Optimal Control of Queueing Systems with Multiple Heterogeneous Facilities

Rob Shone

School of Mathematics

Cardiff University

A thesis submitted for the degree of

Doctor of Philosophy

2014

Abstract

This thesis discusses queueing systems in which decisions are made when customers arrive, either by individual customers themselves or by a central controller. Decisions are made concerning whether or not customers should be admitted to the system (admission control) and, if they are to be admitted, where they should go to receive service (routing control). An important objective is to compare the effects of "selfish" decision-making, in which customers make decisions aimed solely at optimising their own outcomes, with those of "socially optimal" control policies, which optimise the economic performance of the system as a whole. The problems considered are intended to be quite general in nature, and the resulting findings are therefore broad in scope.

Initially, $M/M/1$ queueing systems are considered, and the results presented establish novel connections between two distinct areas of the literature. Subsequently, a more complicated problem is considered, involving routing control in a system which consists of heterogeneous, multiple-server facilities arranged in parallel. It is shown that the multiple-facility system can be formulated mathematically as a Markov Decision Process (MDP), and this enables a fundamental relationship to be proved between individually optimal and socially optimal policies which is of great theoretical and practical importance. Structural properties of socially optimal policies are analysed rigorously, and it is found that 'simple' characterisations of socially optimal policies are usually unattainable in systems with heterogeneous facilities. Finally, the feasibility of finding 'near-optimal' policies for large-scale systems by using heuristics and simulation-based methods is considered.

Declaration

This work has not been submitted in substance for any other degree or award at this or any other university or place of learning, nor is being submitted concurrently in candidature for any degree or other award.

Signed Date

STATEMENT 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of PhD.

Signed Date

STATEMENT 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references. The views expressed are my own.

Signed Date

STATEMENT 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed Date

Acknowledgements

I would like to thank my supervisors Vincent Knight, Paul Harper and Janet Williams for their valuable support and guidance during my research. I would also like to thank John Minty and Kevin Glazebrook for helpful discussions which assisted my progress.

I would also like to extend my gratitude to Cardiff School of Mathematics for funding my research and enabling me to spend 3 years working on enjoyable and challenging problems.

List of publications

- R. Shone, V.A. Knight and J.E. Williams. Comparisons between observable and unobservable $M/M/1$ queues with respect to optimal customer behavior. *European Journal of Operational Research*, 228:133-141, 2013. [161]
- R. Shone, V.A. Knight, P.R. Harper, J.E. Williams and J. Minty. Containment of socially optimal policies in multiple-facility Markovian queueing systems. *Under submission.*

List of presentations

- R. Shone, V.A. Knight, P.R. Harper, J.E. Williams. Comparing observable and unobservable $M/M/1$ queues. *SCOR conference*, April 2012, Nottingham.
- R. Shone, V.A. Knight, P.R. Harper, J.E. Williams. Price of anarchy in queueing systems. *OR54 conference*, September 2012, Edinburgh.
- R. Shone, V.A. Knight, P.R. Harper, J.E. Williams. Optimal control of queueing systems. *EURO conference*, July 2013, Rome, Italy.
- R. Shone, V.A. Knight, P.R. Harper, J.E. Williams. Heuristics for the optimal routing of customers in queueing systems with heterogeneous service stations. *IFORS conference*, July 2014, Barcelona, Spain.

Contents

1 Introduction

This thesis will discuss queueing systems in which the behaviour of individual customers is subject to some form of *control*. More specifically, these systems require *decisions* to be made at regular intervals which correspond to the inter-arrival times of customers who enter the system. Naturally, given a choice between various possible decisions at a particular point in time, one would wish to choose the option which would yield the best expected result with respect to a particular criterion. As such, the vast majority of this thesis is concerned with a particular type of *optimisation problem*, which will be discussed and analysed in rigorous detail in the chapters that follow.

Queueing systems can be formulated mathematically as stochastic processes which evolve according to random probability distributions. The analytical study of these processes has been a popular subject of mathematical research for many decades, and the resulting body of knowledge is usually referred to as *queueing theory*; some notable texts include [11, 67, 100, 125, 132, 178].

Classical queueing theory often involves the characterisation of the limiting or *equilibrium* behaviour of a particular system, assuming that such behaviour exists. For example, one might wish to determine the expected length of a queue at an arbitrary point in time, or an individual customer's expected waiting time in the system. Throughout this thesis, the queueing systems considered will be associated with certain costs and rewards accrued when particular events take place (for example, the fulfilments of customers' service requirements), and an important measure of a system's performance will be the *average net reward* (after subtraction of costs) earned per unit time. Given a full description of the dynamics of a particular system, *and* a complete specification of the behaviour of individual customers, it will usually be possible to use well-established techniques from classical queueing theory to compute performance measures such as average queue lengths, average waiting times and average net rewards. However, in order to seek a decision-making strategy which *optimises* the system's performance with respect to a particular performance measure, it will be necessary to go slightly beyond the realms of traditional queueing theory and instead rely upon *behavioural* queueing theory and the rather broader field of stochastic optimisation.

Prior to the groundbreaking work of Naor in the 1960s (see [2, 131]), applied queueing theory was somewhat limited in scope. This state of affairs is reflected in a "letter to the editor" [115] published

by Leeman in 1964, which provides a qualitative discussion of the limitations of the field. Leeman suggests that operational researchers seldom consider the possibility of introducing or changing prices in order to optimise queueing systems, despite the existence of an accepted economic principle which states that the optimal allocation of scarce resources requires that a cost be charged to users of these resources. Leeman discusses how, for example, 'check-out' fees charged by supermarkets or 'boarding fees' charged by taxi firms might be utilised in order to reduce queue lengths, and concludes his argument by remarking that the role of price-setting in controlling queueing systems ought to be an important consideration for administrators in a capitalist economy.

Another letter by Saaty [151] responds directly to Leeman's arguments by acknowledging the potential of a pricing scheme to alter behaviour in queueing systems. However, he also raises concerns over whether these methods might have a detrimental effect on those "under-privileged" members of society who, he suggests, should not be burdened by having to arrange their purchasing of essential items such as food in such a way as to avoid having to pay exorbitant charges. Saaty suggests that applying congestion-related charges to "luxury-type" queues should be a feasible option, but that queues associated with basic human needs should be treated with due caution.

Naor [131] refers to the qualitative discussions of Leeman and Saaty in his paper on the regulation of queue sizes by levying tolls. Naor's paper considers an observable single-server queue which evolves according to Markovian distributions; specifically, customer arrivals occur according to a time-homogeneous Poisson process, and service times are exponentially distributed (in Kendall's notation, this is an $M/M/1$ queue; see [98]). Customers arriving in the system observe the length of the queue and then decide either to join it and await service, or 'balk' by leaving the system immediately. Naor distinguishes between two different scenarios: one in which arriving customers make decisions which optimise their *own* economic outcomes, and another in which customers adopt a strategy which maximises the collective welfare of *all* customers entering the system. The former scenario is referred to as "individually optimal", while the latter is "socially optimal".

In Naor's system, the assumption of *observability* implies that the length of the queue is always known when decisions are made, and this enables individually optimal and socially optimal customer strategies to be characterised in a relatively simple way; this will be discussed further in Chapter 2 of this thesis. In fact, Naor's results establish the general principle that exercise of narrow self-

interest by all customers does not optimise public good (a recurring theme in much of the related literature; see for example, [103, 203, 204]). This implies that some form of intervention is required on the part of a central administrator in order to maximise the overall welfare of society. Indeed, Naor's results justify the arguments of Leeman by showing that self-interested customers can be induced to behave in a socially optimal manner via the levying of an admission toll.

The task of finding a socially optimal strategy for customers to follow is not especially difficult in the case of an observable $M/M/1$ queue, and Naor's findings will be discussed further in Chapter 2. However, the situation becomes more complicated when one considers systems with *multiple queues*. Chapter 3 will introduce a formulation for a multiple-facility queueing system, in which the number of queues $N \geq 2$ is arbitrary, and every customer who arrives may join (or be directed to) any one of the N queues; in addition, the option of balking will also be available. The task of finding a socially optimal strategy for an observable multiple-facility system is far from trivial; in fact, this represents an optimisation problem which can be tackled effectively within the mathematical framework of a *Markov Decision Process* (MDP). MDPs will be discussed in much greater detail in Chapter 3. In the meantime, this introduction will provide some background notes on the historical development of MDPs and the related area of stochastic dynamic programming.

According to Puterman [141], the study of "sequential decision processes" can be traced back to the work of Bellman [10] and Howard [88] in the late 1950s. Bellman himself, in his ground-breaking work *Dynamic Programming* [10], refers to MDPs as "multi-stage decision processes" and coins the term "dynamic programming" to describe their analytical treatment. He goes on to observe that multi-stage decision processes arise in a "multitude of diverse fields", including "economic, industrial, scientific and even political spheres". He also comments that such problems are "too vast in portent and extent to be treated in the haphazard fashion that was permissible in a more leisurely bygone era", and advocates the development of a more rigorous approach.

Bellman demonstrates remarkable foresight by suggesting, in [10], several of the most intensely-studied problems in operational research today as possible application areas for his work. These areas include optimal inventory control, scheduling of patients in clinics, servicing of aircraft, investment policies and sequential testing procedures. He describes the optimal solution of such problems as a "vast untamed jungle", but recognises the difficulties involved and explicitly refers

to the "curse of dimensionality" as just one example of the challenges to be faced.

Howard [88] comments that Bellman's exposition of dynamic programming in [10] has given "hope" to "those engaged in the analysis of complex systems", but notes that this hope has been "quickly diminished by the realisation that more problems could be formulated by this technique than could be solved". Consequently, Howard states that his objective is to provide an "analytic structure for a decision-making system that is at the same time both general enough to be descriptive and yet computationally feasible". In [88], he defines many of the most familiar concepts associated with MDPs today, including discounted costs, value iteration and policy improvement.

Further contributions from the early 1960s include those of de Ghellinck [34], who discusses the formulation of an MDP as a linear program, d'Epenoux [36], who presents a synthesis of linear and dynamic programming techniques applied to a problem of production and inventory control, and Manne [124], who also focuses on the relationship between the "traditionally distinct" areas of dynamic and linear programming in the context of an inventory control model. Other notable contributions during the 1960s were made by White [195], Odoni [137], MacQueen [122], Strauch [172] and Smallwood [163]. Following the pioneering works of Bellman and Howard, several other authors published books on MDPs in the 1970s; see, for example, [37, 110, 130, 144].

Puterman's book, *Markov Decision Processes* [141] has been described as "the current high-water mark" of MDP literature by Powell [140], and is notable for its strong theoretical approach. Other notable books on MDPs which post-date Puterman's include those of Bertsekas [13], Filar and Vrieze [50], Hernandez-Lerma and Lasserre [82], Hu and Yue [90] and Guo and Hernandez-Lerma [71]. The latter is notable for its exclusive focus on continuous-time MDPs (CTMDPs). Another text which has particular relevance to Chapter 7 of this thesis is *Approximate Dynamic Programming* by Powell [140]. Powell focuses on methods for overcoming the so-called "curse of dimensionality" in MDPs by making large-scale optimisation problems more computationally tractable.

Typical 'real-life' applications of MDPs are discussed in literature surveys including [94, 170, 197]. Beginning in the late 1960s and early 1970s, numerous MDP-related papers were published in areas including inventory and production (see [99, 164, 176, 179, 196]), maintenance and repair (see [33, 42, 84, 165]) and finance and investment (see [38, 128, 136, 150]). This thesis will consider problems involving the control of queueing systems, and a comprehensive survey of the related literature has

been provided by Stidham and Weber [170]. Examples of queueing systems frequently arise in areas including telecommunications, traffic systems, computer systems, manufacturing processes and others. The field of *routing problems*, in which traffic must be routed through a network which is adversely affected by congestion, may be seen as a related area (see [94]). Most of the problems considered in this thesis will be related to *admission control* and *routing control*. Some notable papers with relevance to this area include [46, 93, 108, 118, 129, 149, 169, 192, 201].

The preceding discussion has provided only a brief overview of the literature relevant to this thesis. Further important references of specific interest to the topics discussed in this thesis will be provided in the individual chapters and sections to which they have the greatest relevance.

Chapter 2 of this thesis will consider an $M/M/1$ queueing system. Chapter 3 will introduce a formulation for a multiple-facility queueing system, which will form the basis of most of the work in Chapters 3-7. Some variations of the multiple-facility system will be considered at various stages of this thesis, but the assumptions listed below will remain consistent throughout.

- All of the queueing systems considered are subject to a cost and reward structure whereby rewards are earned when customers are served, but 'holding costs' are incurred while customers are kept waiting in the system. Full details will be given in Chapters 2 and 3.
- When a customer arrives in the system, a routing decision is made which is *permanent and irrevocable*; so for example, a customer cannot join a queue and then leave the system without receiving service. In this respect, customers are assumed to be *infinitely patient*.
- The option of *balking* (i.e. departing from the system immediately, without joining a queue) is available when a customer arrives. Balking does not earn any reward or cost.

The queueing systems considered throughout this thesis are intended to (potentially) be applicable in a wide range of real-world contexts; for example, the multiple-facility queueing system introduced in Chapter 3 may be regarded as a suitable model for a supermarket check-out area, or a telephone call centre. It should be noted that the rewards and holding costs in a queueing system need not necessarily represent monetary values; for example, the reward earned after a customer is served at a particular queue might simply quantify the quality or desirability of the service received, and similarly a 'holding cost' might simply quantify the amount of inconvenience or discomfort suffered

by a customer while they experience delays. Furthermore, the 'customers' in a queueing system might not necessarily represent *people*; for example, they might be items processed on a production line. Indeed, the assumption of 'infinite patience' would tend to suggest that this type of context would be more appropriate, since inanimate objects cannot become impatient! Due to the general nature of the queueing systems considered in this thesis, very few references to specific real-world applications will be made. However, this introduction will briefly discuss one particular application which in fact provided much of the original motivation for the work in this thesis.

Individually optimal and *socially optimal* customer strategies were briefly discussed earlier in the context of Naor's paper [131]. Comparisons between these two types of customer behaviour are particularly relevant to recent developments in the English healthcare system. Since 2006, patients requiring specialist healthcare treatment in England have been entitled to a choice between different providers under the National Health Service (NHS) constitution. A report published by the King's Fund [40] has examined the various factors which may be important to patients when choosing between different healthcare providers, and the implications for the NHS as a whole. The report predicts that patients will increasingly request a choice as they become more aware of their right to choose, and that an increasing number of different factors will influence their choices. Knight and Harper [101] have used game theoretical analyses to investigate the implications of allowing a similar freedom of choice to patients requiring knee replacement surgery in Wales.

If one makes the plausible assumption that patients seeking hospital treatment will be inclined to choose the most convenient option for themselves (as opposed to taking into account the possible interests of *other* patients), then it is clear that allowing patients to choose their own healthcare provider creates a queueing system in which the behaviour of customers (in this case, patients) is *individually optimal*. On the other hand, if patients are denied a free choice and are instead directed by a central authority, then it is conceivable that decisions might be chosen in such a way as to optimise some measure of the system's performance; this would correspond to the *socially optimal* case. Thus, it is clear that comparisons between individually and socially optimal customer behaviour may be relevant in healthcare systems and other public service settings, in which it is important to consider the effects of allowing individuals to make their own choices.

Realistically, the option of *balking* can be accommodated in most queueing systems which operate

without any constraints on the proportion of customers which must receive service; for example, a customer in a supermarket might decide to skip the queue and go home without any shopping, or the manager of a production line might decide to withhold some items in order to avoid overburdening the system. In addition, the assumption that balking does not incur any cost or reward can be made without loss of generality in the context of the optimisation problems considered in this thesis. In many real-world contexts, one might wish to associate balking with a certain *penalty* or *rejection* cost; for example, a shopper who enters a supermarket and then leaves without purchasing any items has clearly wasted some of their own time, whether or not they are able to conveniently acquire the required items from elsewhere. However, in this thesis it will be possible to assume that balking is *always costless* by relying upon the principle that the rewards earned when customers receive service can, if necessary, be adjusted by a fixed amount which corresponds to any cost associated with balking. This will be clarified in the subsequent chapters.

In the context of patient choice in the NHS, 'balking' might represent the decision of a patient to seek private treatment, or to rely upon other remedies outside the provision of the NHS. Whether or not these options might be associated with a *penalty cost* is unimportant from a mathematical perspective, due to the explanation given in the previous paragraph. However, it is also important to note that the cost of balking should always remain the same, regardless of how often it is chosen; in this respect, it may be regarded as a 'fail-safe option' which allows customers to avoid incurring expensive waiting costs as a result of high levels of congestion in the system. In a healthcare context, therefore, the association between balking and seeking outside treatment must rely on an implicit assumption that the 'attractiveness' of outside treatment to an individual patient is not affected in any significant way by the tendency of other patients to choose the same option. Of course, there are always modelling issues to consider in real-world applications, and (as stated previously) these issues are somewhat peripheral to the content of this thesis, since it considers well-defined optimisation problems which are fairly general in terms of their physical applicability.

Brief descriptions of the remaining chapters in this thesis are provided below.

- Chapter 2 will address a problem involving admission control in an $M/M/1$ queue. Two variants of the problem will be considered: one in which the queue is observable, and one in which it is unobservable. In the latter case, decisions must be made independently of the

length of the queue. The observable and unobservable cases will be compared with respect to individually optimal customer behaviour, and also with respect to socially optimal behaviour. Various novel results will be proved involving comparisons between the observable and unobservable systems. However, Chapters 3-7 of this thesis will tend to focus on *observable* queueing systems, and therefore the results in Chapter 2 should be regarded as being largely self-contained, without a strong bearing on the results in later chapters.

- Chapter 3 will introduce a mathematical formulation for the multiple-facility queueing system which forms the basis for most of the research in this thesis. It will be shown that the queueing system can be formulated as a discrete-time Markov Decision Process (MDP). This enables the techniques of dynamic programming to be applied, provided that one assumes a *finite* state space. Some well-known computational algorithms from the literature will be presented. The final part of this chapter will discuss analytical techniques for finite-state MDPs including proofs based on dynamic programming and stochastic coupling arguments.

- Chapter 4 will prove an important relationship between individually optimal and socially optimal policies in a multiple-facility queueing system. This property may be regarded as a generalisation of a similar property which is already known (due to the results of Naor in [131]) to hold in the case of an $M/M/1$ queue. The effect of the system demand rate (i.e. the rate per unit time at which customers arrive in the system) on socially optimal policies will also be investigated. The final section in this chapter will consider an extension involving heterogeneous customer classes, which will later be revisited in Chapter 7.

- Chapter 5 will investigate the structural properties of socially optimal policies. It will be shown that, unfortunately, certain 'common-sense' properties of optimal policies may be difficult (or even impossible) to prove using arguments based on dynamic programming. Certain results will be proved for special cases of the N-facility system introduced in Chapter 3, which will prove to be useful in later chapters. The final section in this chapter will present computational algorithms (based on results from earlier sections) which appear to be capable of improving upon the efficiency of standard dynamic programming algorithms.

- Chapter 6 will consider heuristic methods for finding *near-optimal* policies in systems where dynamic programming algorithms are rendered ineffective due to the size of the finite state

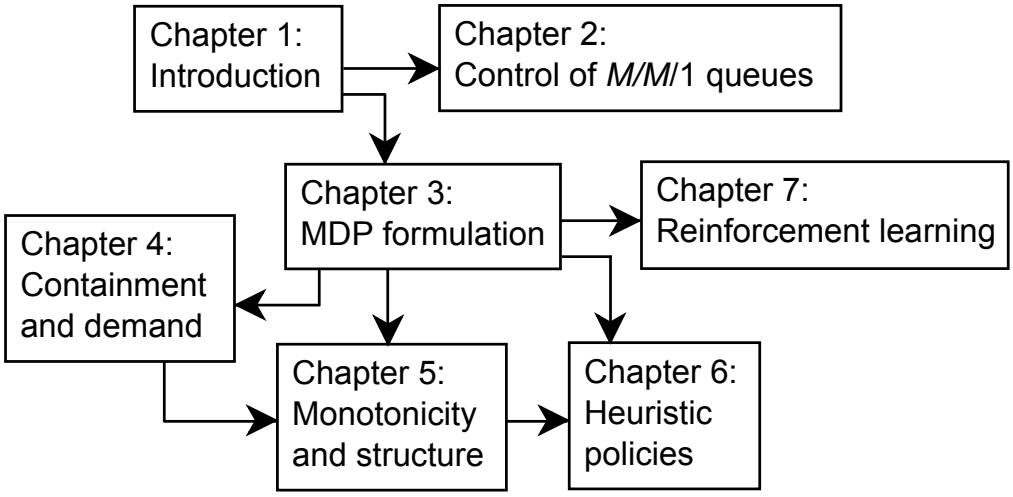


Figure 1.1: A structural map of this thesis, showing how the first seven chapters are related.

space that must be explored. These methods will be based on techniques which have been discussed in the literature. Various interesting properties of these heuristic policies will be proved analytically, and their ability to consistently find near-optimal policies will be tested via numerical experiments involving randomly-generated problem instances.

- Chapter 7 will discuss an alternative method for finding near-optimal policies, referred to as *reinforcement learning* (RL). Essentially, this approach involves simulating the random evolution of a system and gradually 'learning' a near-optimal policy through exploration and experience. Numerical experiments, similar to those in Chapter 6, will be conducted in order to test the performances of certain RL algorithms. Later sections in this chapter will examine how RL algorithms might be adapted in order to cope with extremely vast state spaces, non-exponential distributions and systems with heterogeneous customer classes.

- Finally, Chapter 8 will present conclusions and suggest possible further work.

Throughout the chapters of this thesis, numerous theoretical results (theorems, lemmas etc.) will be presented. Many of these will be original results for which proofs will be given, either in the main text itself or in one of the appendices. In some cases, however, it will be necessary to state a result which has already been published in the literature. In order to enable a distinction to be made, an asterisk (*) will precede the statement of any theorem, lemma or corollary which is *not* an original result of this thesis. In these cases, appropriate references will be provided.

2 Control of $M/M/1$ queues

2.1 Introduction

The study of classical queueing theory usually begins with $M/M/1$ queues. The basic dynamics of $M/M/1$ queues are described in many texts; see, for example, [67, 132]. Essentially, customers arrive at random intervals and await service at a single facility which serves one customer a time. The random process by which customers arrive is a *Poisson process* (see, e.g. [146, 167, 171]) with an intensity rate $\lambda > 0$, and service times are exponentially distributed with mean $\mu^{-1} > 0$. Customers are served in order of arrival, and leave the system permanently after their service is complete. If one assumes that all customers join the queue and that $\lambda < \mu$, then it is possible to formulate the system as a continuous-time Markov chain (see Appendix A.2) and obtain its stationary or *steady-state* distribution, which in turn can be used to derive remarkably simple formulae for steady-state performance measures such as the expected length of the queue or the expected waiting time of a customer in the system; these formulae can be found in the texts mentioned above.

Of course, the assumption that all customers join the queue implies that no *control* is exercised over the system. Control, in this context, refers to the act of making a decision which influences future events. Only one form of control will be considered in this chapter: *admission control*. Suppose that when a customer arrives at the $M/M/1$ queue, there are two options available: they may join the queue and await service, in which case it is assumed that they remain in the system until their service is complete, or alternatively they may exit from the system immediately without receiving service. The act of joining the queue will be referred to from this point onwards as *joining*, and the act of leaving the system will be referred to as *balking*. Hence, the only points in time at which control may be exercised are the random points in time at which new customers arrive, since the decision for each individual customer (either 'join' or 'balk') is assumed irrevocable.

The question of whether decisions are made by individual customers themselves or whether there is a central controller who makes decisions on their behalf has no physical bearing on the mathematical results in this chapter. However, it is natural to suppose that if individual customers are responsible for their own outcomes, they will be inclined to make decisions which serve their own interests. On the other hand, if decisions are controlled centrally, then these decisions may be taken with a

broader objective in mind, such as the optimisation of a particular performance measure for the system as a whole. These contrasting objectives allude to the game-theoretic concept of *selfish* versus *non-selfish* decision-making, which is an important theme throughout this thesis. The sub-optimality of greedy or 'selfish' customer behaviour in the context of overall social welfare has been studied in many of the classical queueing system models, including $M/M/1$, $GI/M/1$, $GI/M/c$ and others; see, for example, [44, 103, 120, 121, 131, 168, 203, 204]. More recently, this theme has been explored in applications including queues with setup and closedown times [174], queues with server breakdowns and delayed repairs [189], vacation queues with partial information [68], queues with compartmented waiting space [43] and patient flow in healthcare systems [101, 102].

Problems involving admission control in queueing systems have been studied extensively in the literature; a comprehensive survey of this work is provided by Hassin and Haviv [76]. However, in the preface of their work, Hassin and Haviv comment that the field of behavioural queueing theory is "lacking continuity" and "leaves many issues uncovered". This is hardly surprising, since almost every possible queueing system formulation that one might conceive has the potential to be modified or generalised in some respect, and as such it is rare to find two independent pieces of work which address exactly the same problem. One important consideration to be made when formulating an admission control problem is the amount of information that should be available to customers (or, in the case of central control, the central decision-maker). In this chapter, the term *'observable'* is used to refer to a system in which the decision-maker is always aware of the number of customers present (referred to as the *state* of the system), and can use this information to inform their decision-making. On the other hand, in an *'unobservable'* system, decisions must be taken independently of the system state, based only on knowledge of the system parameters and the distribution of waiting times; further details will be presented in the next section.

Comparisons between observable and unobservable $M/M/1$ queues are related to a broad category of problems in which the amount of information disclosed to customers (and, possibly, its completeness or reliability) is at the discretion of the service provider. There has been considerable recent interest in this area. For example, Allon et al. [3] consider an $M/M/1$ system in which a firm can influence its customers' behaviour by using "delay announcements". It is found that some level of "intentional vagueness" on the part of the firm may be beneficial in certain circumstances. Guo and Zipkin [69] (see also [70, 71]) also study a single-server Markovian system and find that providing

"more accurate delay information" may improve system performance, but this is dependent upon other factors. Hassin [75] considers a number of sub-models of $M/M/1$ queues and, in each case, examines the question of whether or not the service provider is motivated to reveal information. The fact that all of these publications adopt the classic $M/M/1$ model for their analyses might arguably be seen as an indication that the theme of restricting customer information in queueing systems is a young and emerging one, with strong potential for future development.

Naor [131] is usually recognised as the first author to compare "self-optimisation" with "overall optimisation" (or social optimisation) in the case of an observable $M/M/1$ queue with linear waiting costs and a fixed value of service. Edelson and Hildebrand [44] consider a model similar to Naor's, but without the assumption of observability. While the respective properties of observable and unobservable $M/M/1$ queueing systems have been analysed extensively by Naor, Edelson and Hildebrand and many others (see [76] for further references), comparisons between the two types of system are not abundant in the literature. Indeed, one might observe that both system types constitute their own sub-discipline of the field. This is logical to some extent, as the mathematical techniques that one employs will depend on whether or not the state of the system can be observed exactly. For example, modern analysis of unobservable queueing systems often takes place in a game theoretical setting involving flow control (see, for example, [135] for a discussion of routing games), while a more natural framework for the modelling of an observable queueing system is a continuous-time Markov Decision Process, as discussed in Chapter 3 of this thesis.

While the comparison of observable and unobservable queueing systems might involve the bridging of two very different methodological areas, it is a worthy endeavour due to the potential insights that can be gained into problems involving the optimal control of information. The results in this chapter offer some insight into the effects of suppressing information on queue lengths from newly-arrived customers (or, if decisions are controlled centrally, the central decision-maker). For example, suppose there is a third party who earns a fixed amount of revenue for every customer who joins the queue, as opposed to balking. It is not trivial to determine whether average customer throughput rates will be greater in the *observable* case, or whether the interests of the third party would be better served by making the system *unobservable*. Indeed, this will depend on whether or not customers make decisions selfishly, among other factors. The first task in this chapter will be to provide a mathematical formulation for the queueing system under consideration. It will then be

appropriate to summarise known results from the literature concerning selfishly and socially optimal customer behaviour, before proceeding to compare the observable and unobservable models with respect to optimal queue-joining rates and other relevant performance measures.

2.2 Model formulation

The queueing system to be considered throughout this chapter is an $M/M/1$ queue with linear holding costs and a fixed value of service, similar to Naor's model in [131]; however, there is no prior assumption that the system is observable, since an objective in later sections will be to compare the observable and unobservable cases. Customers arrive according to a *Poisson process* with parameter $\lambda > 0$. The definition below can be found in Ross [146], p. 304.

Definition 2.2.1. ***(Poisson process)***

The counting process $\{N(t), t \geq 0\}$ *is said to be a* Poisson process *with parameter* $\lambda > 0$ *if:*

1. $N(0) = 0$*;*
2. *The process has independent increments;*
3. *The number of events occurring in any interval of length* t *has a Poisson distribution with mean* λt*. That is, for all values* $t \geq 0$ *and* $u \geq 0$*:*

$$P\big(N(t+u) - N(u) = n\big) = e^{-\lambda t}\frac{(\lambda t)^n}{n!} \qquad (n = 0, 1, ...).$$

It should be noted that throughout this thesis, Poisson processes are assumed to be time-homogeneous unless stated otherwise; that is, the parameter λ does not vary with time.

Customers' service times are independently and identically distributed according to an exponential distribution with parameter $\mu > 0$. There is a holding cost $\beta > 0$ per customer per unit time for keeping customers waiting in the system, and a fixed reward $\alpha > 0$ is earned after a service completion. It is assumed that $\alpha > \beta/\mu$ in order to avoid the case where a customer would be unwilling to wait even for their service, which would lead to trivialities. The queue discipline is First-Come-First-Served (FCFS) and each newly-arrived customer either joins the queue, in which case they remain in the system until their service is complete, or balks from the system and does

not return. A diagrammatic representation of the system is provided in Figure 2.1.

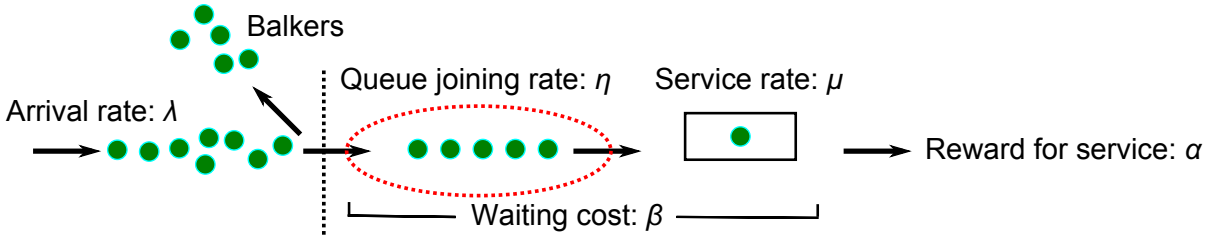


Figure 2.1: A diagrammatic representation of the queueing system.

The relative traffic intensity for the system is denoted $\rho = \lambda/\mu$. As mentioned in the introduction, a typical assumption which ensures stability in an $M/M/1$ queue is that $\lambda < \mu$ and hence $\rho < 1$; however, this assumption is *not* made in this chapter. This is because the attainment of steady-state conditions merely requires the effective queue-joining rate (as opposed to the system arrival rate) to be smaller than the service rate μ. It will be shown in the next section that the effective queue-joining rate always satisfies this condition under both types of customer behaviour to be considered in this chapter, regardless of whether or not the system is observable.

Let $\tilde{\eta}^{[O]}$ and $\eta^{*[O]}$ denote, respectively, the steady-state selfishly and socially optimal queue-joining rates (per unit time) when the system is observable, and let $\tilde{\eta}^{[U]}$ and $\eta^{*[U]}$ denote the corresponding measures when the system is unobservable. Of course, it is necessary to define 'selfish optimality' and 'social optimality' more concretely in order to determine what these optimal joining rates should be; this will be done in Section 2.3. Similar notation is adopted for other system performance measures such as expected waiting times, etc.; this is summarised in Table 2.1.

Note on terminology: The emphasis in the remainder of this chapter will be on comparisons between observable and unobservable systems. As such, when "two types of system" are mentioned, these two types are understood to be *observable* and *unobservable*. When "two types of optimal joining rate" are mentioned, these two types are *selfishly optimal* rates and *socially optimal* rates, both of which will be defined in the next section. When "the equality of selfishly (or socially) optimal joining rates" is discussed, this refers to an observable and an unobservable system sharing the same selfishly (or socially) optimal effective queue-joining rate per unit time.

Output measures	**Observable system**		**Unobservable system**	
	Selfish opt.	Social opt.	Selfish opt.	Social opt.
Optimal joining rate per unit time	$\tilde{\eta}^{[O]}$	$\eta^{*[O]}$	$\tilde{\eta}^{[U]}$	$\eta^{*[U]}$
Prob. of n customers in system	$\tilde{P}_n^{[O]}$	$P_n^{*[O]}$	$\tilde{P}_n^{[U]}$	$P_n^{*[U]}$
Mean Busy Period (MBP)	$\widetilde{MBP}^{[O]}$	$MBP^{*[O]}$	$\widetilde{MBP}^{[U]}$	$MBP^{*[U]}$
Expected no. of customers in system	$\tilde{L}^{[O]}$	$L^{*[O]}$	$\tilde{L}^{[U]}$	$L^{*[U]}$
Mean waiting time in system	$\tilde{W}^{[O]}$	$W^{*[O]}$	$\tilde{W}^{[U]}$	$W^{*[U]}$

Table 2.1: Summary of notation for system output measures (assuming steady-state conditions in all cases).

2.3 Summary of known results

In an *observable* $M/M/1$ queue, each customer arriving in the system is able to calculate their expected cost of waiting as a function of the number of customers already present (that is, the number of customers either waiting in the queue or being served) upon their arrival. As mentioned previously, the number of customers present is referred to as the *state* of the system. It is assumed in this thesis that customers are risk-neutral, and hence customers acting *selfishly* will choose to join the queue if their expected cost of waiting is smaller than the reward for service α. In order to avoid ambiguity, it will also be assumed throughout this chapter (as in [131]) that self-interested customers choose to join the queue if their expected cost of waiting is equal to α.

The expected total cost incurred by a customer for joining under state $n \in \mathbb{N}_0$ is given by $(n+1)\beta/\mu$.

It follows that there exists an integer n_s such that newly-arrived self-interested customers choose to join the queue if and only if there are fewer than n_s customers present when they arrive. Naor [131] derives the following expression for n_s in terms of the system parameters:

$$n_s = \left\lfloor \frac{\alpha\mu}{\beta} \right\rfloor, \tag{2.3.1}$$

where $\lfloor \cdot \rfloor$ denotes the 'floor' function, i.e. the integer part. Equivalently, it may be said that selfish customers follow a *threshold strategy* with threshold n_s. In the case of *social* optimisation, one aims to find a strategy for customers to follow which maximises some quantifiable measure of the 'overall social welfare'. In this chapter, the overall social welfare is measured by the *expected long-run average reward per unit time* earned by the system. Suppose customers follow a common threshold strategy with threshold $n \in \mathbb{N}$; that is, joining is chosen if and only if there are fewer than n customers in the system when the decision is made. Using standard results for finite-capacity $M/M/1$ queues (see [67], p. 74), the expected long-run average reward g_n is then:

$$g_n = \begin{cases} \lambda\alpha \left(\dfrac{1-\rho^n}{1-\rho^{n+1}} \right) - \beta \left(\dfrac{\rho}{1-\rho} - \dfrac{(n+1)\rho^{n+1}}{1-\rho^{n+1}} \right), & \text{if } \rho \neq 1, \\ \lambda\alpha \left(\dfrac{n}{n+1} \right) - \beta \left(\dfrac{n}{2} \right), & \text{if } \rho = 1, \end{cases} \tag{2.3.2}$$

where (in each of the two cases) the expression in the first set of parentheses is the steady-state probability that fewer than n customers are present in the system, and the expression in the second set of parentheses is the expected number of customers in the system given that a threshold n is in effect. Naor showed that g_n is maximised by $n = n_o = \lfloor v_o \rfloor$, where v_o satisfies:

$$\begin{cases} \dfrac{v_o(1-\rho) - \rho(1-\rho^{v_o})}{(1-\rho)^2} = \dfrac{\alpha\mu}{\beta}, & \text{if } \rho \neq 1, \\ \dfrac{v_o(v_o+1)}{2} = \dfrac{\alpha\mu}{\beta}, & \text{if } \rho = 1. \end{cases} \tag{2.3.3}$$

Importantly, Naor also showed that $n_o \leq n_s$, which is in keeping with the general principle that selfish users create busier systems; this principle will be seen again in later chapters of this thesis. The selfishly and socially optimal thresholds, n_s and n_o, can be used in conjunction with results from finite-buffer queueing theory (see [67], p. 77) to derive expressions for the effective queue-joining rates $\tilde{\eta}^{[O]}$ and $\eta^{*[O]}$ which appear in Table 2.1. These expressions are:

$$\tilde{\eta}^{[O]} = \begin{cases} \lambda\left(\dfrac{1-\rho^{n_s}}{1-\rho^{n_s+1}}\right), & \text{if } \rho \neq 1, \\ \lambda\left(\dfrac{n_s}{n_s+1}\right), & \text{if } \rho = 1. \end{cases} \tag{2.3.4}$$

$$\eta^{*[O]} = \begin{cases} \lambda\left(\dfrac{1-\rho^{n_o}}{1-\rho^{n_o+1}}\right), & \text{if } \rho \neq 1, \\ \lambda\left(\dfrac{n_o}{n_o+1}\right), & \text{if } \rho = 1. \end{cases} \tag{2.3.5}$$

Naturally, it must be the case that $\tilde{\eta}^{[O]} \leq \mu$, since otherwise the system would be unstable and queues would become infinitely long, which would imply that customers were deviating from the threshold strategy. In fact, from (2.3.4) it follows that if $\lambda > \mu$ (equivalently, $\rho > 1$), then $\tilde{\eta}^{[O]} \to \mu$ as $n_s \to \infty$. From Naor's results it then follows that $\eta^{*[O]} \leq \tilde{\eta}^{[O]} \leq \mu$.

Next, suppose the system is *unobservable*. In this case, no information is available about the state of the system when decisions are made. It is therefore reasonable to assume that a common randomised strategy determines the actions of all customers to arrive; see, for example, [8, 9, 44]. Specifically, the common strategy followed by all customers may be represented by a value $p \in [0, 1]$ such that a customer joins the queue with probability p, and balks with probability $(1 - p)$. Due to elementary properties of Poisson processes (see [146], p. 310), it then follows that the process by which customers join the queue is a Poisson process with parameter $\eta = p\lambda$.

The arguments used to derive the optimal queue-joining rates $\tilde{\eta}^{[U]}$ and $\eta^{*[U]}$ are game theoretical in nature; a complete explanation is provided by Bell and Stidham [9]. Firstly, the selfishly optimal queue-joining rate $\tilde{\eta}^{[U]}$ is derived from the (Nash) equilibrium strategy. Let $w(\eta)$ be the expected net reward earned by an individual customer for joining the queue when the common queue-joining rate of all customers (determined by the strategy p) is η. In the trivial case where $w(\lambda) \geq 0$ (so that even the largest possible queue-joining rate, $\eta = \lambda$, results in customers incurring an expected waiting cost which does not exceed the reward α), the equilibrium strategy is $p = 1$, since no customer has an incentive *not* to join the queue. Hence, in this case, $\tilde{\eta}^{[U]} = \lambda$.

Next, consider a non-trivial case where $w(\lambda) < 0$. If $\eta \geq \mu$ (which requires $p > 0$), the expected waiting time of a customer is infinite, and a customer acting in their own interests will join the

queue with probability $p = 0$. It follows that an equilibrium solution requires $\eta < \mu$. In fact, it is evident that in order for the common strategy of customers to be in equilibrium, $w(\eta) = 0$ is required; otherwise, the best response of an individual customer to the strategy $p \in (0, 1)$ followed by others would be to choose either $p = 1$ (if $w(\eta) > 0$) or $p = 0$ (if $w(\eta) < 0$).

Assuming $\eta < \mu$, standard results for infinite-capacity $M/M/1$ queues (see [67], p. 61) imply that an expression for the individual expected net reward $w(\eta)$ is given by:

$$w(\eta) = \alpha - \frac{\beta}{\mu - \eta}. \tag{2.3.6}$$

Hence, by solving the equation $w\left(\tilde{\eta}^{[U]}\right) = 0$, one finds:

$$\tilde{\eta}^{[U]} = \min\left(\mu - \frac{\beta}{\alpha},\ \lambda\right). \tag{2.3.7}$$

As in the observable case, the *socially* optimal joining rate $\eta^{*[U]}$ is defined as the joining rate which maximises the expected long-run average reward per unit time for the system. Let $g(\eta)$ denote the expected average reward given a queue-joining rate η. It is only necessary to consider $\eta < \mu$, since $\eta \geq \mu$ would cause expected waiting costs to tend to infinity, which clearly would not be socially optimal. Assuming $\eta < \mu$, the average reward $g(\eta)$ is given by:

$$g(\eta) = \eta\alpha - \beta\left(\frac{\eta}{\mu - \eta}\right). \tag{2.3.8}$$

By differentiating, one finds that $g(\eta)$ is maximised by setting $\eta = \eta^{*[U]}$, where:

$$\eta^{*[U]} = \min\left(\mu\left(1 - \sqrt{\frac{\beta}{\alpha\mu}}\right),\ \lambda\right). \tag{2.3.9}$$

Note that both (2.3.7) and (2.3.9) are valid joining rates only if $\alpha \geq \beta/\mu$, which follows by one of the initial assumptions in this chapter. Finally, it is easy to verify that $\eta^{*[U]} \leq \tilde{\eta}^{[U]}$ by direct comparison, which again shows that selfish behaviour causes a busier system.

Example 2.3.1. ***(Optimal joining rates in an unobservable*** $M/M/1$ ***queue)***

Consider an *unobservable* $M/M/1$ system with demand rate $\lambda = 1.9$, service rate $\mu = 2$, holding cost $\beta = 5$ per unit time and reward for service $\alpha = 25$. Figure 2.2 shows the results of a simulation experiment, in which 100,000 customer arrivals have been simulated for each of 180 different,

uniformly-spaced values of the Poisson queue-joining rate η between 1 and 1.9. For each value of η, the average net reward $w(\eta)$ earned by a customer (resulting from a full simulation run) is plotted. These simulated results may be compared to the expected theoretical values given by the formula in (2.3.6), which have also been plotted. The simulated results closely match the expected values, and show that the unique value of η which equates the average net reward of a customer to zero is indeed given by $\tilde{\eta}^{[O]} = \mu - \beta/\alpha = 1.8$; this is the selfishly optimal joining rate.

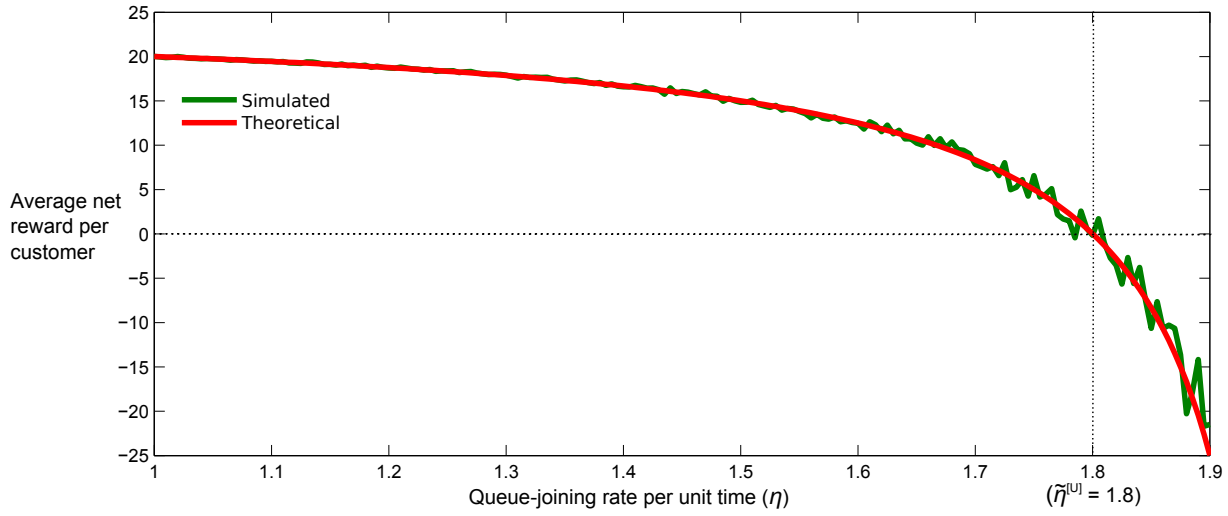


Figure 2.2: Simulated values of $w(\eta)$ for 180 different queue-joining rates η.

On the other hand, consider the problem of social optimisation. Figure 2.3 shows the results of another simulation experiment (using the same values for the system parameters) in which $100,000$ customer arrivals have been simulated for 250 different, uniformly-spaced values of the queue-joining rate η between 0.5 and 1.75. For each value of η, the long-run average net reward per unit time $g(\eta)$ has been plotted. Again, these simulation results may be compared to the theoretical values given by (2.3.8), which have also been plotted. The results confirm that the function $g(\eta)$ is maximised by $\eta^{*[U]} = \mu\left(1 - \sqrt{\beta/(\alpha\mu)}\right) \approx 1.368$, which is the socially optimal joining rate.

This example has assumed an *unobservable* system. One may be interested to determine whether the effective queue-joining rates $\tilde{\eta}^{[O]}$ and $\eta^{*[O]}$ in an *observable* system would take the same values (1.8 and 1.368 respectively), given the same configuration for the system parameters. The next section will address the question of whether or not it is possible for the two types of system to share the same selfishly and socially optimal queue-joining rates. ⊠

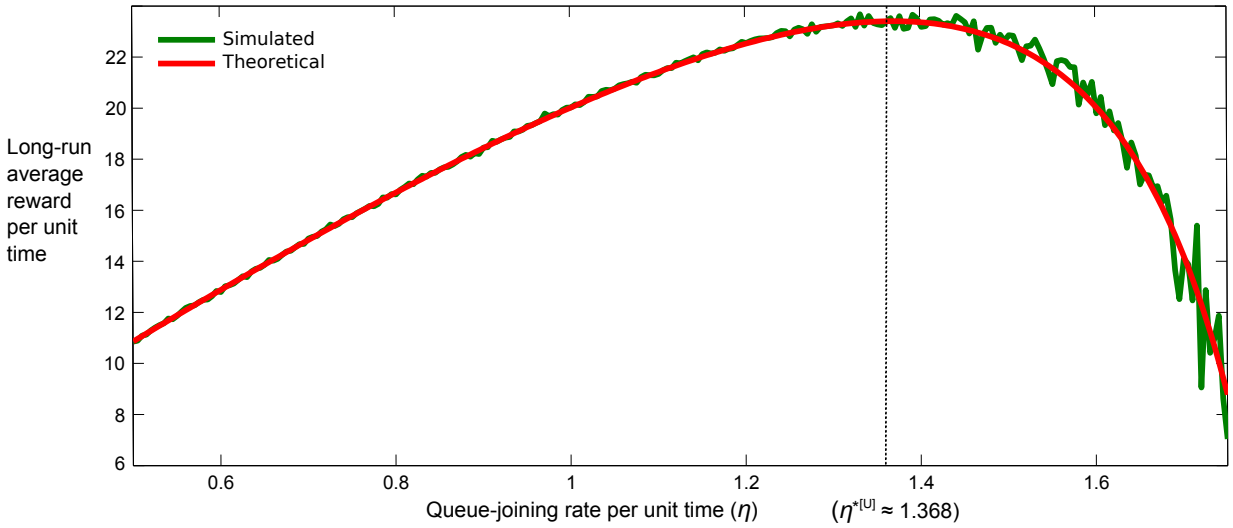


Figure 2.3: Simulated values of $g(\eta)$ for 250 different queue-joining rates η.

2.4 Equality of queue-joining rates

Summarising the results from Section 2.3, the selfishly optimal queue-joining rates $\tilde{\eta}^{[O]}$ and $\tilde{\eta}^{[U]}$ for the observable and unobservable system respectively are given by:

$$\tilde{\eta}^{[O]} = \begin{cases} \lambda\left(\dfrac{1-\rho^{n_s}}{1-\rho^{n_s+1}}\right), & \text{if } \rho \neq 1, \\ \lambda\left(\dfrac{n_s}{n_s+1}\right), & \text{if } \rho = 1, \end{cases} \tag{2.4.1}$$

$$\tilde{\eta}^{[U]} = \min\left(\mu - \beta/\alpha,\ \lambda\right). \tag{2.4.2}$$

On the other hand, the *socially* optimal joining rates are given by:

$$\eta^{*[O]} = \begin{cases} \lambda\left(\dfrac{1-\rho^{n_o}}{1-\rho^{n_o+1}}\right), & \text{if } \rho \neq 1, \\ \lambda\left(\dfrac{n_o}{n_o+1}\right), & \text{if } \rho = 1, \end{cases} \tag{2.4.3}$$

$$\eta^{*[U]} = \min\left(\mu\left(1-\sqrt{\beta/(\alpha\mu)}\right),\ \lambda\right). \tag{2.4.4}$$

In this section, the focus is on investigating the conditions which prescribe the equality of optimal queue-joining rates for the two types of system. The next result establishes the useful fact that, given $\lambda > 0$, the equalities $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ and $\eta^{*[O]} = \eta^{*[U]}$ both require $0 < \rho < 1$.

Lemma 2.4.1.

1. *The equality of selfishly optimal joining rates,* $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$*, requires* $0 < \rho < 1$.

2. *The equality of socially optimal joining rates,* $\eta^{*[O]} = \eta^{*[U]}$*, requires* $0 < \rho < 1$.

Proof. The proof begins with the selfishly optimal case. First, consider $\rho = 1$. Here, the equality of selfishly optimal joining rates implies $\lambda n_s/(n_s+1) = \mu - \beta/\alpha$. Since $\rho = \lambda/\mu = 1$, this is equivalent to $n_s + 1 = \alpha\mu/\beta$. However, $n_s = \lfloor \alpha\mu/\beta \rfloor$ by definition, so this is not possible. Indeed, $n_s + 1$ is strictly greater than $\alpha\mu/\beta$, which implies that $\tilde{\eta}^{[O]} > \tilde{\eta}^{[U]}$ when $\rho = 1$.

Now suppose $\rho > 1$. Since the input parameters α, β and μ are assumed finite, n_s is also finite and hence $\tilde{\eta}^{[O]} < \lambda$, which in turn implies that the equality $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ cannot hold in the case where $\tilde{\eta}^{[U]} = \lambda$. It therefore suffices to consider solutions to the equation $\lambda(1-\rho^{n_s})/(1-\rho^{n_s+1}) = \mu - \beta/\alpha$. Upon dividing by μ, this equation may be cast into a simpler form:

$$\frac{\alpha\mu}{\beta} = \frac{1-\rho^{n_s+1}}{1-\rho} = \sum_{i=0}^{n_s} \rho^i. \tag{2.4.5}$$

Given that $\rho > 1$, (2.4.5) implies $\alpha\mu/\beta > n_s + 1$. However, the definition of n_s in (2.3.1) implies $n_s \leq \alpha\mu/\beta < n_s + 1$, so there is no solution to (2.4.5) when $\rho > 1$. From these arguments it follows that $\tilde{\eta}^{[O]} > \tilde{\eta}^{[U]}$ when $\rho > 1$, which completes the proof of the first statement.

Remark. For the case $\rho = 1$, the argument given above relies on the assumption that selfish customers join the queue if their expected net reward is exactly equal to zero. If it were assumed that customers opted to balk under this scenario, then the selfishly optimal threshold (in the observable case) would be given by $n_s = \lceil \alpha\mu/\beta \rceil - 1$, where $\lceil \cdot \rceil$ denotes the ceiling function. In this case the equality $n_s + 1 = \alpha\mu/\beta$ would be satisfied for all integer values of $\alpha\mu/\beta$, and it would be necessary to change "$0 < \rho < 1$" in the statement of the lemma to "$0 < \rho \leq 1$".

Next, consider the *socially* optimal joining rates. First, suppose $\rho = 1$. In this case, the equality $\eta^{*[O]} = \eta^{*[U]}$ implies $\lambda n_o/(n_o+1) = \mu\left(1 - \sqrt{\beta/(\alpha\mu)}\right)$, which simplifies to:

$$\frac{\alpha\mu}{\beta} = (n_o + 1)^2. \tag{2.4.6}$$

However, by setting $\rho = 1$ in (2.3.3), it follows that $\alpha\mu/\beta$ must also satisfy:

$$\frac{\alpha\mu}{\beta} = \frac{v_o(v_o+1)}{2}. \tag{2.4.7}$$

Since $n_o = \lfloor v_o \rfloor$ by definition, it follows that $v_o < n_o + 1$ and hence $v_o(v_o+1)/2 < (n_o+1)(n_o+2)/2$. Since $(n_o+1)(n_o+2)/2 \leq (n_o+1)^2$ for all $n_o \in \mathbb{N}$, it then follows that (2.4.6) fails to hold when $\rho = 1$. In fact, these arguments imply that $\eta^{*[O]} > \eta^{*[U]}$ when $\rho = 1$.

Next, suppose $\rho > 1$. In this case, the equality $\eta^{*[O]} = \eta^{*[U]}$ implies $\lambda(1-\rho^{n_o})/(1-\rho^{n_o+1}) = \mu\left(1 - \sqrt{\beta/(\alpha\mu)}\right)$, which may be cast into the more simple form:

$$\left(\frac{1-\rho^{n_o+1}}{1-\rho}\right)^2 = \frac{\alpha\mu}{\beta}. \tag{2.4.8}$$

Suppose, for a contradiction, that (2.4.8) holds with $\rho > 1$. Due to (2.3.3), this implies:

$$(1-\rho^{n_o+1})^2 = v_o(1-\rho) - \rho(1-\rho^{v_o}). \tag{2.4.9}$$

Hence, since $v_o < n_o + 1$ and the right-hand side of (2.4.9) is strictly increasing in v_o:

$$(1-\rho^{n_o+1})^2 < (n_o+1)(1-\rho) - \rho(1-\rho^{n_o+1}). \tag{2.4.10}$$

Noting that $(1-\rho) < 0$, (2.4.10) may be put into the following form:

$$(1-\rho)\left(\sum_{i=0}^{n_o}\rho^i\right)^2 > n_o + 1 - \rho\sum_{i=0}^{n_o}\rho^i.$$

It is therefore sufficient (in order to obtain a contradiction) to show:

$$(1-\rho)\left(\sum_{i=0}^{n}\rho^i\right)^2 \leq n + 1 - \rho\sum_{i=0}^{n}\rho^i, \tag{2.4.11}$$

where $n \in \mathbb{N}$ is arbitrary and $\rho > 1$. One may proceed to show that (2.4.11) holds for all integers $n \geq 1$ using induction. When $n = 1$, (2.4.11) simplifies to:

$$\rho^3 - 2\rho + 1 \geq 0. \tag{2.4.12}$$

Let $f(\rho) = \rho^3 - 2\rho + 1$. Then $f(1) = 0$ and $f'(\rho) = 3\rho^2 - 2$, which is positive for $\rho > 1$, so (2.4.11) holds when $n = 1$. As an inductive hypothesis, assume that for arbitrary $k \in \mathbb{N}$:

$$(1-\rho)\left(\sum_{i=0}^{k}\rho^i\right)^2 \leq k + 1 - \rho\sum_{i=0}^{k}\rho^i. \tag{2.4.13}$$

Let $f_n(\rho) = (1-\rho)\left(\sum_{i=0}^{n}\rho^i\right)^2 - (n+1) + \rho\sum_{i=0}^{n}\rho^i$ for $n \geq 1$. In order to show that (2.4.11) holds when $n = k+1$, then (using (2.4.13)) it suffices to show $f_{k+1}(\rho) \leq f_k(\rho)$. That is:

$$(1-\rho)\left(\rho^{2(k+1)} + 2\rho^{k+1}\sum_{i=0}^{k}\rho^i\right) - 1 + \rho^{k+2} \leq 0.$$

Given that $(1-\rho)\rho^{2(k+1)} < 0$ for $\rho > 1$, it is then sufficient to show:

$$2\rho^{k+1}(1-\rho)\sum_{i=0}^{k}\rho^i - 1 + \rho^{k+2} \leq 0. \tag{2.4.14}$$

It will be convenient to write (2.4.14) in the following equivalent form:

$$\left(2\rho^{k+1} - 1\right)\sum_{i=0}^{k}\rho^i - \rho^{k+1} \geq 0. \tag{2.4.15}$$

Induction can be used for a second time to show that, given $\rho > 1$, the inequality (2.4.15) holds for all integers $k \geq 1$. Indeed, when $k = 1$, (2.4.15) reduces to:

$$2\rho^3 + \rho^2 - \rho - 1 \geq 0,$$

which obviously holds when $\rho > 1$. Let $g_k(\rho) := \left(2\rho^{k+1} - 1\right)\sum_{i=0}^{k}\rho^i - \rho^{k+1}$ for $k \geq 1$, and assume (as an inductive hypothesis) that $g_k(\rho) \geq 0$. It is then sufficient to show that $g_{k+1}(\rho) \geq g_k(\rho)$. Indeed, one may show that $g_{k+1}(\rho) \geq g_k(\rho)$ is equivalent to the following:

$$2\rho^{k+1}(\rho - 1)\sum_{i=0}^{k}\rho^i + \rho^{k+2}\left(2\rho^{k+1} - 1\right) \geq 0,$$

which clearly holds for $\rho > 1$. This completes the inductive proof that (2.4.11) holds for all $n \in \mathbb{N}$ and $\rho > 1$. By the previous arguments, this is sufficient to imply that the equality $\eta^{*[O]} = \eta^{*[U]}$ cannot hold when $\rho > 1$; in fact, it is the case that $\eta^{*[O]} > \eta^{*[U]}$ when $\rho > 1$. This completes the proof of the second statement (the socially optimal case) in the lemma. □

The proof of Lemma 2.4.1 provides a necessary *and* sufficient condition for the equality of selfishly optimal joining rates ($\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$), and a similar condition for the equality of socially optimal joining rates ($\eta^{*[O]} = \eta^{*[U]}$). These conditions are now stated as a theorem.

Theorem 2.4.2.

1. *The equality $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ holds if and only if:*

$$\frac{1-\rho^{n_s+1}}{1-\rho} = \frac{\alpha\mu}{\beta}.$$

2. *The equality $\eta^{*[O]} = \eta^{*[U]}$ holds if and only if:*

$$\frac{1-\rho^{n_o+1}}{1-\rho} = \sqrt{\frac{\alpha\mu}{\beta}}.$$

Proof. By Lemma 2.4.1, the equalities $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ and $\eta^{*[O]} = \eta^{*[U]}$ both require $\rho < 1$, and the results follow immediately by referring to (2.4.1)-(2.4.4). □

The quantity $\alpha\mu/\beta$ could, in many types of applications, be controlled as part of the system design; for example, a firm will have some measure of control over the quality of the product that it produces (represented by the value of α). Suppose ρ is fixed at some value in the range $(0, 1)$. The value of $\alpha\mu/\beta$ which prescribes the equality $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ need not be unique. This is due to the fact that $\tilde{\eta}^{[O]}$ behaves as a step function of $\alpha\mu/\beta$, as shown by the next example.

Example 2.4.3. ***(Intersections between selfishly optimal joining rates)***

Consider a system with $\lambda = 0.9$, $\mu = 1$ (hence $\rho = 0.9$) and suppose at least one of the parameters α and β can be chosen arbitrarily. Figure 2.4 shows the selfishly optimal joining rates $\tilde{\eta}^{[O]} = \lambda(1 - \rho^{n_s})/(1 - \rho^{n_s+1})$ and $\tilde{\eta}^{[U]} = \mu - \beta/\alpha$, plotted for $1 \leq \alpha\mu/\beta \leq 5$.

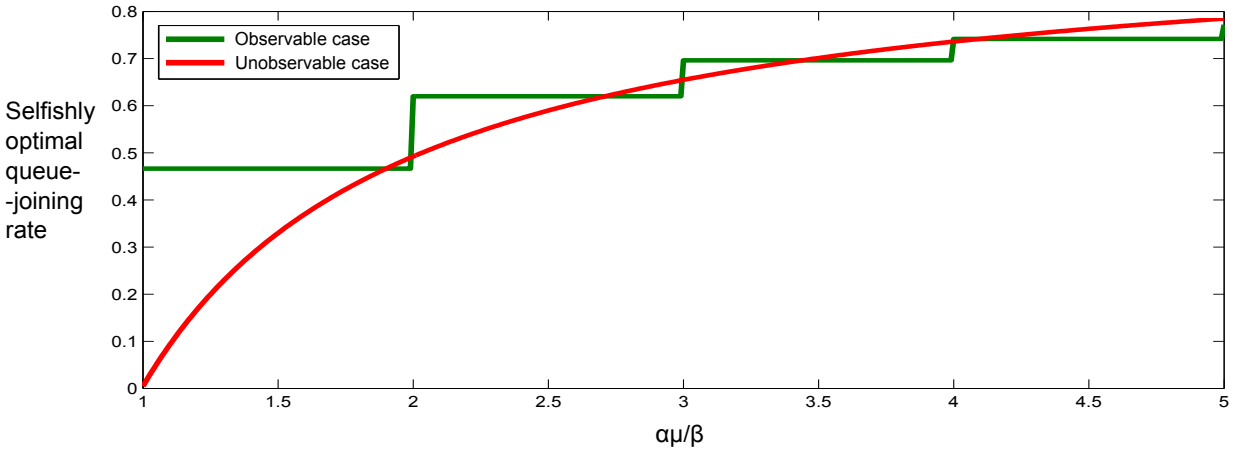


Figure 2.4: Intersections between the selfishly optimal joining rates $\tilde{\eta}^{[O]}$ and $\tilde{\eta}^{[U]}$.

Figure 2.4 shows that there are four intersection points between $\tilde{\eta}^{[O]}$ and $\tilde{\eta}^{[U]}$. The values of $\alpha\mu/\beta$ that result in these intersections are (approximately) 1.90, 2.71, 3.44 and 4.10 (it is as expected that these values represent four distinct values of the integer $n_s = \lfloor \alpha\mu/\beta \rfloor$). The selfishly optimal rates at these points are, respectively, 0.47, 0.63, 0.71 and 0.76. It can be checked that multiple intersection points are also possible in the case of the socially optimal rates. ⊠

The next result concerns the number of distinct values of the quantity $\alpha\mu/\beta$ for which it is possible

to have $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$. These are referred to informally as intersection points.

Theorem 2.4.4. *Let $J(\rho)$ be the number of distinct values of $\alpha\mu/\beta$ for which $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$. Then $J(\rho)$ is monotonically increasing for $\rho \in (0,1)$, and $J(\rho) \to \infty$ as $\rho \to 1$.*

(The number of intersection points between the selfishly optimal joining rates is monotonically increasing with the traffic intensity ρ, and tends to infinity as $\rho \to 1$.)

Proof. By Theorem 2.4.2, a necessary and sufficient condition for $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ is:

$$\frac{1-\rho^{n_s+1}}{1-\rho} = \frac{\alpha\mu}{\beta},$$

where $n_s = \lfloor \alpha\mu/\beta \rfloor$. It therefore makes sense to consider solutions to the following inequalities, where $n \geq 1$ is an integer and (in view of Lemma 2.4.1) $\rho \in (0,1)$:

$$n \leq \frac{1-\rho^{n+1}}{1-\rho} < n+1. \tag{2.4.16}$$

Let $f_n(\rho) = (1-\rho^{n+1})/(1-\rho)$ for $n \in \mathbb{N}$, $\rho \in (0,1)$. It can be shown that $f_n(\rho)$ is strictly increasing and convex in ρ and $\lim_{\rho\to 1} f_n(\rho) = n+1$. Hence, for any $n \in \mathbb{N}$, there exists a value $\rho_n \in [0,1)$ such that the inequalities in (2.4.16) are satisfied if and only if $\rho \in [\rho_n, 1)$.

It may be observed that ρ_n, which is the value of ρ for which $f_n(\rho) = n$, actually increases with n. In the case $n = 1$, one finds $\rho_1 = 0$ since (2.4.16) simplifies to $1 \leq 1+\rho < 2$, which is satisfied for all $\rho \in [0,1)$. For $n = 2$, (2.4.16) implies $\rho^3 - 2\rho + 1 \leq 0 < 3(1-\rho)$, which is satisfied for $\rho \in [\phi^{-1}, 1)$, where ϕ is the well-known golden ratio; hence, $\rho_2 = \phi^{-1} \approx 0.618$. Similarly it can be checked that $\rho_3 \approx 0.811$, $\rho_4 \approx 0.888$, etc. Each value of n for which (2.4.16) holds implies the existence of a *distinct* value of $\alpha\mu/\beta$ for which the selfishly optimal joining rates are equal for the two types of system. Therefore, in order to show that $J(\rho)$ is monotonically increasing for $\rho \in (0,1)$, it suffices to show that the sequence $(\rho_n)_{n\in\mathbb{N}}$ (where ρ_n is the unique solution to the equation $f_n(\rho) = n$) is strictly increasing. This will imply that any value of $\rho \in (0,1)$ satisfying (2.4.16) for a particular integer $n \in \mathbb{N}$ also satisfies the same inequalities for all smaller values of n.

Given that $f_n(\rho_n) = n$, (2.4.16) implies that ρ_n is the unique solution to:

$$\rho^{n+1} - n\rho + n - 1 = 0.$$

Let $g_n(\rho) := \rho^{n+1} - n\rho + n - 1$ for $n \in \mathbb{N}$, $\rho \in (0,1)$. It may be shown that:

$$g_n(\rho) = (1-\rho)\left(n - 1 - \sum_{i=1}^{n} \rho^i\right). \tag{2.4.17}$$

Here, let $h_n(\rho) := n - 1 - \sum_{i=1}^{n} \rho^i$ for $n \in \mathbb{N}$, $\rho \in (0,1)$. Since $(1-\rho) \neq 0$ and in view of the fact that $g_n(\rho_n) = 0$, ρ_n must satisfy $h_n(\rho_n) = 0$. One has the recurrence relation:

$$h_{n+1}(\rho) = h_n(\rho) + 1 - \rho^{n+1}. \tag{2.4.18}$$

If $\rho = \rho_n$ then $h_n(\rho) = 0$ but then, since $1 - \rho^{n+1} > 0$, one finds that $h_{n+1}(\rho) > 0$ and so the equality $h_{n+1}(\rho_n) = 0$ does not hold. Since $h_n(\rho)$ is a decreasing function of ρ, one must have $\rho_{n+1} > \rho_n$ in order for the equality $h_{n+1}(\rho_{n+1}) = 0$ to hold and therefore $(\rho_n)_{n\in\mathbb{N}}$ is a strictly increasing sequence, which completes the proof. The fact that $J(\rho) \to \infty$ as $\rho \to 1$ follows from the fact that $f_n(\rho) = (1-\rho^{n+1})/(1-\rho)$ attains a value of $n+1$ at $\rho = 1$, implying that ρ can always be chosen to be large enough to satisfy $f_n(\rho) \geq n$ for arbitrarily large $n \in \mathbb{N}$. □

Notably, Theorem 2.4.4 differs from most of the results in this chapter in that it is exclusive to the case of *selfishly* optimal queue-joining rates. Investigating whether or not a similar result holds for the socially optimal joining rates is a potential avenue for further work.

Theorem 2.4.4 implies that if the value of $\rho \in (0,1)$ is sufficiently large, it is possible to find multiple values of $\alpha\mu/\beta$ for which $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$. On the other hand, if one considers *fixed* values of $\alpha\mu/\beta$, it is clear that there is a *unique* value of $\rho \in (0,1)$ which satisfies $(1-\rho^{n_s+1})/(1-\rho) = \alpha\mu/\beta$. This is because $f_n(\rho) = (1-\rho^{n+1})/(1-\rho)$ is a strictly increasing function of $\rho > 0$.

By Theorem 2.4.2, the condition required for the equality of *socially* optimal joining rates is $(1-\rho^{n_o+1})/(1-\rho) = \sqrt{\alpha\mu/\beta}$. However, recalling (2.3.3), n_o (unlike n_s) has a dependence on ρ. Increasing the value of ρ may result in a decrease in the integer value n_o. Therefore it cannot be said that, for a fixed value of $\alpha\mu/\beta$, there is a *unique* value of $\rho \in (0,1)$ which yields $\eta^{*[O]} = \eta^{*[U]}$. Indeed, consider an example with $\alpha\mu/\beta = 2.5$. In this case, the equation $\eta^{*[O]} = \eta^{*[U]}$ is satisfied with $\rho \approx 0.581$, in which case $n_o = 1$, but also $\rho \approx 0.412$, in which case $n_o = 2$.

Theorem 2.4.2 established a condition on the system input parameters which ensures $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$, and a separate condition which ensures $\eta^{*[O]} = \eta^{*[U]}$. The remainder of this section addresses the question of whether or not it is possible for an observable system and an unobservable system

which share the same input parameters λ, μ, α and β to *simultaneously* share the same selfishly and socially optimal joining rates. In posing this question, it is important to emphasise that the parameter values λ, μ, α and β are assumed common to both systems; indeed, it is not difficult to construct an example of an observable system which has a selfishly optimal joining rate $\tilde{\eta} > 0$ and a socially optimal joining rate $\eta^* > 0$ (with $\eta^* \leq \tilde{\eta} \leq \lambda$) and a separate example of an unobservable system which has the same two values, $\tilde{\eta}$ and η^*, for its respective optimal joining rates if the parameters λ, μ, α and β are not required to be identical for the two systems.

Theorem 2.4.5. *It is not possible for an observable and an unobservable system with the common parameter values λ, μ, α and β to share the same selfishly and socially optimal joining rates; that is, the equalities $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$, and $\eta^{*[O]} = \eta^{*[U]}$ cannot hold simultaneously.*

Proof. In view of Lemma 2.4.1, it is only necessary to consider $\rho \in (0, 1)$. By Theorem 2.4.2, in order to have $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$, and $\eta^{*[O]} = \eta^{*[U]}$ both of the following must hold:

$$\frac{1-\rho^{n_s+1}}{1-\rho} = \frac{\alpha\mu}{\beta},$$
$$\frac{1-\rho^{n_o+1}}{1-\rho} = \sqrt{\frac{\alpha\mu}{\beta}}.$$

Hence, the simultaneous equalities $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ and $\eta^{*[O]} = \eta^{*[U]}$ would require:

$$\left(\frac{1-\rho^{n_o+1}}{1-\rho}\right)^2 = \frac{1-\rho^{n_s+1}}{1-\rho}.$$

Therefore, in order to prove the theorem, it is sufficient to show that for all $\rho \in (0, 1)$:

$$\left(1-\rho^{n_o+1}\right)^2 > (1-\rho)(1-\rho^{n_s+1}). \tag{2.4.19}$$

Consider the relationship between n_s and n_o. On one hand, $n_s = \lfloor \alpha\mu/\beta \rfloor$ (hence $n_s \leq \alpha\mu/\beta$) and from (2.3.3) one obtains $\alpha\mu/\beta < \left((n_o+1)(1-\rho) - \rho(1-\rho^{n_o+1})\right)/(1-\rho)^2$. Hence:

$$n_s < \frac{(n_o+1)(1-\rho) - \rho(1-\rho^{n_o+1})}{(1-\rho)^2}. \tag{2.4.20}$$

Here, let $f_n(\rho) := \left((n+1)(1-\rho) - \rho(1-\rho^{n+1})\right)/(1-\rho)^2$ for $n \in \mathbb{N}$ and note that $f_n(\rho)$ is an increasing function of ρ; indeed, it may be shown that $f_n(\rho) = \sum_{i=0}^{n}(n+1-i)\rho^i$. Also, $\lim_{\rho\to 1} f_n(\rho) = (n+1)(n+2)/2$. Therefore (2.4.20) implies:

$$n_s < \frac{(n_o+1)(n_o+2)}{2}.$$

Moreover, since n_s and n_o are integers, one may also write:

$$n_s + 1 \leq \frac{(n_o + 1)(n_o + 2)}{2}.$$

So, in order to show that the inequality (2.4.19) holds, it is sufficient to establish the more general fact that for all integers $n \geq 2$ and $\rho \in (0, 1)$:

$$(1 - \rho^n)^2 > (1 - \rho)(1 - \rho^{n(n+1)/2}). \tag{2.4.21}$$

It will be convenient to write (2.4.21) in the equivalent form:

$$\left(\sum_{i=0}^{n-1} \rho^i\right)^2 > \sum_{i=0}^{n(n+1)/2-1} \rho^i. \tag{2.4.22}$$

This will enable a proof by induction. Indeed, for $n = 2$, (2.4.22) reduces to:

$$(1 + \rho)^2 > 1 + \rho + \rho^2,$$

which holds for all $\rho > 0$. Let $g_n(\rho) := (\sum_{i=0}^{n-1} \rho^i)^2 - \sum_{i=0}^{n(n+1)/2-1} \rho^i$ for $n \in \mathbb{N}$ and assume, as an inductive hypothesis, that $g_k(\rho) > 0$ for some $k \in \mathbb{N}$. It will then be sufficient to show that $g_{k+1}(\rho) \geq g_k(\rho)$. Indeed, one can show that this is equivalent to:

$$\rho^k + 2\sum_{i=0}^{k-1} \rho^i - \rho^{k(k-1)/2} \sum_{i=0}^{k} \rho^i \geq 0.$$

Further manipulations yield the equivalent condition:

$$(2 - \rho^{k(k-1)/2}) \sum_{i=0}^{k-1} \rho^i + (1 - \rho^{k(k-1)/2})\rho^k \geq 0. \tag{2.4.23}$$

It is clear that (2.4.23) holds for $0 < \rho < 1$, which completes the inductive proof that (2.4.22) holds for all $n \in \mathbb{N}$. In view of the previous arguments, this completes the proof. □

As an additional note, the fact that $(1 - \rho^{n_o+1})^2 > (1 - \rho)(1 - \rho^{n_s+1})$ implies that the value of $\alpha\mu/\beta$ required for the equality of socially optimal joining rates is greater than the corresponding value required for the equality of selfishly optimal joining rates.

Corollary 2.4.6.

1. *If* $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$, *then* $\eta^{*[O]} > \eta^{*[U]}$.

2. *If* $\eta^{*[O]} = \eta^{*[U]}$, *then* $\tilde{\eta}^{[O]} < \tilde{\eta}^{[U]}$.

Proof. From the proof of Theorem 2.4.5, it may be inferred that the inequality $\left((1-\rho^{n_o+1})/(1-\rho)\right)^2 > (1-\rho^{n_s+1})/(1-\rho)$ holds for all $\rho \in (0,1)$ and all choices of $\alpha\mu/\beta > 1$. If $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ then $(1-\rho^{n_s+1})/(1-\rho) = \alpha\mu/\beta$, hence $\left((1-\rho^{n_o+1})/(1-\rho)\right)^2 > \alpha\mu/\beta$, which implies $\eta^{*[O]} > \eta^{*[U]}$. On the other hand, if $\eta^{*[O]} = \eta^{*[U]}$ then $\left((1-\rho^{n_o+1})/(1-\rho)\right)^2 = \alpha\mu/\beta$, hence $(1-\rho^{n_s+1})/(1-\rho) < \alpha\mu/\beta$, which implies $\tilde{\eta}^{[O]} < \tilde{\eta}^{[U]}$ as required. □

The result of Corollary 2.4.6 is illustrated by Figure 2.5. The figure shows that, under both types of optimal customer behaviour, it is possible to make some general observations about the relationship between the joining rates for the two types of system if $0 < \rho < 1$. (Recall that, from the proof of Lemma 2.4.1, $\rho \geq 1$ implies $\tilde{\eta}^{[O]} > \tilde{\eta}^{[U]}$ and $\eta^{*[O]} > \eta^{*[U]}$.) In a *low-reward* system, where $\alpha\mu/\beta \approx 1$, the optimal joining rates will tend to be higher if the system is *observable*. On the other hand, in a *high-reward* system where $\alpha\mu/\beta$ is large, the optimal joining rates will tend to be higher if the queue is *unobservable*. Thus, for a system designer who wishes to maximise the rate at which customers join the queue for service, it appears that the incentive to reveal the queue length is greater when the value of service (relative to the cost of waiting) is low.

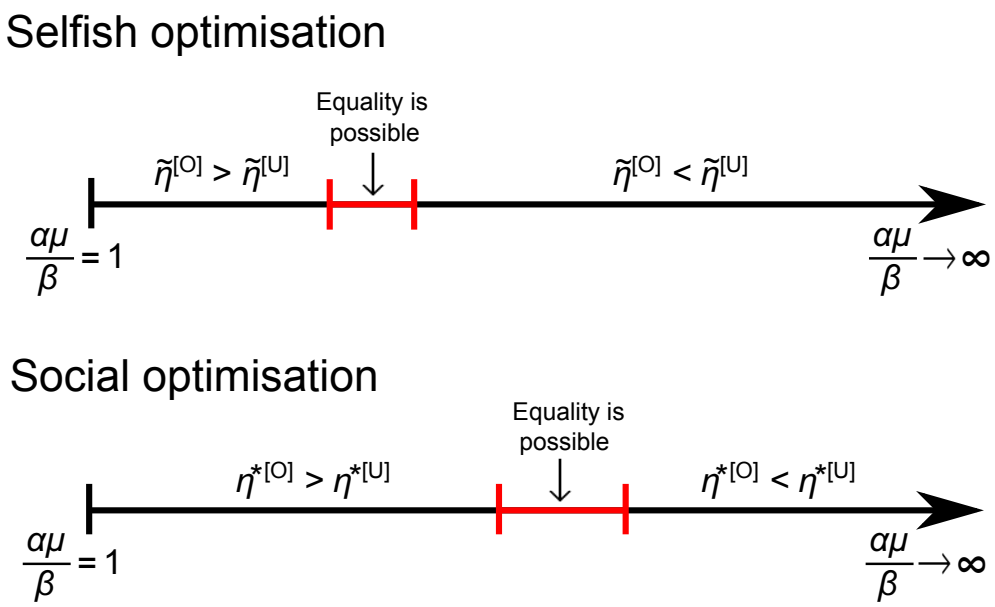


Figure 2.5: Illustration of the result of Corollary 2.4.6.

2.5 Performance measures

The purpose of this section is to compare performance measures for the observable and unobservable systems under selfishly and socially optimal conditions; that is, one assumes either that $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ or that $\eta^{*[O]} = \eta^{*[U]}$ (noting that, due to Theorem 2.4.5, one cannot have both) and considers the implications for steady-state probabilities, mean waiting times etc. Recall that the notation for these performance measures was summarised in Table 2.1. The next result concerns the stationary distributions for the respective system types under both types of optimisation.

Theorem 2.5.1.

1. *If* $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ *then* $\tilde{P}_m^{[U]} \leq \tilde{P}_m^{[O]}$ *for all* $0 \leq m \leq n_s$*, with equality if and only if* $m = 0$*.*

2. *If* $\eta^{*[O]} = \eta^{*[U]}$ *then* $P_m^{*\,[U]} \leq P_m^{*\,[O]}$ *for all* $0 \leq m \leq n_o$*, with equality if and only if* $m = 0$*.*

Proof. Consider the case $m = 0$. Standard results from finite-buffer queueing theory ([67], p. 77) imply, for a buffer size n, $P_0 = (1-\rho)/(1-\rho^{n+1})$. In the case of observable queues with balking, one effectively has (in the selfishly optimal case) a buffer size of n_s and hence $\tilde{P}_0^{[O]} = (1-\rho)/(1-\rho^{n_s+1})$. In the unobservable case, one has a Poisson arrival rate of $\tilde{\eta}^{[U]} = \mu - \beta/\alpha$ to a system with no finite buffer. Using standard results for *infinite* capacity queues, it follows that $\tilde{P}_0^{[U]} = 1 - (\tilde{\eta}^{[U]}/\mu) = \beta/(\alpha\mu)$. Then, recalling Theorem 2.4.2, it may be observed that the necessary and sufficient condition for $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ states exactly that $\tilde{P}_0^{[O]} = \tilde{P}_0^{[U]}$.

Next, consider $m \geq 1$. Using standard results again, $\tilde{P}_m^{[O]} = \rho^m \tilde{P}_0^{[O]}$ for $1 \leq m \leq n_s$. Here, of course, $\rho = \lambda/\mu$ where λ is the arrival rate for the *system*, which is the same as the queue-joining rate provided that there are fewer than n_s customers present. In the case of the unobservable system, the corresponding relation is $\tilde{P}_m^{[U]} = (\tilde{\eta}^{[U]}/\mu)^m \tilde{P}_0^{[U]}$ for $1 \leq m \leq n_o$. Given that $\tilde{\eta}^{[U]} < \lambda$ (which is implied by $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$), it follows that $\tilde{P}_m^{[U]} < \tilde{P}_m^{[O]}$ for $m \geq 1$.

The proof for the socially optimal case is similar, except that in this case one has $P_0^{*[O]} = (1-\rho)/(1-\rho^{n_o+1})$, and $P_0^{*[U]} = 1 - (\eta^{*[U]}/\mu) = \sqrt{\beta/(\alpha\mu)}$. Therefore, recalling Theorem 2.4.2, the equality $P_0^{*[U]} = P_0^{*[O]}$ is equivalent to the necessary and sufficient condition for the equality of socially optimal joining rates, $\eta^{*[O]} = \eta^{*[U]}$. The fact that $P_m^{*\,[U]} < P_m^{*\,[O]}$ for $1 \leq m \leq n_o$ is shown using arguments analogous to those used for the selfishly optimal case. □

Remark. The fact that $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ implies $\tilde{P}_0^{[O]} = \tilde{P}_0^{[U]}$ (with an analogous result for social optimisation) actually follows immediately from a basic property of queueing systems in steady state. Under steady-state conditions, the queue-joining rate must be equal to the rate at which customers exit the system, which is given (for a general system) by $\mu(1 - P_0)$. Hence, $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ implies $\mu(1 - \tilde{P}_0^{[O]}) = \mu(1 - \tilde{P}_0^{[U]})$ and the result follows since μ is a fixed parameter.

The next result of this section concerns the average lengths of idle periods and busy periods for the two types of system under selfishly and socially optimal conditions.

Theorem 2.5.2. *Let $\widetilde{MBP}^{[O]}$ ($\widetilde{MBP}^{[U]}$) and $MBP^{*[O]}$ ($MBP^{*[U]}$) be the mean busy periods for the observable (unobservable) system under selfish and social optimisation respectively. Also let $\widetilde{MIP}^{[O]}$ ($\widetilde{MIP}^{[U]}$) and $MIP^{*[O]}$ ($MIP^{*[U]}$) be the corresponding mean idle periods.*

1. *If $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$, then $\widetilde{MBP}^{[O]} < \widetilde{MBP}^{[U]}$ and $\widetilde{MIP}^{[O]} < \widetilde{MIP}^{[U]}$. Specifically:*

$$\frac{\widetilde{MBP}^{[O]}}{\widetilde{MBP}^{[U]}} = \frac{\widetilde{MIP}^{[O]}}{\widetilde{MIP}^{[U]}} = \frac{1-\rho^{n_s}}{1-\rho^{n_s+1}}.$$

2. *If $\eta^{*[O]} = \eta^{*[U]}$, then $MBP^{*[O]} < MBP^{*[U]}$ and $MIP^{*[O]} < MIP^{*[U]}$. Specifically:*

$$\frac{MBP^{*[O]}}{MBP^{*[U]}} = \frac{MIP^{*[O]}}{MIP^{*[U]}} = \frac{1-\rho^{n_o}}{1-\rho^{n_o+1}}.$$

(Under the equality of selfishly or socially optimal joining rates, the mean busy period and mean idle period are both shorter in the observable case than in the unobservable case.)

Proof. The proof is given for the selfishly optimal case (the socially optimal case is analogous). Consider the mean idle periods. If the system is observable and it is empty, then (assuming $\alpha > \beta/\mu$) the next customer to arrive in the system will join the queue. The mean idle period is thus $1/\lambda$. In the unobservable case, customers join the queue at the Poisson rate $\tilde{\eta}^{[U]}$ regardless of the state of the system. Given that $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$, the mean idle period is $1/\tilde{\eta}^{[O]} = (1-\rho^{n_s+1})/(\lambda(1-\rho^{n_s}))$. The result $\widetilde{MIP}^{[O]}/\widetilde{MIP}^{[U]} = (1-\rho^{n_s})/(1-\rho^{n_s+1})$ then follows.

For the mean busy periods, it can be established using standard results for $M/M/1$ queues that $\widetilde{MBP}^{[O]} = (1-\rho^{n_s})/(\mu(1-\rho))$ and $\widetilde{MBP}^{[U]} = (1-\rho^{n_s+1})/(\mu(1-\rho))$, which gives the required result. However, in order to show that $\widetilde{MBP}^{[O]}/\widetilde{MBP}^{[U]} = (1-\rho^{n_s})/(1-\rho^{n_s+1})$, it is sufficient

to observe that, by Theorem 2.5.1, the equalities $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ and $\tilde{P}_0^{[O]} = \tilde{P}_0^{[U]}$ are equivalent. In order for the equality $\tilde{P}_0^{[O]} = \tilde{P}_0^{[U]}$ to hold, the mean busy periods for the observable and unobservable systems must be in the same ratio as the mean idle periods. That is, one must have $\widetilde{MBP}^{[O]}/\widetilde{MBP}^{[U]} = \widetilde{MIP}^{[O]}/\widetilde{MIP}^{[U]} = (1-\rho^{n_s})/(1-\rho^{n_s+1})$ as required. □

Finally, consider the mean number of customers present in the system. The next result shows that, in the observable case, the congestion in the system is lower on average under selfish (or social) optimisation than in the unobservable case. This is a consequence of the fact that in the observable case, customers may join the queue only in the most favourable circumstances. The fact that mean waiting times are shorter in the observable case is implied by this result.

Theorem 2.5.3. *Let $\tilde{L}^{[O]}$ ($\tilde{L}^{[U]}$) and $L^{*[O]}$ ($L^{*[U]}$) be the steady-state expected numbers of customers in the observable (unobservable) system under selfish and social optimisation respectively. Also, let $\tilde{W}^{[O]}$ ($\tilde{W}^{[U]}$) and $W^{*[O]}$ ($W^{*[U]}$) be the corresponding mean waiting times. Then:*

1. *If $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ then $\tilde{L}^{[O]} < \tilde{L}^{[U]}$ and $\tilde{W}^{[O]} < \tilde{W}^{[U]}$.*
2. *If $\eta^{*[O]} = \eta^{*[U]}$ then $L^{*[O]} < L^{*[U]}$ and $W^{*[O]} < W^{*[U]}$.*

Proof. The proof is given for the selfishly optimal case (the socially optimal case is analogous). Assume $\rho \in (0,1)$, which is implied by the condition $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$. Then:

$$\tilde{L}^{[O]} = \frac{\rho}{1-\rho} - \frac{(n_s+1)\rho^{n_s+1}}{1-\rho^{n_s+1}},$$

$$\tilde{L}^{[U]} = \frac{\tilde{\eta}^{[U]}}{\mu - \tilde{\eta}^{[U]}} = \frac{\tilde{\eta}^{[O]}}{\mu - \tilde{\eta}^{[O]}} = \frac{\rho(1-\rho^{n_s})}{1-\rho}.$$

One can show that the inequality $\tilde{L}^{[O]} < \tilde{L}^{[U]}$ is equivalent to:

$$\rho^{n_s+1} - (n_s+1)\rho + n_s > 0.$$

Let $f_n(\rho) := \rho^{n+1} - (n+1)\rho + n$ for $n \in \mathbb{N}$. Then $f_n'(\rho) = (\rho^n - 1)(n+1) < 0$. Thus, $f_n(\rho)$ is strictly decreasing for $\rho \in (0,1)$, and since $f_n(1) = 0$, the result $\tilde{L}^{[O]} < \tilde{L}^{[U]}$ follows. Little's formula (see [67], p. 10) gives $\tilde{L}^{[O]} = \tilde{\eta}^{[O]}\tilde{W}^{[O]}$ and, similarly, $\tilde{L}^{[U]} = \tilde{\eta}^{[U]}\tilde{W}^{[U]}$. Hence, by the previous arguments, $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ implies $\tilde{L}^{[O]} < \tilde{L}^{[U]}$, which in turn implies $\tilde{W}^{[O]} < \tilde{W}^{[U]}$. □

Figure 2.6 illustrates the result of Theorem 2.5.3 for the selfishly optimal case. In the figure, the relationship $\alpha\mu/\beta = 1 + \rho$ has been assumed in order to ensure that the equality $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ holds.

It can be seen that for the parameter values used, $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$ implies $\tilde{L}^{[O]} < \tilde{L}^{[U]}$. The other results of this section can be illustrated using similar diagrams.

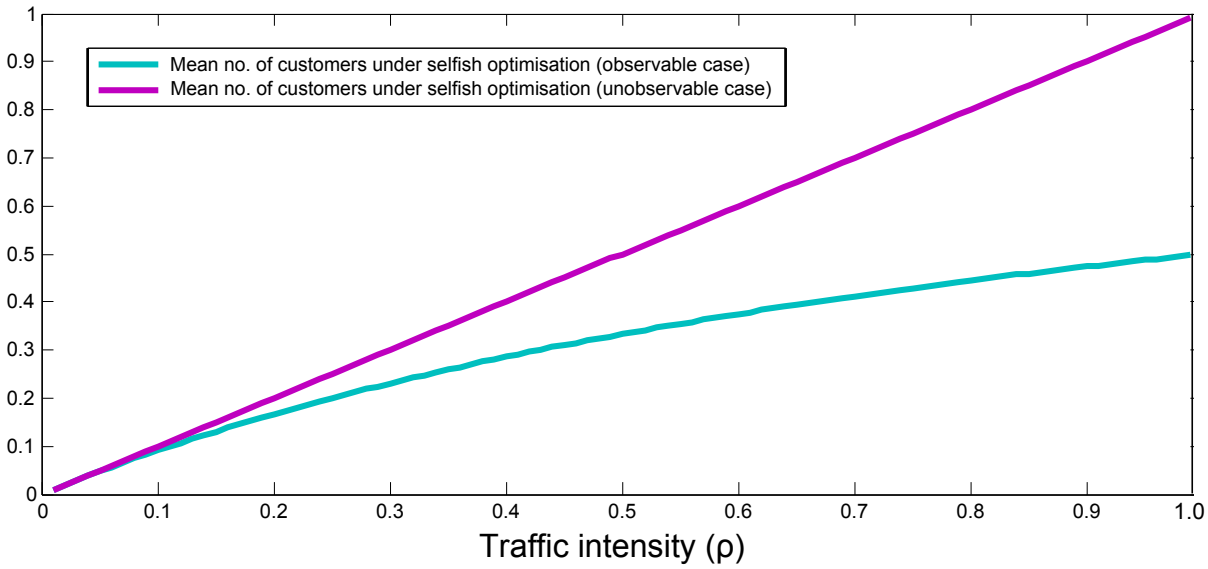


Figure 2.6: Relationships of $\tilde{L}^{[O]}$ and $\tilde{L}^{[U]}$ with ρ assuming the equality $\tilde{\eta}^{[O]} = \tilde{\eta}^{[U]}$.

2.6 Conclusions

As discussed in Section 2.1, the strategic behaviour of customers (or the authority making decisions on their behalf) under varying levels of information has received considerable recent attention in the literature due to the insights that may be gained into whether or not a service provider should restrict the amount of information available to its users. This chapter has considered an $M/M/1$ queueing system under two levels of information. The main results are as follows:

- The equality of selfishly optimal queue-joining rates is achievable only if $\rho \in (0, 1)$, and the same also applies to the equality of socially optimal joining rates.
- A necessary and sufficient condition can be derived for the equality of selfishly optimal queue-joining rates in terms of the system parameters λ, μ, α and β, and a similar condition can also be found for the equality of socially optimal joining rates.
- The number of intersection points between the selfishly optimal joining rates is monotonically increasing with ρ in the interval $(0, 1)$ and tends to infinity as $\rho \to 1$.

- It is not possible to find a set of system parameters for which the selfishly optimal joining rates are equal and the socially optimal joining rates are also equal.
- Small values of $\alpha\mu/\beta$ will tend to imply that selfishly and socially optimal joining rates are greater in the observable case. On the other hand, large values of $\alpha\mu/\beta$ imply that (on average) customers will join more frequently in the unobservable case.
- Various performance measures, including mean busy periods, expected queue lengths and average waiting times are smaller in the observable case than in the unobservable case, assuming the equality of either selfishly or socially optimal joining rates.

The extent to which the system input parameters are controllable via the system design will vary according to particular applications. It has been shown that the decision of a service provider to reveal or suppress information on queue length may or may not affect customer queue-joining rates, depending on the values of the system parameters and the type of customer behaviour assumed. The general principle is that the incentive to reveal the queue length is greater if the value of service (relative to the cost of waiting) is low. However, in general it is not possible to specify a unique 'cut-off' value of $\alpha\mu/\beta$ which defines the optimal strategy of a decision-maker.

One might conjecture that, in many practical situations, if a revenue-maximising server (with the singular aim of maximising customer throughput rates) finds itself in the particular scenario where the average customer throughput rate per unit time is independent of the decision to reveal or suppress information on queue length, then the most sensible decision will be to reveal the queue length in order to generate greater goodwill from customers. The results in Section 2.5 have shown that in the observable case, customers can expect to enjoy shorter waiting times, and greater net benefits for receiving service as a result. One might explore this further by modelling a situation in which the quality of service enjoyed by customers is worth some quantifiable benefit to the server. This would obviously broaden the range of circumstances in which revealing the queue length would be the optimal decision for the revenue-maximising server. The analysis of optimal server policies in models with this type of extra complexity is a possible avenue for future work.

3 MDP formulation

3.1 The basic queueing system

This chapter introduces a mathematical model for the multiple-facility queueing system to be analysed throughout the rest of this thesis. Various modifications to the model will be made in later chapters, but at this early stage it is desirable to adopt a formulation which enables the classical techniques of dynamic programming to be employed without any major difficulty. The queueing system consists of a finite number N of heterogeneous *service facilities*, each of which has a dedicated queue and is subject to a cost and reward scheme similar to that of Naor's single-server model in [131]. The fundamental assumptions of the model are as follows:

- Customers arrive according to a Poisson process with demand rate $\lambda > 0$.
- Each facility $i \in \{1, 2, ..., N\}$ possesses c_i identical service channels, and service times at any channel of facility i are exponentially distributed with mean $\mu_i^{-1} > 0$.
- Newly-arrived customers may proceed to any one of the N service facilities or, alternatively, exit the system immediately without incurring any cost or reward (referred to as *balking*). Thus, there are $N + 1$ possible destinations for any individual customer. Customers who proceed to facility $i \in \{1, 2, ..., N\}$ will either wait in the queue for facility i or go directly to an empty service channel and begin their service immediately, depending on whether or not they find all of the c_i service channels occupied upon their arrival.
- The queue discipline at each facility is First-Come-First-Served (FCFS).
- The system incurs a *linear holding cost* $\beta_i > 0$ per unit time for each customer waiting at facility $i \in \{1, 2, ..., N\}$ (whether in the queue or in service).
- The system earns a *fixed reward* $\alpha_i > 0$ for each customer who completes service at facility $i \in \{1, 2, ..., N\}$. The units of α_i and β_i are assumed to be comparable.
- The system is *fully observable*, in the sense that the number of customers present at each facility is always known and can be used to inform decision-making.

- The decision made upon a customer's arrival is *irrevocable*; the system does not allow reneging, jockeying between queues or postponement of decision-making.
- The decision made upon a customer's arrival is made *without* any knowledge of the exact times at which arrivals or service completions will occur in the future.
- For each $i \in \{1, 2, ..., N\}$, α_i, β_i and μ_i satisfy the relationship $\alpha_i > \beta_i/\mu_i$.

Essentially, the model may be regarded as an N-fold generalisation of the $M/M/1$ system considered in Chapter 2, in which customers may be served at any one of N different facilities arranged in parallel (with the option of balking also available) and each facility i is capable of serving up to c_i customers at the same time. As alluded to in Chapter 2, the assumption that $\alpha_i > \beta_i/\mu_i$ for all facilities $i \in \{1, 2, ..., N\}$ is made in order to avoid degeneracy in the system. If one had $\alpha - \beta_i/\mu_i < 0$ for some facility i, then it would never be worthwhile in an economic sense for a customer to receive service there, since the expected cost incurred during their service time would exceed the reward obtained for service. In fact, all of the results in this thesis remain valid under the weaker assumption that $\alpha_i \geq \beta_i/\mu_i$ for each facility i, with only some extra discussion required in some cases in order to deal with trivialities that occur if $\alpha_i - \beta_i/\mu_i = 0$. In order to avoid these trivialities, it will be simpler to assume the strict inequality $\alpha_i > \beta_i/\mu_i$ throughout.

Another point to be made is that, in the queueing theoretic literature, it is common practice to make an assumption concerning the rate of demand λ in order to ensure system stability. Typical assumptions might be that $\lambda < \sum_i c_i\mu_i$, or (more restrictively) $\lambda < c_i\mu_i$ for all $i \in \{1, 2, ..., N\}$. However, no such assumption is made in this thesis. This is due to the fact that the option of balking is always available, and therefore it is always possible to find a means of controlling the system such that the system is *stable*, in the sense that the total number of customers present is not at risk of escalating in an unbounded fashion. It is therefore not necessary to consider any scenario in which customers 'flood' the system at a rate which exceeds its maximum service completion rate, since this type of behaviour does not achieve any economic objective of interest.

A diagrammatic representation of the system is given in Figure 3.1. It is clear that the dynamics of the system are influenced not only by the input parameters (arrival rate, service rates etc.) but also by the decisions made at the 'routing point' labelled in the diagram. As in Chapter 2, there is no explicit assumption as to whether these decisions are made by customers themselves, or whether

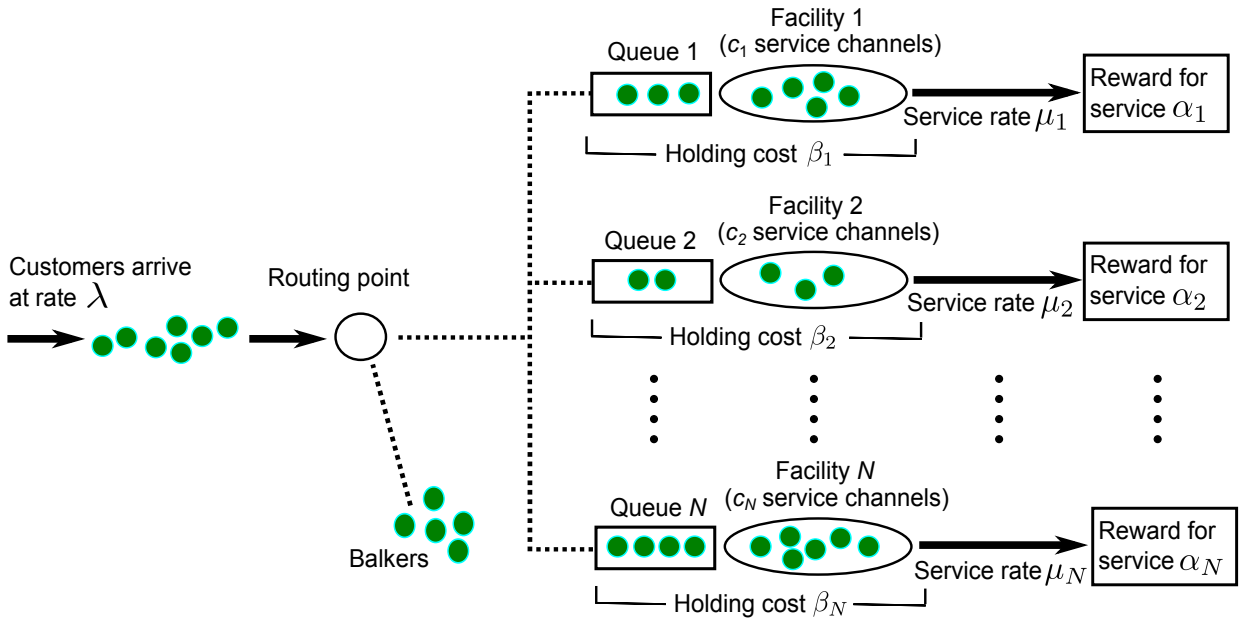


Figure 3.1: A diagrammatic representation of the queueing system.

there is a central decision-maker who controls the destination of each individual; that is, a customer may be viewed as an intelligent entity with the ability to exercise control over his/her own outcome, or on the other hand he/she may be compliant with directions received from a greater authority. The implications of allowing customers to act of their own free will are of great interest when the plausible assumption is made that individuals will tend to make short-sighted decisions aimed at optimising their own individual outcomes; this topic will be expounded upon later.

3.2 Continuous-time MDPs

The queueing system described in Section 3.1 may be controlled by means of a decision-making scheme which determines the routing decision to be made upon each customer's arrival. Such a scheme is referred to as a *policy* for controlling the system. In general, a policy may be as elaborate as desired; the decision to be made upon a customer's arrival may depend on several factors, including the amount of time elapsed since the initialisation of the process, the congestion levels at the various facilities at the time of decision-making, the history of decisions made in the past and their consequences. One might even make a decision according to a random probability

distribution. Various types of policies will be discussed in greater detail in Section 3.6, but at this stage it suffices to adopt the general notion of a policy as some form of decision-making scheme which efficiently determines the actions of customers arriving in the system. The next example illustrates how the consequences of adopting a particular policy may be analysed.

Example 3.2.1. ***(Joining the shortest queue)***

Suppose there is a demand rate $\lambda = 1$ and only two service facilities ($N = 2$), each with only one server available ($c_1 = c_2 = 1$). Parameters for the two facilities are as follows:

$$\begin{aligned} \mu_1 &= 0.8, \qquad \beta_1 = 2, \qquad \alpha_1 = 6, \\ \mu_2 &= 0.4, \qquad \beta_2 = 1, \qquad \alpha_2 = 4. \end{aligned} \tag{3.2.1}$$

For the purposes of this example it will be assumed that an extra constraint is imposed, whereby the number of customers present at the first facility may not exceed two at any time, and there may be at most one customer present at the second facility (equivalently, the second facility will not allow anyone to queue, while its neighbour will permit a queue size of one).

Let it be assumed that the system begins with no customers present. Then the *state* of the system at any given time may be denoted by a vector $\mathbf{x} \in S$, where S is given by:

$$S = \big\{(x_1, x_2) : x_1, x_2 \in \mathbb{N}_0,\ x_1 \leq 2,\ x_2 \leq 1\big\}.$$

Let it be assumed that the system is controlled by means of the following simple policy, which depends only on the state of the system upon arrival of a customer:

- If the system is in state $(2, 1)$, customers are forced to balk.
- If the system is in any *other* state in S, customers go to the facility with the fewest customers already present. In the case of a tie, they always go to facility 1.

The policy described above may be represented as a function $\theta : S \to A_{\mathbf{x}}$ which associates a unique action $a \in A_{\mathbf{x}}$ with each state $\mathbf{x} \in S$, where $A_{\mathbf{x}}$ is the set of actions available at state $\mathbf{x}$. It is clear that the policy θ induces a continuous-time Markov chain (CTMC) over the 6 states in S, with state-dependent sojourn times determined by the arrival and service rates. It is natural to adopt a lexicographical ordering of the states, with the component x_1 taking precedence over x_2. Using the

approach described in Appendix A.2, the *infinitesimal generator matrix* Q, consisting of transition rates $q(\mathbf{x}, \mathbf{y})$ (for states $\mathbf{x}, \mathbf{y} \in S$), may be constructed as follows:

$$
\begin{aligned}
Q &= \begin{pmatrix}
q((0,0),(0,0)) & q((0,0),(0,1)) & \dots & q((0,0),(2,1)) \\
q((0,1),(0,0)) & q((0,1),(0,1)) & \dots & q((0,1),(2,1)) \\
q((1,0),(0,0)) & q((1,0),(0,1)) & \dots & q((1,0),(2,1)) \\
q((1,1),(0,0)) & q((1,1),(0,1)) & \dots & q((1,1),(2,1)) \\
q((2,0),(0,0)) & q((2,0),(0,1)) & \dots & q((2,0),(2,1)) \\
q((2,1),(0,0)) & q((2,1),(0,1)) & \dots & q((2,1),(2,1))
\end{pmatrix} \\
&= \begin{pmatrix}
-\lambda & 0 & \lambda & 0 & 0 & 0 \\
\mu_2 & -(\lambda+\mu_2) & 0 & \lambda & 0 & 0 \\
\mu_1 & 0 & -(\lambda+\mu_1) & \lambda & 0 & 0 \\
0 & \mu_1 & \mu_2 & -(\lambda+\mu_1+\mu_2) & 0 & \lambda \\
0 & 0 & \mu_1 & 0 & -(\lambda+\mu_1) & \lambda \\
0 & 0 & 0 & \mu_1 & \mu_2 & -(\mu_1+\mu_2)
\end{pmatrix}.
\end{aligned}
$$

This is a finite-state CTMC consisting of a single communicating class. Theorem A.2.1 in Appendix A.2 implies that the chain has a stationary distribution $\{\pi(\mathbf{x})\}_{\mathbf{x}\in S}$ satisfying:

$$
\sum_{\mathbf{y}\in S} \pi(\mathbf{y}) q(\mathbf{y}, \mathbf{x}) = 0 \qquad (\mathbf{x} \in S), \tag{3.2.2}
$$

where $\pi(\mathbf{x})$ is the limiting (stationary) probability for state $\mathbf{x} \in S$. Note that the stationary probabilities $\pi(\mathbf{x})$ and transition rates $q(\mathbf{x}, \mathbf{y})$ actually have a dependence on the policy θ, but since there is only one policy of interest in this example it is reasonable to suppress this θ-dependence in order to simplify the notation. The equations in (3.2.2) may be written:

$$
\begin{aligned}
&\lambda\pi((0,0)) = \mu_1\pi((1,0)) + \mu_2\pi((0,1)), \\
&(\lambda+\mu_2)\pi((0,1)) = \mu_1\pi((1,1)), \\
&(\lambda+\mu_1)\pi((1,0)) = \lambda\pi((0,0)) + \mu_1\pi((2,0)) + \mu_2\pi((1,1)), \\
&(\lambda+\mu_1+\mu_2)\pi((1,1)) = \lambda\big(\pi((0,1)) + \pi((1,0))\big) + \mu_1\pi((2,1)), \\
&(\lambda+\mu_1)\pi((2,0)) = \mu_2\pi((2,1)), \\
&(\mu_1+\mu_2)\pi((2,1)) = \lambda\big(\pi((1,1)) + \pi((2,0))\big).
\end{aligned} \tag{3.2.3}
$$

Upon imposing the normalising condition $\sum_{\mathbf{x}\in S}\pi(\mathbf{x})=1$, the equations in (3.2.3) may be solved to obtain the stationary distribution for the CTMC. In most of the examples considered in this thesis, it will not be practical (or useful) to give expressions for the steady-state probabilities $\pi(\mathbf{x})$ in terms of the parameters λ, μ_1, μ_2 etc. Indeed, even in this simple example involving only 6 states, the expressions for $\pi((0,0))$, $\pi((0,1))$ etc. in terms of λ, μ_1 and μ_2 are not 'neat' enough to be worth reproducing here. It is feasible, however, to simply calculate the approximate numerical values (to 4 decimal places) using the values for λ, μ_1 and μ_2 given in (3.2.1):

$$\begin{aligned}&\pi((0,0))\approx 0.1945, \quad \pi((0,1))\approx 0.1267, \quad \pi((1,0))\approx 0.1797,\\ &\pi((1,1))\approx 0.2218, \quad \pi((2,0))\approx 0.0504, \quad \pi((2,1))\approx 0.2268.\end{aligned} \tag{3.2.4}$$

For each state $\mathbf{x}\in S$, (3.2.4) provides an approximation for the expected long-run proportion of time $\pi(\mathbf{x})$ spent in state $\mathbf{x}$ when the policy θ is followed. So far, the holding costs β_i and rewards for service α_i have not featured in this analysis. In order to assess whether or not the policy θ is a sensible one, it would be useful to have a measure of its economic performance; such a measure could then be compared with those of other (possibly more desirable) policies. For this purpose, the concept of *expected long-run average reward per unit time* is required.

Consider an arbitrary state $\mathbf{x}\in S$. For every unit of time spent in $\mathbf{x}$, the holding costs incurred are given by $\beta_1x_1+\beta_2x_2$. However, if $x_i\geq 1$ (for $i=1,2$), then service completions occur at a rate μ_i at facility i, and each completion earns a reward α_i. It therefore makes sense to define the *net reward function* (referred to more simply as the *reward function*) for $\mathbf{x}\in S$ as:

$$r(\mathbf{x}):=\sum_{i=1}^{2}\left(\min(x_i,1)\alpha_i\mu_i-\beta_ix_i\right). \tag{3.2.5}$$

Thus, $r(\mathbf{x})$ is a measure of the expected net reward for each unit of time spent in state $\mathbf{x}$. The *expected long-run average net reward* (more simply, *average reward*) g_θ is then:

$$g_\theta:=\sum_{\mathbf{x}\in S}\pi(\mathbf{x})\,r(\mathbf{x})\approx 1.6913. \tag{3.2.6}$$

The value of g_θ quantifies the performance of the policy θ. There are other performance measures that may be used as alternatives to the expected long-run average reward, including those based on *discounting* of future costs and rewards; these will be discussed later. ⊠

Example 3.2.1 introduced the concepts of system states, decisions, transition rates and net rewards. Of course, the steady-state probabilities in (3.2.4) and average reward calculated in (3.2.6) were consequences of the particular policy θ being followed. In Chapter 2, policies for observable $M/M/1$ queues were represented by *thresholds*; for example, it was shown that a *socially optimal* policy could be defined by a threshold $n_o \in \mathbb{N}$, such that customers join the facility if and only if they observe fewer than n_o customers in the system when they arrive. In the case of multiple facilities, one may still adopt the notion of a socially optimal policy as a policy which maximises expected average reward (as in the $M/M/1$ case); however, the search for such a policy becomes more complicated, as the problem involves both routing control and admission control.

The problem of finding a socially optimal policy which maximises average reward (or optimises some alternative performance measure of interest) over all policies may be tackled within the framework of a *continuous-time Markov Decision Process* (CTMDP). This is defined below.

Definition 3.2.2. ***(Continuous-time Markov Decision Process)***

A continuous-time Markov Decision Process (CTMDP) is a collection of objects:

$$\Psi = \Big\{\mathcal{S}, \mathcal{A}_{\mathbf{x}}, \tau(\mathbf{x}, a), \sigma(\mathbf{x}, a, \mathbf{y}), \xi(\mathbf{x}, a), \xi_\gamma(\mathbf{x}, a), \gamma\Big\},$$

where $\mathcal{S}$ is the state space of the process (assumed either finite or countably infinite), $\mathcal{A}_{\mathbf{x}}$ is the set of actions available at state $\mathbf{x} \in \mathcal{S}$ (also assumed either finite or countably finite), $\tau(\mathbf{x}, a)$ is the sojourn time parameter which determines the distribution of the time spent in state $\mathbf{x} \in \mathcal{S}$ given that action $a \in \mathcal{A}_{\mathbf{x}}$ is chosen, $\sigma(\mathbf{x}, a, \mathbf{y})$ is the probability that the process transfers from state $\mathbf{x} \in \mathcal{S}$ to $\mathbf{y} \in \mathcal{S}$ at its next transition given that action $a \in \mathcal{A}_{\mathbf{x}}$ is chosen, $\xi(\mathbf{x}, a)$ and $\xi_\gamma(\mathbf{x}, a)$ are (respectively) the undiscounted and discounted net rewards earned for choosing action $a \in \mathcal{A}_{\mathbf{x}}$ at state $\mathbf{x} \in \mathcal{S}$, and $\gamma \in (0, \infty)$ is a continuous discount rate applied to future rewards.

Some generalisations of Definition 3.2.2 are certainly possible. The use of vectors $\mathbf{x}$, $\mathbf{y}$ etc. to represent system states is specific to the needs of this thesis, but in the general field of CTMDPs, system states may be scalar quantities, matrices or various other objects; moreover, continuous state and action spaces may be considered (see [14, 41, 81]). In Chapter 4 and onwards, system states will usually be denoted by vectors $\mathbf{x} = (x_1, x_2, ..., x_N)$ except in the case where only a single facility is being considered ($N = 1$), in which case it will be appropriate to use a scalar $x \in \mathbb{N}_0$

to denote the state. In addition, some authors include a subscript t on the reward function and transition rates to indicate possible time-dependence. In this thesis it will be assumed that rewards and transition rates are *stationary*; that is, they remain the same at all times.

The continuous discount rate $\gamma \in (0, \infty)$ is relevant only in problems where the economic performance measure of interest is the *expected total discounted reward*. Further details of this criterion will be given later in this section. If one is interested only in the expected long-run average reward for the system (as in Example 3.2.1), then the discount rate γ may be ignored.

The remainder of this section examines how the N-facility queueing system presented in Section 3.1 fits within the framework of a CTMDP. First, however, some general comments about the evolution of a CTMDP are in order. In fact, a CTMDP is a special case of a *Semi-Markov Decision Process* (SMDP) (see [141, 197]), in which the time spent in a particular state follows an arbitrary probability distribution, and the system controller is required to make a decision every time the system state changes. Essentially, an SMDP evolves by remaining in a particular state for a randomly distributed amount of time and then 'jumping' to a different state, also determined randomly. Typical applications of SMDPs include the control of $GI/M/c$ and $M/G/c$ queueing systems (examples may be found in [157] and [159]). If the inter-transition times are *exponentially distributed* and actions are chosen at every transition, then the SMDP is a CTMDP.

Let Ψ be a CTMDP in some arbitrary state $\mathbf{x} \in S$ at time $t_0 \geq 0$, and suppose an action $a \in A_{\mathbf{x}}$ is chosen by the decision-maker. Then the time until the next state transition is exponentially distributed with parameter $\tau(\mathbf{x}, a)$, which depends on the action chosen as well as the state. Let $t_1 > t_0$ denote the time of the next transition. Then, for each $\mathbf{y} \in S$, $\sigma(\mathbf{x}, a, \mathbf{y})$ denotes the probability that the system jumps to state $\mathbf{y}$ at time t_1, given that action a is chosen at t_0. The transition times t_0, t_1 etc. are the *decision epochs* at which control may be exercised.

Returning to the queueing system described in Section 3.1, it is clear that the designated decision epochs for the system should be the times at which customers arrive, since these are the only times at which decisions are made. Furthermore, assuming that a newly-arrived customer always has the option to join any of the N facilities (or balk), it would seem logical to use $\{0, 1, 2, ..., N\}$ as the set of actions available at every state, where 0 denotes balking and $i \geq 1$ denotes joining facility i ($i = 1, 2, ..., N$). However, some thought must be applied to the formulation of the state

space S. It would seem natural to simply define the state space as the (countably infinite) set of N-vectors $(x_1, x_2, ..., x_N)$, where $x_i \in \mathbb{N}_0$ is the number of customers present at facility i, but in a continuous-time setting this approach has some limitations, as described below.

The general formulation of a CTMDP requires that the action chosen at a particular decision epoch (equivalently, transition time) $t_0 \geq 0$ determines the expected sojourn time in the present state, and also the probability distribution for the state reached at the next transition. Suppose the system state at time t_0 is represented by a vector $(x_1, x_2, ..., x_N)$ as described above, an action is chosen at t_0, and the next transition occurs at some future point in time $t_1 > t_0$. A problem arises in that, according to the sequence of events in a CTMDP, the state transition at t_1 takes place before the decision at t_1 is made. Thus, there is no option but to use the action chosen at t_0 to determine the fate of a customer who arrives at t_1. Essentially, by adopting this formulation, one requires the system controller to make anticipative decisions in advance of customer arrivals; he/she must decide what action will be taken *if* the next random event to take place is an arrival (as opposed to a service completion), irrespective of when this arrival might actually occur.

In fact, while this might initially appear to be a serious drawback, in purely mathematical terms it only presents a problem if the system controller wishes to implement a *time-dependent* policy. If a policy is employed which is *not* time-dependent and selects actions based only on the state of the system upon each customer's arrival, then the controller is always aware of the action he/she will choose *if* an arrival occurs before any other transition; in other words, the time until the next transition is irrelevant for decision-making purposes. Thus, the policy θ considered in Example 3.2.1, being a simple state-dependent policy, can be modelled within this framework. On the other hand, if the system controller wishes to implement a time-dependent policy, the question of whether or not to send a customer to a particular facility i may depend on whether or not that customer arrives before some 'cut-off' point $t' > t_0$; for example, a customer may be sent to facility i if their arrival occurs before t', but refused admission otherwise. Hence, the state space formulation described earlier must be refined in order to accommodate time-dependent policies.

It is sensible to follow an approach similar to that of Puterman [141] (p. 568), who considers an admission control problem for an $M/M/1$ queue. The principle is to consider decision epochs somewhat more generally as the points in time at which random events (arrivals and service completions)

take place, and amend the system state description so that it includes information about the latest random event to have occurred. Specifically, let the system state at some arbitrary decision epoch be given by $((x_1, x_2, ..., x_N), \omega)$, where (as before) $(x_1, x_2, ..., x_N)$ is a vector of head counts at the various facilities, and ω is the random event that occurs at the epoch in question. The symbol Λ will be used to denote the event that an arrival has occurred, and M_i (for $i = 1, 2, ..., N$) will denote a service completion at facility i. The state space $\mathcal{S}$ is then given by:

$$\mathcal{S} = \Big\{ \big((x_1, x_2, ..., x_N), \omega\big) : x_1, x_2, ..., x_N \in \mathbb{N}_0, \, \omega \in \{\Lambda, M_1, M_2, ..., M_N\} \Big\}. \tag{3.2.7}$$

It is important to specify that if $w = M_i$ for some $i \in \{1, 2, ..., N\}$, then this indicates that the number of customers present at facility i has just decreased from $x_i + 1$ to x_i as a result of a service completion, and *not* that the number of customers at i is 'about' to decrease from x_i to $x_i - 1$. For example, suppose one has a system with only two facilities ($N = 2$), and both facilities have one customer present at some arbitrary point in time. If the next random event to occur is a service completion at facility 1, then the system state at the next decision epoch is $\big((0, 1), M_1\big)$, and *not* $\big((1, 1), M_1\big)$. Indeed, if the latter convention was adopted then some of the states in $\mathcal{S}$ (as defined in (3.2.7)) would become redundant, as it would not be possible for the random event $\omega = M_i$ to accompany any vector $(x_1, x_2, ..., x_N)$ with $x_i = 0$. On the other hand, consider the same two-facility scenario and suppose the next event to occur is an arrival. Then the state at the next decision epoch is $\big((1, 1), \Lambda\big)$, indicating that the head count at one of the facilities *may* be about to increase by one, depending on the action chosen by the decision-maker.

Let $(\mathbf{x}, \omega)$ denote a generic state in $\mathcal{S}$, where $\mathbf{x} = (x_1, x_2, ..., x_N)$. The action sets $\mathcal{A}_{(\mathbf{x},\omega)}$ for the N-facility queueing system under consideration may be described as follows:

$$\mathcal{A}_{(\mathbf{x},\omega)} = \begin{cases} \{0, 1, ..., N\}, & \text{if } \omega = \Lambda, \\ \{0\}, & \text{otherwise.} \end{cases} \tag{3.2.8}$$

At any state $(\mathbf{x}, \omega) \in \mathcal{S}$ associated with an arrival (i.e. $\omega = \Lambda$), one assumes that all of the actions $0, 1, ..., N$ are available to the decision-maker. On the other hand, if $\omega = M_i$ for some facility $i \in \{1, 2, ..., N\}$ (indicating a service completion), then in physical terms there is no decision to make and as such one may use the degenerate set $A_{(\mathbf{x},\omega)} = \{0\}$ to represent the set of actions available, where 0 indicates that no customer is admitted to the system (since there is no customer

to admit). Thus, the action sets $A_{(\mathbf{x},\omega)}$ in (3.3.7) have no $\mathbf{x}$-dependence. However, in later sections it will sometimes be appropriate to consider a *finite* state space, in which case the action sets for states $(\mathbf{x},\omega)$ which lie on the 'boundary' of the finite space will need to be restricted in order to ensure that the process remains 'contained' within the finite state space of interest.

Next, consider the sojourn time parameters $\tau((\mathbf{x},\omega),a)$ for the system. If action $a \in \mathcal{A}_{(\mathbf{x},\omega)}$ is chosen at state $(\mathbf{x},\omega) \in \mathcal{S}$, then the time until the next decision epoch should be exponentially distributed with mean $(\tau((\mathbf{x},\omega),a))^{-1} > 0$. This implies the following definition:

$$\tau((\mathbf{x},\omega),a) = \begin{cases} \lambda + \sum_{j=1}^{N} \min(x_j, c_j)\mu_j + \mu_i, & \text{if } \omega = \Lambda,\ a = i \text{ for some } i \in \{1,2,...,N\} \text{ and } x_i < c_i, \\ \lambda + \sum_{j=1}^{N} \min(x_j, c_j)\mu_j, & \text{otherwise.} \end{cases} \tag{3.2.9}$$

Thus, the sojourn time parameter $\tau((\mathbf{x},\omega),a)$ is given by adding the total rate at which service completions take place in the system immediately following a visit to state $(\mathbf{x},\omega)$ (which depends on the choice of action a) to the system demand rate λ. After a sojourn in state $(\mathbf{x},\omega) \in \mathcal{S}$, the system 'jumps' to a new state $(\mathbf{x}',\omega') \in \mathcal{S}$ with transition probability $\sigma((\mathbf{x},\omega),a,(\mathbf{x}',\omega'))$. These transition probabilities can be expressed most intelligibly by letting $(\mathbf{x},\omega)$ be an arbitrary state in $\mathcal{S}$ and writing the probabilities $\sigma((\mathbf{x},\omega),a,(\mathbf{x}',\omega'))$ for all states $(\mathbf{x}',\omega') \in \mathcal{S}$ which are accessible from $(\mathbf{x},\omega)$ in a single transition. Before doing so, it will be convenient to define $\mathbf{x}^{i+}$ as the N-vector which is identical to $\mathbf{x} = (x_1, x_2, ..., x_N)$ except that the i^{th} component is increased by one; similarly, for N-vectors $\mathbf{x}$ with $x_i \geq 1$ for some arbitrary $i \in \{1,2,...,N\}$, let $\mathbf{x}^{i-}$ denote the vector which is identical to $\mathbf{x}$ except that the i^{th} component is reduced by one. That is:

$$\mathbf{x}^{i+} := \mathbf{x} + \mathbf{e}_i,$$
$$\mathbf{x}^{i-} := \mathbf{x} - \mathbf{e}_i,$$

where $\mathbf{e}_i$ is the i^{th} vector in the standard orthonormal basis of $\mathbb{R}^N$. The components of the vector $\mathbf{x}^{i+}$ (respectively, $\mathbf{x}^{i-}$) will be written $x_1^{i+}, x_2^{i+}, ..., x_N^{i+}$ (resp. $x_1^{i-}, x_2^{i-}, ..., x_N^{i-}$), so that x_j^{i+} (resp. x_j^{i-}) denotes the j^{th} component of the vector $\mathbf{x}^{i+}$ (resp. $\mathbf{x}^{i-}$) for $j \in \{1,2,...,N\}$. Let $(\mathbf{x},\omega)$ be an arbitrary state in $\mathcal{S}$. The cases $\omega = \Lambda$ and $\omega \neq \Lambda$ will be considered separately. If $\omega = \Lambda$, then the

probabilities $\sigma((\mathbf{x},\Lambda),a,(\mathbf{x}',\omega'))$ may be summarised for $i,j\in\{1,2,...,N\}$ as follows:

$$\begin{aligned}
&\sigma((\mathbf{x},\Lambda),i,(\mathbf{x}^{i+},\Lambda)) = \lambda/\tau((\mathbf{x},\Lambda),i),\\
&\sigma((\mathbf{x},\Lambda),0,(\mathbf{x},\Lambda)) = \lambda/\tau((\mathbf{x},\Lambda),0),\\
&\sigma((\mathbf{x},\Lambda),i,((\mathbf{x}^{i+})^{j-},M_j)) = \min(x_j^{i+},c_j)\mu_j/\tau((\mathbf{x},\Lambda),i),\\
&\sigma((\mathbf{x},\Lambda),0,(\mathbf{x}^{j-},M_j)) = \min(x_j,c_j)\mu_j/\tau((\mathbf{x},\Lambda),0),
\end{aligned}\tag{3.2.10}$$

and $\sigma((\mathbf{x},\Lambda),a,(\mathbf{x}',\omega'))=0$ for any destination state $(\mathbf{x}',\omega')\in\mathcal{S}$ and action $a\in\mathcal{A}_{(\mathbf{x},\omega)}$ not already accounted for in (3.2.10). On the other hand, for states $(\mathbf{x},\omega)$ with $\omega=M_i$ for some $i\in\{1,2,...,N\}$, the only action available is $a=0$. Hence, for $j\in\{1,2,...,N\}$:

$$\begin{aligned}
&\sigma((\mathbf{x},M_i),0,(\mathbf{x},\Lambda)) = \lambda/\tau((\mathbf{x},M_i),0),\\
&\sigma((\mathbf{x},M_i),0,(\mathbf{x}^{j-},M_j)) = \min(x_j,c_j)\mu_j/\tau((\mathbf{x},M_i),0),
\end{aligned}\tag{3.2.11}$$

and $\sigma((\mathbf{x},M_i),0,(\mathbf{x}',\omega'))=0$ for any destination state $(\mathbf{x}',\omega')\in\mathcal{S}$ not accounted for in (3.2.11). The subtlety of the CTMDP formulation given thus far is that the action chosen at a particular decision epoch determines (in conjunction with the underlying probabilistic structure) what happens only up to the point when the next decision is due to be made, i.e. when the next customer arrival occurs. In a sense, the naive formulation that was briefly discussed earlier (in which the system state was simply an N-vector $(x_1,x_2,...,x_N)$) went a step too far, by using the action chosen at time $t_0\geq 0$ to dictate the routing decision made at the *next* decision epoch $t_1>t_0$.

Referring to Definition 3.2.2, it is also necessary to define suitable reward functions $\xi((\mathbf{x},\omega),a)$ and $\xi_\gamma((\mathbf{x},\omega),a)$ in order to complete the CTMDP formulation. In Example 3.2.1, the expected long-run average reward per unit time under policy θ was defined as $g_\theta=\sum_{\mathbf{x}\in S}\pi(\mathbf{x})\,r(\mathbf{x})$. This definition for g_θ was made possible by the fact that the policy θ under consideration chose actions based only on the current system state (thereby enabling states to be represented simply by vectors $\mathbf{x}$, as opposed to pairs $(\mathbf{x},\omega)$). Consequently, it was possible to represent the evolution of the system under θ using a time-homogeneous CTMC and derive the stationary distribution $\{\pi(\mathbf{x})\}_{\mathbf{x}\in S}$. However, a somewhat more general definition for g_θ is required in order to accommodate policies which do not enable such an approach. Consider a new definition for the expected long-run average reward $g_\theta((\mathbf{x},\omega))$ under an arbitrary policy θ, given some initial state $(\mathbf{x},\omega)\in\mathcal{S}$:

$$g_\theta((\mathbf{x},\omega)) = \liminf_{t\to\infty} \frac{1}{t} E_\theta \left[\sum_{n=0}^{N_t} \xi((\mathbf{x}_n,\omega_n),a_n) \middle| (\mathbf{x}_0,\omega_0) = (\mathbf{x},\omega) \right], \tag{3.2.12}$$

where N_t denotes the number of decision epochs (equivalently, the number of transitions or 'jumps') occurring in time $[0,t]$, $(\mathbf{x}_n,\omega_n)$ is the state of the system at the n^{th} decision epoch, a_n is the corresponding action chosen, and $\xi((\mathbf{x},\omega),a)$ is the (undiscounted) reward earned for a sojourn in state $(\mathbf{x},\omega) \in \mathcal{S}$ given that action $a \in \mathcal{A}_{(\mathbf{x},\omega)}$ is chosen. Let T_n denote the time of the n^{th} decision epoch, so that $N_t = \max\{n : T_n \leq t\}$. It makes sense to follow Puterman [141] (p. 567) in defining $\xi((\mathbf{x},\omega),a)$ as the *expected* reward accumulated during a sojourn in state $(\mathbf{x},\omega)$ after choosing action a, but in doing so one must be mindful of the non-uniformity of the expected inter-transition times $E[T_{n+1} - T_n]$ caused by the dependence of the sojourn parameters $\tau((\mathbf{x},\omega),a)$ on $(\mathbf{x},\omega)$ and a. In fact, as will become clear throughout this chapter, there are a number of equally valid ways of defining the expected single-sojourn reward $\xi((\mathbf{x},\omega),a)$. Two alternative formulations will be given here, but it is not difficult to conceive other variants which are equally suitable.

It is logical to begin with the reward formulation which most literally follows the description of the system given in Section 3.1. Given that a reward $\alpha_i > 0$ is earned each time a service completion occurs at facility $i \in \{1,2,...,N\}$, it is reasonable to propose that the reward function $\xi((\mathbf{x},\omega),a)$ should incorporate a term α_i if and only if $\omega = M_i$. On the other hand, holding costs are accumulated continuously over time; specifically, the rate per unit time at which holding costs are accrued immediately following a visit to state $(\mathbf{x},\omega) \in \mathcal{S}$ is given by $\sum_{j=1}^N \beta_j x_j$, with the possibility of an extra cost β_i if a new customer is admitted to facility i at the relevant decision epoch. Using the additional property that the length of a sojourn in state $(\mathbf{x},\omega)$ after choosing action a is exponentially distributed with mean $1/\tau((\mathbf{x},\omega),a)$, the expected single-sojourn reward $\xi((\mathbf{x},\omega),a)$ may be defined for states $(\mathbf{x},\omega) \in \mathcal{S}$ and actions $a \in \mathcal{A}_{(\mathbf{x},\omega)}$ as follows:

$$\xi((\mathbf{x},\omega),a) = \begin{cases} \alpha_i - \dfrac{\sum_{j=1}^N \beta_j x_j}{\tau((\mathbf{x},\omega),a)}, & \text{if } \omega = M_i \text{ for some } i \in \{1,2,...,N\}, \\ -\dfrac{\sum_{j=1}^N \beta_j x_j + \beta_i}{\tau((\mathbf{x},\omega),a)}, & \text{if } \omega = \Lambda \text{ and } a = i \text{ for some } i \in \{1,2,...,N\}, \\ -\dfrac{\sum_{j=1}^N \beta_j x_j}{\tau((\mathbf{x},\omega),a)}, & \text{otherwise.} \end{cases} \tag{3.2.13}$$

On the other hand, an alternative approach might involve the system earning rewards corresponding

to the expected (net) payoffs of individual customers themselves, calculated at their times of arrival. Let $w_i(x)$ be the expected net reward earned by an individual customer for joining facility $i \in \{1, 2, ..., N\}$ when there are already $x \in \mathbb{N}_0$ customers present there. Since it is assumed that customers do not renege or jockey between queues, it is simple to calculate $w_i(x)$ for each $x \in \mathbb{N}_0$ as a function of the system parameters. Indeed, if $x < c_i$ then one simply has:

$$w_i(x) = \alpha_i - \frac{\beta_i}{\mu_i} \qquad (x < c_i). \tag{3.2.14}$$

If a customer joins facility i when there are already $x \geq c_i$ customers present (so that they have to wait before beginning service), then the number of service completions that must occur at facility i before their service begins is $x - c_i + 1$. Formally, their expected waiting time in the queue has a gamma distribution with parameters $x - c_i + 1$ and $c_i\mu_i$ (see [103], p. 518). Hence:

$$\begin{aligned} w_i(x) &= \alpha_i - \frac{\beta_i(x - c_i + 1)}{c_i\mu_i} - \frac{\beta_i}{\mu_i} \\ &= \alpha_i - \frac{\beta_i(x+1)}{c_i\mu_i} \qquad\qquad (x \geq c_i). \end{aligned} \tag{3.2.15}$$

Therefore, using (3.2.14) and (3.2.15), one may derive the following single-sojourn reward function as an alternative to the function $\xi((\mathbf{x}, \omega), a)$ given in (3.2.13):

$$\hat{\xi}((\mathbf{x}, \omega), a) = \begin{cases} \alpha_i - \dfrac{\beta_i}{\mu_i}, & \text{if } \omega = \Lambda,\ a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } x_i < c_i, \\ \alpha_i - \dfrac{\beta_i(x_i + 1)}{c_i\mu_i}, & \text{if } \omega = \Lambda,\ a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } x_i \geq c_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3.2.16}$$

It is clear that both of the reward formulations $\xi((\mathbf{x}, \omega), a)$ and $\hat{\xi}((\mathbf{x}, \omega), a)$ are logical. Indeed, the formulation in (3.2.13) is based on the real-time holding costs and rewards accrued during the system's evolution, while the alternative formulation in (3.2.16) is based on an unbiased estimate of each individual customer's contribution to the aggregate net reward, made at the time of their entry to the system. For ease of reference, the function $\xi(\cdot)$ in (3.2.13) will be called the *real-time* reward function, while the function $\hat{\xi}(\cdot)$ in (3.2.16) will be referred to as the *anticipatory* reward function. Intuitively, one would expect both functions to yield an accurate measurement for the expected average reward under a particular policy θ when used within (3.2.12), and a formal equivalence between the two formulations in a discrete-time setting will be proved in Section 3.5.

Figure 3.2 shows the results of a simulation experiment involving the parameters in Example 3.2.1, designed for the purpose of comparing the two reward formulations in (3.2.13) and (3.2.16). For the two-facility system under consideration, a CTMDP was formulated exactly as described in this section thus far (i.e. with the same state space and action sets, etc.) and allowed to evolve randomly according to the sojourn time and transition probability distributions given in (3.2.9) and (3.2.11) respectively. The figure shows the evolution of the average reward per unit time over a short time period of 5000 time units. Two simulation runs were performed, with the reward formulations (3.2.13) and (3.2.16) used alternately and $((0,0),\Lambda)$ chosen as the initial state. Under both formulations, the average reward quickly converges towards the value 1.6913 calculated in Example 3.2.1. When the original reward formulation (3.2.13) is used, the aggregate net reward tends to take negative values for very small values of t (the amount of time elapsed), since the rewards for service α_1 and α_2 are not earned until service completions occur; on the other hand, when the formulation (3.2.16) is used, the aggregate net reward takes comparatively large positive values when t is small, since the system is immediately credited for customers' expected net rewards (which are always non-negative under the policy θ) upon their entry to the system.

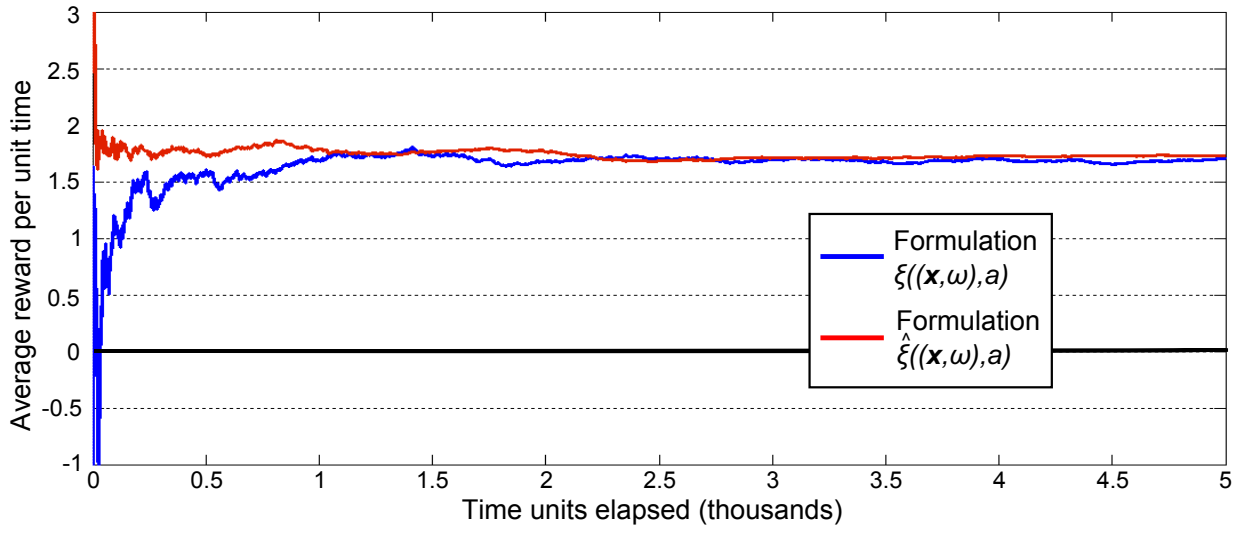


Figure 3.2: Comparison of the average rewards under $\xi(\cdot)$ and $\hat{\xi}(\cdot)$ for the system in Example 3.2.1.

Before concluding this section, it will also be useful to discuss the *discounting* of future rewards. Recall (from Definition 3.2.2) that the parameter $\gamma \in (0,\infty)$ represents a continuous discount rate. The *expected total discounted reward* $v_{\gamma,\theta}((\mathbf{x},\omega))$ under a particular policy θ and initial state

$(\mathbf{x},\omega) \in \mathcal{S}$ may be used as an alternative to the expected long-run average reward in (3.2.12) for measuring the performance of the system under θ. It is defined as follows:

$$v_{\gamma,\theta}((\mathbf{x},\omega)) = E_\theta\left[\sum_{n=0}^{\infty} e^{-\gamma T_n}\,\xi_\gamma((\mathbf{x}_n,\omega_n),a_n)\middle|(\mathbf{x}_0,\omega_0)=(\mathbf{x},\omega)\right], \tag{3.2.17}$$

where T_n is the time of the n^{th} 'jump' of the process to state $(\mathbf{x}_n,\omega_n) \in \mathcal{S}$, and $a_n \in \mathcal{A}_{(\mathbf{x}_n,\omega_n)}$ is the corresponding action chosen. The new function $\xi_\gamma(\cdot)$ is used for the single-sojourn rewards because, in the case of continuous discounting, it is not entirely accurate to use the same reward function as in the average reward case; indeed, the expected single-sojourn reward $\xi_\gamma((\mathbf{x},\omega),a)$ for a sojourn in state $(\mathbf{x},\omega)$ (following selection of action a) must take into account the continuous depreciation of the rewards earned while the sojourn is in progress. The standard approach used in the literature for defining single-sojourn rewards in CTMDPs with discounting (see, for example, [160]) is to allow 'lump sum' rewards to be earned at the beginning and end of a sojourn (which can be set equal to zero if appropriate), and also to allow an additional portion of the reward to be earned at a continuous rate during the sojourn itself. In order to be consistent with this approach, the function $\xi_\gamma((\mathbf{x}_n,\omega_n),a_n)$ in (3.2.17) should take the following general form:

$$\begin{aligned}\xi_\gamma((\mathbf{x}_n,\omega_n),a_n) = {} & R_1((\mathbf{x}_n,\omega_n),a_n) \\ & + R_2((\mathbf{x}_n,\omega_n),a_n)\int_0^\infty \tau((\mathbf{x}_n,\omega_n),a_n)e^{-\tau((\mathbf{x}_n,\omega_n),a_n)t}\int_0^t e^{-\gamma u}\,du\,dt \\ & + R_3((\mathbf{x}_n,\omega_n),a_n)\int_0^\infty \tau((\mathbf{x}_n,\omega_n),a_n)e^{-(\gamma+\tau((\mathbf{x}_n,\omega_n),a_n))t}\,dt,\end{aligned}$$

where $R_1((\mathbf{x}_n,\omega_n),a_n)$ and $R_3((\mathbf{x}_n,\omega_n),a_n)$ are the lump sum portions of the reward received at the start and end of the sojourn respectively, and $R_2((\mathbf{x}_n,\omega_n),a_n)$ is the reward rate while the sojourn is in progress. In the case of the N-facility queueing system under consideration, the rewards for service α_i are earned instantaneously upon completions of service, and therefore these rewards can be paid immediately as lump sums at states $(\mathbf{x},\omega) \in \mathcal{S}$ with $\omega = M_i$. On the other hand, holding costs must be handled in a different way, as these are subject to continuous discounting. For example, suppose the system is in some state $(\mathbf{x}_n,\omega_n) \in \mathcal{S}$ at its n^{th} decision epoch, and suppose also that $\omega_n = M_i$ for some $i \in \{1,2,...,N\}$. Then, noting that the time until the next transition is exponentially distributed with parameter $\tau((\mathbf{x}_n,\omega_n),a_n)$, the expected contribution of the reward

earned during the next sojourn to the cumulative value $v_{\gamma,\theta}((\mathbf{x},\omega))$ is given by:

$$\begin{aligned} & e^{-\gamma T_n}\int_0^\infty \tau((\mathbf{x}_n,\omega_n),a_n)e^{-\tau((\mathbf{x}_n,\omega_n),a_n)t}\left(\alpha_i-\sum_{j=1}^N\beta_j(\mathbf{x}_n)_j\int_0^t e^{-\gamma u}du\right)dt \\ & = e^{-\gamma T_n}\left(\alpha_i-\frac{\sum_{j=1}^N\beta_j(\mathbf{x}_n)_j}{\gamma+\tau((\mathbf{x}_n,\omega_n),a_n)}\right), \end{aligned}$$

where $(\mathbf{x}_n)_j$ denotes the j^{th} component of $\mathbf{x}_n$. In general, if one wishes to compute the expected total discounted reward using a 'real-time' reward formulation similar to (3.2.13), then in order to ensure compatibility with the definition of $v_{\gamma,\theta}((\mathbf{x},\omega))$ in (3.2.17) (in which the factor $e^{-\gamma T_n}$ already appears), $\xi_\gamma((\mathbf{x},\omega),a)$ should be defined for $(\mathbf{x},\omega)\in\mathcal{S}$ and $a\in\mathcal{A}_{(\mathbf{x},\omega)}$ as follows:

$$\xi_\gamma((\mathbf{x},\omega),a)=\begin{cases} \alpha_i-\dfrac{\sum_{j=1}^N\beta_jx_j}{\gamma+\tau((\mathbf{x},\omega),a)}, & \text{if } \omega=M_i \text{ for some } i\in\{1,2,...,N\}, \\ -\dfrac{\sum_{j=1}^N\beta_jx_j+\beta_i}{\gamma+\tau((\mathbf{x},\omega),a)}, & \text{if } \omega=\Lambda \text{ and } a=i \text{ for some } i\in\{1,2,...,N\}, \\ -\dfrac{\sum_{j=1}^N\beta_jx_j}{\gamma+\tau((\mathbf{x},\omega),a)}, & \text{otherwise.} \end{cases} \tag{3.2.18}$$

On the other hand, if one wishes to compute $v_{\gamma,\theta}((\mathbf{x},\omega))$ using an 'anticipatory' reward formulation similar to that given in (3.2.16), then the effect of the discount rate γ on the single-sojourn rewards is even more significant. In this case, it is necessary to find an expression for the expected net reward earned by an individual customer, taking into account the continuous discounting. If a customer joins some facility $i\in\{1,2,...,N\}$ when there are fewer than c_i customers already present (so that they begin service immediately), then their expected discounted reward is given by:

$$\begin{aligned} & \int_0^\infty \mu_ie^{-\mu_it}\left(\alpha_ie^{-\gamma t}-\beta_i\int_0^t e^{-\gamma u}du\right)dt \\ & = \frac{\alpha_i\mu_i-\beta_i}{\gamma+\mu_i}. \end{aligned}$$

On the other hand, if a customer joins facility i when the number of customers present x_i is greater than or equal to c_i, then their expected holding costs incurred while waiting in the queue must also be taken into account. In this case, their expected waiting time in the queue (before entering service) is gamma-distributed with parameters x_i-c_i+1 and $c_i\mu_i$ (please note that 'gamma-distributed'

here refers to the gamma distribution from probability theory, and should not be confused with the discount rate γ). Since their service time must also be accounted for, it follows that the customer's expected discounted net reward may be computed via the following integral:

$$\int_0^\infty \frac{(c_i\mu_i)^{x_i-c_i+1}}{(x_i-c_i)!}\, t^{x_i-c_i} e^{-c_i\mu_i t} \int_0^\infty \mu_i e^{-\mu_i u}\left(\alpha_i e^{-\gamma(t+u)} - \beta_i \int_0^{t+u} e^{-\gamma s} ds\right) du\, dt$$

$$= \frac{1}{\gamma(\gamma+\mu_i)}\left[\left(\frac{c_i\mu_i}{\gamma+c_i\mu_i}\right)^{x_i-c_i+1}(\gamma\alpha_i\mu_i - \beta_i\mu_i) - \beta_i(\gamma+\mu_i)\right].$$

Summarising these arguments, if one wishes to apply an 'anticipatory' reward formulation similar to (3.2.16) to a problem with continuous discounting at a rate $\gamma \in (0,\infty)$, then the reward function $\hat{\xi}_\gamma((\mathbf{x},\omega),a)$ (taking the place of $\xi_\gamma((\mathbf{x},\omega),a)$ in (3.2.17)) should be given by:

$$\hat{\xi}_\gamma((\mathbf{x},\omega),a) = \begin{cases} \dfrac{\alpha_i\mu_i - \beta_i}{\gamma+\mu_i}, & \text{if } \omega=\Lambda,\ a=i \text{ for some } i\in\{1,2,...,N\} \text{ and } x_i < c_i, \\ \dfrac{1}{\gamma(\gamma+\mu_i)}\left[\left(\dfrac{c_i\mu_i}{\gamma+c_i\mu_i}\right)^{x_i-c_i+1}(\gamma\alpha_i\mu_i-\beta_i\mu_i)-\beta_i(\gamma+\mu_i)\right], & \text{if } \omega=\Lambda,\ a=i \text{ for some } i\in\{1,2,...,N\} \text{ and } x_i \geq c_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3.2.19}$$

Figure 3.3 shows the results of another simulation experiment, again using the parameters from Example 3.2.1, designed to compare the discounted reward formulations (3.2.18) and (3.2.19). First, the 'real-time' formulation (3.2.18) was adopted and 50,000 simulation runs were carried out in order to estimate the total discounted reward for the system operating under the policy θ described in the example, with the discount rate γ set to a value of 0.1 and the initial state $(\mathbf{x}_0,\omega_0) = ((0,0),\Lambda)$ used. On each simulation run, sojourn times and transitions were generated randomly according to (3.2.9) and (3.2.10)-(3.2.11) and the process was allowed to continue until the value of $e^{-\gamma t}$ (where t denotes the total amount of time elapsed) eventually became negligibly small. The procedure was then repeated (i.e. another 50,000 simulation runs were performed) under the alternative formulation (3.2.19), with the same discount rate $\gamma = 0.1$ and initial state $((0,0),\Lambda)$ used. For both reward formulations, the figure shows the evolution of the *average* total discounted reward (over all trials) as the number of trials increases from 0 to 50,000.

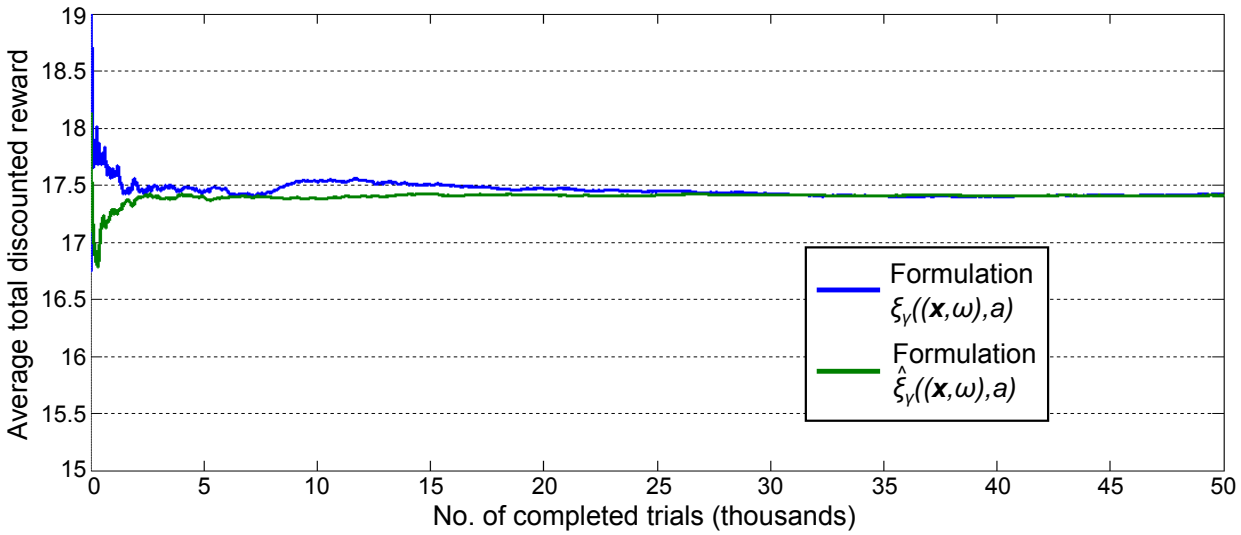


Figure 3.3: Comparison of average total discounted rewards under reward formulations $\xi_\gamma(\cdot)$ and $\hat{\xi}_\gamma(\cdot)$ for the system in Example 3.2.1, with a discount rate $\gamma = 0.1$ and initial state $(\mathbf{x}_0, \omega_0) = ((0,0), \Lambda)$.

Figure 3.3 shows that under both formulations, the average total discounted reward appears to converge towards a value of approximately 17.4. It should always be the case that the expected value $v_{\gamma,\theta}((\mathbf{0}, \Lambda))$ as defined in (3.2.17) takes the same value under both of the formulations $\xi_\gamma(\cdot)$ and $\hat{\xi}_\gamma(\cdot)$, so that there is an equivalence between the two reward formulations similar to the equivalence between the formulations (3.2.13) and (3.2.16) (for average reward problems) discussed previously. However, it is important to note that this equivalence applies *only* when the initial state is $(\mathbf{0}, \Lambda)$; that is, for any given system, it will generally *not* be the case that the expected total discounted reward $v_{\gamma,\theta}((\mathbf{x}, \omega))$ takes the same value under both reward formulations if the initial state $(\mathbf{x}, \omega) \in \mathcal{S}$ is arbitrary. To understand why this is, suppose the system is initialised in some state $(\mathbf{x}_0, \omega_0)$ with $\mathbf{x}_0 \neq \mathbf{0}$, so that there is already at least one customer present in the system. If the *real-time* formulation $\xi_\gamma(\cdot)$ is used, then the single-sojourn rewards earned in the early stages of the process include the holding costs for the customer(s) present initially, and also the rewards earned when their services are finally completed. On the other hand, if the *anticipatory* formulation is used, then these single-sojourn rewards include only the expected holding costs and rewards for *new* customers who enter the system. Similarly, if the initial state is $(\mathbf{0}, M_i)$ for some $i \in \{1, 2, ..., N\}$, then the initial reward under the real-time formulation $\xi_\gamma((\mathbf{0}, M_i))$ includes a 'payment' for the service which has just been completed, whereas the anticipatory reward $\hat{\xi}_\gamma((\mathbf{0}, M_i))$ does not. It is

only when the initial state is $(\mathbf{0}, \Lambda)$ that the two formulations are equivalent.

The specification of the reward functions $\xi((\mathbf{x}, \omega), a)$ and $\xi_\gamma((\mathbf{x}, \omega), a)$ (with $\hat{\xi}((\mathbf{x}, \omega), a)$ and $\hat{\xi}_\gamma((\mathbf{x}, \omega), a)$ as alternatives) completes the formulation of the queueing system described in Section 3.1 as a CTMDP. The next section will introduce an important and useful technique, known as *uniformisation*, whereby a CTMDP can be analysed in a discrete-time setting.

3.3 Uniformisation

Section 3.2 explained how to formulate the queueing system presented in Section 3.1 as a continuous-time MDP. In fact, there exists an equivalence between continuous-time MDPs and discrete-time MDPs which is similar in principle to the equivalence between discrete-time and continuous-time Markov chains discussed in Appendix A.2. This can be exploited using an extremely useful technique, known as *uniformisation*, which will be relied upon extensively throughout this thesis. The result itself is usually credited to Lippman [119], who applied the technique in the context of a queueing control problem involving an $M/M/c$ queueing system. Using what he refers to as a "new device", which involves using uniformisation to enlarge the set of decision epochs, Lippman successfully characterises optimal policies for a number of classical queueing control problems. Serfozo [160] also provides a somewhat more concise explanation of the same technique.

Continuous-time MDPs have already been defined in Section 3.2. Before proceeding with the equivalence result, it is necessary to provide a definition for discrete-time MDPs.

Definition 3.3.1. ***(Discrete-time Markov Decision Process)***

A discrete-time Markov Decision Process (MDP) is a collection of objects:

$$\Phi = \Big\{S, A_{\mathbf{x}}, p(\mathbf{x}, a, \mathbf{y}), r(\mathbf{x}, a), r_\phi(\mathbf{x}, a), \phi\Big\},$$

where S is the state space of the process (assumed either finite or countably infinite), $A_{\mathbf{x}}$ is the set of actions available at state $\mathbf{x} \in S$ (also assumed either finite or countably finite), $p(\mathbf{x}, a, \mathbf{y})$ is the one-step transition probability for transferring from $\mathbf{x} \in S$ to $\mathbf{y} \in S$ given that $a \in A_{\mathbf{x}}$ is chosen, $r(\mathbf{x}, a)$ and $r_\phi(\mathbf{x}, a)$ are (respectively) the undiscounted and discounted net rewards earned by choosing action $a \in A_{\mathbf{x}}$ at state $\mathbf{x} \in S$, and $\phi \in (0, 1)$ is a discount factor.

Thus, the definition of a discrete-time MDP is similar to that of a continuous-time MDP; the essential difference is that transitions (or equivalently, decision epochs) occur at uniformly-spaced intervals, so that sojourn times are fixed as opposed to being randomly distributed. In this thesis, problems will usually be formulated as discrete-time MDPs rather than CTMDPs, and therefore "MDP" will be used to refer to a discrete-time MDP unless otherwise stated. Note that Definition 3.3.1 relates to a general scenario in which the state of the system can be represented adequately by a vector $\mathbf{x} \in S$; however, in the previous section it was deemed necessary to represent the state of the N-facility queueing system under consideration using a pair $(\mathbf{x}, \omega)$, where ω represents a random event. Since the early part of this section summarises some general results from the literature, it will be convenient to retain the 'simplified' notation (with vectors $\mathbf{x}$, $\mathbf{y}$ etc. to represent states) in order to express the relevant theoretical concepts in a more concise manner. Later in this section, the queueing system from earlier in this chapter will be considered again specifically.

The definition of the expected long-run average reward for a discrete-time MDP, given some policy θ and initial state $\mathbf{x} \in S$, differs only slightly from the CTMDP case. Specifically:

$$g_\theta(\mathbf{x}) := \liminf_{m\to\infty} m^{-1} E_\theta \left[\sum_{n=0}^{m-1} r(\mathbf{x}_n, a_n) \Big| \mathbf{x}_0 = \mathbf{x} \right], \tag{3.3.1}$$

where $\mathbf{x}_n$ is the system state at the n^{th} time epoch, and a_n is the corresponding action chosen. Note that due to the limit infimum in (3.3.1), the worst-case scenario is considered when the limit fails to exist. On the other hand, suppose rewards are discounted by a factor $\phi \in (0, 1)$ on each discrete time step. Then the expected total discounted reward under θ is given by:

$$v_{\phi,\theta}(\mathbf{x}) := E_\theta \left[\sum_{n=0}^{\infty} \phi^n r_\phi(\mathbf{x}_n, a_n) \Big| \mathbf{x}_0 = \mathbf{x} \right], \tag{3.3.2}$$

Thus, a reward earned n steps into the future is discounted by a factor ϕ^n (relative to an immediate reward). As observed by Sennott [156], $v_{\phi,\theta}(\mathbf{x})$ may be infinite if the rewards $r_\phi(\mathbf{x}_n, a_n)$ are unbounded. It is also worthwhile to note that the discount factor ϕ in a discrete-time MDP has a somewhat contrary meaning to the discount rate γ used in a CTMDP, in the sense that small values of ϕ (e.g. $\phi \approx 0$) represent very *severe* discounting rates, whereas if $\phi \approx 1$ then the total discounted reward over a finite number of time steps n is almost equal to the total *undiscounted* reward over the same number of time steps. Contrastingly, it is clear from (3.2.17) that in a *continuous-time* process, small values of the discount rate γ (e.g. $\gamma \approx 0$) represent very *mild* discounting rates,

whereas the effect of discounting becomes increasingly severe as γ tends to infinity.

Before proceeding further, it is necessary to introduce a definition for a particular kind of policy which applies to both discrete-time and continuous-time MDPs. So far in this chapter, the term 'policy' has been used in a somewhat vague manner to refer to the general strategy of a decision-maker for controlling the system. In Section 3.6, various types of policies will be defined and classified in a more formal way. However, the next definition is required immediately.

Definition 3.3.2. ***(Stationary policy)***

Let Φ be either a discrete-time or a continuous-time MDP, with state space S and action sets $A_{\mathbf{x}}$ for $\mathbf{x} \in S$. Suppose the system is controlled by means of a policy θ which associates a unique *action $\theta(\mathbf{x}) \in A_{\mathbf{x}}$ with any given state $\mathbf{x} \in S$; that is, at any decision epoch where the process is in state $\mathbf{x}$, the unique action $\theta(\mathbf{x})$ is chosen. Then θ is referred to as a* stationary *policy.*

Thus, a stationary policy may be regarded as a direct mapping $\theta : S \to A_{\mathbf{x}}$. The next result, found in [160], explains the equivalence between MDPs and CTMDPs with respect to expected long-run average rewards and total discounted rewards under a fixed stationary policy.

*** Theorem 3.3.3.** *Let Ψ be a continuous-time Markov Decision Process (CTMDP) with state space S and action sets $A_{\mathbf{x}}$ for $\mathbf{x} \in S$. For each state-action pair $(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$ one associates the sojourn time parameter $\tau(\mathbf{x}, a)$ and transition probabilities $\sigma(\mathbf{x}, a, \mathbf{y})$ (for $\mathbf{y} \in S$). The single-sojourn reward function is denoted by $\xi_\gamma(\mathbf{x}, a)$ if rewards are continuously discounted with rate $\gamma \in (0, \infty)$, and $\xi(\mathbf{x}, a)$ if there is no discounting. Assume that there exists $C > 0$ such that $(1 - \sigma(\mathbf{x}, a, \mathbf{x}))\tau(\mathbf{x}, a) \leq C < \infty$ for all $\mathbf{x} \in S$ and $a \in A_{\mathbf{x}}$. Let $\hat{\Psi}$ be a CTMDP with the same state and action sets as Ψ, with transition probabilities and reward functions given by:*

$$p(\mathbf{x}, a, \mathbf{y}) = \begin{cases} 1 - \dfrac{(1 - \sigma(\mathbf{x}, a, \mathbf{x}))\,\tau(\mathbf{x}, a)}{C}, & \text{if } \mathbf{y} = \mathbf{x}, \\ \dfrac{\sigma(\mathbf{x}, a, \mathbf{y})\,\tau(\mathbf{x}, a)}{C}, & \text{if } \mathbf{y} \neq \mathbf{x}, \end{cases} \tag{3.3.3}$$

$$r(\mathbf{x}, a) = \frac{\xi(\mathbf{x}, a)\tau(\mathbf{x}, a)}{C}, \tag{3.3.4}$$

$$r_\gamma(\mathbf{x}, a) = \xi_\gamma(\mathbf{x}, a) \left(\frac{\gamma + \tau(\mathbf{x}, a)}{\gamma + C} \right), \tag{3.3.5}$$

and suppose $\hat{\Psi}$ *has sojourn time parameter* C *for all* $(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$. *Let* Φ *denote a* discrete-time MDP *with the same state and action spaces as* Ψ *and* $\hat{\Psi}$, *single-step reward function* $r(\mathbf{x}, a)$ *or* $r_\gamma(\mathbf{x}, a)$ *(depending on whether or not discounting is used), transition probabilities* $p(\mathbf{x}, a, \mathbf{y})$ *and discount factor* $\phi := C/(\gamma + C)$, *so that* Φ *is identical in construction to* $\hat{\Psi}$ *except with respect to its discounting structure. If* Ψ, $\hat{\Psi}$ *and* Φ *are all controlled by the same stationary policy* θ, *then* Ψ *and* $\hat{\Psi}$ *yield the same expected long-run average reward under* θ *and this is equal to* $Cg_\theta^\dagger(\mathbf{x})$, *where* $g_\theta^\dagger(\mathbf{x})$ *is the expected long-run average reward for the process* Φ. *Furthermore, all three processes yield the same expected total discounted reward* $v_{\gamma,\theta}(\mathbf{x})$ *for any initial state* $\mathbf{x} \in S$.

For the proof of the theorem, please refer to Serfozo [160].

A valuable observation made by Serfozo is that, while the continuous-time processes $\hat{\Psi}$ and Ψ have a larger class of *non-stationary* policies than the discrete-time process Φ, all three processes have the same class of *stationary* policies. Thus, if a stationary policy can be found for any one of $\hat{\Psi}$, Ψ and Φ which maximises the expected average reward over all stationary policies, then it must also do so for the other two processes. In this sense, the three processes are equivalent.

Typically, the principle of uniformisation is used to treat a continuous-time problem within a discrete-time setting, and indeed this will be the approach taken throughout most of this thesis. However, it is useful to bear in mind that when uniformisation is applied, there are actually three processes involved: the original CTMDP Ψ, the discrete-time MDP Φ and an *intermediate* process $\hat{\Psi}$ which, as specified by Theorem 3.3.3, is a continuous-time process with uniformly-distributed sojourn times. Of course, the fact that sojourn times for $\hat{\Psi}$ are uniformly distributed does *not* imply that all sojourn times for the process are identical; it simply means that each sojourn time follows the same random distribution (specifically, an exponential distribution with parameter C). The technique of uniformisation may be illustrated using a diagram similar to that given in Puterman [141] (p. 567); see Figure 3.4 below. In some of the results later in this chapter, the equivalence between Ψ and $\hat{\Psi}$ (referred to as the "uniformised CTMDP") will prove to be useful.

In the case of the N-facility queueing system described in Section 3.1, the results of Theorem 3.3.3 (with some minor modifications) can be applied in order to convert the continuous-time MDP Ψ formulated in Section 3.2 to a discrete-time MDP Φ which yields the same expected long-run average reward $g_\theta((\mathbf{x}, \omega))$ and expected total discounted reward $v_{\gamma,\theta}((\mathbf{x}, \omega))$ under a given stationary policy

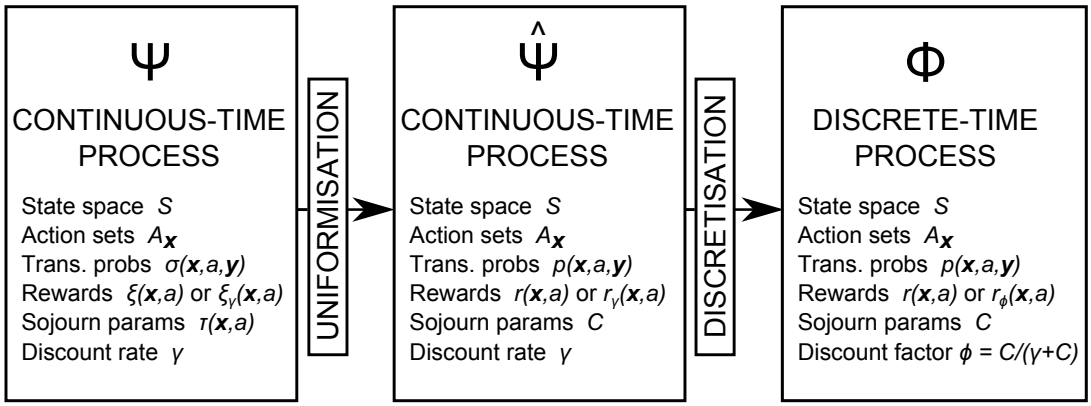


Figure 3.4: The process of uniformisation involving the three processes Ψ, $\hat{\Psi}$ and Φ.

θ and initial state $(\mathbf{x}, \omega) \in \mathcal{S}$. An explicit formulation for Φ is given below.

Discrete-time MDP formulation

The *state space* (denoted $\mathcal{S}$) for Φ is the same as for the CTMDP in Section 3.2. That is:

$$\mathcal{S} = \Big\{ \big((x_1, x_2, ..., x_N), \omega\big) : x_1, x_2, ..., x_N \in \mathbb{N}_0, \, \omega \in \{\Lambda, M_1, M_2, ..., M_N\} \Big\}. \tag{3.3.6}$$

The *action sets* (denoted $\mathcal{A}_{(\mathbf{x},\omega)}$) for states $(\mathbf{x}, \omega) \in \mathcal{S}$ are also the same as in Section 3.2:

$$\mathcal{A}_{(\mathbf{x},\omega)} = \begin{cases} \{0, 1, ..., N\}, & \text{if } \omega = \Lambda, \\ \{0\}, & \text{otherwise.} \end{cases} \tag{3.3.7}$$

In order to define the transition probabilities for the MDP, which will be denoted (in this particular formulation) by $\mathcal{P}((\mathbf{x}, \omega), a, (\mathbf{x}', \omega'))$ for $(\mathbf{x}, \omega), (\mathbf{x}', \omega') \in \mathcal{S}$, it is necessary to find a constant C such that $\tau((\mathbf{x}, \omega), a) \leq C$ for all $(\mathbf{x}, \omega) \in \mathcal{S}$ and $a \in \mathcal{A}_{(\mathbf{x},\omega)}$, where $\tau((\mathbf{x}, \omega), a)$ is the sojourn time parameter defined in (3.2.9). Clearly, it is sufficient to choose any positive value C satisfying $C \geq \lambda + \sum_{i=1}^{N} c_i \mu_i$. Then, by applying the result of Theorem 3.3.3 to the 'jump' probabilities $\sigma((\mathbf{x}, \omega), a, (\mathbf{x}', \omega'))$ defined in (3.2.10)-(3.2.11), the probabilities $\mathcal{P}((\mathbf{x}, \omega), a, (\mathbf{x}', \omega'))$ for states $(\mathbf{x}, \omega) \in \mathcal{S}$ with $\omega = \Lambda$ may be summarised for $i, j \in \{1, 2, ..., N\}$ as follows:

$$\begin{aligned} &\mathcal{P}((\mathbf{x}, \Lambda), i, (\mathbf{x}^{i+}, \Lambda)) = \lambda\Delta, \\ &\mathcal{P}((\mathbf{x}, \Lambda), i, ((\mathbf{x}^{i+})^{j-}, M_j)) = \min(x_j^{i+}, c_j)\mu_j\Delta, \\ &\mathcal{P}((\mathbf{x}, \Lambda), i, (\mathbf{x}, \Lambda)) = 1 - \lambda\Delta - \sum_{k=1}^{N} \min(x_k^{i+}, c_k)\mu_k\Delta, \end{aligned}$$

$$\mathcal{P}((\mathbf{x},\Lambda),0,(\mathbf{x}^{j-},M_j)) = \min(x_j,c_j)\mu_j\Delta,$$
$$\mathcal{P}((\mathbf{x},\Lambda),0,(\mathbf{x},\Lambda)) = 1-\sum_{k=1}^{N}\min(x_k,c_k)\mu_k\Delta, \tag{3.3.8}$$

and $\mathcal{P}((\mathbf{x},\Lambda),0,(\mathbf{x}',\omega')) = 0$ for any state $(\mathbf{x}',\omega') \in \mathcal{S}$ not already accounted for in (3.3.8). Here, $\Delta := 1/C$ is the *discrete-time step size*, which is assumed to satisfy:

$$0 < \Delta \leq \left(\lambda + \sum_{i=1}^{N} c_i\mu_i\right)^{-1}. \tag{3.3.9}$$

Note that (3.3.8) specifies a non-zero value for the *self-transition* probability $\mathcal{P}((\mathbf{x},\Lambda),i,(\mathbf{x},\Lambda))$, whereas the corresponding 'jump' probability $\sigma((\mathbf{x},\Lambda),i,(\mathbf{x},\Lambda))$ in the CTMDP formulation of Section 3.2 was equal to zero. Similarly, although the value of $\sigma((\mathbf{x},\Lambda),0,(\mathbf{x},\Lambda))$ was non-zero in the CTMDP formulation (due to the fact that self-transitions may occur in the continuous-time process when balking is chosen at states with $\omega = \Lambda$), its value was smaller than that of the self-transition probability $\mathcal{P}((\mathbf{x},\Lambda),0,(\mathbf{x},\Lambda))$ in (3.3.8). This illustrates the general principle that uniformisation causes extra 'self-transitions' by reducing the time in between successive decision epochs (see [141], p. 562). For states $(\mathbf{x},\omega) \in \mathcal{S}$ with $\omega = M_i$ for some facility i, the probabilities $\mathcal{P}((\mathbf{x},\omega),a,(\mathbf{x}',\omega'))$ may be summarised for facilities $j \in \{1,2,...,N\}$ as follows:

$$\mathcal{P}((\mathbf{x},M_i),0,(\mathbf{x},\Lambda)) = \lambda\Delta,$$
$$\mathcal{P}((\mathbf{x},M_i),0,(\mathbf{x}^{j-},M_j)) = \min(x_j,c_j)\mu_j\Delta,$$
$$\mathcal{P}((\mathbf{x},M_i),0,(\mathbf{x},M_i)) = 1-\lambda\Delta-\sum_{k=1}^{N}\min(x_k,c_k)\mu_k\Delta, \tag{3.3.10}$$

and $\mathcal{P}((\mathbf{x},M_i),0,(\mathbf{x}',\omega')) = 0$ for any state $(\mathbf{x}',\omega') \in \mathcal{S}$ not accounted for in (3.3.10). Once again, there is a non-zero 'self-transition' probability $\mathcal{P}((\mathbf{x},M_i),0,(\mathbf{x},M_i))$ which does not appear in (3.2.11). The remaining task, in order to complete the discrete-time MDP formulation, is to define the undiscounted and discounted reward functions for the process. These will be denoted here by $\mathcal{R}((\mathbf{x},\omega),a)$ and $\mathcal{R}_\phi((\mathbf{x},\omega),a)$ respectively. As discussed in the previous section, there are at least two different approaches that may be taken as far as the single-transition rewards are concerned; one approach is to define these rewards based on the 'real-time' holding costs and service completions incurred by the system in a single transition, while an alternative method is to adopt an 'anticipatory' formulation and reward the system based on the *expected* net rewards of

individual customers as they enter the system. In this particular case, suppose that a 'real-time' reward formulation is used. By applying the result of Theorem 3.3.3 with the continuous-time reward function $\xi((\mathbf{x},\omega),a)$ as defined in (3.2.13) and $\tau((\mathbf{x},\omega),a)$ as defined in (3.2.9), one obtains the following discretised function $\mathcal{R}((\mathbf{x},\omega),a)$ (for use in *undiscounted* problems):

$$\mathcal{R}((\mathbf{x},\omega),a) = \begin{cases} \alpha_i \left(\lambda + \sum_{j=1}^{N} \min(x_j, c_j)\mu_j \right) - \sum_{j=1}^{N} \beta_j x_j, & \text{if } \omega = M_i \text{ for some } i \in \{1,2,...,N\}, \\ -\sum_{j=1}^{N} \beta_j x_j^{i+}, & \text{if } \omega = \Lambda \text{ and } a = i \text{ for some } i \in \{1,2,...,N\}, \\ -\sum_{j=1}^{N} \beta_j x_j, & \text{otherwise.} \end{cases} \tag{3.3.11}$$

Note that the discrete-time step size $\Delta = 1/C$ (henceforth referred to as the *uniformisation parameter*) has been omitted from (3.3.11). If the definition of the uniformised reward given in Theorem 3.3.3 was adhered to exactly, then the parameter Δ would be included as a multiplicative factor. However, in that case, the expected long-run average reward $g_\theta^\dagger(\mathbf{x})$ for the discretised process Φ operating under some stationary policy θ would (as stated by the theorem) be equal to Δ times the average reward of the continuous-time processes Ψ and $\hat{\Psi}$ under the same policy. Essentially, by omitting the factor Δ in (3.3.11), one may ensure that the expected long-run average reward obtained by using (3.3.11) within (3.3.1) gives the correct value for the average reward *per unit time*, as opposed to the average reward per discrete time step. Next, by applying the result of Theorem 3.3.3 to the continuous-time function $\xi_\gamma((\mathbf{x},\omega),a)$ defined in (3.2.18), one obtains the following reward function $\mathcal{R}_\phi(\cdot)$ for use in *discounted* problems (recall that the discount factor ϕ is obtained from γ via the transformation $\phi = 1/(1+\gamma\Delta)$, hence $\gamma = (1-\phi)/(\phi\Delta)$):

$$\mathcal{R}_\phi((\mathbf{x},\omega),a) = \begin{cases} \alpha_i\big(1-\phi(1-\lambda\Delta)\big) + \phi\Delta \sum_{j=1}^{N} \big(\alpha_i \min(x_j,c_j)\mu_j - \beta_j x_j\big), & \text{if } \omega = M_i \text{ for some } i \in \{1,2,...,N\}, \\ -\phi\Delta \sum_{j=1}^{N} \beta_j x_j^{i+}, & \text{if } \omega = \Lambda \text{ and } a = i \text{ for some } i \in \{1,2,...,N\}, \\ -\phi\Delta \sum_{j=1}^{N} \beta_j x_j, & \text{otherwise.} \end{cases} \tag{3.3.12}$$

Naturally, it is also possible to apply the result of Theorem 3.3.3 to the anticipatory reward functions in (3.2.16) and (3.2.19) in order to obtain alternative reward functions for the discretised process, but the resulting expressions are not particularly elegant and in later sections it will be sufficient to restrict attention to the 'real-time' formulations given in (3.3.11) and (3.3.12).

Before concluding this section, it is appropriate to make a further comment regarding the uniformisation parameter Δ. In applications involving the uniformisation of CTMDPs, it is quite common for the assumption to be made that the supremum of the sojourn time parameters $\tau(\cdot)$ over all state-action pairs is bounded above by one, so that $\Delta = 1$ is a valid choice for the uniformisation parameter (see, for example, [35]). This assumption can usually be made without loss of generality, since the units of time can always be re-scaled; for example, in the queueing system under consideration, one might reasonably assume that the arrival rate λ and service rates μ_i are scaled in such a way that $\lambda + \sum_{i=1}^{N} c_i \mu_i = 1$. By making this assumption, one would then be able to suppress the appearance of the parameter Δ in the transition probabilities (3.3.8)-(3.3.10).

While the assumption that $\lambda + \sum_{i=1}^{N} c_i \mu_i = 1$ is quite reasonable if λ and $\mu_1, \mu_2, ..., \mu_N$ are treated as fixed parameters, it becomes somewhat impractical when these parameters are allowed to vary. Some of the results in later chapters of this thesis will involve treating the demand rate λ as a variable, in order to investigate the effect of increasing or decreasing λ on the performance of the system. When the value of λ changes, there is an implicit effect on the range of positive values which Δ may take; as such, it becomes somewhat awkward to suppress Δ completely from the relevant analyses (although there are ways in which this can be done). Clearly, it would not be desirable to create unnecessary confusion by suppressing the parameter Δ in some results and including it in others. It will therefore be deemed appropriate in this thesis to simply include the uniformisation parameter Δ throughout; that is, there is no assumption made that $\lambda + \sum_{i=1}^{N} c_i \mu_i = 1$, and Δ will be included in all transition probabilities associated with uniformised MDPs.

In some of the results to follow later, it will be necessary to specify a particular value for the uniformisation parameter Δ; for example, $\Delta = (\lambda + \sum_{i=1}^{N} c_i \mu_i)^{-1}$ will be an obvious choice. However, unless otherwise stated, it may be assumed somewhat more generally that:

$$\Delta \in \left(0, \left(\lambda + \sum_{i=1}^{N} c_i \mu_i \right)^{-1} \right].$$

The fact that there always exists a range of possible values for Δ when uniformisation is applied is a very important principle, which can be exploited in a similar way to the principle of introducing extra 'self-transitions' in a discrete-time Markov chain without affecting the underlying stationary distribution of the chain (see, for example, the convergence proof for relative value iteration found in Bertsekas [13] (p. 191), which will be discussed briefly in Section 3.7).

Further results involving stationary policies will be discussed in the next section.

3.4 Optimality of stationary policies

An important theme in later chapters of this thesis will be the identification and characterisation of policies θ which maximise the expected long-run average reward for the system $g_\theta((\mathbf{x}, \omega))$, defined in (3.2.12). In keeping with the terminology introduced in Chapter 2, policies which achieve this objective will be referred to as *socially optimal*. Policies which maximise the expected total discounted reward $v_{\gamma,\theta}((\mathbf{x}, \omega))$ in (3.2.17), given some continuous discounting rate $\gamma \in (0, \infty)$, will not be expressly of interest in this thesis. However, it transpires that various important theoretical results involving *average reward* problems (i.e. those in which the objective is to maximise the average reward $g_\theta((\mathbf{x}, \omega))$) may be proved by treating an average reward problem as a limiting case of a discounted reward problem, as the continuous discount rate γ tends to zero.

The next definition clarifies exactly what is meant by an *optimal policy*.

Definition 3.4.1. ***(Average reward optimal and γ-discount optimal policies)***

An average reward optimal *policy is any admissible policy θ^* for which:*

$$g_{\theta^*}((\mathbf{x}, \omega)) = \sup_\theta g_\theta((\mathbf{x}, \omega)) \qquad \forall\ (\mathbf{x}, \omega) \in \mathcal{S},$$

where the supremum is taken over all admissible policies for the system. Similarly, a γ-discount optimal policy *is any admissible policy θ^*_γ for which:*

$$v_{\gamma,\theta^*_\gamma}((\mathbf{x}, \omega)) = \sup_\theta v_{\gamma,\theta}((\mathbf{x}, \omega)) \qquad \forall\ (\mathbf{x}, \omega) \in \mathcal{S}.$$

In Definition 3.4.1, the term "admissible policy" in fact refers to any strategy for controlling the system which does not fundamentally violate the assumptions of Section 3.1 (by, for example,

postponing the decision made when a customer arrives, or using knowledge of the arrival times of future customers to inform decision-making). In particular, an admissible policy might make decisions based on the history of actions chosen in the past, or the amount of time elapsed since the initialisation of the process. A formal classification for admissible policies will be provided in Section 3.6, but in this section the only important distinction is between *stationary* policies (which select actions based only on the current system state) and *non-stationary* policies.

As discussed in Section 3.2, the augmented state space formulation given in (3.2.7) (with $(\mathbf{x}, \omega)$ used to denote a generic system state) was deemed necessary in order to enable the implementation of time-dependent policies. However, if it were possible to restrict attention to stationary policies *only*, then it would be sufficient to represent the system state using only an N-vector $\mathbf{x} = (x_1, x_2, ..., x_N)$, since this would contain all information required by the decision-maker; in particular, the exact time of a customer's arrival would be irrelevant for decision-making purposes. In general, the analysis of CTMDPs and MDPs becomes much simpler if one considers stationary policies only. As observed by Sennott [159], "the advantages for implementation of a stationary policy are clear, since it necessitates the storage of less information than required to implement a general policy". However, some justification is required in order to simplify the problem in this way.

Recall that Ψ denotes the CTMDP formulated in Section 3.2. The purpose of this section is to prove that there exists an average reward optimal policy for Ψ which is also a *stationary* policy. In order to do this, it will be sufficient to prove certain properties of the *optimal value function* $v_\gamma(\cdot)$ associated with a γ-discounted problem. This function is defined for $(\mathbf{x}, \omega) \in \mathcal{S}$ by:

$$v_\gamma((\mathbf{x}, \omega)) := v_{\theta^*_\gamma}((\mathbf{x}, \omega)) = \sup_\theta v_{\gamma,\theta}((\mathbf{x}, \omega)).$$

Given any continuous discount-rate $\gamma \in (0, \infty)$, results from the literature imply that there exists a γ-discount optimal stationary policy for the process Ψ. The next result, which is adapted from Sennott [157] and relies upon results in [12] and [155], establishes this fact.

*** Theorem 3.4.2.** *Let the discount rate $\gamma \in (0, \infty)$ be fixed. The optimal value function $v_\gamma((\mathbf{x}, \omega)) = \sup_\theta v_{\gamma,\theta}((\mathbf{x}, \omega))$ satisfies the following γ-discount optimality equations:*

$$v_\gamma((\mathbf{x}, \omega)) = \max_{a \in A_{(\mathbf{x},\omega)}} \left\{ \xi_\gamma((\mathbf{x}, \omega), a) + L_\gamma((\mathbf{x}, \omega), a) \sum_{(\mathbf{x}', \omega') \in \mathcal{S}} \sigma((\mathbf{x}, \omega), a, (\mathbf{x}', \omega'))\, v_\gamma((\mathbf{x}', \omega')) \right\}, \quad (3.4.1)$$

where, for states $(\mathbf{x},\omega)\in\mathcal{S}$ *and actions* $a\in\mathcal{A}_{(\mathbf{x},\omega)}$, $L_\gamma((\mathbf{x},\omega),a)$ *is given by:*

$$L_\gamma((\mathbf{x},\omega),a)=\int_0^\infty \tau((\mathbf{x},\omega),a)e^{-(\gamma+\tau((\mathbf{x},\omega),a))t}\,dt=\frac{\tau((\mathbf{x},\omega),a)}{\gamma+\tau((\mathbf{x},\omega),a)}. \tag{3.4.2}$$

Moreover, let θ^*_γ *be any stationary policy such that for all* $(\mathbf{x},\omega)\in\mathcal{S}$, *the action* $\theta^*_\gamma((\mathbf{x},\omega))$ *chosen by* θ^*_γ *attains the maximum in (3.4.1). Then* θ^*_γ *is* γ*-discount optimal.*

Proof. The proof relies upon some assumptions, and therefore it is necessary to verify that these hold in the case of the CTMDP Ψ formulated in Section 3.2. It is assumed that:

1. The single-sojourn rewards are bounded above (but not necessarily bounded below); that is, there exists $R\in\mathbb{R}$ such that $\xi_\gamma((\mathbf{x},\omega),a)\leq R$ for all $((\mathbf{x},\omega),a)\in\mathcal{S}\times\mathcal{A}_{(\mathbf{x},\omega)}$.
2. There exist $\epsilon>0$ and $t_0>0$ such that $e^{-\tau((\mathbf{x},\omega),a)t_0}\geq\epsilon$ for all $((\mathbf{x},\omega),a)\in\mathcal{S}\times\mathcal{A}_{(\mathbf{x},\omega)}$.
3. For all states $(\mathbf{x},\omega)\in\mathcal{S}$, it is the case that $v_\gamma((\mathbf{x},\omega))>-\infty$.

Indeed, let $\alpha^*:=\max(\alpha_1,\alpha_2,...,\alpha_N)$ denote the maximum value of service across all facilities. Then it can be verified from (3.2.18) that $\xi_\gamma((\mathbf{x},\omega),a)\leq\alpha^*$ for all $((\mathbf{x},\omega),a)\in\mathcal{S}\times\mathcal{A}_{(\mathbf{x},\omega)}$ and $\gamma\in(0,\infty)$, so the first assumption holds. In order to verify the second assumption, one may choose $t_0=(\lambda+\sum_{i=1}^N c_i\mu_i)^{-1}>0$. Then, since the sojourn time parameters $\tau((\mathbf{x},\omega),a)$ are uniformly bounded above by $\lambda+\sum_{i=1}^N c_i\mu_i$, it follows that $e^{-\tau((\mathbf{x},\omega),a)t_0}\geq e^{-1}>0$ for all state-action pairs $((\mathbf{x},\omega),a)$. This condition ensures that the process is 'regular', in the sense that it makes only finitely many transitions in any finite amount of time (see [157], p. 250). Finally, let θ_0 denote the degenerate policy which chooses to balk at any state $(\mathbf{x},\omega)$ with $\omega=\Lambda$, and consider an arbitrary initial state $(\mathbf{x},\omega)\in\mathcal{S}$. Obviously, since no customers are admitted under the degenerate policy θ_0, the rate at which holding costs are incurred is steepest at the beginning of the process. Hence, since the rewards α_i are non-negative, it follows from (3.2.17) and (3.2.18) that:

$$v_{\gamma,\theta_0}((\mathbf{x},\omega))\geq-\sum_{j=1}^N\beta_j x_j\int_0^\infty e^{-\gamma t}dt=-\frac{\sum_{j=1}^N\beta_j x_j}{\gamma}.$$

Thus, since $v_\gamma((\mathbf{x},\omega))\geq v_{\gamma,\theta_0}((\mathbf{x},\omega))$ for all $(\mathbf{x},\omega)\in\mathcal{S}$ by definition of v_γ, this confirms that $v_\gamma((\mathbf{x},\omega))$ may always be bounded below by some finite negative value, which verifies the third assumption. The optimality equations (3.4.1) and existence of a γ-discount optimal stationary policy are then established using the arguments in [155] (see also [12], p. 251). □

Recall that, by Theorem 3.3.3, one may apply uniformisation and deduce that the expected total discounted reward $v_{\gamma,\theta}((\mathbf{x},\omega))$ under some stationary policy θ is equivalent to:

$$v_{\phi,\theta}((\mathbf{x},\omega)) := E_\theta\left[\sum_{n=0}^{\infty}\phi^n\mathcal{R}_\phi((\mathbf{x}_n,\omega_n),a_n)\Big|(\mathbf{x}_0,\omega_0)=(\mathbf{x},\omega)\right], \tag{3.4.3}$$

where the state evolution of the discrete-time process in (3.4.3) is determined by the transition probabilities $\mathcal{P}((\mathbf{x},\omega),a,(\mathbf{x}'\,\omega'))$ in (3.3.8)-(3.3.10), the discounted reward function $\mathcal{R}_\phi((\mathbf{x},\omega),a)$ is as defined in (3.3.12) and the discount factor ϕ is given by $\phi = (1+\gamma\Delta)^{-1}$, with $\Delta \in (0,(\lambda+\sum_{i=1}^N c_i\mu_i)^{-1}]$. For clarity, it will be assumed throughout the rest of this section that $v_{\phi,\theta}(\cdot)$ represents the expected total discounted reward function for the discrete-time MDP Φ obtained by applying uniformisation to the CTMDP Ψ, where the policy θ is arbitrary. An explicit formulation for Φ was given in the previous section. Also, $v_\phi(\cdot)$ will be used to denote the corresponding optimal value function, so that $v_\phi((\mathbf{x},\omega)) = \sup_\theta v_{\phi,\theta}((\mathbf{x},\omega))$ for all states $(\mathbf{x},\omega)\in\mathcal{S}$.

The fact that (by Theorem 3.4.2) a stationary γ-discount optimal policy exists for Ψ, together with the equivalence of the expected total discounted reward functions $v_{\gamma,\theta}(\cdot)$ and $v_{\phi,\theta}(\cdot)$ in the case of *stationary* policies θ (established by Theorem 3.3.3), justifies the approach of evaluating $v_\gamma((\mathbf{x},\omega))$ for any given discount rate $\gamma\in(0,\infty)$ and state $(\mathbf{x},\omega)\in\mathcal{S}$ by restricting attention to stationary policies in the uniformised process Φ. Some essential results from the literature concerning discounted rewards in discrete-time MDPs are summarised below.

*** Theorem 3.4.3.** *The optimal value function $v_\phi((\mathbf{x},\omega)) = \sup_\theta v_{\phi,\theta}((\mathbf{x},\omega))$ for the discrete-time MDP Φ satisfies the following ϕ-discount optimality equations:*

$$v_\phi((\mathbf{x},\omega)) = \max_{a\in\mathcal{A}_{(\mathbf{x},\omega)}}\left\{\mathcal{R}_\phi((\mathbf{x},\omega),a) + \phi\sum_{(\mathbf{x}',\omega')\in\mathcal{S}}\mathcal{P}((\mathbf{x},\omega),a,(\mathbf{x}',\omega'))\,v_\phi((\mathbf{x}',\omega'))\right\}, \tag{3.4.4}$$

where $\mathcal{R}_\phi(\cdot)$ and $\mathcal{P}(\cdot)$ are as given in (3.3.12) and (3.3.8)-(3.3.10) respectively. Furthermore, let θ^*_ϕ be any stationary policy such that for all $(\mathbf{x},\omega)\in\mathcal{S}$, the action $\theta^*_\phi((\mathbf{x},\omega))$ chosen by θ^*_ϕ attains the maximum in (3.4.4). Then the stationary policy θ^*_ϕ is ϕ-discount optimal.

Proof. This result actually follows immediately from previous results. Let $\hat{\Psi}$ be the CTMDP referred to in the statement of Theorem 3.3.3, with uniform sojourn time parameter C for all $((\mathbf{x},\omega),a)\in\mathcal{S}\times\mathcal{A}_{(\mathbf{x},\omega)}$ (where $C\geq\lambda+\sum_{i=1}^N c_i\mu_i$) and transition probabilities and rewards defined as in (3.3.12) and (3.3.8)-(3.3.10) respectively. Theorem 3.3.3 implies that, for any stationary policy

θ, the expected total discounted reward function for $\hat{\Psi}$ is equivalent to both $v_{\gamma,\theta}(\cdot)$ and $v_{\phi,\theta}(\cdot)$, so it will be convenient to denote this also by $v_{\phi,\theta}(\cdot)$. The result of Theorem 3.4.2 in fact applies to any CTMDP which satisfies the assumptions stated at the beginning of the proof (refer to [157], p. 251). Since it can easily be verified that the process $\hat{\Psi}$ satisfies these assumptions, the theorem implies that the optimal value function $v_\phi(\cdot)$ satisfies the optimality equations:

$$v_\phi((\mathbf{x},\omega)) = \max_{a\in A_{(\mathbf{x},\omega)}} \left\{ \mathcal{R}_\phi((\mathbf{x},\omega),a) + \left(\frac{C}{\gamma+C}\right) \sum_{(\mathbf{x}',\omega')\in\mathcal{S}} \mathcal{P}((\mathbf{x},\omega),a,(\mathbf{x}',\omega'))\, v_\phi((\mathbf{x}',\omega')) \right\},$$

where the factor $C/(\gamma+C)$ is obtained by analogy to (3.4.2) and the uniformity of the sojourn time parameters in the process $\hat{\Psi}$. By Theorem 3.3.3, the discount factor ϕ used for the discrete-time MDP Φ is also given by $\phi = C/(\gamma+C) = (1+\gamma\Delta)^{-1}$, so the result follows. □

The next result is sometimes referred to as the "method of successive approximations", and may be used to prove properties of $v_\phi(\cdot)$ using an inductive procedure.

*** Theorem 3.4.4.** *Define the function $v_\phi^{(0)} : \mathcal{S} \to \mathbb{R}$ by $v_\phi^{(0)}((\mathbf{x},\omega)) = 0$ for all $(\mathbf{x},\omega) \in \mathcal{S}$. Let the functions $v_\phi^{(n+1)} : \mathcal{S} \to \mathbb{R}$ be defined for integers $n \geq 0$ as follows:*

$$v_\phi^{(n+1)}((\mathbf{x},\omega)) = \max_{a\in\mathcal{A}_{(\mathbf{x},\omega)}} \left\{ \mathcal{R}_\phi((\mathbf{x},\omega),a) + \phi \sum_{(\mathbf{x}',\omega')\in\mathcal{S}} \mathcal{P}((\mathbf{x},\omega),a,(\mathbf{x}',\omega'))\, v_\phi^{(n)}((\mathbf{x}',\omega')) \right\}. \tag{3.4.5}$$

Then, for each $((\mathbf{x},\omega)) \in \mathcal{S}$, $v_\phi^{(n)}((\mathbf{x},\omega))$ converges uniformly to $v_\phi((\mathbf{x},\omega))$ as $n \to \infty$.

Proof. A proof is given by Ross [145] (p. 60). However, the proof depends on some technical details which should be discussed. These details can be found in Appendix A.3, page 408.

As discussed previously, the main objective in this section is to prove the existence of a stationary *average reward optimal* policy for the CTMDP Ψ. However, this can be done by establishing certain properties of the optimal value function $v_\gamma(\cdot)$ (or equivalently, the function $v_\phi(\cdot)$ associated with the uniformised process Φ) in a *discounted* reward problem. The approach taken in the remainder of this section will be to verify that $v_\gamma(\cdot)$ satisfies the conditions proposed by Sennott [157] (p. 250) which imply the existence of a stationary average reward optimal policy. First, it will be useful to establish a property of the discounted reward function $\mathcal{R}_\phi(\cdot)$ defined in (3.3.12).

Lemma 3.4.5. *Let $\alpha^* = \max(\alpha_1, \alpha_2, ..., \alpha_N)$. Then for all states $(\mathbf{x}, \omega) \in \mathcal{S}$, actions $a \in \mathcal{A}_{(\mathbf{x},\omega)}$, facilities $i \in \{1, 2, ..., N\}$ and discount factors $\phi \in (0, 1)$:*

$$\mathcal{R}_\phi((\mathbf{x}^{i+}, \omega), a) - \mathcal{R}_\phi((\mathbf{x}, \omega), a) \leq (\alpha^* \mu_i - \beta_i)\, \Delta. \tag{3.4.6}$$

Proof. Note that the random event ω is the same for the states $(\mathbf{x}, \omega)$ and $(\mathbf{x}^{i+}, \omega)$. Hence, due to the formulation of the action spaces in (3.3.7), any action $a \in \mathcal{A}_{(\mathbf{x},\omega)}$ also belongs to $\mathcal{A}_{(\mathbf{x}^{i+},\omega)}$. If $\omega = \Lambda$ then, for arbitrary $a \in \mathcal{A}_{(\mathbf{x},\omega)}$ and $i \in \{1, 2, ..., N\}$, one simply has:

$$\mathcal{R}_\phi((\mathbf{x}^{i+}, \Lambda), a) - \mathcal{R}_\phi((\mathbf{x}, \Lambda), a) = -\phi \Delta \beta_i, \tag{3.4.7}$$

which is negative for all $\phi \in (0, 1)$. On the other hand, $\alpha^* \mu_i - \beta_i$ is strictly positive due to the assumption in Section 3.1 that $\alpha_i - \beta_i/\mu_i > 0$ for all $i \in \{1, 2, ..., N\}$, so (3.4.6) holds. If $\omega = M_j$ for some $j \in \{1, 2, ..., N\}$, then for $a \in \mathcal{A}_{(\mathbf{x},\omega)} = \{0\}$ and $i \in \{1, 2, ..., N\}$:

$$\mathcal{R}_\phi((\mathbf{x}^{i+}, M_j), a) - \mathcal{R}_\phi((\mathbf{x}, M_j), a) = \begin{cases} \phi \Delta \left(\alpha_j \mu_i - \beta_i\right), & \text{if } x_i < c_i, \\ -\phi \Delta \beta_i, & \text{if } x_i \geq c_i, \end{cases} \tag{3.4.8}$$

and the upper bound in (3.4.6) then follows since $\phi \in (0, 1)$ and $\alpha_j \leq \alpha^*$. $\square$

The next result establishes some properties of the optimal value function $v_\gamma(\cdot)$ which appear at first sight to be somewhat remarkable, but can be reasoned using intuition.

Lemma 3.4.6. *Let θ^*_γ be a stationary γ-discount optimal policy. Consider a state $(\mathbf{x}, \omega) \in \mathcal{S}$ with $\omega = \Lambda$, and let $a^* \in \{0, 1, ..., N\}$ denote the action chosen by θ^*_γ at $(\mathbf{x}, \Lambda)$. Furthermore, let the notational convention be assumed whereby $\mathbf{x}^{0+} = \mathbf{x}$. Then, for $i \in \{1, 2, ..., N\}$:*

$$v_\gamma((\mathbf{x}^{a^*+}, M_i)) - v_\gamma((\mathbf{x}, \Lambda)) = \alpha_i. \tag{3.4.9}$$

In addition, for any two distinct facilities $i, j \in \{1, 2, ..., N\}$ and any vector $\mathbf{x} \in \mathbb{N}_0^N$:

$$v_\gamma((\mathbf{x}, M_i)) - v_\gamma((\mathbf{x}, M_j)) = \alpha_i - \alpha_j. \tag{3.4.10}$$

Proof. The results can be proved directly using the continuous-time optimality equations (3.4.1). However, for the purposes of later results, it will be useful to begin by considering the discrete-time

MDP Φ obtained from Ψ via uniformisation, and prove using the method of successive approximations (Theorem 3.4.4) that the following properties hold at all stages $n \in \mathbb{N}$:

$$\begin{aligned} &v_\phi^{(n)}((\mathbf{x}^{a_n^*+}, M_i)) - v_\phi^{(n)}((\mathbf{x}, \Lambda)) \leq \alpha_i, \\ &v_\phi^{(n)}((\mathbf{x}, M_i)) - v_\phi^{(n)}((\mathbf{x}, M_j)) \leq \max(\alpha_i - \alpha_j,\ 0), \end{aligned} \tag{3.4.11}$$

where $\phi = 1/(1+\gamma\Delta)$ is the discount factor for Φ, and $a_n^* \in \{0, 1, ..., N\}$ is an action attaining the maximum on the right-hand side of (3.4.5) for the state $(\mathbf{x}, \Lambda)$ with n replaced by $n-1$; that is, a_n^* is an optimal action at state $(\mathbf{x}, \Lambda)$ in a finite-stage problem with n stages. These properties may be proved by induction on n; details can be found in Appendix A.3 (p. 409). □

The next result establishes a uniform upper bound for the first-order differences $v_\gamma((\mathbf{x}^{i+}, \omega)) - v_\gamma((\mathbf{x}, \omega))$, applicable for all $(\mathbf{x}, \omega) \in \mathcal{S}$, $i \in \{1, 2, ..., N\}$ and $\gamma \in (0, \infty)$.

Lemma 3.4.7. *For all $(\mathbf{x}, \omega) \in S$, $i \in \{1, 2, ..., N\}$ and $\gamma \in (0, \infty)$:*

$$v_\gamma((\mathbf{x}^{i+}, \omega)) - v_\gamma((\mathbf{x}, \omega)) \leq \alpha^* - \frac{\beta_i}{\mu_i}. \tag{3.4.12}$$

Proof. Again, this result may be proved on induction using the finite-stage functions $v_\phi^{(n)}$ associated with the uniformised process Φ. For details, see Appendix A.3 (p. 412). □

The next two results verify that the optimal value function v_γ satisfies certain conditions which are known from the literature (in particular, [157]) to imply the existence of an *average reward optimal* stationary policy. First it will be shown, using an argument similar to that of Proposition 3 in [157], that the differences $v_\gamma((\mathbf{x}, \omega)) - v_\gamma((\mathbf{0}, \Lambda))$ are uniformly bounded above.

Lemma 3.4.8. *Let $\mathbf{0} = (0, 0, ..., 0)$ denote the zero vector. There exists $\gamma_0 \in (0, \infty)$ and a constant $K \in \mathbb{R}$ such that, for all states $(\mathbf{x}, \omega) \in \mathcal{S}$ and discount rates $\gamma \in (0, \gamma_0)$:*

$$v_\gamma((\mathbf{x}, \omega)) - v_\gamma((\mathbf{0}, \Lambda)) \leq K.$$

Proof. This proof will proceed in stages. An outline of the proof is given below.

1. First, the CTMDP Ψ formulated in Section 3.2 will be transformed into an equivalent CTMDP $\Psi^\dagger$ which has the same class of γ-discount optimal policies. The transformation will be made by re-defining the single-sojourn reward function for the process.

2. By considering the transformed process $\Psi^{\dagger}$, it will be shown that there exists a finite subset U of the infinite state space $\mathcal{S}$ such that the expected total discounted reward $v_{\gamma}((\mathbf{x},\omega))$ for any state $(\mathbf{x},\omega)$ outside U is bounded above by $\max_{(\mathbf{x},\omega)\in U} v_{\gamma}((\mathbf{x},\omega))$.

3. Finally, an explicit upper bound will be found for $\max_{(\mathbf{x},\omega)\in U} v_{\gamma}((\mathbf{x},\omega)) - v_{\gamma}((\mathbf{0},\Lambda))$.

Let $\gamma \in (0,\infty)$ be fixed. As in the previous two lemmas, let $\alpha^{*} = \max(\alpha_1, \alpha_2, ..., \alpha_N)$. Next, define a function $\zeta_{\gamma}((\mathbf{x},\omega),a)$ for $(\mathbf{x},\omega) \in \mathcal{S}$ and $a \in \mathcal{A}_{(\mathbf{x},\omega)}$ as follows:

$$\zeta_{\gamma}((\mathbf{x},\omega),a) := \alpha^{*}\left(\frac{\gamma + C}{\gamma + \tau((\mathbf{x},\omega),a)}\right), \tag{3.4.13}$$

where $\tau((\mathbf{x},\omega),a)$ is defined as in (3.2.9) and, as in Theorem 3.3.3, C is any constant satisfying $C \geq \lambda + \sum_{j=1}^{N} c_j\mu_j$. Consider a transformation of the CTMDP Ψ formulated in Section 3.2 whereby an amount $\zeta_{\gamma}((\mathbf{x},\omega),a)$ is subtracted from the reward $\xi_{\gamma}((\mathbf{x},\omega),a)$ earned every time the process visits state $(\mathbf{x},\omega) \in \mathcal{S}$ at some decision epoch and the action $a \in \mathcal{A}_{(\mathbf{x},\omega)}$ is chosen. Note that $\zeta_{\gamma}((\mathbf{x},\omega),a)$ is effectively a 'lump sum' deduction, rather than a penalty charged continuously while the next sojourn is in progress. Let $\Psi^{\dagger}$ denote the CTMDP formulated as in Section 3.2 (with a real-time reward formulation used, similar to (3.2.18)) but taking into account the new reward structure. Referring to (3.2.17), the expected total discounted reward under an arbitrary policy θ, given some initial state $(\mathbf{x},\omega) \in \mathcal{S}$, may be written for the CTMDP $\Psi^{\dagger}$ as follows:

$$\begin{aligned} v^{\dagger}_{\gamma,\theta}((\mathbf{x},\omega)) &= E_{\theta}\left[\sum_{n=0}^{\infty} e^{-\gamma T_n}\big(\xi_{\gamma}((\mathbf{x}_n,\omega_n),a_n) - \zeta_{\gamma}((\mathbf{x}_n,\omega_n),a_n)\big)\,\middle|\,(\mathbf{x}_0,\omega_0) = (\mathbf{x},\omega)\right] \\ &= E_{\theta}\left[\sum_{n=0}^{\infty} e^{-\gamma T_n}\left(J^{\dagger}_{\gamma}((\mathbf{x}_n,\omega_n),a_n) - \sum_{j=1}^{N}\beta_j(\mathbf{x}_n^{a_n+})_j\int_0^{\delta_{n+1}} e^{-\gamma t}dt\right)\,\middle|\,(\mathbf{x}_0,\omega_0) = (\mathbf{x},\omega)\right], \end{aligned} \tag{3.4.14}$$

where T_n is the time of the n^{th} decision epoch, $(\mathbf{x}_n,\omega_n) \in \mathcal{S}$ is the corresponding state, a_n is the action chosen under the policy θ, $\delta_{n+1} = T_{n+1} - T_n$ is the length of the $(n+1)^{th}$ sojourn of the process, $(\mathbf{x}_n^{a_n+})_j$ denotes the j^{th} component of the vector $\mathbf{x}_n^{a_n+}$, and the non-positive function $J^{\dagger}_{\gamma}((\mathbf{x},\omega),a)$ is defined for state-action pairs $((\mathbf{x},\omega),a) \in \mathcal{S}\times\mathcal{A}_{(\mathbf{x},\omega)}$ as follows:

$$J^{\dagger}_{\gamma}((\mathbf{x},\omega),a) = \begin{cases} \alpha_i - \zeta_{\gamma}((\mathbf{x},\omega),a), & \text{if } \omega = M_i \text{ for some } i \in \{1,2,...,N\}, \\ -\zeta_{\gamma}((\mathbf{x},\omega),a), & \text{otherwise.} \end{cases}$$

By Theorem 3.3.3, $v^{\dagger}_{\gamma,\theta}((\mathbf{x},\omega))$ may be obtained equivalently by considering a CTMDP with uniformly-distributed sojourn times (these being exponentially distributed with parameter C), but in order for this to be a valid procedure the single-sojourn rewards must be multiplied by a factor $(\gamma+\tau((\mathbf{x},\omega),a))/(\gamma+C)$. By applying this procedure, one obtains the following single-sojourn reward function $\mathcal{R}^{\dagger}_{\gamma}((\mathbf{x},\omega),a)$ for use in the uniformised version of $\Psi^{\dagger}$:

$$\begin{aligned}\mathcal{R}^{\dagger}_{\gamma}((\mathbf{x},\omega),a) &= \left(\frac{\gamma+\tau((\mathbf{x},\omega),a)}{\gamma+C}\right)\big(\xi_{\gamma}((\mathbf{x},\omega),a)-\zeta_{\gamma}((\mathbf{x},\omega),a)\big)\\ &= \mathcal{R}_{\gamma}((\mathbf{x},\omega),a)-\alpha^{*},\end{aligned}$$

where $\mathcal{R}_{\gamma}((\mathbf{x},\omega),a)$ is the reward function obtained by applying uniformisation to the single-sojourn rewards $\xi_{\gamma}((\mathbf{x},\omega),a)$ for the *original* process Ψ. Hence, due to the equivalence of the uniformised CTMDP proved by Theorem 3.3.3, an alternative expression for $v^{\dagger}_{\gamma,\theta}((\mathbf{x},\omega))$ is:

$$\begin{aligned}v^{\dagger}_{\gamma,\theta}((\mathbf{x},\omega)) &= E_{\theta}\left[\sum_{n=0}^{\infty}e^{-\gamma T_n}\big(\mathcal{R}_{\gamma}((\mathbf{x}_n,\omega_n),a_n)-\alpha^{*}\big)\,\middle|\,(\mathbf{x}_0,\omega_0)=(\mathbf{x},\omega)\right]\\ &= v_{\gamma,\theta}((\mathbf{x},\omega))-\alpha^{*}E_{\theta}\left[\sum_{n=0}^{\infty}e^{-\gamma T_n}\,\middle|\,(\mathbf{x}_0,\omega_0)=(\mathbf{x},\omega)\right],\end{aligned}\tag{3.4.15}$$

where the expectation is now taken with respect to the random (uniformly distributed) transition times and jump probabilities $\mathcal{P}((\mathbf{x},\omega),a,(\mathbf{x}',\omega'))$ for the *uniformised* process operating under θ. Since, in the uniformised process, the time of the n^{th} decision epoch (for $n\geq 1$) is gamma-distributed with parameters n and C, (3.4.15) can be simplified further:

$$\begin{aligned}v^{\dagger}_{\gamma,\theta}((\mathbf{x},\omega)) &= v_{\gamma,\theta}((\mathbf{x},\omega))-\alpha^{*}\left(1+\sum_{n=1}^{\infty}\int_{0}^{\infty}\frac{C^n}{(n-1)!}\,t^{n-1}e^{-(\gamma+C)t}dt\right)\\ &= v_{\gamma,\theta}((\mathbf{x},\omega))-\alpha^{*}\left(1+\sum_{n=1}^{\infty}\left(\frac{C}{\gamma+C}\right)^{n}\right)\\ &= v_{\gamma,\theta}((\mathbf{x},\omega))-\alpha^{*}\left(\frac{\gamma+C}{\gamma}\right).\end{aligned}\tag{3.4.16}$$

It follows from (3.4.16) that, for any two states $(\mathbf{x},\omega),(\mathbf{x}',\omega')\in\mathcal{S}$:

$$v^{\dagger}_{\gamma,\theta}((\mathbf{x},\omega))-v^{\dagger}_{\gamma,\theta}((\mathbf{x}',\omega'))=v_{\gamma,\theta}((\mathbf{x},\omega))-v_{\gamma,\theta}((\mathbf{x}',\omega')).\tag{3.4.17}$$

In particular, since this principle applies to a γ-discount optimal policy:

$$v^{\dagger}_{\gamma}((\mathbf{x},\omega))-v^{\dagger}_{\gamma}((\mathbf{x}',\omega'))=v_{\gamma}((\mathbf{x},\omega))-v_{\gamma}((\mathbf{x}',\omega')),\tag{3.4.18}$$

where $v_{\gamma}^{\dagger}((\mathbf{x},\omega)) = \sup_{\theta} v_{\gamma,\theta}^{\dagger}((\mathbf{x},\omega))$ for $(\mathbf{x},\omega) \in \mathcal{S}$. Essentially, the purpose of making the transformation from Ψ to $\Psi^{\dagger}$ is that in the transformed process $\Psi^{\dagger}$, all of the single-sojourn rewards (given by $\xi_{\gamma}((\mathbf{x},\omega),a) - \zeta_{\gamma}((\mathbf{x},\omega),a)$) are non-positive, which can easily be seen from the fact that $\xi_{\gamma}((\mathbf{x},\omega),a)$ is uniformly bounded above by α^{*} and $\zeta_{\gamma}((\mathbf{x},\omega),a) \geq \alpha^{*}$ for all $(\mathbf{x},\omega) \in \mathcal{S}$ and $a \in \mathcal{A}_{(\mathbf{x},\omega)}$. This non-positivity of the rewards will be required later. The results given in [157] (which relate to a cost minimisation problem) assume that all costs are non-negative, and as the author observes, this assumption can be made without loss of generality since it is always possible to make a transformation which eradicates negative costs without affecting the optimality criterion of interest. Analogously, (3.4.16) implies that any policy θ which is γ-discount optimal in the original process Ψ must remain likewise optimal in the transformed process $\Psi^{\dagger}$.

In order to proceed, some general theoretical results for CTMDPs are needed. Let $g_{\theta}^{\dagger}((\mathbf{x},\omega))$ denote the expected long-run average (undiscounted) reward for the transformed process $\Psi^{\dagger}$ under policy θ. For clarification, when average rewards are considered the single-sojourn rewards for $\Psi^{\dagger}$ are given by $\xi((\mathbf{x},\omega),a) - \zeta((\mathbf{x},\omega),a)$, where $\xi((\mathbf{x},\omega),a)$ is as defined in (3.2.13) and:

$$\zeta((\mathbf{x},\omega),a) = \lim_{\gamma \downarrow 0} \zeta_{\gamma}((\mathbf{x},\omega),a) = \frac{\alpha^{*}C}{\tau((\mathbf{x},\omega),a)}.$$

By applying uniformisation, one can show that $g_{\theta}^{\dagger}((\mathbf{x},\omega))$ is related to $g_{\theta}((\mathbf{x},\omega))$ (the expected long-run average reward for the original process Ψ) as follows:

$$g_{\theta}^{\dagger}((\mathbf{x},\omega)) = g_{\theta}((\mathbf{x},\omega)) - \alpha^{*}C, \tag{3.4.19}$$

and thus any policy which is average reward optimal for Ψ will also be similarly optimal for $\Psi^{\dagger}$. Naturally, both $g_{\theta}^{\dagger}((\mathbf{x},\omega))$ and $v_{\gamma,\theta}^{\dagger}((\mathbf{x},\omega))$ are non-positive since all rewards for $\Psi^{\dagger}$ are non-positive. The following result is found in [157] (p. 251) and the proof relies upon certain results from analysis which can be found in [199]. It is the case that for any policy θ and $(\mathbf{x},\omega) \in \mathcal{S}$:

$$\liminf_{\gamma \downarrow 0} \gamma v_{\gamma,\theta}^{\dagger}((\mathbf{x},\omega)) \geq g_{\theta}^{\dagger}((\mathbf{x},\omega)). \tag{3.4.20}$$

Furthermore, let θ be any stationary policy which induces an irreducible, ergodic Markov chain on some finite set of states $\mathcal{S}_{\theta} \subset \mathcal{S}$ with steady-state probability distribution $\{\pi_{\theta}((\mathbf{x},\omega))\}_{(\mathbf{x},\omega)\in\mathcal{S}_{\theta}}$. Then it can be shown that for all states $(\mathbf{x},\omega) \in \mathcal{S}$, the following holds:

$$g_{\theta}^{\dagger}((\mathbf{x},\omega)) = g_{\theta}^{\dagger}, \tag{3.4.21}$$

where $g_\theta^\dagger$ is a constant. Verification of this fact may be carried out by first noting that under the policy θ, the following two (somewhat trivial) conditions are satisfied:

$$\sum_{(\mathbf{x},\omega)\in\mathcal{S}_\theta} \pi_\theta((\mathbf{x},\omega))\big(\xi((\mathbf{x},\omega),\theta) - \zeta((\mathbf{x},\omega),\theta)\big) > -\infty, \tag{3.4.22}$$

$$\sum_{(\mathbf{x},\omega)\in\mathcal{S}_\theta} \frac{\pi_\theta((\mathbf{x},\omega))}{\tau((\mathbf{x},\omega),\theta)} > -\infty, \tag{3.4.23}$$

where the notation $((\mathbf{x},\omega),\theta)$ denotes a state-action pair in which the action chosen is prescribed by the policy θ. Indeed, (3.4.22) is implied by the finiteness of $\mathcal{S}_\theta$, while (3.4.23) follows from the fact that $\tau((\mathbf{x},\omega),a) \leq C$ for all $((\mathbf{x},\omega),a) \in \mathcal{S}\times\mathcal{A}_{(\mathbf{x},\omega)}$. Using these properties, it can then be shown using the arguments given in [83] (p. 326) that the continuous-time Markov chain induced by θ on $\mathcal{S}_\theta$ is ergodic, which in turn implies (3.4.21) (see [157], p. 254). From (3.4.16) it follows that $\gamma v_{\gamma,\theta}^\dagger((\mathbf{x},\omega)) = \gamma v_{\gamma,\theta}((\mathbf{x},\omega)) - \alpha^*(\gamma + C)$. Since $\alpha^*(\gamma+C) \to \alpha^* C$ as $\gamma \downarrow 0$, it can then be shown easily using (3.4.19) and (3.4.20) that, given any initial state $(\mathbf{x},\omega)\in\mathcal{S}$, policy θ and positive number ϵ, there exists a value $\gamma_1 \in (0,\infty)$ such that for all $\gamma\in(0,\gamma_1)$:

$$\gamma v_{\gamma,\theta}^\dagger((\mathbf{x},\omega)) > g_\theta^\dagger - \epsilon, \tag{3.4.24}$$

The next part of the proof proceeds similarly to that of Proposition 3 in [157]. Let θ be a fixed stationary policy which induces an ergodic, irreducible Markov chain defined on a finite set $\mathcal{S}_\theta \subset \mathcal{S}$, so that (3.4.21) applies for all $(\mathbf{x},\omega)\in\mathcal{S}$. Let the set U_θ be defined as follows:

$$U_\theta := \left\{ (\mathbf{x},\omega)\in\mathcal{S} : -\sum_{j=1}^{N} \beta_j x_j \geq g_\theta^\dagger - \epsilon \right\}, \tag{3.4.25}$$

where $\epsilon > 0$ is arbitrary. That is, U_θ is the set of states in $\mathcal{S}$ for which the associated holding costs are bounded above by the non-negative quantity $\epsilon - g_\theta^\dagger$. Clearly, for any fixed $\epsilon > 0$, U_θ is a finite set. Hence, due to (3.4.24), there exists $\gamma_0 \in (0,\infty)$ such that $\gamma v_{\gamma,\theta}^\dagger((\mathbf{x},\omega)) > g_\theta^\dagger - \epsilon$ for all $(\mathbf{x},\omega)\in U_\theta$ and $\gamma\in(0,\gamma_0)$. The remainder of the proof will consider an arbitrary, fixed discount rate $\gamma\in(0,\gamma_0)$. Since U_θ is finite, there exists a state $(\mathbf{x},\omega)^* \in U_\theta$ such that:

$$v_\gamma^\dagger((\mathbf{x},\omega)^*) \geq v_\gamma^\dagger((\mathbf{x},\omega)) \quad \forall\,(\mathbf{x},\omega)\in U_\theta. \tag{3.4.26}$$

(Note that the optimal value function $v_\gamma^\dagger$ is used in (3.4.26), as opposed to $v_{\gamma,\theta}^\dagger$.) Suppose the transformed process $\Psi^\dagger$ is initialised in some initial state $(\mathbf{x},\omega)\notin U_\theta$ and follows a γ-discount

optimal policy θ^*_γ. The claim of this proof is that, given $(\mathbf{x},\omega) \notin U_\theta$, the quantity $v^\dagger_\gamma((\mathbf{x},\omega))$ cannot possibly exceed $v^\dagger_\gamma((\mathbf{x},\omega)^*)$, and hence the state $(\mathbf{x},\omega)^*$ referred to in (3.4.26) must be a maximiser of $v^\dagger_\gamma(\cdot)$ over *all* states in $\mathcal{S}$. Let the random variable Z be defined as follows:

$$Z := \min\left\{n \in \mathbb{N} : (\mathbf{x}_n,\omega_n) \in U_\theta \,\middle|\, (\mathbf{x}_0,\omega_0) = (\mathbf{x},\omega)\right\}.$$

Hence, Z is the first decision epoch at which the system state is an element of U_θ. Then, in view of (3.4.14), the (optimal) expected total discounted reward $v^\dagger_\gamma((\mathbf{x},\omega))$ satisfies:

$$\begin{aligned}
v^\dagger_\gamma((\mathbf{x},\omega)) &= E_{\theta^*_\gamma}\left[\sum_{n=0}^{\infty} e^{-\gamma T_n}\left(J^\dagger_\gamma((\mathbf{x}_n,\omega_n),a_n) - \sum_{j=1}^{N}\beta_j(\mathbf{x}_n^{a_n+})_j\int_0^{\delta_{n+1}} e^{-\gamma t}dt\right)\,\middle|\,(\mathbf{x}_0,\omega_0)=(\mathbf{x},\omega)\right] \\
&\leq E_{\theta^*_\gamma}\left[\left(\sum_{n=0}^{Z-1} e^{-\gamma T_n}(g^\dagger_\theta-\epsilon)\int_0^{\delta_{n+1}} e^{-\gamma t}dt + e^{-\gamma T_Z}v^\dagger_\gamma((\mathbf{x},\omega)^*)\right)\,\middle|\,(\mathbf{x}_0,\omega_0)=(\mathbf{x},\omega)\right] \\
&= E_{\theta^*_\gamma}\left[\left((g^\dagger_\theta-\epsilon)\int_0^{T_Z} e^{-\gamma t}dt + e^{-\gamma T_Z}v^\dagger_\gamma((\mathbf{x},\omega)^*)\right)\,\middle|\,(\mathbf{x}_0,\omega_0)=(\mathbf{x},\omega)\right] \\
&= E_{\theta^*_\gamma}\left[\left((g^\dagger_\theta-\epsilon)\left(\frac{1-e^{-\gamma T_Z}}{\gamma}\right) + e^{-\gamma T_Z}v^\dagger_\gamma((\mathbf{x},\omega)^*)\right)\,\middle|\,(\mathbf{x}_0,\omega_0)=(\mathbf{x},\omega)\right], \qquad (3.4.27)
\end{aligned}$$

where the inequality is due to the non-positivity of $J^\dagger_\gamma((\mathbf{x}_n,\omega_n),a_n)$, the definition of U_θ in (3.4.25) and (3.4.26). By the previous arguments, the following holds for all $(\mathbf{x},\omega) \in U_\theta$:

$$\gamma v^\dagger_\gamma((\mathbf{x},\omega)^*) \geq \gamma v^\dagger_\gamma((\mathbf{x},\omega)) \geq \gamma v^\dagger_{\gamma,\theta}((\mathbf{x},\omega)) \geq g^\dagger_\theta - \epsilon. \qquad (3.4.28)$$

Hence, (3.4.27) implies:

$$\begin{aligned}
v^\dagger_\gamma((\mathbf{x},\omega)) &\leq E_{\theta^*_\gamma}\left[\left(\gamma v^\dagger_\gamma((\mathbf{x},\omega)^*)\left(\frac{1-e^{-\gamma T_Z}}{\gamma}\right) + e^{-\gamma T_Z}v^\dagger_\gamma((\mathbf{x},\omega)^*)\right)\,\middle|\,(\mathbf{x}_0,\omega_0)=(\mathbf{x},\omega)\right] \\
&= v^\dagger_\gamma((\mathbf{x},\omega)^*). \qquad (3.4.29)
\end{aligned}$$

Note that in (3.4.27) and (3.4.29), the assumption is made that Z is finite (and hence T_Z is also finite); however, by considering the limiting behaviour as $T_Z \to \infty$, it is clear that the result also holds when Z is infinite. Thus, due to (3.4.18), one may conclude that $v_\gamma((\mathbf{x},\omega)) \leq v_\gamma((\mathbf{x},\omega)^*)$ for all $(\mathbf{x},\omega) \notin U_\theta$. The proof of the lemma will therefore be complete if it can be shown that there exists some constant K such that $v_\gamma((\mathbf{x},\omega)^*) - v_\gamma((\mathbf{0},\Lambda)) \leq K$ (recall that $\gamma \in (0,\gamma_0)$ is arbitrary). This may be done by taking an approach similar to that in [157], and arguing that

$v_\gamma^\dagger((\mathbf{0},\Lambda)) - v_\gamma^\dagger((\mathbf{x},\omega)^*)$ is bounded below by the expected total (non-positive) *undiscounted* reward accumulated during a sojourn from state $(\mathbf{0},\Lambda)$ to $(\mathbf{x},\omega)^*$ in the process $\Psi^\dagger$ operating under the discount-optimal policy θ_γ^*; however, in order to use this approach one must establish that this quantity is finite. In fact, it will be simpler to take a somewhat more direct approach here and make use of Lemma 3.4.7. For all $j \in \{1,2,...,N\}$, let $u_j \in \mathbb{N}_0$ be defined as follows:

$$u_j := \max\left\{n \in \mathbb{N}_0 : x_j = n \text{ for some } \mathbf{x} \in U_\theta\right\}.$$

Clearly, each integer u_j is finite due to the finiteness of U_θ. By using (3.4.18) and also making use of a telescoping sum, one can establish the following upper bound:

$$\begin{aligned}
&v_\gamma((\mathbf{x},\omega)^*) - v_\gamma((\mathbf{0},\omega)) \\
&= \sum_{i=1}^{N}\sum_{j=0}^{x_i^*-1}\left(v_\gamma\left(\left(\mathbf{0}+\sum_{k=1}^{i-1}x_k^*\mathbf{e}_k + (j+1)\mathbf{e}_i,\,\omega\right)\right) - v_\gamma\left(\left(\mathbf{0}+\sum_{k=1}^{i-1}x_k^*\mathbf{e}_k + j\mathbf{e}_i,\,\omega\right)\right)\right), \\
&\leq \sum_{i=1}^{N} x_i^*\left(\alpha^* - \frac{\beta_i}{\mu_i}\right) \\
&\leq \sum_{i=1}^{N} u_i\left(\alpha^* - \frac{\beta_i}{\mu_i}\right), \qquad (3.4.30)
\end{aligned}$$

where x_i^* denotes the i^{th} component of the vector $\mathbf{x}$ associated with the state $(\mathbf{x},\omega)^* \in U_\theta$, and (as previously) $\mathbf{e}_i$ denotes the vector with i^{th} component equal to one and all other components equal to zero. It should also be clarified that the event ω associated with the state $(\mathbf{0},\omega)$ in the first line of (3.4.30) corresponds to the event ω associated with $(\mathbf{x},\omega)^*$. The first inequality in (3.4.30) results from applying the result of Lemma 3.4.7 a total of $\sum_{i=1}^{N} x_i^*$ times, while the second inequality is due to the fact that $x_i^* \leq u_i$ for each facility $i \in \{1,2,...,N\}$. If the random event ω is equal to Λ then the claim of the lemma is proved, since one may then write for any $(\mathbf{x},\omega) \in \mathcal{S}$:

$$v_\gamma((\mathbf{x},\omega)) - v_\gamma((\mathbf{0},\Lambda)) \leq v_\gamma((\mathbf{x},\Lambda)^*) - v_\gamma((\mathbf{0},\Lambda)) \leq \sum_{i=1}^{N} u_i\left(\alpha^* - \frac{\beta_i}{\mu_i}\right). \qquad (3.4.31)$$

On the other hand, if the random event ω associated with $(\mathbf{x},\omega)^*$ is equal to M_i for some $i \in \{1,2,...,N\}$, then one may still apply (3.4.30) and write for any $(\mathbf{x},\omega) \in \mathcal{S}$:

$$v_\gamma((\mathbf{x},\omega)) - v_\gamma((\mathbf{0},\Lambda)) \leq v_\gamma((\mathbf{x},M_i)^*) - v_\gamma((\mathbf{0},\Lambda))$$

$$= v_\gamma((\mathbf{x}, M_i)^*) - v_\gamma((\mathbf{0}, M_i)) + v_\gamma((\mathbf{0}, M_i)) - v_\gamma((\mathbf{0}, \Lambda))$$
$$\leq \sum_{i=1}^{N} u_i \left(\alpha^* - \frac{\beta_i}{\mu_i}\right) + v_\gamma((\mathbf{0}, M_i)) - v_\gamma((\mathbf{0}, \Lambda)). \tag{3.4.32}$$

In this case, it remains to establish a γ-independent upper bound for the extra term $v_\gamma((\mathbf{0}, M_i)) - v_\gamma((\mathbf{0}, \Lambda))$. This is not a difficult task, since the only possible transition from state $(\mathbf{0}, M_i)$ is to $(\mathbf{0}, \Lambda)$ and hence the discount optimality equations in (3.4.1) imply:

$$v_\gamma((\mathbf{0}, M_i)) = \alpha_i + \frac{\lambda}{\gamma + \lambda} v_\gamma((\mathbf{0}, \Lambda))$$
$$\leq \alpha^* + v_\gamma((\mathbf{0}, \Lambda)). \tag{3.4.33}$$

Note that (3.4.33) assumes that $v_\gamma((\mathbf{0}, \Lambda))$ is non-negative, which follows from the fact that in the original (non-transformed) process Ψ, $v_\gamma((\mathbf{0}, \Lambda))$ is bounded below by the expected total discounted reward $v_{\gamma,\theta_0}((\mathbf{0}, \Lambda))$ for the policy θ_0 which always chooses to balk, and clearly this quantity is equal to zero. From (3.4.33) it follows that $v_\gamma((\mathbf{0}, M_i)) - v_\gamma((\mathbf{0}, \Lambda)) \leq \alpha^*$ for all $i \in \{1, 2, ..., N\}$. Hence, in view of (3.4.31) and (3.4.32), one can write for any state $(\mathbf{x}, \omega) \in \mathcal{S}$:

$$v_\gamma((\mathbf{x}, \omega)) - v_\gamma((\mathbf{0}, \Lambda)) \leq \alpha^* + \sum_{i=1}^{N} u_i \left(\alpha^* - \frac{\beta_i}{\mu_i}\right). \tag{3.4.34}$$

The quantity on the right-hand side of (3.4.34) may be regarded as a constant whose value depends only on the system parameters and the choice of the policy θ which determines the finite set U_θ. Any policy which induces an ergodic, irreducible CTMC on some finite subset of $\mathcal{S}$ will yield, via (3.4.34), a valid upper bound for $v_\gamma((\mathbf{x}, \omega)) - v_\gamma((\mathbf{0}, \Lambda))$ which applies for all $(\mathbf{x}, \omega) \in \mathcal{S}$. The choice of θ (and $\epsilon > 0$) effectively determines the tightness of the upper bound obtained. For example, if a 'poor' policy θ is chosen under which the expected long-run average reward $g_\theta^\dagger$ for the transformed process $\Psi^\dagger$ is very small, then the set U_θ defined in (3.4.18) will be relatively large, and hence the upper bound in (3.4.34) will also be large; on the other hand, if the policy θ is close to (or attains) average reward optimality, then the upper bound will be smaller. Since the same choice of θ and ϵ will work for any discount rate $\gamma \in (0, \gamma_0)$, this completes the proof of the lemma. □

Before stating the main result of this section, one further property of v_γ must also be proved; specifically, it must be shown that, given any $(\mathbf{x}, \omega) \in \mathcal{S}$, a state-dependent lower bound can be found for $v_\gamma((\mathbf{x}, \omega)) - v_\gamma((\mathbf{0}, \Lambda))$ which holds for any discount rate $\gamma \in (0, \infty)$.

Lemma 3.4.9. *For every state $(\mathbf{x},\omega)\in\mathcal{S}$, there exists a non-negative value $f((\mathbf{x},\omega))<\infty$ such that for all discount rates $\gamma\in(0,\infty)$, the following holds:*

$$v_\gamma((\mathbf{x},\omega))-v_\gamma((\mathbf{0},\Lambda))\geq -f((\mathbf{x},\omega)). \tag{3.4.35}$$

Proof. Let $\gamma\in(0,\infty)$ be fixed. Consider the same transformation as described in the proof of Lemma 3.4.8, whereby an amount $\zeta_\gamma((\mathbf{x},\omega),a)=\alpha^*(\gamma+C)/(\gamma+\tau((\mathbf{x},\omega),a))$ is deducted from the reward $\xi_\gamma((\mathbf{x},\omega),a)$ earned at each decision epoch. Given that $\alpha^*=\max(\alpha_1,\alpha_2,...,\alpha_N)$ and $C\geq\lambda+\sum_{i=1}^N c_i\mu_i$, this ensures that each single-sojourn reward is non-positive and hence the expected total discounted reward $v^\dagger_{\gamma,\theta}((\mathbf{x},\omega))$ for the transformed process $\Psi^\dagger$ is non-positive under any policy θ and initial state $(\mathbf{x},\omega)\in\mathcal{S}$. It will be convenient to begin by proving the result for states $(\mathbf{0},M_i)$, where $i\in\{1,2,...,N\}$ is arbitrary. Using the non-positivity of $v^\dagger_\gamma((\mathbf{0},\Lambda))$, the discount optimality equations (3.4.1) for the transformed process $\Psi^\dagger$ imply:

$$\begin{aligned}
v^\dagger_\gamma((\mathbf{0},M_i)) &= \xi_\gamma((\mathbf{0},M_i),0)-\zeta_\gamma((\mathbf{0},M_i),0)+\frac{\tau((\mathbf{0},M_i),0)}{\gamma+\tau((\mathbf{0},M_i),0)}\,v^\dagger_\gamma((\mathbf{0},\Lambda))\\
&= \alpha_i-\alpha^*\left(\frac{\gamma+C}{\gamma+\lambda}\right)+\frac{\lambda}{\gamma+\lambda}\,v^\dagger_\gamma((\mathbf{0},\Lambda))\\
&\geq \alpha_i-\frac{\alpha^*C}{\lambda}+v^\dagger_\gamma((\mathbf{0},\Lambda))\\
&= \xi((\mathbf{0},M_i),0)-\zeta((\mathbf{0},M_i),0)+v^\dagger_\gamma((\mathbf{0},\Lambda)),
\end{aligned} \tag{3.4.36}$$

where $\xi((\mathbf{x},\omega),a)$ is the undiscounted reward function defined in (3.2.13) and (as in Lemma 3.4.8) $\zeta((\mathbf{x},\omega),a)=\alpha^*C/\tau((\mathbf{x},\omega),a)$ for $(\mathbf{x},\omega)\in\mathcal{S}$. Hence, (3.4.36) implies:

$$v^\dagger_\gamma((\mathbf{0},M_i))-v^\dagger_\gamma((\mathbf{0},\Lambda))\geq\xi((\mathbf{0},M_i),0)-\zeta((\mathbf{0},M_i),0)=:-f((\mathbf{0},M_i)),$$

so a negative lower bound for $v^\dagger_\gamma((\mathbf{0},M_i))-v^\dagger_\gamma((\mathbf{0},\Lambda))$ has been found, which is independent of γ as required. Due to the property (3.4.17), the same lower bound also applies for $v_\gamma((\mathbf{0},M_i))-v_\gamma((\mathbf{0},\Lambda))$. The rest of this proof will use induction over the state space $\mathcal{S}$. Consider an arbitrary state $(\mathbf{x},\omega)\in\mathcal{S}$, and suppose (as an inductive assumption) that a non-negative value $f((\mathbf{y},\omega'))$ satisfying (3.4.35) has been found for all states $(\mathbf{y},\omega')\in\mathcal{S}$ such that the componentwise inequality $\mathbf{y}<\mathbf{x}$ holds (i.e. $y_i\leq x_i$ for all $i\in\{1,2,...,N\}$, with strict inequality in at least one component). The claim has been shown to be true when $\mathbf{x}=\mathbf{0}^{i+}$ for any $i\in\{1,2,...,N\}$, due to the fact that $\mathbf{0}$ is the only vector strictly smaller than $\mathbf{0}^{i+}$ in the partial order. It will then be sufficient to show

that there exists a γ-independent lower bound for $v_\gamma((\mathbf{x},\omega)) - v_\gamma((\mathbf{0},\Lambda))$. First, suppose one has $\omega = \Lambda$. The optimality equations (3.4.1) imply that for any action $a \in \{0,1,...,N\}$:

$$v_\gamma^\dagger((\mathbf{x},\Lambda)) \geq \xi_\gamma((\mathbf{x},\Lambda),a) - \zeta_\gamma((\mathbf{x},\Lambda),a) + L_\gamma((\mathbf{x},\Lambda),a) \sum_{(\mathbf{x}',\omega')\in\mathcal{S}} \sigma((\mathbf{x},\Lambda),a,(\mathbf{x}',\omega'))\, v_\gamma^\dagger((\mathbf{x}',\omega')). \tag{3.4.37}$$

Hence, by considering the action $a = 0$ (i.e. balking) in (3.4.37):

$$\begin{aligned} v_\gamma^\dagger((\mathbf{x},\Lambda)) &\geq \xi_\gamma((\mathbf{x},\Lambda),0) - \zeta_\gamma((\mathbf{x},\Lambda),0) \\ &\quad + \frac{\lambda v_\gamma^\dagger((\mathbf{x},\Lambda))}{\gamma + \tau((\mathbf{x},\Lambda),0)} + \frac{\sum_{j=1}^N \min(x_j,c_j)\mu_j v_\gamma^\dagger((\mathbf{x}^{j-},M_j))}{\gamma + \tau((\mathbf{x},\Lambda),0)} \\ &\geq \xi((\mathbf{x},\Lambda),0) - \zeta((\mathbf{x},\Lambda),0) \\ &\quad + \frac{\lambda v_\gamma^\dagger((\mathbf{x},\Lambda))}{\tau((\mathbf{x},\Lambda),0)} + \frac{\sum_{j=1}^N \min(x_j,c_j)\mu_j v_\gamma^\dagger((\mathbf{x}^{j-},M_j))}{\tau((\mathbf{x},\Lambda),0)}, \end{aligned} \tag{3.4.38}$$

where the second inequality is due to the non-positivity of $v_\gamma^\dagger(\cdot)$ and the fact that $\xi_\gamma((\mathbf{x},\Lambda),0) - \zeta_\gamma((\mathbf{x},\Lambda),0) \geq \xi((\mathbf{x},\Lambda),0) - \zeta((\mathbf{x},\Lambda),0)$ which may be easily verified using (3.2.18) and (3.4.13). Hence, using the fact that $\tau((\mathbf{x},\Lambda),0) = \lambda + \sum_{j=1}^N \min(x_j,c_j)\mu_j$:

$$\begin{aligned} \frac{\sum_{j=1}^N \min(x_j,c_j)\mu_j v_\gamma^\dagger((\mathbf{x},\Lambda))}{\tau((\mathbf{x},\Lambda),0)} &\geq \xi((\mathbf{x},\Lambda),0) - \zeta((\mathbf{x},\Lambda),0) \\ &\quad + \frac{\sum_{j=1}^N \min(x_j,c_j)\mu_j v_\gamma^\dagger((\mathbf{x}^{j-},M_j))}{\tau((\mathbf{x},\Lambda),0)}. \end{aligned} \tag{3.4.39}$$

Then, by subtracting $\sum_{j=1}^N \min(x_j,c_j)\mu_j v_\gamma^\dagger((\mathbf{0},\Lambda))/\tau((\mathbf{x},\Lambda),0)$ from both sides of (3.4.39) and applying the inductive assumption to the differences $v_\gamma^\dagger((\mathbf{x}^{j-},M_j)) - v_\gamma^\dagger((\mathbf{0},\Lambda))$:

$$\begin{aligned} \frac{\sum_{j=1}^N \min(x_j,c_j)\mu_j \left(v_\gamma^\dagger((\mathbf{x},\Lambda)) - v_\gamma^\dagger((\mathbf{0},\Lambda))\right)}{\tau((\mathbf{x},\Lambda),0)} &\geq \xi((\mathbf{x},\Lambda),0) - \zeta((\mathbf{x},\Lambda),0) \\ &\quad - \frac{\sum_{j=1}^N \min(x_j,c_j)\mu_j f((\mathbf{x}^{j-},M_j))}{\tau((\mathbf{x},\Lambda),0)}. \end{aligned} \tag{3.4.40}$$

By multiplying both sides of (3.4.40) by $\tau((\mathbf{x},\Lambda),0)/(\sum_{j=1}^N \min(x_j,c_j)\mu_j)$, one obtains:

$$v_\gamma^\dagger((\mathbf{x},\Lambda)) - v_\gamma^\dagger((\mathbf{0},\Lambda)) \geq \frac{\tau((\mathbf{x},\Lambda),0)\,(\xi((\mathbf{x},\Lambda),0) - \zeta((\mathbf{x},\Lambda),0)) - \sum_{j=1}^N \min(x_j,c_j)\mu_j f((\mathbf{x}^{j-},M_j))}{\sum_{j=1}^N \min(x_j,c_j)\mu_j}. \tag{3.4.41}$$

This implies that one can define $f((\mathbf{x},\Lambda))$ as follows:

$$f((\mathbf{x},\Lambda)) := \frac{\tau((\mathbf{x},\Lambda),0)\,(\zeta((\mathbf{x},\Lambda),0) - \xi((\mathbf{x},\Lambda),0)) + \sum_{j=1}^N \min(x_j,c_j)\mu_j f((\mathbf{x}^{j-},M_j))}{\sum_{j=1}^N \min(x_j,c_j)\mu_j}, \tag{3.4.42}$$

with the result that $v_\gamma^\dagger((\mathbf{x},\Lambda)) - v_\gamma^\dagger((\mathbf{0},\Lambda))$ is bounded below by $-f((\mathbf{x},\Lambda))$. Note that $f((\mathbf{x},\Lambda))$ is non-negative since $\zeta((\mathbf{x},\Lambda),0) \geq \xi((\mathbf{x},\Lambda),0)$ and each value $f((\mathbf{x}^{j-},M_j))$ is non-negative by the inductive assumption. Again, due to (3.4.17), $-f((\mathbf{x},\Lambda))$ will also suffice as a lower bound for the difference $v_\gamma((\mathbf{x},\Lambda)) - v_\gamma((\mathbf{0},\Lambda))$ associated with the non-transformed process Ψ.

It remains to consider the case where $\omega = M_i$ for some $i \in \{1,2,...,N\}$. However, in this case the only possible action is $a = 0$, and therefore one may apply exactly the same arguments as in the case $\omega = \Lambda$; that is, the inequalities (3.4.38)-(3.4.41) remain valid with $(\mathbf{x},\Lambda)$ replaced by $(\mathbf{x},M_i)$. In conclusion, for any state $(\mathbf{x},\omega) \in \mathcal{S}$ it is possible to find a state-dependent lower bound $-f((\mathbf{x},\omega))$ satisfying (3.4.35) which is independent of the discount rate γ, provided that the same can be said for any state $(\mathbf{y},\omega') \in \mathcal{S}$ with $\mathbf{y} < \mathbf{x}$ in a partial ordering sense. Moreover, suitable values $f((\mathbf{x},\omega))$ can be obtained explicitly using (3.4.42) (which remains valid with Λ replaced by ω), by setting $f((\mathbf{0},\Lambda)) = 0$ and $f((\mathbf{0},M_i)) = \zeta((\mathbf{0},M_i),0) - \xi((\mathbf{0},M_i),0)$ for each $i \in \{1,2,...,N\}$ and then using recursive substitution. This completes the proof of the lemma. □

The final result of this section states that an average reward optimal policy for the continuous-time process Ψ may be found by restricting attention to *stationary* policies.

*** Theorem 3.4.10.** *There exists a stationary policy for Ψ which is average reward optimal. Moreover, there exists a constant g^* satisfying $g^* = \lim_{\gamma \downarrow 0} \gamma v_\gamma((\mathbf{x},\omega))$ for all $(\mathbf{x},\omega) \in \mathcal{S}$ and a function $h : \mathcal{S} \to \mathbb{R}$ with $-f((\mathbf{x},\omega)) \leq h((\mathbf{x},\omega)) \leq K$ for all $(\mathbf{x},\omega) \in \mathcal{S}$ (where K and $f((\mathbf{x},\omega))$ are as stated in Lemmas 3.4.8 and 3.4.9 respectively) such that for all $(\mathbf{x},\omega) \in \mathcal{S}$:*

$$h((\mathbf{x},\omega)) = \max_{a \in \mathcal{A}_{(\mathbf{x},\omega)}} \left\{ \xi((\mathbf{x},\omega),a) - \frac{g^*}{\tau((\mathbf{x},\omega),a)} + \sum_{(\mathbf{x}',\omega') \in \mathcal{S}} \sigma((\mathbf{x},\omega),a,(\mathbf{x}',\omega'))h((\mathbf{x}',\omega')) \right\}. \tag{3.4.43}$$

Proof. The proof relies upon showing that Ψ satisfies a set of conditions which have been shown by Sennott [157] to be sufficient for the existence of a stationary average reward optimal policy in a general Semi-Markov Decision Process (SMDP). Sennott's paper concerns the minimisation of expected long-run average costs, assuming that costs are bounded below (but not necessarily above). After translation to the equivalent problem of maximising long-run average rewards (with rewards bounded above), Sennott's conditions may be stated as follows:

1. The single-sojourn rewards are bounded above (but not necessarily bounded below); that is,

there exists $R \in \mathbb{R}$ such that $\xi_\gamma((\mathbf{x},\omega),a) \leq R$ for all $((\mathbf{x},\omega),a) \in \mathcal{S} \times \mathcal{A}_{(\mathbf{x},\omega)}$.

2. There exist $\epsilon > 0$ and $t_0 > 0$ such that $e^{-\tau((\mathbf{x},\omega),a)t_0} \geq \epsilon$ for all $((\mathbf{x},\omega),a) \in \mathcal{S} \times \mathcal{A}_{(\mathbf{x},\omega)}$.

3. For all states $(\mathbf{x},\omega) \in \mathcal{S}$, it is the case that $v_\gamma((\mathbf{x},\omega)) > -\infty$.

4. There exists $H \in \mathbb{R}$ such that $1/\tau((\mathbf{x},\omega),a) \leq H$ for all $((\mathbf{x},\omega),a) \in \mathcal{S} \times \mathcal{A}_{(\mathbf{x},\omega)}$.

5. There exists $\gamma_0 \in (0,\infty)$ and a constant $K \in \mathbb{R}$ such that, for all states $(\mathbf{x},\omega) \in \mathcal{S}$ and discount rates $\gamma \in (0,\gamma_0)$, $v_\gamma((\mathbf{x},\omega)) - v_\gamma((\mathbf{0},\Lambda)) \leq K$.

6. For every state $(\mathbf{x},\omega) \in \mathcal{S}$, there exists a non-negative value $f((\mathbf{x},\omega)) < \infty$ such that for all discount rates $\gamma \in (0,\infty)$, $v_\gamma((\mathbf{x},\omega)) - v_\gamma((\mathbf{0},\Lambda)) \geq -f((\mathbf{x},\omega))$.

7. For every $(\mathbf{x},\omega) \in \mathcal{S}$, and $a \in \mathcal{A}_{(\mathbf{x},\omega)}$ it is the case that $\sum_{(\mathbf{x}',\omega') \in \mathcal{S}} \sigma((\mathbf{x},\omega),a,(\mathbf{x}',\omega'))f((\mathbf{x}',\omega')) < \infty$, where $f((\mathbf{x}',\omega'))$ is as stated in the previous condition.

Conditions (1), (2) and (3) have already been verified in the proof of Theorem 3.4.2. The fact that condition (4) holds is immediate by choosing $H = 1/\lambda$. Conditions (5) and (6) are implied by the results of Lemmas 3.4.8 and 3.4.9 respectively. Finally, condition (7) follows by noting that the number of states $(\mathbf{x}',\omega')$ that can be reached in a single transition from an arbitrary state $(\mathbf{x},\omega) \in \mathcal{S}$ is finite and each value $f((\mathbf{x}',\omega'))$ is finite, regardless of the action $a \in \mathcal{A}_{(\mathbf{x},\omega)}$ chosen. Details of the rest of the proof may be found in [157] (see also [141], p. 559). □

The existence of a stationary average reward optimal policy for the continuous-time MDP Ψ justifies the approach of uniformising the process as described by Theorem 3.3.3 and then searching for an average reward optimal policy for the discrete-time MDP Φ, since any policy which maximises the expected long-run average reward among stationary policies in Φ must also do so in Ψ, and therefore (by Theorem 3.4.10) such a policy must be average reward optimal among *all* admissible policies in Ψ. In addition, restricting attention to stationary policies will enable a simplification of the state space to be made. Further details will be given in the next section.

3.5 Alternative MDP formulations

In this section and onwards, attention will be focused on the discrete-time MDP which is obtained by uniformising (and discretising) the continuous-time MDP Ψ formulated in Section 3.2. In later chapters, it will simplify matters considerably to use an MDP formulation in which the system state at any discrete time epoch is simply an N-vector $\mathbf{x} = (x_1, x_2, ..., x_N)$, as opposed to a pair $(\mathbf{x}, \omega)$ as in previous sections. As discussed in Section 3.2, this simplification of the state space is justified if one wishes to consider only *stationary* policies; indeed, the previous formulation was used only in order to accommodate time-dependent (non-stationary) policies. The results in Section 3.4 have established that it is possible to find both an average reward optimal policy and a γ-discount optimal policy for Ψ (given any discount rate $\gamma \in (0, \infty)$) by considering only stationary policies. This is an important fact, since if it were not known to be true then (in view of the fact that Theorem 3.3.3 applies specifically to stationary policies) one would not be able to rule out the possibility of a *non-stationary* policy being strictly superior to any stationary policy in the process Ψ, but being either sub-optimal or inadmissible in the discretised process Φ, and therefore the application of uniformisation would not have rigorous theoretical justification.

In order to avoid unnecessarily lengthy discussions, the approach taken in this section will be to simply introduce a new MDP formulation and then prove that it is equivalent (with respect to expected long-run average rewards earned by stationary policies) to the previous discrete-time MDP formulation Φ given in (3.3.6)-(3.3.12), as opposed to explaining the rationale of the new formulation in detail. However, some explanatory comments will be given following the equivalence proof. The previous MDP formulation inherited the same state space description as Ψ, but the new formulation will involve a simplified state space. Details are given below.

Simplified discrete-time MDP formulation

- The state space S and action sets $A_{\mathbf{x}}$ (for $\mathbf{x} \in S$) are given by:

$$S = \big\{(x_1, x_2, ..., x_N) : x_1, x_2, ..., x_N \in \mathbb{N}_0\big\}, \tag{3.5.1}$$

$$A_{\mathbf{x}} = A = \big\{0, 1, 2, ..., N\big\} \qquad \forall\, \mathbf{x} \in S. \tag{3.5.2}$$

- The one-step transition probabilities $p(\mathbf{x}, a, \mathbf{y})$ for $\mathbf{x}, \mathbf{y} \in S$ and $a \in A$ are given by:

$$p(\mathbf{x}, a, \mathbf{y}) = \begin{cases} \lambda\Delta, & \text{if } a = i \text{ for some} \\ & i \in \{1, 2, ..., N\} \text{ and } \mathbf{y} = \mathbf{x}^{i+}, \\ \min(x_i, c_i)\mu_i\Delta, & \text{if } \mathbf{y} = \mathbf{x}^{i-} \\ & \text{for some } i \in \{1, 2, ..., N\}, \\ 1 - I(a \neq 0)\lambda\Delta - \sum_{i=1}^{N} \min(x_i, c_i)\mu_i\Delta, & \text{if } \mathbf{y} = \mathbf{x}, \\ 0, & \text{otherwise,} \end{cases} \tag{3.5.3}$$

  where $\Delta \in \left(0, (\lambda + \sum_{i=1}^{N} c_i\mu_i)^{-1}\right]$ represents the discrete time step size and I is the indicator function; that is, $I(a \neq 0)$ takes a value of 1 if $a \neq 0$ and 0 otherwise.

- The reward function $r(\mathbf{x})$ is independent of the action a, and is given for $\mathbf{x} \in S$ by:

$$r(\mathbf{x}) = \sum_{i=1}^{N} \big(\alpha_i \min(x_i, c_i)\mu_i - \beta_i x_i\big). \tag{3.5.4}$$

.

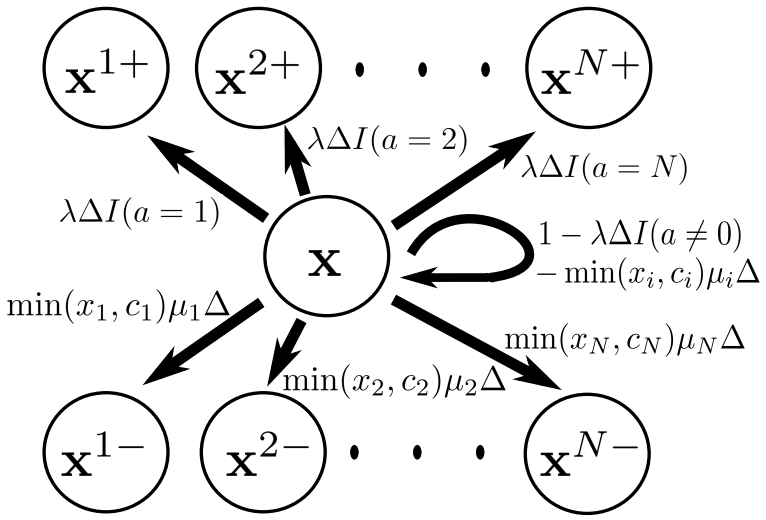


Figure 3.5: Transition probabilities (marked next to arrows) from an arbitrary state $\mathbf{x} \in S$ in Υ.

The formulation in (3.5.1)-(3.5.4) will be denoted by Υ, whereas Φ denotes the formulation given in (3.3.6)-(3.3.12). The next lemma establishes an equivalence between the two.

Lemma 3.5.1. *Suppose the process Φ operates under a stationary policy θ which induces an ergodic, irreducible Markov chain on some set of states $S_\theta \subseteq \mathcal{S}$, with $\mathcal{S}$ as defined in (3.2.7). Assume also that the stationary distribution $\{\pi_\theta((\mathbf{x},\omega))\}_{(\mathbf{x},\omega)\in\mathcal{S}}$ exists under θ, where $\pi_\theta((\mathbf{x},\omega))$ is the steady-state probability of the process Φ being in state $(\mathbf{x},\omega) \in \mathcal{S}$ and $\sum_{(\mathbf{x},\omega)} \pi_\theta((\mathbf{x},\omega)) = 1$. For each N-vector $\mathbf{x} = (x_1, x_2, ..., x_N)$, let the 'aggregate set' $H_\mathbf{x}$ be defined as follows:*

$$H_\mathbf{x} := F_\mathbf{x} \cup G_\mathbf{x},$$

where the disjoint sets $F_\mathbf{x}$ and $G_\mathbf{x}$ are given by:

$$\begin{aligned} F_\mathbf{x} &:= \big\{(\mathbf{y},\omega) \in \mathcal{S} : \mathbf{y} = \mathbf{x}^{i-} \text{ for some } i \in \{1,2,...,N\},\ \omega = \Lambda \text{ and } \theta((\mathbf{y},\omega)) = i\big\}, \\ G_\mathbf{x} &:= \big\{(\mathbf{y},\omega) \in \mathcal{S} : \mathbf{y} = \mathbf{x} \text{ and } \omega = M_i \text{ for some } i \in \{1,2,...,N\}\big\}, \end{aligned} \tag{3.5.5}$$

where $\theta((\mathbf{y},\omega))$ is the action chosen by θ at state $(\mathbf{y},\omega) \in \mathcal{S}$. Let $(H_0, H_1, ...)$ denote the sequence of aggregate states visited by the process Φ under the stationary policy described above. Then the sequence $(H_0, H_1, ...)$ is a Markov chain. That is, for $\mathbf{x} \in S$ and $n \geq 0$:

$$P\left(H_{n+1} = H_\mathbf{x} \mid H_n, H_{n-1}..., H_1, H_0\right) = P\left(H_{n+1} = H_\mathbf{x} \mid H_n\right).$$

Furthermore, suppose the 'simplified' process Υ operates under a stationary policy whereby the action chosen at state $\mathbf{x} \in S$ is equal to the action chosen by θ at state $(\mathbf{x},\Lambda) \in \mathcal{S}$ in the process Φ, and let $\{\pi_\theta(\mathbf{x})\}_{\mathbf{x}\in S}$ denote the resulting stationary distribution for Υ. Then:

$$\pi_\theta(\mathbf{x}) = \sum_{(\mathbf{y},\omega)\in H_\mathbf{x}} \pi_\theta((\mathbf{y},\omega)) =: \pi_\theta(H_\mathbf{x}) \qquad \forall\, \mathbf{x} \in S. \tag{3.5.6}$$

That is, the processes Φ and Υ have analogous steady-state distributions.

Proof. The proof can be accomplished using partitioning arguments. The details are somewhat laborious, and therefore they have been placed in Appendix A.3 (page 416).

Using the result of Lemma 3.5.1, it is possible to establish an equivalence between the processes Φ and Υ with respect to expected long-run average rewards under a stationary policy.

Lemma 3.5.2. *Suppose the processes Φ and Υ operate in the same manner as described in Lemma 3.5.1; that is, Φ follows a stationary policy θ which induces an ergodic, irreducible Markov chain with stationary distribution $\{\pi_\theta((\mathbf{x},\omega))\}_{(\mathbf{x},\omega)\in\mathcal{S}}$, and Υ follows a policy whereby the action chosen at any state $\mathbf{x} \in S$ is equal to the action chosen by the policy θ at state $(\mathbf{x},\Lambda) \in \mathcal{S}$ in the process Φ. Then the processes Φ and Υ earn the same expected long-run average reward.*

Proof. It will be convenient to consider the rewards for service α_i and holding costs β_i separately. The fact that the two processes incur the same expected long-run average holding costs per unit time can be shown very easily, since (3.3.11) implies that if the state $(\mathbf{x}_n, \omega_n)$ of process Φ at some epoch of time $n \geq 0$ is *any* state belonging to the aggregate set $H_{\mathbf{x}}$ (for some $\mathbf{x} \in S$), then the 'negative portion' of the reward $\mathcal{R}((\mathbf{x}_n, \omega_n), \theta((\mathbf{x}_n, \omega_n)))$ earned by Φ is given by $-\sum_{j=1}^{N} \beta_j x_j$. Hence, the expected long-run average holding costs incurred by Φ are given by:

$$\sum_{\mathbf{x} \in S} \pi_\theta(H_{\mathbf{x}}) \sum_{j=1}^{N} \beta_j x_j. \tag{3.5.7}$$

On the other hand, (3.5.4) implies that the long-run average holding costs for Υ are:

$$\sum_{\mathbf{x} \in S} \pi_\theta(\mathbf{x}) \sum_{j=1}^{N} \beta_j x_j. \tag{3.5.8}$$

The fact that (3.5.7) and (3.5.8) are equal then follows by the equivalence of the stationary distributions for the two processes, established by Lemma 3.5.1. The problem of verifying that the two processes earn the same long-run average rewards for service completions is slightly more difficult. Consider an arbitrary facility $i \in \{1, 2, ..., N\}$ and vector $\mathbf{x} \in S$, and then consider the rate per unit time at which service completions occur at facility i and cause the process Φ to enter some state $(\mathbf{x}_n, \omega_n)$ belonging to the aggregate set $H_{\mathbf{x}}$. This steady-state rate is given by:

$$\pi_\theta(H_{\mathbf{x}}) \lim_{n \to \infty} P\Big(\omega_n = M_i \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big) = \pi_\theta((\mathbf{x}, M_i)). \tag{3.5.9}$$

Consider the possible ways in which one might have $(\mathbf{x}_n, \omega_n) = (\mathbf{x}, M_i)$ at some time step $n \geq 1$. By definition of $H_{\mathbf{x}}$, it is clear that this can only occur if the state of the process Φ belongs to either $H_{\mathbf{x}^{i+}}$ or $H_{\mathbf{x}}$ at time step $n-1$. If $(\mathbf{x}_{n-1}, \omega_{n-1}) \in H_{\mathbf{x}^{i+}}$, then by (3.3.8)-(3.3.10) there is a probability $\min(x_i^{i+}, c_i)\mu_i\Delta$ that one will have $(\mathbf{x}_n, \omega_n) = (\mathbf{x}, M_i)$; on the other hand, if $(\mathbf{x}_{n-1}, \omega_{n-1}) \in H_{\mathbf{x}}$ then it can be verified using (3.3.8)-(3.3.10) that a transition to $(\mathbf{x}, M_i)$ is only possible if $(\mathbf{x}_{n-1}, \omega_{n-1}) = (\mathbf{x}, M_i)$, in which case the relevant probability is the 'self-transition' probability $\mathcal{P}((\mathbf{x}, M_i), 0, (\mathbf{x}, M_i)) = 1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j, c_j)\mu_j\Delta$. Hence:

$$\begin{aligned} P\Big((\mathbf{x}_n, \omega_n) = (\mathbf{x}, M_i)\Big) &= \min(x_i^{i+}, c_i)\mu_i\Delta P\Big((\mathbf{x}_{n-1}, \omega_{n-1}) \in H_{\mathbf{x}^{i+}}\Big) \\ &\quad + \left(1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j, c_j)\mu_j\Delta\right) P\Big((\mathbf{x}_{n-1}, \omega_{n-1}) = (\mathbf{x}, M_i)\Big). \end{aligned} \tag{3.5.10}$$

By then taking limits as $n \to \infty$, re-arranging and cancelling out Δ:

$$\pi_\theta((\mathbf{x}, M_i)) = \frac{\pi_\theta(H_{\mathbf{x}^{i+}}) \min(x_i^{i+}, c_i)\mu_i}{\lambda + \sum_{j=1}^{N} \min(x_j, c_j)\mu_j}.$$

Let $\mathcal{R}^+((\mathbf{x}, \omega), a)$ be the 'positive portion' of the single-step reward $\mathcal{R}((\mathbf{x}, \omega), a)$ defined in (3.3.11). That is, for state-action pairs $((\mathbf{x}, \omega), a) \in \mathcal{S} \times \mathcal{A}_{(\mathbf{x},\omega)}$:

$$\mathcal{R}^+((\mathbf{x}, \omega), a) = \begin{cases} \alpha_i \left(\lambda + \sum_{j=1}^{N} \min(x_j, c_j)\mu_j \right), & \text{if } \omega = M_i \text{ for some } i \in \{1, 2, ..., N\}, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, by the previous results, the expected long-run average rate at which the process Φ earns positive rewards as a result of service completions at facility i is given by:

$$\begin{aligned} &\sum_{\mathbf{x} \in S} \pi_\theta((\mathbf{x}, M_i)) \mathcal{R}^+((\mathbf{x}, M_i), 0) \\ &= \sum_{\mathbf{x} \in S} \frac{\pi_\theta(H_{\mathbf{x}^{i+}}) \min(x_i^{i+}, c_i)\mu_i}{\lambda + \sum_{j=1}^{N} \min(x_j, c_j)\mu_j} \alpha_i \left(\lambda + \sum_{j=1}^{N} \min(x_j, c_j)\mu_j \right) \\ &= \sum_{\mathbf{x} \in S} \pi_\theta(H_{\mathbf{x}^{i+}}) \alpha_i \min(x_i^{i+}, c_i)\mu_i. \end{aligned} \tag{3.5.11}$$

On the other hand, (3.5.4) implies that the expected long-run average rewards for the process Υ attributable to service completions at facility i are given by:

$$\sum_{\mathbf{x} \in S} \pi_\theta(\mathbf{x}) \alpha_i \min(x_i, c_i)\mu_i. \tag{3.5.12}$$

Note that the infinite summations $\sum_{\mathbf{x} \in S} \pi_\theta(H_{\mathbf{x}^{i+}}) \min(x_i^{i+}, c_i)$ and $\sum_{\mathbf{x} \in S} \pi_\theta(H_{\mathbf{x}}) \min(x_i, c_i)$ differ only in that the first summation excludes terms of the form $\pi_\theta(H_{\mathbf{x}}) \min(x_i, c_i)$ where the i^{th} component of $\mathbf{x}$ is zero, but these terms are equal to zero. Hence, due to (3.5.6):

$$\sum_{\mathbf{x} \in S} \pi_\theta(H_{\mathbf{x}^{i+}}) \alpha_i \min(x_i^{i+}, c_i)\mu_i = \sum_{\mathbf{x} \in S} \pi_\theta(H_{\mathbf{x}}) \alpha_i \min(x_i, c_i)\mu_i = \sum_{\mathbf{x} \in S} \pi_\theta(\mathbf{x}) \alpha_i \min(x_i, c_i)\mu_i.$$

Therefore (3.5.11) and (3.5.12) are equal. Since these arguments can be repeated for any facility $i \in \{1, 2, ..., N\}$, it follows that the two processes Φ and Υ earn the same expected long-run average rewards as a result of service completions. Since their equivalence with respect to expected long-run average holding costs has already been established, this completes the proof. $\square$

Obviously, in the discretised processes Φ and Υ, arrivals and service completions can only occur at discrete epochs of time $n = 0, 1, 2$, etc. An interesting feature of the 'simplified' MDP formulation Υ described in (3.5.1)-(3.5.4) is that rewards are applied 'retrospectively', in the sense that at any time step n, the reward $r(\mathbf{x}_n)$ is based on the service completion rates and holding costs incurred by the system over the *previous* time interval $[n-1, n]$ as opposed to the *next* interval $[n, n+1]$. It is this characteristic which enables the single-step reward $r(\mathbf{x}_n)$ to be determined independently of the action a_n chosen at time n. For example, consider a scenario in which a new customer arrives at time $n \geq 0$ when the system is in state $\mathbf{x} \in S$, and joins some facility $i \in \{1, 2, ..., N\}$. The rate at which holding costs are incurred at facility i immediately *prior* to time n is $\beta_i x_i$, whereas the corresponding rate immediately *after* time n is $\beta_i(x_i + 1)$. The reward $r(\mathbf{x})$ earned at time n incorporates the former quantity, rather than the latter. Similarly, the reward $r(\mathbf{x})$ also includes a term $\alpha_i \min(x_i, c_i)\mu_i$ (representing the rate of service completions at facility i), which takes no account of the new customer who arrives and joins facility i at the n^{th} time step.

The retrospective nature of the reward function $r(\mathbf{x})$ offers some advantages, not least of which is the fact that its lack of explicit dependence on actions chosen creates the appearance of greater simplicity (obviously, the actions chosen *do* affect the rewards indirectly, since they determine the transition probabilities for the process). However, some trivialities can occur when one considers *finite time horizon* problems, since the action chosen by the decision-maker with only one stage of the process remaining will have no bearing on the total reward earned (assuming that there are no terminal rewards earned when the horizon is reached); as such, the concept of an 'optimal one-stage policy' becomes somewhat meaningless. This is not a major issue, and is relevant to the results in the thesis only to the extent that it will sometimes be appropriate to ignore the case $n = 1$ when proving properties of optimal policies over a finite number of stages $n \in \mathbb{N}$.

In the remainder of this section it will be shown that the reward function $r(\mathbf{x})$ defined in (3.5.4) may be replaced with a different function $\hat{r}(\mathbf{x}, a)$ (to be defined shortly), without affecting the expected long-run average reward earned by the process Υ under a fixed stationary policy. The interchangeability of the two functions $r(\mathbf{x})$ and $\hat{r}(\mathbf{x}, a)$ will be extremely useful for proving various results later in this thesis. Suppose a new customer joins facility $i \in \{1, 2, ..., N\}$ under the state $\mathbf{x} \in S$. Recalling (3.2.14)-(3.2.15), the customer's *individual expected net reward* $w_i(x_i)$, taking into

account their expected waiting costs and the value of service α_i, is given by:

$$w_i(x_i) = \begin{cases} \alpha_i - \dfrac{\beta_i}{\mu_i}, & \text{if } x_i < c_i, \\ \alpha_i - \dfrac{\beta_i(x_i+1)}{c_i\mu_i}, & \text{if } x_i \geq c_i. \end{cases} \tag{3.5.13}$$

The transition probabilities (3.5.3) for Υ imply that for any $i \in \{1, 2, ..., N\}$, the process makes a transition from state $\mathbf{x}$ to $\mathbf{x}^{i+}$ with probability $\lambda\Delta$ if action $a = i$ is chosen. There is an implicit assumption here that an action is chosen by the decision-maker at *every discrete time step*. However, this action only affects the state-evolution of the process if a customer arrives at the time step in question (which occurs with probability $\lambda\Delta$); if no arrival occurs at a particular time step, then the action chosen at that step has no effect on the state transitions of the system. This echoes the discussion in Section 3.2, in which it was observed that when a simplified state space formulation such as (3.5.1) is used, the decision-maker is effectively required to make an *anticipative* decision at each decision epoch, without any knowledge of when the next arrival will occur. In this section, a discrete-time formulation is being considered rather than a continuous-time formulation, which effectively provides the decision-maker with extra opportunities to make decisions (since the effect of uniformisation is to increase the frequency of decision epochs by introducing self-transitions). As such, the decision-maker is not required to 'look as far ahead' as in the continuous-time case, but nevertheless the principle of making anticipative decisions remains the same.

To use an analogy, one may imagine the choice of an action in the discretised process Υ as being similar to the setting of points on a railway track. The switching of points on a track has no effect on railway traffic until the next train passes through. Similarly, in the process Υ, the decision-maker chooses an action at a particular time step which determines the route of any customer to arrive at that step, but if no customer arrives then the decision made has no effect.

Although the transition probabilities for Υ are defined in such a way that actions may not necessarily have consequences, the new reward function $\hat{r}(\cdot)$ will be defined in such a way that the reward $\hat{r}(\mathbf{x}_n, a_n)$ earned at a particular time step $n \geq 0$ *always* depends on the action chosen, regardless of whether or not an arrival occurs at time n. This will be achieved by setting the reward $\hat{r}(\mathbf{x}_n, a_n)$ equal to the *expected total of individuals' expected net rewards* at time step n. Obviously, the phrase "expected total of individuals' expected net rewards" is somewhat unwieldy; fortunately, it

is required here only for the purpose of motivating the definition of $\hat{r}(\mathbf{x}, a)$, and will not need to be used again. Suppose the system is in some state $\mathbf{x} \in S$ at a particular time step and an action $a \in A$ is chosen. The number of customer arrivals at that step will either be one or zero, with probabilities $\lambda\Delta$ and $1 - \lambda\Delta$ respectively. If a customer arrives, then their individual expected net reward is equal to $w_i(x_i)$ if $a = i$ for some $i \in \{1, 2, ..., N\}$ and zero otherwise. Hence, referring to (3.5.13), the expected total of individuals' expected net rewards is simply given by:

$$\begin{cases} \lambda\Delta\left(\alpha_i - \dfrac{\beta_i}{\mu_i}\right), & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } x_i < c_i, \\ \lambda\Delta\left(\alpha_i - \dfrac{\beta_i(x_i + 1)}{c_i\mu_i}\right), & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } x_i \geq c_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3.5.14}$$

If (3.5.14) was used as the definition for $\hat{r}(\mathbf{x}, a)$, then the formula for the expected long-run average reward in (3.3.1) would yield the *average reward per discrete time step*, as opposed to the average reward per unit time. This minor problem can be resolved by simply dividing by Δ in (3.5.14) (recall that a similar adjustment was made to the reward function $\mathcal{R}((\mathbf{x}, \omega), a)$ for the process Φ, defined in (3.3.11)). Accordingly, the new reward function $\hat{r}(\mathbf{x}, a)$ is given by:

$$\hat{r}(\mathbf{x}, a) = \begin{cases} \lambda\left(\alpha_i - \dfrac{\beta_i}{\mu_i}\right), & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } x_i < c_i, \\ \lambda\left(\alpha_i - \dfrac{\beta_i(x_i + 1)}{c_i\mu_i}\right), & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } x_i \geq c_i, \\ 0, & \text{otherwise.} \end{cases} \tag{3.5.15}$$

It is important to emphasise once again that when the reward formulation (3.5.15) is used for Υ, a reward $\hat{r}(\mathbf{x}, a)$ is earned for choosing $a \in A$ at state $\mathbf{x} \in S$, *regardless of whether or not an arrival occurs*. Although this may result in the system earning 'fictitious' rewards based on the expected individual rewards of customers who do not arrive, the mathematical validity of this formulation is ensured by the fact that it is based on an *expectation*, as explained previously.

In fact, it is possible to define the rewards in the discretised process so that they are associated with *transitions* between states. One may define $\hat{r}(\mathbf{x}, a, \mathbf{y})$ as the (anticipatory) reward for transferring from state $\mathbf{x} \in S$ to $\mathbf{y} \in S$, given that action $a \in A$ is chosen. In this case, in order to obtain an

equivalence with (3.5.15), it is appropriate to define $\hat{r}(\mathbf{x}, a, \mathbf{y})$ as follows:

$$\hat{r}(\mathbf{x}, a, \mathbf{y}) = \begin{cases} w_i(x_i), & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } \mathbf{y} = \mathbf{x}^{i+}, \\ 0, & \text{otherwise,} \end{cases}$$

where $w_i(x_i)$ is as defined in (3.5.13). The equivalence with (3.5.15) may be easily verified; indeed, the *expected* reward for choosing facility i at state $\mathbf{x}$ is then given by:

$$\sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y})\, \hat{r}(\mathbf{x}, a, \mathbf{y}) = \lambda \Delta w_i(x_i),$$

which corresponds with the (fixed) reward for choosing facility i at state $\mathbf{x}$ in (3.5.14). One might suggest that the $\hat{r}(\mathbf{x}, a, \mathbf{y})$ formulation is somewhat more natural, since it enables rewards to be earned only at the times at which customers arrive; however, the $\hat{r}(\mathbf{x}, a)$ formulation is clearly simpler from a notational point of view. In later chapters it will occasionally prove useful to switch between the real-time and anticipatory reward functions $r(\mathbf{x})$ and $\hat{r}(\mathbf{x}, a)$, but it will not be necessary to use any transition-based reward functions of the form $\hat{r}(\mathbf{x}, a, \mathbf{y})$.

The reward formulations $r(\mathbf{x})$ and $\hat{r}(\mathbf{x}, a)$ in (3.5.4) and (3.5.15) respectively are both logical for the process Υ, and their relationship bears analogy to the relationship between the continuous-time reward formulations $\xi((\mathbf{x}, \omega), a)$ and $\hat{\xi}((\mathbf{x}, \omega), a)$ in (3.2.13) and (3.2.16). Specifically, (3.5.4) is based on the real-time holding costs and rewards incurred during the system's evolution, while (3.5.15) is based on an unbiased estimate of each arriving customer's contribution to the aggregate net reward. Using the same terminology as in Section 3.2, the formulation (3.5.4) will be referred to as the *real-time* reward formulation, while its counterpart (3.5.15) will be referred to as the *anticipatory* formulation. The next result establishes an equivalence between the two formulations which, as mentioned earlier, will be of tremendous use throughout this thesis.

Theorem 3.5.3. *Assume the process Υ operates under a stationary policy θ. Let $g_\theta(\mathbf{x}, r)$ and $g_\theta(\mathbf{x}, \hat{r})$ denote the expected long-run average rewards under the reward formulations $r(\mathbf{x})$ and $\hat{r}(\mathbf{x}, a)$ respectively, given that the policy θ is followed and the initial state is $\mathbf{x} \in S$. Then:*

$$g_\theta(\mathbf{x}, r) = g_\theta(\mathbf{x}, \hat{r}), \tag{3.5.16}$$

That is, the average reward under θ is the same under either reward formulation.

Proof. Let it be assumed that the policy θ induces an irreducible, ergodic Markov chain with stationary distribution $\{\pi_\theta(\mathbf{x})\}_{\mathbf{x}\in S}$, where $\pi_\theta(\mathbf{x})$ is the steady state probability of being in state $\mathbf{x} \in S$ and $\sum_{\mathbf{x}\in S} \pi_\theta(\mathbf{x}) = 1$. If this is not the case, then the system is unstable and both quantities in (3.5.16) are (negatively) infinite. Hence, $g_\theta(\mathbf{x}, r)$ and $g_\theta(\mathbf{x}, \hat{r})$ are given by:

$$g_\theta(\mathbf{x}, r) = \sum_{\mathbf{x}\in S} \pi_\theta(\mathbf{x})\, r(\mathbf{x}),$$
$$g_\theta(\mathbf{x}, \hat{r}) = \sum_{\mathbf{x}\in S} \pi_\theta(\mathbf{x})\, \hat{r}\big(\mathbf{x}, \theta(\mathbf{x})\big),$$

where $\theta(\mathbf{x})$ denotes the action chosen by θ under state $\mathbf{x}$. (Recall that r, unlike $\hat{r}$, is independent of the action chosen.) For each $\mathbf{x} \in S$, the steady-state probability $\pi_\theta(\mathbf{x})$ is the same under either reward formulation since the policy θ is fixed. The objective is to show:

$$\sum_{\mathbf{x}\in S} \pi_\theta(\mathbf{x})\, r(\mathbf{x}) = \sum_{\mathbf{x}\in S} \pi_\theta(\mathbf{x})\, \hat{r}\big(\mathbf{x}, \theta(\mathbf{x})\big).$$

The state space S may be partitioned into disjoint subsets. For each facility $i \in \{1, 2, .., N\}$, let S_i denote the set of states at which the action chosen (under the policy θ) is to join i. Then $S_i := S_{i-} \cup S_{i+}$, where S_{i-} and S_{i+} are defined in the following way:

$$S_{i-} := \big\{\mathbf{x} \in S : \theta(\mathbf{x}) = i \text{ and } x_i < c_i\big\},$$

$$S_{i+} := \big\{\mathbf{x} \in S : \theta(\mathbf{x}) = i \text{ and } x_i \geq c_i\big\}.$$

Also, let S_0 denote the set of states at which the action chosen under θ is to balk. Now let $g_\theta(\mathbf{x}, r)$ and $g_\theta(\mathbf{x}, \hat{r})$ be divided into 'positive' and 'negative' constituents as follows:

$$g_\theta^+(\mathbf{x}, r) := \sum_{\mathbf{x}\in S} \sum_{i=1}^{N} \pi_\theta(\mathbf{x}) \min(x_i, c_i)\alpha_i\mu_i,$$

$$g_\theta^-(\mathbf{x}, r) := -\sum_{\mathbf{x}\in S} \sum_{i=1}^{N} \pi_\theta(\mathbf{x})\beta_i x_i, \tag{3.5.17}$$

$$g_\theta^+(\mathbf{x}, \hat{r}) := \lambda \sum_{i=1}^{N} \sum_{\mathbf{x}\in S_i} \pi_\theta(\mathbf{x})\alpha_i, \tag{3.5.18}$$

$$g_\theta^-(\mathbf{x}, \hat{r}) := -\lambda \sum_{i=1}^{N} \left(\sum_{\mathbf{x}\in S_{i-}} \pi_\theta(\mathbf{x}) \frac{\beta_i}{\mu_i} + \sum_{\mathbf{x}\in S_{i+}} \pi_\theta(\mathbf{x}) \frac{\beta_i(x_i + 1)}{c_i\mu_i} \right). \tag{3.5.19}$$

By referring to (3.5.4) and (3.5.15), it can be checked that $g_\theta(\mathbf{x}, r) = g_\theta^+(\mathbf{x}, r) + g_\theta^-(\mathbf{x}, r)$ and $g_\theta(\mathbf{x}, \hat{r}) = g_\theta^+(\mathbf{x}, \hat{r}) + g_\theta^-(\mathbf{x}, \hat{r})$. It will be sufficient to show that $g_\theta^+(\mathbf{x}, r) = g_\theta^+(\mathbf{x}, \hat{r})$ and $g_\theta^-(\mathbf{x}, r) = g_\theta^-(\mathbf{x}, \hat{r})$. Let $S_{i,k} \subseteq S_i$ (for $k = 0, 1, 2, ...$) be the set of states at which the action chosen under θ is to join facility i, given that there are k customers present at i. That is:

$$S_{i,k} := \big\{\mathbf{x} \in S : \theta(\mathbf{x}) = i \text{ and } x_i = k\big\}.$$

Using the detailed balance equations for ergodic Markov chains under steady state conditions (see [171] p. 148) it follows that for every facility i and $k \geq 0$, the 'rate of flow' exiting the class of states in S with $x_i \leq k$ must be equal to the rate of flow entering this class, hence:

$$\lambda \sum_{\mathbf{x} \in S_{i,k}} \pi_\theta(\mathbf{x}) = \sum_{\substack{\mathbf{x} \in S \\ x_i = k+1}} \pi_\theta(\mathbf{x}) \min(x_i, c_i)\mu_i. \tag{3.5.20}$$

Summing over all $k \in \mathbb{N}_0$, the following relationship is obtained:

$$\lambda \sum_{\mathbf{x} \in S_i} \pi_\theta(\mathbf{x}) = \sum_{\mathbf{x} \in S} \pi_\theta(\mathbf{x}) \min(x_i, c_i)\mu_i, \tag{3.5.21}$$

for $i \in \{1, 2, ..., N\}$. The physical interpretation of (3.5.21) is that, under steady state conditions, the rate at which customers join facility i is equal to the rate at which service completions occur at i. Multiplying both sides of (3.5.21) by α_i and summing over $i \in \{1, 2, ..., N\}$ gives:

$$\lambda \sum_{i=1}^{N} \sum_{\mathbf{x} \in S_i} \pi_\theta(\mathbf{x})\alpha_i = \sum_{i=1}^{N} \sum_{\mathbf{x} \in S} \pi_\theta(\mathbf{x}) \min(x_i, c_i)\alpha_i\mu_i,$$

which states that $g_\theta^+(\mathbf{x}, \hat{r}) = g_\theta^+(\mathbf{x}, r)$ as required. It remains to show that $g_\theta^-(\mathbf{x}, \hat{r}) = g_\theta^-(\mathbf{x}, r)$. In (3.5.20) (which holds for all $k \in \mathbb{N}$ and $i \in \{1, 2, .., N\}$), putting $k = c_i$ yields:

$$\lambda \sum_{\mathbf{x} \in S_{i,c_i}} \pi_\theta(\mathbf{x}) = \sum_{\substack{\mathbf{x} \in S \\ x_i = c_i+1}} \pi_\theta(\mathbf{x}) c_i\mu_i. \tag{3.5.22}$$

Suppose both sides of (3.5.22) are multiplied by $c_i + 1$. Since the sum on the left-hand side is over $\mathbf{x} \in S_{i,c_i}$ and the sum on the right-hand side is over states with $x_i = c_i + 1$, this is equivalent to multiplying each summand on the left-hand side by $x_i + 1$ and each summand on the right-hand side by x_i. In addition, multiplying both sides by $\beta_i/(c_i\mu_i)$ yields:

$$\lambda \sum_{\mathbf{x} \in S_{i,c_i}} \pi_\theta(\mathbf{x}) \frac{\beta_i(x_i + 1)}{c_i\mu_i} = \sum_{\substack{\mathbf{x} \in S \\ x_i = c_i+1}} \pi_\theta(\mathbf{x})\beta_i x_i. \tag{3.5.23}$$

Similar expressions result from putting $k = c_i + 1,\ c_i + 2$ etc. Recall that $\bigcup_{k=c_i}^{\infty} S_{i,k} = S_{i+}$ by definition. Hence, summing over all $k \geq c_i$ in (3.5.23) gives:

$$\lambda \sum_{\mathbf{x} \in S_{i+}} \pi_\theta(\mathbf{x}) \frac{\beta_i(x_i + 1)}{c_i \mu_i} = \sum_{\substack{\mathbf{x} \in S \\ x_i \geq c_i + 1}} \pi_\theta(\mathbf{x}) \beta_i x_i. \tag{3.5.24}$$

Note also that multiplying both sides of (3.5.20) by β_i/μ_i and summing over $k \in \{1, 2, ..., c_i - 1\}$ (and also recalling that $\bigcup_{k=0}^{c_i - 1} S_{i,k} = S_{i-}$) gives:

$$\lambda \sum_{\mathbf{x} \in S_{i-}} \pi_\theta(\mathbf{x}) \frac{\beta_i}{\mu_i} = \sum_{\substack{\mathbf{x} \in S \\ x_i \leq c_i}} \pi_\theta(\mathbf{x}) \beta_i x_i. \tag{3.5.25}$$

Hence, from (3.5.24) and (3.5.25) the following is obtained:

$$\lambda \left(\sum_{\mathbf{x} \in S_{i-}} \pi_\theta(\mathbf{x}) \frac{\beta_i}{\mu_i} + \sum_{\mathbf{x} \in S_{i+}} \pi_\theta(\mathbf{x}) \frac{\beta_i(x_i + 1)}{c_i \mu_i} \right) = \sum_{\mathbf{x} \in S} \pi_\theta(\mathbf{x}) \beta_i x_i.$$

Summing over $i \in \{1, 2, ..., N\}$ gives $g_\theta^-(\mathbf{x}, \hat{r}) = g_\theta^-(\mathbf{x}, r)$ as required. It has already been shown that $g_\theta^+(\mathbf{x}, \hat{r}) = g_\theta^+(\mathbf{x}, r)$, so this completes the proof that $g_\theta(\mathbf{x}, \hat{r}) = g_\theta(\mathbf{x}, r)$. □

It follows from Theorem 3.5.3 that any stationary policy which maximises the expected long-run average reward for Υ when the real-time reward formulation (3.5.4) is used must also do so when the anticipatory formulation (3.5.15) is adopted instead. Both formulations offer certain practical advantages, and (as stated earlier) both will be relied upon to prove later results.

As a further point, in Theorem 3.5.3 it was proved that $g_\theta^+(\mathbf{x}, r) = g_\theta^+(\mathbf{x}, \hat{r})$ (with $g_\theta^+(\mathbf{x}, r)$ and $g_\theta^+(\mathbf{x}, \hat{r})$ as defined in (3.5.19)), and similarly $g_\theta^-(\mathbf{x}, r) = g_\theta^-(\mathbf{x}, \hat{r})$. These facts, together with the fact that $g_\theta(\mathbf{x}, \hat{r}) = g_\theta(\mathbf{x}, r)$, make it possible to 'mix' the two formulations by noting:

$$g_\theta(\mathbf{x}, r) = g_\theta(\mathbf{x}, \hat{r}) = g_\theta^+(\mathbf{x}, r) + g_\theta^-(\mathbf{x}, \hat{r}) = g_\theta^-(\mathbf{x}, r) + g_\theta^+(\mathbf{x}, \hat{r}).$$

This relationship enables 'hybrid' reward functions to be created using the 'positive' and 'negative' components of r and $\hat{r}$. For example, by taking the positive component of $\hat{r}(\mathbf{x}, a)$ and the negative component of $r(\mathbf{x})$, a new reward function $\bar{r}(\mathbf{x}, a)$ may be defined as follows:

$$\bar{r}(\mathbf{x}, a) = \begin{cases} \lambda \alpha_i - \sum_{j=1}^{N} \beta_j x_j, & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\}, \\ -\sum_{j=1}^{N} \beta_j x_j, & \text{otherwise.} \end{cases} \tag{3.5.26}$$

The arguments in the proof of Theorem 3.5.3 show that the average reward $g_\theta(\mathbf{x}, \bar{r})$ under the new 'hybrid' function $\bar{r}$ is equal to both $g_\theta(\mathbf{x}, r)$ and $g_\theta(\mathbf{x}, \hat{r})$; that is, the three reward functions r, $\hat{r}$ and $\bar{r}$ can be used interchangeably. It is somewhat interesting to note that the 'real-time' reward function r in (3.5.4) has no explicit dependence on λ, while the 'hybrid' reward function $\bar{r}$ in (3.5.26) has no explicit dependence on any of the service rates μ_i or service capacities c_i. Of course, there is an implicit dependence on these parameters due to the detailed balance equations for Markov chains, which imply that the steady-state relationship (3.5.21) holds.

Much of the work in this chapter so far has been concerned with presenting and comparing different CTMDP and MDP formulations. Comparisons have been made between continuous-time and discrete-time formulations, the two state space formulations (3.2.7) and (3.5.1), and different reward formulations such as (3.5.4) and (3.5.15) (or, in the CTMDP case, (3.2.13) and (3.2.16)). Throughout the vast majority of the remainder of this thesis, attention will be restricted to the 'simplified' discrete-time MDP Υ, with the two reward formulations $r(\mathbf{x})$ and $\hat{r}(\mathbf{x}, a)$ being interchanged at various points. However, one should not lose sight of the fact that the queueing system described in Section 3.1 is a *continuous-time* system; essentially, the discrete-time MDP Υ is simply a vehicle through which analysis of the continuous-time system can be performed.

Broadly speaking, the problems of interest in later chapters will involve either the analysis of certain stationary policies, or the identification and characterisation of *average reward optimal* policies. It is considered self-evident that the continuous-time MDP Ψ formulated in Section 3.2 is an accurate mathematical model for the real-world process described in Section 3.1. As such, identifying an optimal control policy for this real-world process should be equivalent to finding an optimal policy for the process Ψ. The fact that this task may be accomplished via analysis of the discrete-time MDP Υ may be argued by citing the main results of this chapter so far:

- Any policy θ^* which is average reward optimal among all stationary policies for the process Υ under the anticipatory reward formulation $\hat{r}(\mathbf{x}, a)$ must also be likewise optimal under the real-time formulation $r(\mathbf{x})$, and vice versa. This is due to Theorem 3.5.3.

- If θ^* maximises the average reward among stationary policies in Υ then, by Lemma 3.5.2, it must also do so in Φ, the MDP formulated in Section 3.3 (with a more complicated state space representation). This assumes that actions are mapped so that the action chosen at

state $(\mathbf{x}, \Lambda) \in \mathcal{S}$ in Φ matches the action chosen at state $\mathbf{x} \in S$ in Υ by θ^*.

- If θ^* is average reward optimal among stationary policies in Φ then it must also be average reward optimal among stationary policies in the continuous-time process Ψ. This is due to Theorem 3.3.3, noting that Φ is obtained directly from Ψ via uniformisation.
- If θ^* maximises the average reward among stationary policies in Ψ then, by Theorem 3.4.10, it must also maximise average reward among *all* admissible policies in Ψ.

Thus, a chain of arguments can be used to show that optimising the average reward over stationary policies in Υ (using either reward formulation) is equivalent to optimising the average reward over the (much larger) class of all admissible policies in Ψ. If one only wishes to investigate the properties of a fixed stationary policy (or compare certain stationary policies), then the equivalence between Υ and Ψ is much easier to argue; this requires only Lemma 3.5.2 and Theorem 3.3.3.

The next section will provide a formal classification for admissible policies in an MDP.

3.6 Decision rules and policies

So far, the term *policy* has been used in a somewhat informal manner to refer to the decision-making scheme used by the system controller in a CTMDP or MDP. It is important to provide a proper definition for this term, and also the related concept of a *decision rule*. As mentioned in Section 3.2, it is generally assumed that the rewards and transition rates are *stationary*, and have no dependence on the time index (i.e. the amount of time for which the process has been running); however, there is not necessarily any similar assumption for the decision-making criteria used by the system controller. In fact, the action chosen when the system is in a particular state may depend not only on the state itself, but also on the time index and the entire history of states visited and actions chosen up to that point. Furthermore, an action may be chosen according to a random probability distribution as opposed to being selected deterministically.

This thesis will adopt the classification of decision rules and policies proposed by Puterman [141] (p. 533), with one exception: a stationary policy (see Definition 3.3.2) will always be assumed to be *non-randomised*. By contrast, Puterman allows stationary policies to be either randomised or

deterministic. The classification here will be given in the context of a general *discrete-time* MDP (see Definition 3.3.1), but the corresponding classification for a CTMDP would be analogous. As discussed in the previous section, attention will generally be restricted to discrete-time MDPs from this point onwards, so the continuous-time classification is somewhat less relevant.

Suppose an MDP is initialised in some state $\mathbf{x}_0 \in S$, and an action $a_0 \in A_{\mathbf{x}}$ is chosen. The *history* of the process up to the n^{th} decision epoch is defined as a sequence $\mathcal{H}_n$, given by:

$$\mathcal{H}_n = (\mathbf{x}_0, a_0, \omega_0, \mathbf{x}_1, a_1, \omega_1, ..., \mathbf{x}_n), \tag{3.6.1}$$

where $\mathbf{x}_m \in S$ (for $m = 0, 1, ..., n$) is the state of the process at the n^{th} decision epoch, $a_m \in A_{\mathbf{x}_m}$ is the corresponding action chosen, and ω_m is the random event that occurs. In the case of the MDP Υ (see (3.5.1)-(3.5.4)) the random events ω_m may be represented using similar notation to that in Section 3.2, except that since a discrete-time MDP is being considered, the event that occurs at a particular time step may not necessarily be either an arrival or a service completion; one must also consider the possibility that no event occurs (i.e. the MDP makes a 'self-transition'). As such, the set of possible events should also include an extra symbol to denote a 'non-event'. It will be convenient to use 0 for this purpose, so that each random event ω_m satisfies:

$$\omega_m \in \{\Lambda, M_1, M_2, ..., M_N, 0\}.$$

It should be noted that including the random events ω_m in the history of the process $\mathcal{H}_n$ does not complicate the formulation or analysis of the MDP Υ in any way. In fact, the random events are included for the sole purpose of ensuring that the history $\mathcal{H}_n$ includes the arrival times of all customers since the initialisation of the process. Obviously, it is sometimes possible to deduce a customer's arrival time from the state-transitions of the process; for example, if the state of the system is $\mathbf{x} \in S$ at time step n and $\mathbf{x}^{i+}$ at time step $n+1$ (for some $i \in \{1, 2, ..., N\}$), then this implies a customer arrival at the n^{th} time step. However, if the history records the process being in state $\mathbf{x}$ at both of the time steps n and $n+1$, with the action $a_n = 0$ (i.e. balking) having been chosen at time n, then it is impossible to deduce from this information whether a customer has arrived at time n, or whether a 'non-event' has occurred (recall that actions $a \in A$ are always chosen at every discrete time step, regardless of whether or not a customer arrives). There is no reason why the history of the process should *not* record the arrival times of customers, so for this reason it is desirable for the events ω_m to be 'logged' as part of the process history.

Let d_n be the *decision rule* which prescribes the procedure for selecting an action at time n. Puterman's categorisation scheme for decision rules ([141], p. 21) is summarised below.

Definition 3.6.1. ***(Classification of decision rules)***

Let d_n be the decision rule applied by the system controller at the n^{th} decision epoch.

- *If d_n is a function $d_n : S \to A_{\mathbf{x}_n}$, i.e. it chooses an action with certainty depending only on the present state $\mathbf{x}_n$, then d_n is* deterministic Markovian (*abbreviated as* MD).
- *If d_n is a function $d_n : \mathcal{H}_n \to A_{\mathbf{x}_n}$, i.e. it chooses an action with certainty depending on the history of past states, actions and random events (including the present state $\mathbf{x}_n$), then the decision rule is* deterministic and history-dependent (HD).
- *If d_n specifies a probability $p_{d_n(\mathbf{x}_n)}(a)$ for choosing each action $a \in A_{\mathbf{x}_n}$ which depends only on the present state $\mathbf{x}_n$, then d_n is* randomised Markovian (MR).
- *If d_n specifies a probability $p_{d_n(\mathcal{H}_n)}(a)$ for choosing each action $a \in A_{\mathbf{x}_n}$, and this probability distribution depends on the history of past states, actions and random events (including the present state $\mathbf{x}_n$), then d_n is* randomised and history-dependent (HR).

It follows that a deterministic policy is a special case of a randomised policy in which the probability distribution is *degenerate*, i.e. $p_{d_n(\mathbf{x}_n)}(a) = 1$ (or $p_{d_n(\mathcal{H}_n)}(a) = 1$) for some $a \in A_{\mathbf{x}_n}$. Clearly, a deterministic Markovian (MD) decision rule is the easiest to apply, but this may be too restrictive in some applications. A randomised and history dependent (HR) rule is the most general type of rule. Note that in general, there is no requirement for the same decision rule to be used at each decision epoch; that is, a deterministic decision rule may be applied at one point in time, but a randomised rule may be chosen at another. A *policy* is a means of specifying the decision rules employed at different epochs of time during the evolution of the process.

Definition 3.6.2. ***(Policies)*** *A* policy θ *is a sequence of decision rules $(d_0, d_1, ...)$, where d_n is the decision rule applied by the system controller at the n^{th} decision epoch.*

Thus, according to Definition 3.3.2, a *stationary* policy is one in which the same deterministic Markovian decision rule is applied at every decision epoch. The results in earlier sections have

established that an average reward optimal policy for the process Ψ can always be found by restricting attention to stationary policies in Υ. However, this does not mean that *non-stationary* policies (which fit somewhere within the broader classification of policies provided in Definition 3.6.2) will not be relevant to consider in later chapters. In fact, it is sometimes possible to prove that a particular stationary policy is *not* average reward optimal by showing that it is inferior to a carefully-constructed *non-stationary* policy. Examples will be provided later.

The next section will present some general results from the literature related to average reward optimal policies for discrete-time MDPs, including computational algorithms.

3.7 Finite state spaces

The purpose of this section is to discuss methods for computing average reward optimal policies for discrete-time MDPs, but before doing so it is necessary to discuss some of the related theory. Consider the discrete-time MDP Υ formulated in (3.5.1)-(3.5.4). By direct analogy to (3.3.1), the expected long-run average reward for Υ under an arbitrary policy θ, given an initial state $\mathbf{x}_0 \in S$ and assuming that the real-time reward formulation (3.5.4) is used, is defined as:

$$g_\theta(\mathbf{x}) = \liminf_{t\to\infty} t^{-1} E_\theta \left[\sum_{n=0}^{t-1} r(\mathbf{x}_n) \Big| \mathbf{x}_0 = \mathbf{x} \right], \tag{3.7.1}$$

where the expectation is with respect to the random transitions that occur under θ. Due to Theorem 3.5.3, the rewards $r(\mathbf{x}_n)$ in (3.7.1) may be replaced by the rewards $\hat{r}(\mathbf{x}_n, a_n)$ defined in (3.5.15) without affecting the value of $g_\theta(\mathbf{x})$. Indeed, it should be noted that all of the results in this section remain valid under either of the reward formulations $r(\cdot)$ and $\hat{r}(\cdot)$.

The state space S for Υ, defined in (3.5.1), is countably infinite. Unfortunately, it is the case that certain theoretical results for MDPs rely on the assumption of a *finite* state space. Furthermore, the computational algorithms presented later in this section cannot be applied, in their conventional forms, to a problem in which the state space S is infinite. In practice, one can work around this problem by truncating the state space, but only if it can safely be assumed that there exists a finite subset $R \subset S$ and an optimal policy θ^* such that R contains any state that could possibly be visited under θ^*. This approach will be discussed further in the next chapter.

Let $\tilde{\Upsilon}$ denote an MDP which is similar to Υ except that the action sets $A_{\mathbf{x}}$ for $\mathbf{x} \in S$ are restricted as follows: for each facility $i \in \{1, 2, ..., N\}$, there exists a threshold $B_i \in \mathbb{N}$ such that the action i is not available at any state $\mathbf{x}$ with $x_i \geq B_i$. Then, assuming that the system is initialised in state $\mathbf{0}$, the state space for the process may be described by the *finite* set $\tilde{S}$ given by:

$$\tilde{S} := \big\{(x_1, x_2, ..., x_N) : x_i \in \mathbb{N}_0,\ x_i \leq B_i\ (i = 1, 2, ..., N)\big\}. \tag{3.7.2}$$

All of the results in this section will assume a finite state space $\tilde{S}$, with $\tilde{S}$ as defined in (3.7.2). For the time being, the choice of non-negative integers $B_1, B_2, ..., B_N$ is understood to be arbitrary. Two immediate consequences of the finiteness of $\tilde{S}$ are as follows:

1. Any stationary policy induces a Markov chain with an irreducible, aperiodic class of states.
2. The rewards are bounded; that is, there exists $M > 0$ such that $|r(\mathbf{x})| \leq M$ for all $\mathbf{x} \in \tilde{S}$.

Indeed, to establish the first of the above properties, let θ be a stationary policy and let $R_\theta \subseteq \tilde{S}$ denote the set of states accessible from state $\mathbf{0}$ in the Markov chain induced by θ (please refer to Appendix A.1 for the definition of accessibility). Then R_θ is a single communicating class and hence is *irreducible*. It is also *aperiodic*, since the state $\mathbf{0}$ can be reached from itself via a single transition. The second property follows directly from (3.5.4) and the finiteness of $\tilde{S}$.

It will be useful to refer to some results from the literature for finite-state MDPs. Let θ be a stationary policy which induces a Markov chain with transition probabilities $p_\theta(\mathbf{x}, \mathbf{y})$ for $\mathbf{x}, \mathbf{y} \in \tilde{S}$ and let the *expected total reward* over n time epochs, $V_\theta^{(n)}(\mathbf{x})$, be defined as:

$$V_\theta^{(n)}(\mathbf{x}) := E_\theta \left[\sum_{k=0}^{n-1} r(\mathbf{x}_k) \Big| \mathbf{x}_0 = \mathbf{x} \right]. \tag{3.7.3}$$

The next result addresses the existence of the limit in (3.7.1).

*** Lemma 3.7.1.** *Let θ be any stationary policy in the MDP $\tilde{\Upsilon}$ with finite state space $\tilde{S}$. Then, for each state $\mathbf{x} \in \tilde{S}$, the limit $\lim_{n\to\infty} V_\theta^{(n)}(\mathbf{x})/n$ exists, and:*

$$g_\theta(\mathbf{x}) = \lim_{n\to\infty} \frac{1}{n} V_\theta^{(n)}(\mathbf{x}) = \sum_{\mathbf{y}\in\tilde{S}} \pi_\theta(\mathbf{y})\, r(\mathbf{y}), \tag{3.7.4}$$

where $\big\{\pi_\theta(\mathbf{y})\big\}_{\mathbf{y}\in\tilde{S}}$ is the stationary distribution of the MDP under θ.

Proof. Please refer to Bertsekas [12] (p. 330) or Puterman [141] (p. 333).

From Lemma 3.7.1 it follows that, for any *stationary* policy θ, the limit in (3.7.1) is guaranteed to exist, so it is not necessary to write 'lim inf'. Furthermore, it is immediate from (3.7.4) that $g_\theta(\mathbf{x})$ is independent of $\mathbf{x}$; that is, g_θ is a constant function, so the average reward is independent of the initial state. Henceforth, g_θ will be regarded as a scalar quantity in this section.

The quantity g_θ is sometimes referred to as the *gain* of the stationary policy θ. Puterman [141] (p. 334) explains that this term originates from control engineering, in which it refers to "the ratio of the output of a system to its input". The next result introduces a functional equation which can be used to compute the gain, or average reward, of a fixed policy in $\hat{\Upsilon}$.

*** Theorem 3.7.2.** *Consider a fixed stationary policy θ. Suppose there exists a scalar quantity g and a function $h : \tilde{S} \to \mathbb{R}$, where $\tilde{S}$ is the finite state space, such that:*

$$g + h(\mathbf{x}) = r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p_\theta(\mathbf{x}, \mathbf{y}) h(\mathbf{y}) \qquad \forall\, \mathbf{x} \in \tilde{S}. \tag{3.7.5}$$

Then $g = g_\theta$, i.e. g is the average reward associated with the policy θ.

Proof. The simplest method of proof, as given in Bertsekas [12] (p. 334), is to consider a problem where, for each state $\mathbf{x} \in \tilde{S}$, the action set $A_{\mathbf{x}}$ is replaced by $\{\theta(\mathbf{x})\}$; that is, $A_{\mathbf{x}}$ consists of only a single element corresponding to the action prescribed by the stationary policy θ. Then Theorem 3.7.2 actually follows as a corollary from Theorem 3.7.3 (to be stated shortly). □

The result of Theorem 3.7.2 can be used to evaluate the performance of a fixed policy. Indeed, (3.7.5) is a system of $|\tilde{S}|$ equations in $|\tilde{S}| + 1$ unknowns, namely the set of $h(\mathbf{x})$ values and the average reward g itself. One may impose the extra condition $h(\mathbf{z}) = 0$, where $\mathbf{z} \in \tilde{S}$ is an arbitrary 'reference state' ($\mathbf{z} = \mathbf{0}$ is the simplest choice), in order to obtain a unique solution. In principle, one may use Gaussian elimination or similar techniques to solve (3.7.5); however, if $|\tilde{S}|$ is large, it is somewhat easier computationally to use an algorithmic approach. The *Policy Evaluation Algorithm* detailed below has its origins in the early MDP literature; see, for example, [88, 195].

Policy Evaluation Algorithm (PEA)

1. Input θ, the stationary policy to be evaluated.

2. Set $n = 0$, $h_0(\mathbf{x}) = 0$ and $w_0(\mathbf{x}) = 0$ for all $\mathbf{x} \in \tilde{S}$.

3. For each $\mathbf{x} \in \tilde{S}$, let $w_{n+1}(\mathbf{x}) = r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h_n(\mathbf{y})$.

4. Set $h_{n+1}(\mathbf{x}) = w_{n+1}(\mathbf{x}) - w_{n+1}(\mathbf{z})$ for all $\mathbf{x} \in \tilde{S}$, where $\mathbf{z} \in \tilde{S}$ is a fixed 'reference state'.

5. If $n = 0$, set $\delta = 1$; otherwise set $\delta = \max_{\mathbf{x} \in \tilde{S}} |h_{n+1}(\mathbf{x}) - h_n(\mathbf{x})|$. If $\delta < \epsilon$, where ϵ is a small positive number, go to step 6; otherwise, increment n by 1 and return to step 3.

6. Output the value $h_n(\mathbf{x})$ for each $\mathbf{x} \in \tilde{S}$. The average reward g_θ under policy θ can be evaluated by putting $\mathbf{x} = \mathbf{0}$ in (3.7.5). Then, since $r(\mathbf{0}) = 0$ and $h_n(\mathbf{0}) = 0$:

$$g_\theta = \lambda \Delta h_n(\mathbf{0}^{a+}),$$

where $a = \theta(\mathbf{0})$, i.e. a is the action chosen by θ at state $\mathbf{0}$.

The policy evaluation equations in (3.7.5) may be employed in a creative manner to compute various performance measures for the system. Indeed, consider two reward functions $r_L(\mathbf{x})$ and $r_W(\mathbf{x}, a)$ (where only the latter has an action-dependence), defined as follows:

$$r_L(\mathbf{x}) := \sum_{i=1}^{N} x_i,$$

$$r_W(\mathbf{x}, a) = \begin{cases} 1/\mu_i, & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } x_i < c_i, \\ (x_i + 1)/(c_i \mu_i), & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } x_i \geq c_i, \\ 0, & \text{otherwise.} \end{cases}$$

Then, given that $r_L(\mathbf{x})$ represents the total number of customers present in the system under state $\mathbf{x}$, replacing $r(\mathbf{x})$ with $r_L(\mathbf{x})$ in (3.7.5) enables g_θ to measure the average number of customers present in the system under policy θ. Similarly, replacing $r(\mathbf{x})$ with $r_W(\mathbf{x}, \theta(\mathbf{x}))$ in (3.7.5) enables g_θ to measure the average time spent by customers in the system (taking into account customers who balk and thereby spend no time in the system). One might even extend this notion by introducing a family of $|\tilde{S}|$ reward functions $r_{\mathbf{y}}(\mathbf{x})$ (one function for each $\mathbf{y} \in \tilde{S}$), defined by:

$$r_{\mathbf{y}}(\mathbf{x}) := \begin{cases} 1, & \text{if } \mathbf{x} = \mathbf{y}, \\ 0, & \text{otherwise.} \end{cases}$$

Then replacing $r(\mathbf{x})$ with $r_{\mathbf{y}}(\mathbf{x})$ in (3.7.5) enables g_θ to measure the stationary probability $\pi_\theta(\mathbf{y})$ under policy θ. One might modify the steps in the Policy Evaluation Algorithm so that each state $\mathbf{y} \in \tilde{S}$ has its own sets of values $v_{\mathbf{y}}(\mathbf{x})$, $w_{\mathbf{y}}(\mathbf{x})$ and $h_{\mathbf{y}}(\mathbf{x})$ which are updated on each iteration, then a set of values $g_{\mathbf{y}}(\mathbf{x})$ is output at the end of the algorithm which gives the entire stationary distribution $\{\pi_\theta(\cdot)\}$; however, it is easy to see that this is somewhat cumbersome.

The Policy Evaluation Algorithm considers only a *fixed* stationary policy. The next topic to be discussed is *optimality*. Recall that average reward optimal policies have already been defined in Section 3.4 (see Definition 3.4.1). The next result introduces the *average reward optimality equations*, which form a theoretical basis for various algorithms to be discussed later.

*** Theorem 3.7.3.** *Suppose there exists a scalar quantity g^* and a function $h : \tilde{S} \to \mathbb{R}$ such that:*

$$g^* + h(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h(\mathbf{y}) \right\} \qquad \forall\, \mathbf{x} \in \tilde{S}. \tag{3.7.6}$$

Let θ^* be a stationary policy such that, for all states $\mathbf{x} \in \tilde{S}$, the action $\theta^*(\mathbf{x})$ chosen by θ^* attains the maximum in (3.7.6). Then the stationary policy θ^* is average reward optimal for the process $\hat{\Upsilon}$, and $g_{\theta^*}(\mathbf{x}) = g^*$ for every $\mathbf{x} \in \tilde{S}$, i.e. g^* is the optimal average reward.

Proof. This result is found in many texts. See, for example, Bertsekas [12] (p. 332).

Clearly, it is desirable to establish sufficient conditions for the existence of a constant g^* and function h satisfying the average reward optimality equations (3.7.6). Several such conditions have been proposed in the literature, some of which can easily be shown to hold in the case of the finite-state MDP $\tilde{\Upsilon}$. For example, since the state $\mathbf{0}$ is accessible from any state in $\tilde{S}$, there cannot be two disjoint, closed communicating classes under any stationary policy; it then follows that any stationary policy induces a Markov chain which is *unichain* (see Appendix A.1) and the existence of a solution to (3.7.6) then follows from Proposition 2.6 in [13] (p. 198). For more examples of sufficient existence conditions, see [12] (pp. 335-340), [13] (p. 198) or [141] (p. 358).

It is possible to take a somewhat more constructive approach to establishing the existence of a solution to (3.7.6). A popular technique for computing average reward optimal policies in finite-state MDPs is known as *relative value iteration*; this technique was first proposed in undiscounted problems by White [195] (see also [12, 47]). The details of the *Relative Value Iteration Algorithm*

(abbreviated as RVIA) are given below. A proof that the algorithm converges to a solution to the optimality equations (3.7.6) when applied to a finite-state MDP will be cited later.

Relative Value Iteration Algorithm (RVIA)

1. Set $n = 0$, $h_0(\mathbf{x}) = 0$ and $w_0(\mathbf{x}) = 0$ for all $\mathbf{x} \in \tilde{S}$.
2. For each $\mathbf{x} \in \tilde{S}$, let $w_{n+1}(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h_n(\mathbf{y}) \right\}$.
3. Set $h_{n+1}(\mathbf{x}) = w_{n+1}(\mathbf{x}) - w_{n+1}(\mathbf{z})$ for all $\mathbf{x} \in \tilde{S}$, where $\mathbf{z} \in \tilde{S}$ is a fixed 'reference state'.
4. If $n = 0$, set $\delta = 1$; otherwise set $\delta = \max_{\mathbf{x} \in \tilde{S}} \left| h_{n+1}(\mathbf{x}) - h_n(\mathbf{x}) \right|$. If $\delta < \epsilon$, where ϵ is a small positive number, go to step 5; otherwise, increment n by 1 and return to step 2.
5. Output $h_n(\mathbf{x})$ for all $\mathbf{x} \in \tilde{S}$ and a stationary policy θ^* such that, for each $\mathbf{x} \in \tilde{S}$, $\theta^*(\mathbf{x})$ maximises $\left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h_n(\mathbf{y}) \right\}$ over all actions $a \in A_{\mathbf{x}}$. The optimal average reward g^* can be evaluated by putting $\mathbf{x} = 0$ in (3.7.6). Then, since $r(\mathbf{0}) = h_n(\mathbf{0}) = 0$:

$$g^* = \lambda \Delta h_n(\mathbf{0}^{a+}),$$

where $a = \theta^*(\mathbf{0})$, i.e. a is the action chosen by θ^* at state $\mathbf{0}$.

The subtraction of $w_{n+1}(\mathbf{z})$ on step 3 of the algorithm ensures that the $h_n(\cdot)$ and $w_n(\cdot)$ values remain bounded, rather than diverging to infinity. The choice of the reference state $\mathbf{z} \in \tilde{S}$ is arbitrary, but in the case of the MDP $\tilde{\Upsilon}$, taking $\mathbf{z} = \mathbf{0}$ will be most convenient. Note that the Policy Evaluation Algorithm on page 98 is essentially a modified version of the RVIA which considers only one allowable action (dependent on the fixed policy θ) at each state $\mathbf{x} \in \tilde{S}$. The next result guarantees the convergence of the RVIA, and also provides a constructive proof of the existence of a constant g^* and function h satisfying the average reward optimality equations (3.7.6).

*** Theorem 3.7.4.** *Consider the MDP $\tilde{\Upsilon}$ with finite state space $\tilde{S}$. Let $h_0(\mathbf{x}) = 0$ for all $\mathbf{x} \in \tilde{S}$. For each state $\mathbf{x} \in \tilde{S}$ and integer $k \geq 0$, let $h_{k+1}(\mathbf{x})$ be defined as follows*:

$$h_{k+1}(\mathbf{x}) := \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h_k(\mathbf{y}) \right\} - \max_{a \in A_{\mathbf{0}}} \left\{ r(\mathbf{0}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{0}, a, \mathbf{y}) h_k(\mathbf{y}) \right\}. \qquad (3.7.7)$$

Then, for each $\mathbf{x} \in \tilde{S}$*, the following limit exists:*

$$h(\mathbf{x}) := \lim_{k\to\infty} h_k(\mathbf{x}).$$

Moreover, let g^* *be a scalar quantity defined by*:

$$g^* := \max_{a\in A_{\mathbf{0}}} \left\{ r(\mathbf{0}) + \sum_{\mathbf{y}\in\tilde{S}} p(\mathbf{0}, a, \mathbf{y}) h(\mathbf{y}) \right\}.$$

The constant g^* *and set of values* $\left\{h(\mathbf{x})\right\}_{\mathbf{x}\in\tilde{S}}$ *satisfy the average reward optimality equations*:

$$g^* + h(\mathbf{x}) = \max_{a\in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y}\in\tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h(\mathbf{y}) \right\} \qquad \forall\, \mathbf{x} \in \tilde{S}. \tag{3.7.8}$$

Furthermore, the solution obtained is the unique *solution to (3.7.8) with* $h(\mathbf{0}) = 0$*.*

Proof. This result has been proved by Bertsekas [13] (p. 206) (see also [62, 141, 182]). However, the proof relies upon an additional assumption which should be discussed here.

Let $\theta = (d_0, d_1, ...)$ denote an arbitrary policy, where d_n $(n = 0, 1, 2, ...)$ is the decision rule used at the n^{th} discrete time epoch, and let P_{d_n} denote the transition matrix associated with the rule d_n. The assumption relied upon by the proof is that there exists an integer $m > 0$ and a positive number $\epsilon > 0$ such that, for all admissible policies θ and all states $\mathbf{x} \in \tilde{S}$:

$$\begin{aligned} \left(P_{d_m} P_{d_{m-1}} ... P_{d_1}\right)_{\mathbf{x0}} &\geq \epsilon, \\ \left(P_{d_{m-1}} P_{d_{m-2}} ... P_{d_0}\right)_{\mathbf{x0}} &\geq \epsilon, \end{aligned} \tag{3.7.9}$$

where $P_{\mathbf{x0}}$ denotes the element in the row corresponding to state $\mathbf{x}$ and the column corresponding to state $\mathbf{0}$ of the matrix P. Indeed, given a finite state space $\tilde{S}$ with the property that for each $i \in \{1, 2, ..., N\}$ there exists $B_i \in \mathbb{N}_0$ such that $x_i \leq B_i$ for all $\mathbf{x} \in \tilde{S}$, it is sufficient to take $m = \sum_{i=1}^{N} B_i$ in order to establish (3.7.9). To see this, note that at each state $\mathbf{x} \in \tilde{S} \setminus \{\mathbf{0}\}$ there is a positive probability of a service completion. Moreover, according to the transition probabilities formulated in (3.5.3), if the system is in state $\mathbf{0}$ at any time step, then there is a probability $(1 - \lambda\Delta) \geq \sum_{i=1}^{N} c_i \mu_i \Delta$ that it remains in state $\mathbf{0}$ at the next step. Suppose the process is initialised in an arbitrary state $\mathbf{x} \in \tilde{S}$ and an unbroken sequence of service completions occurs on consecutive time steps, causing the process to reach state $\mathbf{0}$, *and* no further arrivals occur until a total of m

time steps have elapsed (since initialisation). By the previous arguments, the probability of this chain of events occurring is at least $(\mu_{\min}\Delta)^m$, where $\mu_{\min}$ is the minimum of the N service rates. (Note that the decision rules $d_0, d_1, ..., d_m$ are irrelevant to this argument, since it considers a chain of events in which no arrivals occur, therefore the actions chosen have no effect.)

It follows that for all $\mathbf{x} \in \tilde{S}$, the conditions (3.7.9) can be satisfied by choosing:

$$\begin{aligned} m &:= \sum_{i=1}^{N} B_i, \\ \epsilon &:= (\mu_{\min}\Delta)^m. \end{aligned} \tag{3.7.10}$$

Bertsekas [13] (p. 213) proves that the conditions (3.7.9) hold in the more general case of a finite-state MDP where every stationary policy is unichain. His arguments involve making a 'data transformation' which effectively ensures that $p(\mathbf{x}, a, \mathbf{x}) > 0$ for all $\mathbf{x} \in \tilde{S}$ and $a \in A_{\mathbf{x}}$; i.e. each state in $\tilde{S}$ has a positive probability of a 'self-transition', regardless of the action chosen. The aforementioned data transformation can be made without fundamentally altering the original problem, since it transpires that the average reward under any stationary policy is unaffected. In the case of the process $\tilde{\Upsilon}$ considered in this section, it is actually possible to ensure that $p(\mathbf{x}, a, \mathbf{x}) > 0$ for all $\mathbf{x} \in \tilde{S}$ and $a \in A_{\mathbf{x}}$ by simply choosing the uniformisation parameter Δ to be any positive number *smaller* than $\left(\lambda + \sum_{i=1}^{N} c_i\mu_i\right)^{-1}$; in view of (3.5.3), this achieves exactly the same effect as the data transformation discussed in [13]. Moreover, Theorem 3.3.3 implies that the average reward under any stationary policy is unaffected by reducing the value of Δ in this way.

For details of the remainder of the proof, please refer to [13] (p. 206). □

By Theorems 3.7.3 and 3.7.4, it follows that that the stationary policy obtained in step 5 of the RVIA is average reward optimal. Note that although the values g^* and $h(\mathbf{x})$ which satisfy (3.7.6) (assuming the extra condition $h(\mathbf{0}) = 0$) are unique, a stationary average reward optimal *policy* θ^* need not be unique; indeed, there may be more than one action which attains the maximum on the right-hand side of (3.7.6) for a particular state $\mathbf{x} \in \tilde{S}$. The next example illustrates how comparisons may be drawn between two different average reward optimal policies.

Example 3.7.5. ***(Bias optimality)***

Consider a single-facility system ($N = 1$) with only one service channel, a demand rate $\lambda = 1$, a

service rate $\mu = 2$ and holding cost rate $\beta = 5$. In keeping with the assumption of a finite state space, a constraint will be imposed whereby joining the queue is not allowed at any state $x \in \mathbb{N}_0$ with $x \geq 5$. As a result, the state space may be described by the set $\tilde{S} = \{0, 1, ..., 5\}$, with two possible actions (joining or balking) permitted at every state except $x = 5$.

Given that the system is $M/M/1$, the results from Section 2.3 imply that an average reward optimal policy (referred to as a *socially* optimal policy in Chapter 2) is characterised by a threshold n_o such that joining the queue is chosen at a state $x \in \tilde{S}$ if and only if $x < n_o$. Furthermore, (2.3.3) states that a socially optimal threshold n_o is given by $\lfloor v_o \rfloor$, where v_o satisfies:

$$\frac{v_o(1-\rho) - \rho(1-\rho^{v_o})}{(1-\rho)^2} = \frac{\alpha\mu}{\beta}, \tag{3.7.11}$$

where $\rho = \lambda/\mu = 1/2$. Naor's analysis [131] shows that although the value of v_o satisfying (3.7.11) will be unique (with the other parameters fixed), if v_o is a positive integer then the thresholds v_o and $v_o - 1$ will both be socially optimal. As such, it is possible to manipulate the value of α in order to ensure the existence of non-unique optimal policies. For example, setting $\alpha = 645/32$ leads to a solution $v_o = 5$ in (3.7.11), and hence the thresholds 4 and 5 are both optimal.

Let θ_4 and θ_5 denote the stationary policies with thresholds 4 and 5 respectively, and also let it be assumed that $\alpha = 645/32$ and the system is formulated as an MDP with reward function $r(x)$ defined in (3.5.4) and transition probabilities given by (3.5.3) with $\Delta = (\lambda + \mu)^{-1}$. Using relative value iteration (page 101), one may obtain the optimal average reward $g^* = 245/16$ and the unique set of values $h(x)$ (with $h(0) = 0$) satisfying the average reward optimality equations (3.7.6); these values are shown in Table 3.1. The policies θ_4 and θ_5 differ only at the state $x = 4$, and it may be checked that both of the actions $a = 0$ (balking) and $a = 1$ (joining) maximise the expression $r(x) + \sum_{y \in \tilde{S}} p(x, a, y)h(y)$ at $x = 4$. Moreover, both policies maximise the same expression at all other states $x \in \tilde{S}$; that is, they both satisfy the optimality equations as expected.

State x	0	1	2	3	4	5
$h(x)$	0.0000	45.9375	77.8125	96.5625	104.0625	104.0625

Table 3.1: Values of $h(x)$ for states $x \in \{0, 1, ..., 5\}$ with the reward formulation $r(x)$ used.

Naturally, given the relationship between the RVIA and PEA algorithms discussed earlier, applying

the Policy Evaluation Algorithm (page 98) to either of the two policies θ_4 and θ_5 yields the same set of $h(x)$ values shown in Table 3.1. Given that both of the policies θ_4 and θ_5 yield the same average reward and the same set of relative values $h(x)$, one might be tempted to conclude that the two policies are indistinguishable with respect to any performance measure of interest (assuming undiscounted rewards); in other words, there is no way to rank one policy above the other. However, the purpose of this example is to show that the more conservative policy θ_4 is preferable to θ_5 with respect to a criterion which might, in practice, be of interest to a decision-maker.

The expected total reward $V_\theta^{(n)}(x)$ over n stages under an arbitrary policy θ and initial state $x \in \mathbb{N}_0$ was defined in (3.7.3). One may calculate $V_\theta^{(n)}(x)$ for $n \geq 0$ recursively, using:

$$V_\theta^{(n)}(x) = \begin{cases} 0, & \text{if } n = 0, \\ r(x) + \sum_{y \in \tilde{S}} p_\theta(x, y) V_\theta^{(n-1)}(y), & \text{if } n \geq 1. \end{cases}$$

The expected average reward over n stages is then given by $V_\theta^{(n)}(x)/n$ for $n \geq 1$. Figure 3.6 shows the finite-stage average rewards $V_\theta^{(n)}(x)/n$ plotted under both of the policies θ_4 and θ_5 for $n \leq 50$, assuming that the initial state is $x = 4$ (the only state at which the two policies differ). Although both of these average rewards converge towards the optimal value g^* as $n \to \infty$, it transpires that the average reward under θ_4 is *always strictly greater* than the average reward under θ_5 over any finite number of stages $n \geq 1$. This can be explained intuitively. When joining is chosen by θ_5 at state $x = 4$, the process incurs an extra holding cost which would *not* be incurred under the alternative policy θ_4. Thus, θ_4 earns an advantage over θ_5 which is not 'equalised' by θ_5 until, at some point in the future, the process operating under θ_5 has a service in progress which would not have taken place if the policy θ_4 had been followed instead. In other words, θ_4 consistently earns advantages over θ_5 which are not 'equalised' by θ_5 until a later point in time, or (in a sense) θ_5 is constantly trying to 'catch up' with θ_4 with respect to expected total reward. Since θ_5 can never catch up with θ_4 over all possible random trajectories, it makes sense that the expected average reward over a finite number of stages is always strictly greater under θ_4 than under θ_5.

Given that θ_4 is strictly superior to θ_5 over any finite number of stages, one would imagine that a decision-maker would prefer to follow the policy θ_4 if they were interested in optimising the system's performance over a *finite* amount of time (which, realistically, would always be the case in practice!).

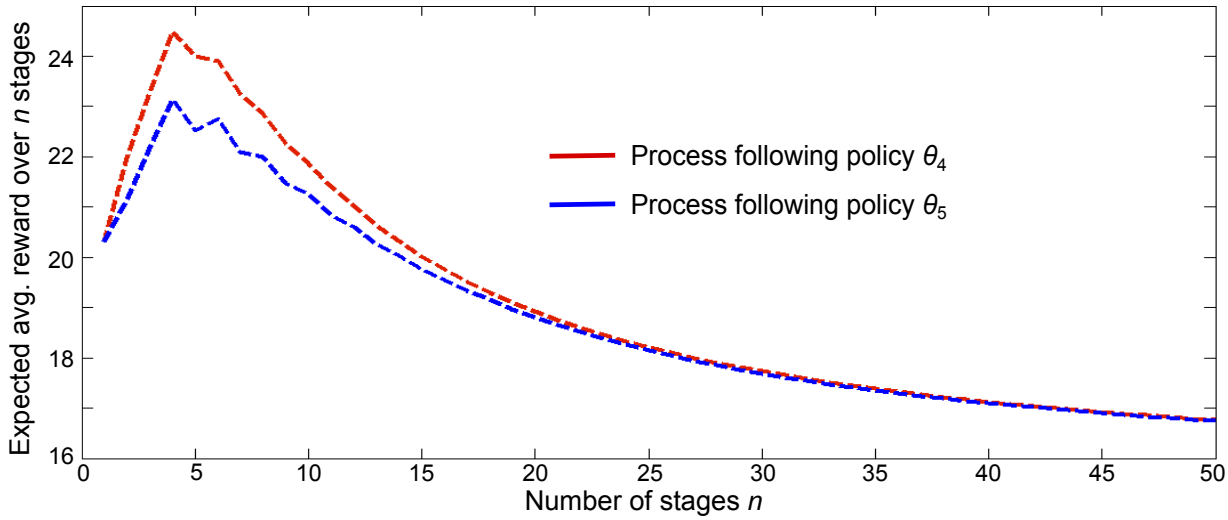


Figure 3.6: Expected average rewards over n stages under the policies θ_4 and θ_5 with initial state $x = 4$, assuming that the 'real-time' reward formulation $r(x) = \alpha\mu \min(x, 1) - \beta x$ is used.

However, the fact remains that both policies are average reward optimal according to Definition 3.4.1. In order to differentiate between θ_4 and θ_5 with respect to performance, the stronger concept of *bias optimality* is needed. In general, suppose one has an average reward optimal stationary policy θ^*. The *bias* of θ^*, given an initial state $x \in \tilde{S}$ (see [78, 117]), is given by:

$$b_{\theta^*}(x) := \lim_{n\to\infty} E_{\theta^*}\left[\sum_{t=0}^{n-1} (r(x_n) - g^*) \,\middle|\, x_0 = x\right].$$

Hence, the bias corresponds to the expected (finite) total reward in a new process where the optimal gain g^* is subtracted from each single-stage reward $r(x_n)$. An average reward optimal policy θ_0^* is said to be *bias optimal* if, for all other average reward optimal policies θ^*:

$$b_{\theta_0^*}(x) \geq b_{\theta^*}(x) \quad \forall\, x \in \tilde{S}.$$

Suitable conditions which imply that a particular average reward optimal policy is also bias optimal have been discussed in the literature; see [78] (p. 146). The easiest way to compute the bias values $b_{\theta^*}(x)$ for an average reward optimal policy θ^* is to use the fact that the function $b_{\theta^*} : \tilde{S} \to \mathbb{R}$ must satisfy, together with some other function $k_{\theta^*} : \tilde{S} \to \mathbb{R}$, all of the following equations:

$$\begin{aligned} b_{\theta^*}(x) &= r(x) - g^* + \sum_{y\in\tilde{S}} p_{\theta^*}(x,y) b_{\theta^*}(y) \quad \forall\, x \in \tilde{S}, \\ k_{\theta^*}(x) &= -b_{\theta^*}(x) + \sum_{y\in\tilde{S}} p_{\theta^*}(x,y) k_{\theta^*}(y) \quad \forall\, x \in \tilde{S}. \end{aligned} \tag{3.7.12}$$

In fact, for a given average reward optimal policy θ^* there will be infinitely many sets of values $\{k_{\theta^*}(x)\}_{x\in\tilde{S}}$ which satisfy (3.7.12), so it is necessary to impose an extra condition such as $k_{\theta^*}(0) = 0$ in order to obtain an unique solution. However, the bias values $b_{\theta^*}(x)$ which satisfy (3.7.12) will always be unique. Table 3.2 shows the values $b_{\theta_4}(x)$ and $b_{\theta_5}(x)$ for the optimal policies θ_4 and θ_5 respectively. Note that, since the functions $b_{\theta_4}(\cdot)$ and $b_{\theta_5}(\cdot)$ both satisfy the first set of equations in (3.7.12), it must be the case that they differ from the relative value function $h(\cdot)$ only by an additive constant. Indeed, comparing Tables 3.1 and 3.2 confirms that this is true.

State x	0	1	2	3	4	5
$b_{\theta_4}(x)$	-31.4819	14.4556	46.3306	65.0806	72.5806	72.5806
$b_{\theta_5}(x)$	-32.6339	13.3036	45.1786	63.9286	71.4286	71.4286

Table 3.2: Values of $b_{\theta_4}(x)$ and $b_{\theta_5}(x)$ for states $x \in \{0, 1, ..., 5\}$ with reward formulation $r(x)$ used.

Table 3.2 shows that $b_{\theta_4}(x) > b_{\theta_5}(x)$ for all $x \in \{0, 1, ..., 5\}$. Since it can easily be verified that θ_4 and θ_5 are the only stationary average reward optimal policies in this example, it follows that θ_4 is bias optimal, whereas θ_5 is not. However, the most interesting aspect of this example is that the same conclusion does *not* hold when the reward function $r(x)$ is replaced by the alternative function $\hat{r}(x, a)$, previously referred to as the *anticipatory* reward function. Indeed, when the alternative function $\hat{r}(x, a)$ is used, the process earns a positive reward by allowing a customer to join at state $x = 4$, whereas a zero reward is earned by balking. Hence, the policy θ_5 earns an advantage over θ_4 by allowing an extra customer to join, but this is compensated for by the fact that customers who join the queue at later points in time suffer increased waiting times (and hence, the process earns smaller single-stage rewards) as a result of an extra customer being present in the system. In a sense, therefore, switching to the alternative reward formulation $\hat{r}(x, a)$ reverses the roles of θ_4 and θ_5 by allowing θ_5 to earn an advantage over θ_4 at state $x = 4$ and then forcing the policy θ_4 to 'play catch-up' to θ_5 with respect to the expected total reward earned over n stages.

Figure 3.7 shows the expected finite-stage average rewards over 50 stages under the policies θ_4 and θ_5, with the reward function $\hat{r}(x, a)$ used instead of $r(x)$. In this case, θ_5 earns a strictly greater expected total reward than θ_4 over any finite number of stages $n \geq 1$, although the difference tends to zero as $n \to \infty$. By Theorem 3.5.3, the optimal average reward g^* is unaffected by switching from

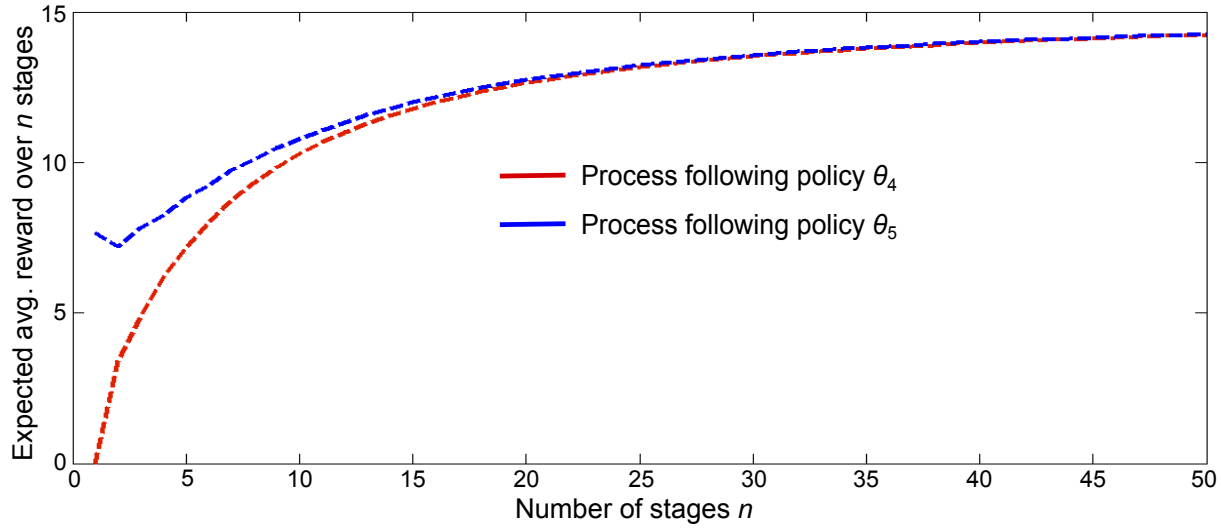


Figure 3.7: Expected average rewards over n stages under the policies θ_4 and θ_5 with initial state $x = 4$, assuming that the 'real-time' reward formulation $\hat{r}(x, a) = I(a = 1)\,(\alpha - \beta(x + 1)/\mu)$ is used.

$r(x)$ to $\hat{r}(x, a)$, but the relative values satisfying (3.7.6) *are* affected and so it will be appropriate to denote these by $\hat{h}(x)$ in the case of anticipatory rewards. Similarly, the bias functions under the policies θ_4 and θ_5 will be denoted by $\hat{b}_{\theta_4}(\cdot)$ and $\hat{b}_{\theta_5}(\cdot)$ respectively. Table 3.3 shows the values of $\hat{h}(x)$, (with $\hat{h}(0) = 0$), $\hat{b}_{\theta_4}(x)$ and $\hat{b}_{\theta_5}(x)$ for $x \in \{0, 1, ..., 5\}$. It can be seen that $\hat{b}_{\theta_4}(x) < \hat{b}_{\theta_5}(x)$ for all $x \in \tilde{S}$, confirming that the policy θ_5 is bias optimal in this case, whereas θ_4 is not.

State x	0	1	2	3	4	5
$\hat{h}(x)$	0.00000	-7.03125	-20.62500	-39.84375	-52.81250	-85.78125
$\hat{b}_{\theta_4}(x)$	9.07258	2.04133	-11.55242	-30.77117	-53.73992	-76.70867
$\hat{b}_{\theta_5}(x)$	10.29018	3.25893	-10.33482	-29.55357	-52.52232	-75.49107

Table 3.3: Values of $\hat{h}(x)$, $\hat{b}_{\theta_4}(x)$ and $\hat{b}_{\theta_5}(x)$ for $x \in \{0, 1, ..., 5\}$ with reward formulation $\hat{r}(x, a)$ used.

The conclusion of this example is that, although any policy which is average reward optimal for the MDP $\hat{\Upsilon}$ under reward formulation $r(x)$ must also be likewise optimal under the formulation $\hat{r}(x, a)$ due to Theorem 3.5.3, this equivalence does not necessarily hold when a more sensitive optimality criterion such as bias optimality is considered. Indeed, one may find two average reward optimal policies such that only one is bias optimal, with the identity of the bias optimal policy depending on the reward formulation used. The results of the present example appear to suggest that a

more conservative policy will be favoured by the bias optimality criterion when the function $r(x)$ is used, whereas the converse statement should apply when $\hat{r}(x, a)$ is used. Indeed, some results have been published in the literature which lend support to this notion, although the results in question involve slightly different problem formulations; see [78, 116] for further details. ⊠

Another well-established method for computing average reward optimal policies in finite-state MDPs is known as *policy improvement* (or sometimes *policy iteration*). This method was introduced by Howard [88], although a similar method called "approximation in policy space" was also described by Bellman [10]. Essentially, one begins with an arbitrary stationary policy θ, and then finds corresponding values g_θ and $h_\theta(\mathbf{x})$ which satisfy the evaluation equations in (3.7.5). Each action $\theta(\mathbf{x})$ prescribed by θ is then checked to see whether it maximises the expression $r(\mathbf{x}) + \sum_{\mathbf{y}\in\tilde{S}} p(\mathbf{x}, a, \mathbf{y})h_\theta(\mathbf{y})$ over all permissible actions $a \in A_{\mathbf{x}}$. If this test is passed for all $\mathbf{x} \in \tilde{S}$, then by Theorem 3.7.3, θ is average reward optimal; otherwise, the policy is modified and the procedure is repeated until eventually an optimal policy is obtained. The *Policy Improvement Algorithm*, in the form presented by Puterman [141] (p. 378), is given as follows:

Policy Improvement Algorithm (PIA)

1. Set $n = 0$ and select an arbitrary stationary policy θ_0.

2. (*Policy evaluation.*) Find a scalar g_n and a function h_n which satisfy the equations:

$$g_n + h_n(\mathbf{x}) = r(\mathbf{x}) + \sum_{\mathbf{y}\in\tilde{S}} p_{\theta_n}(\mathbf{x}, \mathbf{y})h_n(\mathbf{y}) \qquad \forall\, \mathbf{x} \in \tilde{S}, \tag{3.7.13}$$

   together with the extra condition $h_n(\mathbf{0}) = 0$. (Note: The Policy Evaluation Algorithm given on page 98 may be used to perform this step.)

3. (*Policy improvement.*) For each $\mathbf{x} \in \tilde{S}$, choose an action $\theta_{n+1}(\mathbf{x})$ to satisfy:

$$\theta_{n+1}(\mathbf{x}) \in \operatorname*{arg\,max}_{a\in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y}\in\tilde{S}} p(\mathbf{x}, a, \mathbf{y})h_n(\mathbf{y}) \right\}, \tag{3.7.14}$$

   setting $\theta_{n+1}(\mathbf{x}) = \theta_n(\mathbf{x})$ if possible.

4. If $\theta_{n+1}(\mathbf{x}) = \theta_n(\mathbf{x})$ for all $\mathbf{x} \in \tilde{S}$ then stop; the policy θ_{n+1} is average reward optimal. Otherwise, increment n by 1 and return to step 2.

The specification that, if possible, $\theta_{n+1}(\mathbf{x})$ should be chosen to be equal to $\theta_n(\mathbf{x})$ is made to avoid cycling between policies which yield the same average reward. The algorithm is designed to terminate if it finds *any* policy which is optimal, rather than cycling between multiple optimal policies. The convergence of the PIA is ensured by the following result.

*** Theorem 3.7.6.** *The Policy Improvement Algorithm, applied to the finite-state MDP $\tilde{\Upsilon}$, converges in a finite number of iterations to a solution of the optimality equations*:

$$g^* + h(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h(\mathbf{y}) \right\} \qquad \forall\, \mathbf{x} \in \tilde{S}, \tag{3.7.15}$$

and an average reward optimal stationary policy θ^*.

Proof. Please refer to Bertsekas [13] (p. 214), or Puterman [141] (p. 383).

Example 3.7.7. ***(Policy improvement)***

This example revisits the two-facility system in Example 3.2.1 in order to demonstrate how the Policy Improvement Algorithm (PIA) can be used to find an average reward optimal policy. Recall that the demand rate is $\lambda = 1$, there are two service facilities with one service channel each, and the service rates, holding costs and fixed rewards for the two facilities are given by:

$$\begin{aligned} &\mu_1 = 0.8, \qquad \beta_1 = 2, \qquad \alpha_1 = 6, \\ &\mu_2 = 0.4, \qquad \beta_2 = 1, \qquad \alpha_2 = 4. \end{aligned}$$

The state space S consists of only 6 states, and the stationary policy θ operates as follows:

$$\begin{aligned} &\theta((0,0)) = 1, \qquad \theta((0,1)) = 1, \qquad \theta((1,0)) = 2, \\ &\theta((1,1)) = 1, \qquad \theta((2,0)) = 2, \qquad \theta((2,1)) = 0. \end{aligned}$$

Let the system be uniformised as described in Section 3.3, with $\Delta = (\lambda + \mu_1 + \mu_2)^{-1} = 5/11$. After initialising the PIA with $\theta_0 = \theta$, using the condition $h((0,0)) = 0$ enables the system of equations (3.7.13) to be solved uniquely. The solution, with figures rounded to 4 d.p., is:

$$\begin{aligned} g_0 = 1.6913, \qquad & h_0((0,0)) = 0, \qquad h_0((0,1)) = 1.3268, \qquad h_0((1,0)) = 3.7209, \\ & h_0((1,1)) = 4.2584, \qquad h_0((2,0)) = 2.2641, \qquad h_0((2,1)) = 3.0596. \end{aligned}$$

Consider the state $(0,0)$. In step 3 of the PIA, It is necessary to identify the action $a \in A_{(0,0)}$ which maximises $r((0,0)) + \sum_{\mathbf{y}} p((0,0),a,\mathbf{y})h_0(\mathbf{y})$. The reward $r((0,0))$ is independent of the action chosen, so (enumerating the transition probabilities) a must satisfy:

$$a \in \arg\max_{a \in A_{(0,0)}} \left\{ (1-\lambda\Delta)h_0((0,0)) + \lambda\Delta h_0\big((0,0)^{a+}\big) \right\},$$

Since $\max\left(h_0((0,0)),\ h_0((0,1)),\ h_0((1,0))\right) = h_0((1,0))$, it transpires that the action $a=1$ chosen by policy θ_0 at $\mathbf{x}=(0,0)$ attains the maximum in step 3. Similarly, it can be checked that the actions chosen by θ_0 at states $(0,1)$, $(1,0)$, $(2,0)$ and $(2,1)$ pass the optimality test; hence, the new policy θ_1 derived in step 3 chooses the same actions as θ_0 at each of these states. On the other hand, checking the permissible actions $a \in A_{(1,1)} = \{0,1\}$ at the state $(1,1)$, one finds:

$$\begin{aligned}
&\max_{a \in A_{(1,1)}} \left\{ r((1,1)) + \sum_{\mathbf{y}} p((1,1),a,\mathbf{y})h_0(\mathbf{y}) \right\} \\
&= r((1,1)) + \mu_1\Delta h_0((0,1)) + \mu_2\Delta h_0((1,0)) + \lambda\Delta \max\left(h_0((1,1)), h_0((2,1))\right),
\end{aligned}$$

and since $h_0((1,1)) > h_0((2,1))$, the maximum is attained by the action $a=0$; that is, the new policy θ_1 differs from θ_0 by choosing to balk at $\mathbf{x}=(1,1)$. Replacing the policy θ_0 with θ_1 and re-solving the equations in (3.7.13) (with $h_1((0,0))=0$) yields the solution:

$$\begin{aligned}
g_1 = 1.8772, \qquad & h_1((0,0)) = 0, \qquad && h_1((0,1)) = 1.8526, \qquad && h_1((1,0)) = 4.1298, \\
& h_1((1,1)) = 5.4035, && h_1((2,0)) = 2.4965, && h_1((2,1)) = 3.5596.
\end{aligned}$$

As expected, the average reward is greater under the new policy θ_1. Furthermore, it can be checked that each action $\theta_1(\mathbf{x})$ maximises $r(\mathbf{x}) + \sum_{\mathbf{y}} p(\mathbf{x},a,\mathbf{y})h_1(\mathbf{y})$, so by Theorem 3.7.3 the new policy θ_1 is optimal. To summarise, the original policy θ_0 differs from the optimal policy θ_1 only in the action chosen at the state $(1,1)$, and the PIA finds the optimal policy θ_1 on its first improvement step. The sub-optimality of the original policy θ_0 is given by $(g_1 - g_0)/g_1 \approx 10\%$. $\boxtimes$

The PEA, RVIA and PIA algorithms presented in this section are examples of *dynamic programming* (DP) algorithms which can be useful for theoretical purposes as well as practical ones, as later results will demonstrate. Before concluding this chapter, it will be useful to introduce a further technique which will be relied upon in later chapters; this is known as *stochastic coupling*.

3.8 Stochastic coupling

Section 3.7 introduced the technique of *relative value iteration* as a means of obtaining a constant g^* and set of values $\{h(\mathbf{x})\}_{\mathbf{x}\in\tilde{S}}$ satisfying the average reward optimality equations in (3.7.6). By Theorem 3.7.4, the values $h(\mathbf{x})$ can be obtained as limits of the finite-stage values $h_n(\mathbf{x})$ defined in (3.7.7) as $n \to \infty$. This creates the possibility of using inductive arguments based on the finite-stage functions h_n to establish properties of the limiting function h, which in turn can be used to deduce properties of average reward optimal policies and potentially gain other interesting insights into the evolution of a particular process; examples of this technique will be given in later chapters. However, the purpose of this section is to introduce an alternative method of proof, known as *stochastic coupling*, which bears certain analogies to inductive arguments based on value iteration while at the same time being somewhat more transparent and intuitively appealing.

Stochastic coupling is an important topic in probability theory, although there are relatively few texts dedicated entirely to the subject. An excellent reference is Thorisson [181], in which a broad range of applications are described. In [181], Thorisson describes a coupling as a "joint construction of two or more random variables (or processes), usually in order to deduce properties of the individual variables or gain insight into distributional similarities or relations between them." Naturally, a formal mathematical definition is possible; however, for the purposes of this thesis, it will be desirable to define a coupling in a somewhat simplified way in order to fit the requirements of the queueing systems under consideration without being unnecessarily general. The following definition introduces the notion of a stochastic coupling as a device for establishing relationships and similarities between two processes in the manner alluded to by Thorisson.

Definition 3.8.1. ***(Coupling)***

Let Φ_1 and Φ_2 be discrete-time MDPs as defined in (3.3.1), with state spaces S_1 and S_2 respectively, and suppose both processes may be uniformised with a common uniformisation parameter $\Delta > 0$. The two processes are assumed to follow arbitrary (separate) policies. Let $(\mathbf{x}_n)_{n\in\mathbb{N}_0}$ and $(\mathbf{y}_n)_{n\in\mathbb{N}_0}$ denote the state-time evolutions of Φ_1 and Φ_2 respectively; that is, $\mathbf{x}_n \in S_1$ (respectively, $\mathbf{y}_n \in S_2$) is the state of Φ_1 (respectively, Φ_2) at the n^{th} discrete time step. The initial states of Φ_1 and Φ_2 are, respectively, $\mathbf{x}_0 \in S_1$ and $\mathbf{y}_0 \in S_2$. Consider a new discrete-time Markov chain Φ_c with state space $S_1 \times S_2$, also uniformised with the same parameter Δ, whose state-time evolution is denoted

by $(\hat{\mathbf{x}}_n, \hat{\mathbf{y}}_n)_{n\in\mathbb{N}_0}$. *Furthermore, assume that the initial state of* Φ_c, $(\hat{\mathbf{x}}_0, \hat{\mathbf{y}}_0)$, *satisfies* $\hat{\mathbf{x}}_0 = \mathbf{x}_0$ *and* $\hat{\mathbf{y}}_0 = \mathbf{y}_0$. *Then* Φ_c *is said to be a* coupling *of* Φ_1 *and* Φ_2 *if, for all* $n \in \mathbb{N}_0$, *the probability distributions of* $\hat{\mathbf{x}}_n$ *and* $\hat{\mathbf{y}}_n$ *are identical to those of* $\mathbf{x}_n$ *and* $\mathbf{y}_n$ *respectively.*

For convenience, the pair $(\hat{\mathbf{x}}_n, \hat{\mathbf{y}}_n)$ will occasionally be referred to as a coupling rather than the process Φ_c itself; that is, a coupling may be referred to by its state-time evolution. Also, the terms "coupled process" and "coupling" will be used interchangeably in this section.

Assuming that Φ_1 and Φ_2 can both be uniformised as described in Section 3.3, the subsequent assumption that the same uniformisation parameter Δ can be used for both processes is non-restrictive. Indeed, suppose $\Delta_1 > 0$ and $\Delta_2 > 0$ are valid choices for the uniformisation parameters of Φ_1 and Φ_2 respectively; then, due to Theorem 3.3.3, any value $\Delta \in (0, \Delta_1)$ may also be used for the uniformisation of Φ_1, and an analogous statement applies to Φ_2. Therefore one can simply choose $\Delta := \min(\Delta_1, \Delta_2)$ as a uniformisation parameter for both processes.

Note that, although the distributions of $\hat{\mathbf{x}}_n$ and $\hat{\mathbf{y}}_n$ must be identical to those of $\mathbf{x}_n$ and $\mathbf{y}_n$ respectively for all $n \in \mathbb{N}_0$, there is no requirement for the *joint* distribution of $\hat{\mathbf{x}}_n$ and $\hat{\mathbf{y}}_n$ to be the same as that of $\mathbf{x}_n$ and $\mathbf{y}_n$. For example, given two processes Φ_1 and Φ_2, one may be able to construct a coupling $(\hat{\mathbf{x}}_n, \hat{\mathbf{y}}_n)$ satisfying Definition 3.8.1 such that the componentwise inequality $\hat{\mathbf{x}}_n \geq \hat{\mathbf{y}}_n$ is guaranteed to hold for all $n \in \mathbb{N}_0$, and yet the inequality $\mathbf{x}_n \geq \mathbf{y}_n$ may not necessarily hold with Φ_1 and Φ_2 operating independently. It is the fact that couplings may be created using artificial rules for their transition probabilities and yet still preserve the required distributional equalities that makes them such a useful tool. In practice, the equivalence between the distributions of $\hat{\mathbf{x}}_n$ and $\mathbf{x}_n$ (similarly for $\hat{\mathbf{y}}_n$ and $\mathbf{y}_n$) may be accomplished by defining the transition probabilities for the coupling in such a way that for all $\mathbf{x}, \mathbf{x}' \in S_1$, $\mathbf{y}, \mathbf{y}' \in S_2$ and stages $n \in \mathbb{N}_0$:

$$P\left(\mathbf{x}_{n+1} = \mathbf{x}' | \mathbf{x}_n = \mathbf{x}\right) = P\left(\hat{\mathbf{x}}_{n+1} = \mathbf{x}' | \hat{\mathbf{x}}_n = \mathbf{x}\right), \tag{3.8.1}$$

$$P\left(\mathbf{y}_{n+1} = \mathbf{y}' | \mathbf{y}_n = \mathbf{y}\right) = P\left(\hat{\mathbf{y}}_{n+1} = \mathbf{y}' | \hat{\mathbf{y}}_n = \mathbf{y}\right). \tag{3.8.2}$$

The probabilities on the left-hand sides of (3.8.1) and (3.8.2) in fact depend on the policies θ_1 and θ_2 followed by Φ_1 and Φ_2; on the other hand, Φ_c is merely a Markov chain rather than an MDP, whose transition probabilities are assumed to be chosen based on the policies followed by Φ_1 and Φ_2 in order to ensure the required equivalence. An example will be presented shortly in order to

demonstrate the technique. First, however, it will be useful to introduce the concept of a *sample path* as a means of representing the random trajectory of a stochastic process.

Definition 3.8.2. ***(Sample path)***

Let $\mathbf{\Omega} = (\Omega_0, \Omega_1, \Omega_2, ...)$ *be a sequence of i.i.d. random variables, and let the support (i.e. set of all possible values) of each* Ω_i *be denoted by* W*, where it is assumed that* W *is non-empty and finite. A* sample path *is a vector* $\boldsymbol{\omega} = (\omega_0, \omega_1, \omega_2, ...)$ *belonging to the set* $W \times W \times W \times ...$ *which completely determines the evolution of a given discrete-time stochastic process.*

The random variables Ω_i $(i = 0, 1, 2, ...)$ are in fact *categorical* random variables whose values govern the random transitions of a given process. At any given time step n, the value $\omega_n \in W$ (henceforth referred to as the *event* that occurs at time n) is completely independent of the state of the system, and also of the events that preceded it. One might imagine simply rolling a dice at each discrete time step to determine the random event that occurs, with the same dice used at all times, regardless of the system state. If the common distribution of the Ω_i is chosen appropriately, then each random transition of a discrete-time MDP can be determined as a consequence of the state, action chosen and random event occurring at the relevant point in time; the same applies to discrete-time Markov chains, except that obviously actions are not chosen. It is the fact that a sample path may be used to determine the evolution of a *coupling* (which, according to Definition 3.8.1, is simply a discrete-time Markov chain) that is of primary interest here.

A number of results in this thesis which involve comparisons between two or more processes will be established by first constructing a coupling for the relevant processes, and then examining the possible sample paths that may be followed by the coupled process. In practice, the construction of the coupling will involve listing the possible events that may occur (i.e. the events in the finite set W) at any time step n and, for each event, specifying the probability of occurrence and also the effect on the state of the coupled process if the event occurs. Several of the results in later chapters will involve comparisons between different policies in order to prove or disprove optimality, but in this chapter it will be useful to provide an example of how an argument based on coupling and sample paths can be used to prove an elementary property of $M/M/c$ queues.

Example 3.8.3. ***(Coupling of M/M/c queues)***

Consider two $M/M/c$ queues operating independently. Both queues have a common arrival rate $\lambda > 0$ and serve customers at a rate $\mu > 0$ at any service channel. Let c_1 and c_2 be the service capacities at the first and second queue respectively. It is assumed that $c_1 > c_2 \geq 1$; that is, the first queue has more service channels than the second. Rewards and holding costs are not relevant in this example. Suppose both queues operate a trivial policy of always allowing customers to join the queue, i.e. there is no balking; it will be assumed that $\lambda < c_2\mu$ in order to ensure system stability. let L_1 and L_2 denote the expected numbers of customers present at the first and second queues respectively under steady-state conditions. The aim is to show:

$$L_1 \leq L_2,$$

using a coupling argument. Of course, this fact may be established using more direct arguments based on standard formulae for $M/M/c$ queues (see, e.g. [67], p. 69), but the purpose of this example is simply to demonstrate the coupling technique. Let Φ_1 and Φ_2 be MDPs formulated using the transition probabilities defined in (3.5.3), with $c = c_1$ for the first process and $c = c_2$ for the second; naturally, both processes have state space $\mathbb{N}_0$. In order to create a coupling, it will be assumed that both processes are uniformised with parameter $\Delta = (\lambda + c_1\mu)^{-1}$. Let $(x_n)_{n\in\mathbb{N}_0}$ and $(y_n)_{n\in\mathbb{N}_0}$ denote the state-time evolutions of Φ_1 and Φ_2 respectively (there is some conflict here with earlier notation since x_i has previously been used to denote the i^{th} component of the vector $\mathbf{x}$, but this does not cause problems in this particular example since the states are not vectors.) It may be assumed that $x_0 = y_0 = 0$, since an $M/M/c$ queue will eventually converge to its stationary distribution and hence L_1 and L_2 are not affected by the choice of initial states.

Let $(\hat{x}_n, \hat{y}_n)$ be a coupling with state space $\mathbb{N}_0^2$, initialised in state $(0, 0)$. At any time step n, the random event ω_n that occurs is an element of the following set:

$$W := \left\{\Lambda, M^{(1)}, M^{(2)}, ..., M^{(c_1)}\right\},$$

where the events $\Lambda, M^{(1)}, M^{(2)}, ..., M^{(c_1)}$ are defined as follows:

- Λ is the event that a customer arrives, which occurs with probability $\lambda\Delta$. This event is seen by both of the marginal processes $(\hat{x}_n)$ and $(\hat{y}_n)$. If $\omega_n = \Lambda$, then $\hat{x}_{n+1} = \hat{x}_n + 1$ and $\hat{y}_{n+1} = \hat{y}_n + 1$; that is, both processes gain an extra customer.

- $M^{(i)}$ (for $i = 1, 2, ..., c_1$) is the event that a service completes at the i^{th} service channel, which occurs with probability $\mu\Delta$. This event is seen by the marginal process $(\hat{x}_n)$ if and only if $i \leq \hat{x}_n$, and it is seen by the process $(\hat{y}_n)$ if and only if $i \leq \min(\hat{y}_n, c_2)$. If it is seen by either marginal process, then that process loses a customer. That is:
  - $\hat{x}_{n+1} = \hat{x}_n - 1$ if and only if $\omega_n = M^{(i)}$ for some $i \leq \hat{x}_n$;
  - $\hat{y}_{n+1} = \hat{y}_n - 1$ if and only if $\omega_n = M^{(i)}$ for some $i \leq \min(\hat{y}_n, c_2)$.

The use of the phrase "seen by" is quite common in the literature to indicate that a particular event may affect one process without affecting another. For example, if a service completion is "seen by" one process but not another, this means that only the process which "sees" the service completion has one fewer customer present at the next time step; the other process is unaffected. If an event is not "seen" by a particular marginal process, then that process remains in the same state; that is, $\hat{x}_{n+1} = \hat{x}_n$ if $(\hat{x}_n)$ is the marginal process in question. As a point of terminology, it should be understood that even if an event is not "seen" by any process, this does not alter the fact that the event has occurred; for example, a service completion is said to "occur" even if it is not seen by any process, in which case there is no change in the state of the coupling. In future coupling proofs, the descriptions of the various events in W (as given in the two bullet points above) will be somewhat shortened in order to avoid extraneous detail; for example, if a customer arrival is seen by both of the marginal processes $(\hat{x}_n)$ and $(\hat{y}_n)$, then this will be stated without the extra information that $\hat{x}_{n+1} = \hat{x}_n + 1$ and $\hat{y}_{n+1} = \hat{y}_n + 1$ as a result. The marginal processes $(\hat{x}_n)$ and $(\hat{y}_n)$ will also be referred to simply as processes; so, for example, it will be deemed sufficient to state that "the event Λ occurs with probability $\lambda\Delta$ and causes an arrival to be seen by both processes".

The coupled process $(\hat{x}_n, \hat{y}_n)$ evolves according to a sample path $(\omega_0, \omega_1, \omega_2, ...)$, where each ω_i is one of the random events in the set $W = \{\Lambda, M^{(1)}, M^{(2)}, ..., M^{(c_1)}\}$. According to the definitions of these events given previously, there is a probability $\lambda\Delta$ of an arrival at any time step. Also, it can easily be checked from the definition of the events $M^{(i)}$ (noting that each $M^{(i)}$ has the same probability of occurrence, $\mu\Delta$) that the probability of a service completion being seen by process $(\hat{x}_n)$ at time step n is $\min(\hat{x}_n, c_1)\mu\Delta$, and the corresponding probability for the process $(\hat{y}_n)$ is $\min(\hat{y}_n, c_2)\mu\Delta$. If no event is seen by a particular process at time step n, then it remains in the same state at time step $n + 1$. It follows that the transition probabilities $P(\hat{x}_{n+1} = \hat{x}' | \hat{x}_n = \hat{x})$ and $P(\hat{y}_{n+1} = \hat{y}' | \hat{y}_n = \hat{y})$ correspond exactly with the transition probabilities defined in (3.5.3) which

govern the transitions of the MDPs Φ_1 and Φ_2 respectively. Therefore, for all $n \in \mathbb{N}_0$, $\hat{x}_n$ has the same distribution as x_n and $\hat{y}_n$ has the same distribution as y_n. Of course, this is a requirement of the coupling construction. In order to show that $L_1 \leq L_2$, it will therefore be sufficient to restrict attention to the coupled variables and prove that $E[\hat{x}_n] \leq E[\hat{y}_n]$ for all $n \in \mathbb{N}_0$.

In fact, in this particular example one can prove the stronger property that, given any sample path $\boldsymbol{\omega} = (\omega_0, \omega_1, \omega_2, ...)$, one must have $\hat{x}_n \leq \hat{y}_n$ for all $n \in \mathbb{N}_0$. This can be shown using a somewhat trivial inductive argument. It is assumed that the $\hat{x}_0 = \hat{y}_0 = 0$, so the inequality holds when $n = 0$. Assume that $\hat{x}_k \leq \hat{y}_k$, for arbitrary $k \in \mathbb{N}_0$. If the event Λ occurs at stage k, then it is seen by both processes, so $\hat{x}_{k+1} \leq \hat{y}_{k+1}$ follows. Next, suppose $\hat{x}_k < \hat{y}_k$ holds with strict inequality. It is not possible for an event to occur which causes $\hat{x}_{k+1}$ to be greater than $\hat{x}_k$ while simultaneously causing $\hat{y}_{k+1}$ to be smaller than $\hat{y}_k$, and since the state of $(\hat{y}_n)$ can decrease by at most one in a single time step, it must be the case that $\hat{x}_k < \hat{y}_k$ implies $\hat{x}_{k+1} \leq \hat{y}_{k+1}$. The only remaining case to check is the case where $\hat{x}_k = \hat{y}_k$ and an event other than Λ occurs. If $\hat{y}_k \geq c_2$, then the only service completion events that would be seen by $(\hat{y}_n)$ at stage k are the events $M^{(1)}, M^{(2)}, ..., M^{(c_2)}$, but due to the definition of these events and the fact that $\hat{x}_k = \hat{y}_k$, these events are also seen by the process $(\hat{x}_n)$ at the same stage. Similarly, if $\hat{y}_k < c_2$ then the only service completion events that would be seen by $(\hat{y}_n)$ are the events $M^{(1)}, M^{(2)}, ..., M^{(\hat{y}_k)}$, but these events are also seen by $(\hat{x}_n)$ at the same stage. In summary, it is not possible for the process $(\hat{y}_n)$ to see a service completion which is not seen by $(\hat{x}_n)$ unless $(\hat{y}_n)$ has more customers present than $(\hat{x}_n)$ at the stage in question. It follows that $\hat{x}_{k+1} \leq \hat{y}_{k+1}$, which proves by induction that $\hat{x}_n \leq \hat{y}_n$ for all $n \in \mathbb{N}_0$.

Given that $\hat{x}_n \leq \hat{y}_n$ for all $n \in \mathbb{N}_0$ along any random sample path $\boldsymbol{\omega}$, it follows trivially that $E[\hat{x}_n] \leq E[\hat{y}_n]$ for all $n \in \mathbb{N}_0$. Hence, due to the coupling construction, $E[x_n] \leq E[y_n]$ for all $n \in \mathbb{N}_0$, where (x_n) and (y_n) represent the respective evolutions of the original MDPs Φ_1 and Φ_2 which are naturally independent of each other. It follows that $\lim_{n\to\infty} E[x_n] \leq \lim_{n\to\infty} E[y_n]$, or equivalently $L_1 \leq L_2$, which completes the proof of the required property. $\boxtimes$

Figure 3.8 illustrates the transition probabilities for the coupled process $(\hat{x}_n, \hat{y}_n)$ in the case where $c_1 = 2$ and $c_2 = 1$. Self-transitions (i.e. transitions from a state to itself) are not shown in the diagram in order to avoid over-cluttering, and neither are transitions to or from non-recurrent states. Given that the initial state is $(0, 0)$, any state $(\hat{x}, \hat{y})$ with $\hat{x} > \hat{y}$ can never be visited and

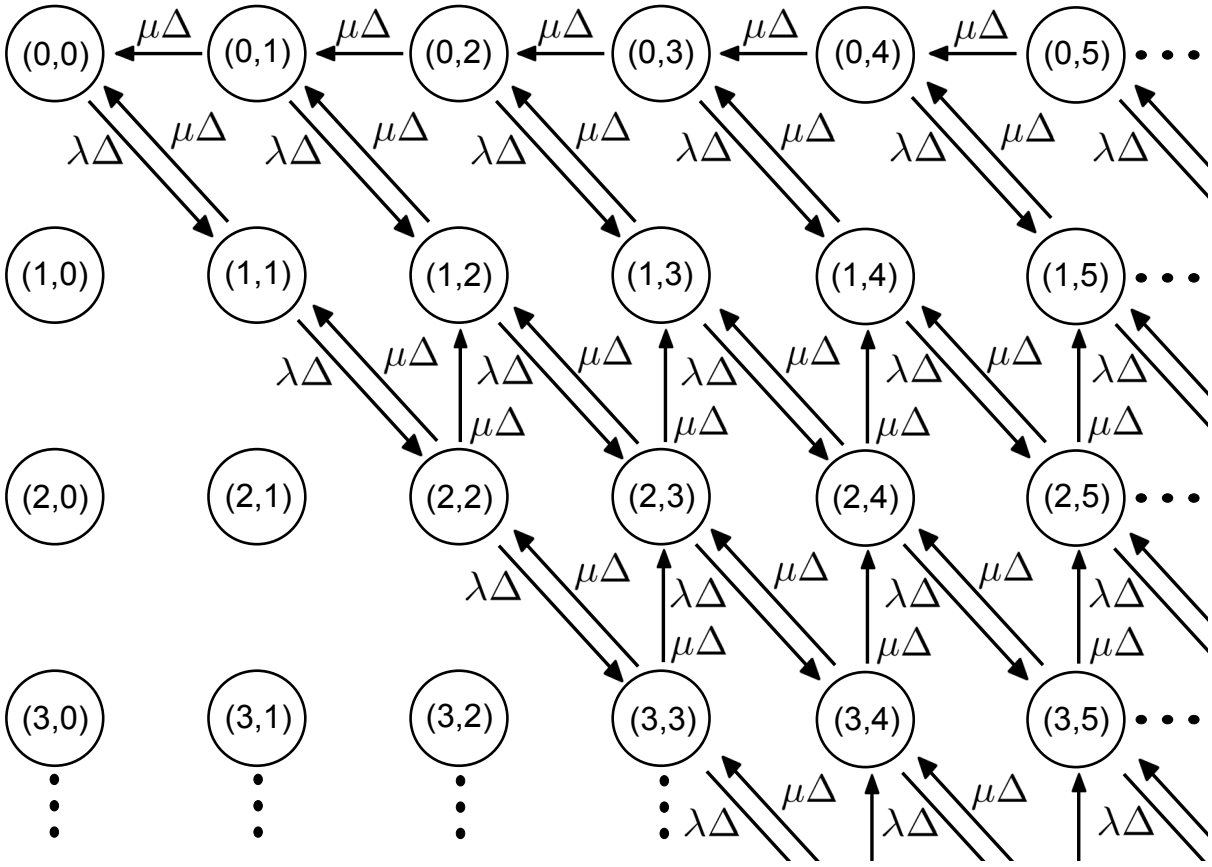


Figure 3.8: The transitions probabilities for the coupled process $(\hat{x}_n, \hat{y}_n)$ in Example 3.8.3 with $c_1 = 2$ and $c_2 = 1$, not including self-transitions or transitions to or from non-recurrent states.

therefore the Markov chain has an 'upper triangular' structure. The reason for states with $\hat{x} < \hat{y}$ being accessible is that, in any state $(\hat{x}, \hat{y})$ with $\hat{x} \geq 2$, there is a positive probability that a service completion will be the seen by the first process but not by the second process; this is the reason for the 'upward' transitions in the diagram at states with $\hat{x} \geq 2$. Note that customer arrivals are always seen by both processes, so these events always cause 'diagonal' transitions.

It was mentioned earlier that proofs involving the construction of a coupling for two discrete-time processes are in some sense analogous to inductive proofs based on the functions $h_n(\cdot)$ derived using relative value iteration. Indeed, although various results later in this thesis will be proved using coupling formulations, these results are generally not beyond the realms of being achievable using dynamic programming methods. It is not considered necessary to prove any formal equivalence between the two methodologies here; in fact, it will be quite evident from the style of some of the

coupling-based proofs in later chapters that the analogous proofs based on relative value iteration would follow extremely similar arguments. To some extent, therefore, the question of whether to prove certain results using coupling constructions or to rely merely on the finite-stage value functions of dynamic programming is simply a question of presentation.

The proof given in Example 3.8.3 relies on the principle that, after the coupling $(\hat{x}_n, \hat{y}_n)$ has been constructed, the marginal processes $(\hat{x}_n)$ and $(\hat{y}_n)$ evolve according to identical sequences of random events. Essentially, the sole purpose of the coupling construction is to provide a theoretical justification for the strategy of 'forcing' the two processes to follow the same random trajectory. This suggests that any proof based on a stochastic coupling argument may be considerably shortened by *not explicitly formulating the coupling itself*; while this might sound somewhat bizarre, in fact it is quite reasonable (and much more convenient from a notational point of view) to write a proof which directly tackles the processes (x_n) and (y_n) by assuming that these processes follow the same sample path, as opposed to performing an analysis on the coupled process $(\hat{x}_n, \hat{y}_n)$. To make this idea more clear, the proof given in Example 3.8.3 could have avoided an explicit coupling construction by proceeding (roughly) along the following lines:

- Assume that the processes (x_n) and (y_n) follow the same (arbitrary) sample path $\boldsymbol{\omega}$.
- Show that $x_n \leq y_n$ for all $n \in \mathbb{N}_0$, assuming (naturally) that the two processes are both initialised in state 0 and evolve according to the same sequence of events.
- Conclude that $L_1 \leq L_2$ by considering expectations over all possible sample paths.

The idea of a coupling between (x_n) and (y_n) would be implicit in such an argument, but the coupling would not need to be constructed explicitly, or even referred to in the proof. The coupling-based proofs given in later chapters will take advantage of this 'shortcut strategy' in order to avoid tedious details as much as possible, but the general methodology explained in this chapter will be referred to as and when it becomes appropriate to be more rigorous.

Before concluding this section, it will be useful to address a minor issue that may arise in the conversion of a stochastic coupling argument to a dynamic programming argument. The issue concerns the effect of the value of Δ (the uniformisation parameter) on the values $h(\mathbf{x})$ obtained from relative value iteration, which (due to Theorem 3.7.4) are known to satisfy the average reward

optimality equations in (3.7.6). Recall that, due to Theorem 3.3.3, the parameter Δ may take any value between zero and a certain positive upper bound (dependent upon the transition rates of the CTMDP) without affecting the expected long-run average reward under a particular stationary policy. Theorem 3.7.4 establishes the existence of a stationary optimal policy, assuming that the state space is finite. It therefore seems reasonable to conclude that the optimal long-run average reward g^* should be unaffected by the exact value of Δ, and indeed this is the case. One might also assume that the relative values $h(\mathbf{x})$ satisfying the equations (3.7.6) should be independent of the choice of Δ; however, this assumption would be incorrect. In fact, the $h(\mathbf{x})$ values *do* depend on the value of Δ. The following lemma clarifies the nature of this dependence.

Lemma 3.8.4. *Consider the finite-state MDP $\tilde{\Upsilon}$ discussed in Section 3.7, and assume that $\tilde{\Upsilon}$ is uniformised with parameter $\Delta \in \big(0, (\lambda + \sum_i c_i\mu_i)^{-1}\big]$. Let $h : \tilde{S} \to \mathbb{R}$ be a function which, together with the constant g^*, satisfies the average reward optimality equations:*

$$g^* + h(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h(\mathbf{y}) \right\} \qquad \forall\, \mathbf{x} \in \tilde{S}. \tag{3.8.3}$$

Furthermore, assume that $h(\mathbf{0}) = 0$ in order to determine the $h(\mathbf{x})$ values uniquely. Let $\Delta^\dagger \in (0, \Delta)$ be arbitrary, and consider an MDP $\Upsilon^\dagger$ identical to $\tilde{\Upsilon}$ except that it is uniformised with parameter $\Delta^\dagger$. Let $h^\dagger$ be a function which, together with the constant $g^\dagger$, satisfies:

$$g^\dagger + h^\dagger(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p^\dagger(\mathbf{x}, a, \mathbf{y}) h^\dagger(\mathbf{y}) \right\} \qquad \forall\, \mathbf{x} \in \tilde{S}, \tag{3.8.4}$$

where $h^\dagger(\mathbf{0}) = 0$ is assumed, and the transition probabilities $p^\dagger(\mathbf{x}, a, \mathbf{y})$ are obtained by replacing Δ with $\Delta^\dagger$ in (3.5.3). Then $g^\dagger = g^*$, and for each $\mathbf{x} \in \tilde{S}$:

$$h^\dagger(\mathbf{x}) = (\Delta/\Delta^\dagger) h(\mathbf{x}).$$

Proof. It is clear from the definition of the transition probabilities $p(\mathbf{x}, a, \mathbf{y})$ in (3.5.3) that the effect of reducing the value of Δ is to increase the probability of a 'self-transition'. To be specific, the probabilities $p^\dagger(\mathbf{x}, a, \mathbf{y})$ satisfy the following for all $\mathbf{x}, \mathbf{y} \in \tilde{S}$ and $a \in A_{\mathbf{x}}$:

$$p^\dagger(\mathbf{x}, a, \mathbf{y}) = \begin{cases} (\Delta^\dagger/\Delta) p(\mathbf{x}, a, \mathbf{y}), & \text{if } \mathbf{y} \neq \mathbf{x}, \\ 1 - \displaystyle\sum_{\mathbf{y} \neq \mathbf{x}} (\Delta^\dagger/\Delta) p(\mathbf{x}, a, \mathbf{y}), & \text{if } \mathbf{y} = \mathbf{x}. \end{cases}$$

However, the expression for $p^\dagger(\mathbf{x}, a, \mathbf{x})$ can be re-written as follows:

$$1 - \sum_{\mathbf{y} \neq \mathbf{x}} (\Delta^\dagger/\Delta) p(\mathbf{x}, a, \mathbf{y}) = 1 - (\Delta^\dagger/\Delta)\,(1 - p(\mathbf{x}, a, \mathbf{x})) = (\Delta^\dagger/\Delta) p(\mathbf{x}, a, \mathbf{x}) + 1 - (\Delta^\dagger/\Delta).$$

This enables the optimality equations in (3.8.4) to be put into the form:

$$g^\dagger + h^\dagger(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) (\Delta^\dagger/\Delta) h^\dagger(\mathbf{y}) \right\} + (1 - \Delta^\dagger/\Delta) h^\dagger(\mathbf{x}) \quad \forall\, \mathbf{x} \in \tilde{S}, \qquad (3.8.5)$$

Then, after cancellation of $h^\dagger(\mathbf{x})$:

$$g^\dagger + (\Delta^\dagger/\Delta) h^\dagger(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) (\Delta^\dagger/\Delta) h^\dagger(\mathbf{y}) \right\} \qquad \forall\, \mathbf{x} \in \tilde{S}, \qquad (3.8.6)$$

Recall that the values $h(\mathbf{x})$ satisfying the equations (3.8.3) are unique, assuming that $h(\mathbf{0}) = 0$. Hence, by direct comparison between (3.8.3) and (3.8.6):

$$h^\dagger(\mathbf{x}) = (\Delta/\Delta^\dagger) h(\mathbf{x}) \qquad \forall\, \mathbf{x} \in \tilde{S},$$

and $g^\dagger$ is a constant satisfying the average reward optimality equations, which must therefore (by Theorem 3.7.3) be equal to the optimal long-run average reward, g^*. □

Lemma 3.8.4 states that the effect of reducing the value of the uniformisation parameter from Δ to $\Delta^\dagger \in (0, \Delta)$ is that the relative values $h(\mathbf{x})$ are *increased* by the multiplicative factor $\Delta/\Delta^\dagger$. However, from (3.8.5) it may be seen that the terms within the maximisation operator are unaffected by the transformation from $h(\mathbf{y})$ to $h^\dagger(\mathbf{y})$, since $(\Delta^\dagger/\Delta) h^\dagger(\mathbf{y}) = h(\mathbf{y})$ for all $\mathbf{y} \in \tilde{S}$. It follows that any action a which attains the maximum in the original optimality equations (3.8.3) also does so in the transformed optimality equations (3.8.4). In conclusion, changing the value of Δ modifies the relative values for all $\mathbf{x} \in \tilde{S}$, but does not affect the class of optimal policies.

3.9 Conclusions

This chapter began by introducing a mathematical model for a multiple-facility queueing system with heterogeneous service facilities, which evolves according to Markovian probability distributions. Most of the assumptions listed in Section 3.1 will be retained throughout this thesis, although

some will occasionally be relaxed in order to allow a broader range of problems to be addressed; for example, heterogeneous customers will be considered in Chapters 4 and 7.

It is important not to lose sight of the fact that the queueing system in Section 3.1 operates in *continuous time* rather than discrete time, and this is why a continuous-time MDP formulation was given in Section 3.2. However, the various results in Sections 3.3, 3.4 and 3.5 have shown that any *stationary* policy θ earns the same expected long-run average reward in the continuous-time process Ψ as an equivalent stationary policy (related to θ via a bijective relationship, as explained in Lemma 3.5.2) in the discrete-time MDP Υ formulated in Section 3.5. Since Theorem 3.4.10 has established the existence of a stationary policy which maximises the average reward for Ψ, it follows that in order to find an average reward optimal policy for Ψ it is sufficient to restrict attention to stationary policies in Υ. Accordingly, most of the results given in subsequent chapters will assume the context of a discrete-time MDP with a simplified state space given by (3.5.1).

The practical task of finding an average reward optimal stationary policy for Υ can be accomplished using the dynamic programming algorithms discussed in Section 3.7, but this requires a finite truncation to be made to the state space S which can only be justified if it can be shown that there exists an optimal policy θ which induces a Markov chain restricted to some finite set of states $S_\theta \subset S$. The results in the next chapter will examine whether or not it is feasible to make such a truncation, by investigating the characteristics of average reward optimal policies.

4 Containment and demand

Methods for obtaining average reward optimal policies for the queueing system introduced in Section 3.1 will be of interest throughout this thesis. A further objective will be to identify the characteristics of these optimal policies. It will become evident later that certain structural properties of optimal policies can be proved only by making additional assumptions about the system; for example, some of the results in Chapter 5 will require restrictions to be imposed on the number of facilities N. However, throughout this chapter the model described in Section 3.1 will be considered in its full generality, without any extra conditions on the system parameters.

Section 4.1 will revisit the themes of selfish and social optimisation discussed in Chapter 2, and explain how these may be generalised to a system consisting of N facilities. Section 4.2 will establish a set of optimality equations for MDPs defined on an infinite state space. These equations will then be used to prove an important 'containment' property of socially optimal policies which will be of great practical use in later chapters. Section 4.3 will prove a further property of socially optimal policies which complements the main result of the preceding section. Section 4.4 will prove various interesting results involving the effect of the demand rate λ on optimal policies, and finally Section 4.5 will consider an extension of the problem involving heterogeneous customers.

4.1 Selfish and social optimisation

This section explores the themes of *selfish optimisation* and *social optimisation* in the multiple-facility queueing system introduced in Chapter 3. Informally speaking, *selfish* behaviour occurs when individual customers take actions aimed solely at optimising their own outcomes; *social* optimisation, on the other hand, involves the adherence of customers to a common behavioural scheme which optimises their collective welfare. It is natural to suppose that selfish behaviour arises when customers act of their own free will, whereas socially optimal behaviour may be induced by a central authority (essentially, a 'benign dictatorship') which controls the fate of each newly-arrived customer; however, the mathematical results developed in this chapter do not require such assumptions to be made. It will be shown later that socially optimal policies possess an interesting 'containment' property which has useful implications for computational algorithms.

The assumptions of Section 3.1 apply throughout this chapter. By using the technique of uniformisation described in Section 3.3, it is valid to model the N-facility queueing system within a discrete-time framework, using the MDP Υ formulated in Section 3.5. At this point, it is worth noting that the state space S of this discrete-time MDP is *countably infinite*; as such, the various results of Section 3.7 (which assume a finite state space) cannot be applied unless there is a justification for truncating the state space. *Policies* for discrete-time MDPs have been defined in Section 3.6. Before proceeding, it is necessary to establish the exact criteria for determining whether a particular policy is *selfishly optimal* and/or *socially optimal* in a given system.

Selfishly optimal policies will be discussed first. Suppose that each customer arriving in the system is allowed to make his or her own decision, as opposed to being directed by a central decision-maker. As stated in Section 3.1, it is assumed that the queueing system is *fully observable*, and therefore each customer is able to observe the exact state of the system $\mathbf{x} \in S$ before making a decision (the case of *unobservable queues* will be discussed in Chapter 6). Under this scenario, a customer may calculate their expected net reward (taking into account the expected cost of waiting and the value of service) at each facility based on the number of customers present. Let $w(\mathbf{x}, a)$ denote an individual customer's expected net reward for choosing action $a \in \{0, 1, ..., N\}$ under state $\mathbf{x} \in S$ (recall that $a = 0$ denotes balking). Then, by analogy to the results in Section 3.2:

$$w(\mathbf{x}, a) := \begin{cases} \alpha_i - \dfrac{\beta_i}{\mu_i}, & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } x_i < c_i, \\ \alpha_i - \dfrac{\beta_i(x_i + 1)}{c_i \mu_i}, & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } x_i \geq c_i, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.1}$$

If customers act selfishly, they will simply choose the action a which maximises $w(\mathbf{x}, a)$. Naturally, this implies that if the expected net rewards $w(\mathbf{x}, i)$ for all facilities $i \in \{1, 2, ..., N\}$ are negative, then a selfish customer's decision will be to balk. In the case of ties, a convention will be used whereby the facility with the smallest index i is chosen; however, balking will never be preferred to joining any facility i for which $w(\mathbf{x}, i) = 0$. It is interesting to note that since the FCFS queue discipline is assumed at each facility, a selfish customer's behaviour depends only on the existing state, and is not influenced by the knowledge that other customers act selfishly.

With this definition of selfish behaviour it is easy to see that the selfishly optimal policy, denoted by $\tilde{\theta}$, is a stationary policy which, somewhat curiously, is independent of the demand rate λ. Given that selfish customers refuse to choose facility i if $w(\mathbf{x}, i) < 0$, it follows that for $i = 1, 2, ..., N$ there exists an upper threshold $\tilde{B}_i$ which represents the greatest possible number of customers at i under steady state conditions. The threshold $\tilde{B}_i$ can be derived from (4.1.1) as:

$$\tilde{B}_i := \left\lfloor \frac{\alpha_i c_i \mu_i}{\beta_i} \right\rfloor, \tag{4.1.2}$$

where $\lfloor \cdot \rfloor$ denotes the integer part. Thus, a selfishly optimal policy $\tilde{\theta}$ induces an ergodic Markov chain defined on a *finite* set of states $\tilde{S}$. Formally, $\tilde{S}$ is given by:

$$\tilde{S} := \Big\{ (x_1, x_2, ..., x_N) \in S : x_i \leq \tilde{B}_i \text{ for all } i \Big\}. \tag{4.1.3}$$

Henceforth, $\tilde{S}$ will be referred to as the *selfishly optimal state space*, since it consists of all states which are positive recurrent under the policy $\tilde{\theta}$. The size of $\tilde{S}$ is given by:

$$\left| \tilde{S} \right| = \prod_{i=1}^{N} (\tilde{B}_i + 1).$$

As stated previously, a *socially optimal* policy is one which maximises the collective welfare of all customers. As in Chapter 2, *expected long-run average reward* will be used to measure 'collective welfare'. Recall from Chapter 3 that the average reward $g_\theta(\mathbf{x})$ is defined as:

$$g_\theta(\mathbf{x}) := \liminf_{t \to \infty} t^{-1} E_\theta \left[\sum_{n=0}^{t-1} r(\mathbf{x}_n) \Big| \mathbf{x}_0 = \mathbf{x} \right], \tag{4.1.4}$$

where θ is an arbitrary policy, $\mathbf{x}_n \in S$ is the state of the process at time step $n \in \mathbb{N}_0$ and $\mathbf{x} \in S$ is the initial state. In (4.1.4), the *real-time* reward function defined in (3.5.4) is used, but if θ is a stationary policy then this can be replaced by the anticipatory reward function $\hat{r}(\mathbf{x}, a)$ defined in (3.5.15) without affecting the value of $g_\theta(\mathbf{x})$ (this equivalence is due to Theorem 3.5.3). In this chapter, a *socially optimal* policy is defined as any policy θ^* which satisfies:

$$g_{\theta^*}(\mathbf{x}) = \sup_\theta g_\theta(\mathbf{x}) =: g^*(\mathbf{x}) \qquad \forall\, \mathbf{x} \in S,$$

and hence attains average reward optimality as defined in Chapter 3. It should be noted that a socially optimal policy will generally not be *unique*, whereas the selfishly optimal policy $\tilde{\theta}$ and the selfishly optimal state space $\tilde{S}$ are both determined uniquely (assuming the conventions discussed

earlier for dealing with ties, etc.). Changing the ordering of the facilities (and thereby the tie-breaking rules) affects the policy $\tilde{\theta}$, but does *not* alter the boundaries of $\tilde{S}$. Two important ways in which the selfish policy $\tilde{\theta}$ differs from a socially optimal policy are as follows:

1. The decisions made under $\tilde{\theta}$ are entirely independent of the demand rate λ;
2. The threshold $\tilde{B}_i$ (representing the steady-state maximum occupancy at facility i) is independent of the parameters for the other facilities $j \neq i$.

Given an arbitrary stationary policy θ, let S_θ denote the set of states in S which are positive recurrent in the Markov chain induced by θ. A major result to be proved in this chapter is that if θ^* is a socially optimal policy, then the following relationship holds:

$$S_{\theta^*} \subseteq \tilde{S}.$$

That is, the set of states which are positive recurrent under θ^* (equivalently, the set of states accessible from the state $\mathbf{0}$ under θ^*) is contained within the selfishly optimal state space $\tilde{S}$. This principle may be illustrated using a simple example with two facilities.

Example 4.1.1. ***(Containment in two dimensions)***

Consider a system with demand rate $\lambda = 10$ and only two facilities. The first facility has two channels available ($c_1 = 2$) and a service rate $\mu_1 = 8$, holding cost $\beta_1 = 10$ and fixed reward $\alpha_1 = 2$. The parameters for the second facility are $c_2 = 2$, $\mu_2 = 2$, $\beta_2 = 10$ and $\alpha_2 = 6$, so it offers a higher reward but a slower service rate. The system may be uniformised by taking $\Delta = 1/30$, so that $(\lambda + \sum_i c_i\mu_i)\,\Delta = 1$. The selfishly optimal state space $\tilde{S}$ for this system consists of 12 states. Table 4.1 shows the decisions taken at these states under the selfishly optimal policy $\tilde{\theta}$, and also the corresponding decisions taken under a *socially* optimal policy θ^*. (Verification of the fact that θ^* is socially optimal requires results which will be established in Section 4.2.)

By referring to Table 4.1, the differences between the two policies $\tilde{\theta}$ and θ^* may be observed. At the states $(0,1)$, $(1,1)$ and $(2,2)$, the socially optimal policy θ^* deviates from the selfish policy $\tilde{\theta}$. More striking, however, is the fact that under the socially optimal policy, some of the states in $\tilde{S}$ are actually unattainable under steady state conditions. Indeed, the recurrent state space S_{θ^*} of

the socially optimal policy consists of only 9 states (coloured grey in Table 4.1). Thus, for this system, $S_{\theta^*} \subseteq \tilde{S}$ and in this chapter it will be proved that this principle holds in general. ⊠

Selfishly optimal policy $\tilde{\theta}$			
	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	2	2	1
$x_1 = 1$	2	2	1
$x_1 = 2$	2	2	1
$x_1 = 3$	2	2	0

Socially optimal policy θ^*			
	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	2	1	1
$x_1 = 1$	2	1	1
$x_1 = 2$	2	2	0
$x_1 = 3$	2	2	0

Table 4.1: Selfishly and socially optimal policies for the system considered in Example 4.1.1. For each state $\mathbf{x} = (x_1, x_2) \in \tilde{S}$, the corresponding decisions under the respective policies are shown.

In order to develop the topic of socially optimal policies for MDPs defined on an infinite state space, certain extra theoretical results are needed. These are discussed in the next section.

4.2 Containment of socially optimal policies

In Example 4.1.1 it was claimed that the policy θ^* (illustrated in Table 4.1) was socially optimal, but no details were given regarding the method for deriving this policy or proving its optimality. Recall that Section 3.7 introduced two algorithms for computing average reward optimal policies, but these required the assumption of a finite state space. In (3.7.6), the average reward optimality equations for MDPs defined on a *finite* state space $\tilde{S}$ were given as follows:

$$g^* + h(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h(\mathbf{y}) \right\} \qquad \forall\, \mathbf{x} \in \tilde{S}. \tag{4.2.1}$$

The conditions under which these equations can be used to characterise average reward optimal policies for MDPs defined on *infinite* state spaces have been explored by various authors. Derman [37] has shown, under the assumption of bounded rewards, that if a constant g^* and function h can be found which satisfy (4.2.1), then any stationary policy determined by this equation is average reward optimal. Ross [145] considered the optimal discounted value function $v_\phi(\cdot)$ (see Definition 3.4.1) and showed that if the differences $|v_\phi(\mathbf{x}) - v_\phi(\mathbf{z})|$ (where $\mathbf{z} \in S$ is a fixed reference state)

are uniformly bounded in $\mathbf{x}$ and ϕ, then a constant g^* and function h satisfying (4.2.1) must exist. Zijm [205] also assumed bounded rewards and provided a number of equivalent conditions, each of which ensure the existence of a bounded solution to (4.2.1). Other notable references for average reward optimisation in denumerable-state MDPs include [25, 26, 27, 28, 153].

The results in this section will make use of the conditions provided by Sennott [156] for the existence of an average reward optimal stationary policy satisfying the equations (4.2.1) in the case of a countably infinite state space. These conditions rely on properties of the optimal value function $v_\phi(\cdot)$ in a *discounted reward* problem and are largely analogous to the conditions discussed in Section 3.4 which relate to a continuous-time MDP. The expected total discounted reward for the MDP Υ operating under an arbitrary policy θ, given a discount factor $\phi \in (0,1)$, is given by:

$$v_{\phi,\theta}(\mathbf{x}) := E_\theta \left[\sum_{n=0}^{\infty} \phi^n r(\mathbf{x}_n) \Big| \mathbf{x}_0 = \mathbf{x} \right], \tag{4.2.2}$$

where $\mathbf{x} \in S$ is the initial state. In (4.2.2) it is assumed that the real-time reward formulation (3.5.4) is used. Similarly, under the alternative reward formulation (3.5.15), one has:

$$\hat{v}_{\phi,\theta}(\mathbf{x}) := E_\theta \left[\sum_{n=0}^{\infty} \phi^n \hat{r}(\mathbf{x}_n, a_n) \Big| \mathbf{x}_0 = \mathbf{x} \right]. \tag{4.2.3}$$

As discussed in Section 3.5 (and proved by Theorem 3.5.3), the two reward formulations r and $\hat{r}$ are equivalent with respect to the expected long-run average reward earned under a stationary policy. However, in general it is *not* the case that $v_{\phi,\theta}(\mathbf{x})$ and $\hat{v}_{\phi,\theta}(\mathbf{x})$ are equal for a fixed $\mathbf{x} \in S$. Indeed, given some discount factor $\phi \in (0,1)$ and state $\mathbf{x} \in S$, it is perfectly possible to have $v_{\phi,\theta_1}(\mathbf{x}) > v_{\phi,\theta_2}(\mathbf{x})$ and $\hat{v}_{\phi,\theta_1}(\mathbf{x}) < \hat{v}_{\phi,\theta_2}(\mathbf{x})$ for two different stationary policies θ_1 and θ_2. Typically, this scenario would require θ_1 to be a more *conservative* policy than θ_2; that is, the set of states accessible from $\mathbf{0}$ under θ_1 would be smaller than the corresponding set under θ_2.

Despite the fact that $v_{\phi,\theta}(\cdot)$ and $\hat{v}_{\phi,\theta}(\cdot)$ are not equivalent, the existence of an average reward optimal policy satisfying the equations (4.2.1) may be proved under either reward formulation, although the arguments are slightly different depending on the formulation used. In order to avoid unnecessarily lengthy arguments, it will therefore be convenient to concentrate on only one of the two formulations. As such, let it be assumed that the *anticipatory* formulation is used, so that the expected total discounted reward is given by (4.2.3). Then, in order to qualify as ϕ-discount

optimal, a policy θ^* must satisfy the following condition for all admissible policies θ:

$$\hat{v}_{\phi,\theta^*}(\mathbf{x}) \geq \hat{v}_{\phi,\theta}(\mathbf{x}) \qquad \forall\, \mathbf{x} \in S.$$

Furthermore, $\hat{v}_\phi(\mathbf{x}) = \sup_\theta \hat{v}_{\phi,\theta}(\mathbf{x})$ will be used to denote the optimal value function. By analogy to (3.4.4), the function $\hat{v}_\phi(\cdot)$ satisfies the following discount optimality equations:

$$\hat{v}_\phi(\mathbf{x}) = \max_{a \in A} \left\{ \hat{r}(\mathbf{x}, a) + \phi \sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y})\, \hat{v}_\phi(\mathbf{y}) \right\} \qquad (\mathbf{x} \in S). \tag{4.2.4}$$

The next result establishes an important monotonicity property of $\hat{v}_\phi(\mathbf{x})$ (which, incidentally, does not hold for its counterpart $v_\phi(\mathbf{x})$ under the real-time formulation (3.5.4)).

Lemma 4.2.1. *For every state $\mathbf{x} \in S$, discount factor $\phi \in (0,1)$ and facility $i \in \{1, 2, ..., N\}$:*

$$\hat{v}_\phi(\mathbf{x}^{i+}) \leq \hat{v}_\phi(\mathbf{x}).$$

Proof. The proof relies upon the method of successive approximations (see Theorem 3.4.4) and is based on induction. Details can be found in Appendix A.4, page 421.

The next lemma establishes a property of $\hat{v}_\phi(\cdot)$ which will be needed in order to state one of the main results of this section. Specifically, it must be shown that there exists a state-dependent lower bound for the difference $\hat{v}_\phi(\mathbf{x}) - \hat{v}_\phi(\mathbf{0})$ which holds for any discount factor $\phi \in (0,1)$. The proof of the lemma is similar to that of its continuous-time analogue, Lemma 3.4.9.

Lemma 4.2.2. *For every state $\mathbf{x} \in S$, there exists a non-negative value $f(\mathbf{x}) < \infty$ such that for all discount factors $\phi \in (0,1)$, the following holds:*

$$\hat{v}_\phi(\mathbf{x}) - \hat{v}_\phi(\mathbf{0}) \geq -f(\mathbf{x}). \tag{4.2.5}$$

Proof. The proof emulates that of Lemma 3.4.9 by considering a transformed MDP in which all rewards are non-positive. Details can be found in Appendix A.4, page 423.

The results of the previous two lemmas may be used to show that the MDP Υ satisfies Sennott's conditions in [156] for the extension of the average reward optimality equations (4.2.1) to an MDP with an infinite state space. This important result is stated below as a theorem.

*** Theorem 4.2.3.** *Consider the MDP Υ formulated in (3.5.1)-(3.5.4). There exists a constant g^* and a function $\hat{h}(\mathbf{x})$, with $g^* = \lim_{\phi\uparrow 1}(1-\phi)\hat{v}_\phi(\mathbf{x})$ and $-f(\mathbf{x}) \leq \hat{h}(\mathbf{x}) \leq 0$ for $\mathbf{x} \in S$, such that:*

$$g^* + \hat{h}(\mathbf{x}) = \max_{a \in A} \left\{ \hat{r}(\mathbf{x}, a) + \sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y}) \hat{h}(\mathbf{y}) \right\} \qquad \forall\, \mathbf{x} \in S. \tag{4.2.6}$$

Moreover, let θ^* be a stationary policy and suppose $\theta^*(\mathbf{x})$ attains the maximum in (4.2.6) for every state $\mathbf{x} \in S$. Then θ^* is average reward optimal, with an average reward g^*.

Proof. This result is proved in [156], but it requires certain conditions to be met, so it should be checked that these conditions hold for the MDP Υ. The conditions are as follows:

1. Rewards are bounded above, but not below; that is, there exists a constant $R < \infty$ such that for every state $\mathbf{x} \in S$ and action $a \in A$, $\hat{r}(\mathbf{x}, a) \leq R$.

2. For each state $\mathbf{x} \in S$ and discount factor $\phi \in (0, 1)$, $\hat{v}_\phi(\mathbf{x}) > -\infty$.

3. Define $\hat{h}_\phi(\mathbf{x}) = \hat{v}_\phi(\mathbf{x}) - \hat{v}_\phi(\mathbf{z})$ for some fixed reference state $\mathbf{z} \in S$. There exists a constant $K < \infty$ such that, for each state $\mathbf{x} \in S$, $\hat{h}_\phi(\mathbf{x}) \leq K$ for all $\phi \in (0, 1)$.

4. For each $\mathbf{x} \in S$, there exists a non-negative value $f(\mathbf{x}) < \infty$ such that $\hat{h}_\phi(\mathbf{x}) \geq -f(\mathbf{x})$ for all $\phi \in (0, 1)$. Also, $\sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y}) f(\mathbf{y}) < \infty$ for all $\mathbf{x} \in S$ and $a \in A$.

Condition (1) is obviously satisfied by Υ, since $\hat{r}(\mathbf{x}, a) \leq \lambda\alpha^*$ for all state-action pairs $(\mathbf{x}, a) \in S \times A$. In order to verify the second condition, let θ_0 denote the degenerate policy which always chooses to balk. Then, under θ_0, all of the single-step rewards $\hat{r}(\mathbf{x}_n, a_n)$ are equal to zero and hence $\hat{v}_{\phi,\theta_0}(\mathbf{x}) = 0$ for all $\mathbf{x} \in S$, which in turn implies that $\hat{v}_\phi(\mathbf{x}) \geq 0$ since $\hat{v}_\phi(\mathbf{x}) \geq \hat{v}_{\phi,\theta_0}(\mathbf{x})$. Thus, condition (2) holds. In condition (3), one can simply take $K = 0$ and $\mathbf{z} = \mathbf{0}$. Indeed, Lemma 4.2.1 has shown that $\hat{v}_\phi(\mathbf{x}^{i+}) \leq \hat{v}_\phi(\mathbf{x})$ for all $\mathbf{x} \in S$ and $i \in \{1, 2, ..., N\}$, and therefore a trivial inductive argument shows that $\hat{v}_\phi(\mathbf{x}) - \hat{v}_\phi(\mathbf{0}) \leq 0$ for $\mathbf{x} \in S$. This is why the theorem states that $\hat{h}(\mathbf{x}) \leq 0$ for all $\mathbf{x} \in S$. Finally, the existence of a non-negative value $f(\mathbf{x})$ satisfying $\hat{h}_\phi(\mathbf{x}) \geq -f(\mathbf{x})$ for each $\mathbf{x} \in S$ and $\phi \in (0, 1)$ has been proved by Lemma 4.2.2. Since the action set A is finite and each value $f(\mathbf{x})$ is finite, it follows that $\sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y}) f(\mathbf{y}) < \infty$ for each $(\mathbf{x}, a) \in S \times A$.

Under the conditions stated, the theorem can be proved using the arguments given by Sennott [156] (p. 632), which are analogous to those required in the proof of Theorem 3.4.10. □

Interestingly, Sennott's proof of Theorem 4.2.3 (see [156]) involves obtaining an average reward optimal policy as a limit of ϕ-discount optimal policies as the discount factor ϕ tends to 1. The fact that an average reward optimal policy can be obtained in this manner offers the possibility of proving further results by exploiting the properties of discount-optimal policies.

The next result establishes the 'containment principle' of socially optimal policies.

Theorem 4.2.4. *The MDP Υ permits a stationary policy θ^* satisfying the optimality equations (4.2.6) which induces an ergodic Markov chain defined on a finite set of states $S_{\theta^*} \subseteq \tilde{S}$.*

Informally, it may be said that for some socially optimal policy θ^*, "the socially optimal state space is contained within the selfishly optimal state space".

Proof. Due to the definition of $\tilde{S}$ in (4.1.3) it is sufficient to show that for some stationary optimal policy θ^*, the action $\theta^*(\mathbf{x})$ prescribed under state $\mathbf{x} \in S$ is never to join facility i when $x_i = \tilde{B}_i$ (for $i = 1, 2, ..., N$). Let $(\phi_n)_{n\in\mathbb{N}}$ be a sequence of discount factors converging to 1, and let $(\theta^*_{\phi_n})_{n\in\mathbb{N}}$ be a corresponding sequence of discount-optimal policies; that is, for each $n \in \mathbb{N}$, $\theta^*_{\phi_n}$ is ϕ_n-discount optimal. Referring to the proof of Theorem 4.2.3 (see [156], p. 632), an average reward optimal policy θ^* may be obtained as a limit of some sequence $(\theta^*_{\phi_{n_k}})_{k\in\mathbb{N}}$ which is a subsequence of $(\theta^*_{\phi_n})_{n\in\mathbb{N}}$. It follows that for each $\mathbf{x} \in S$, there exists an integer $U(\mathbf{x})$ such that $\theta_{\phi_{n_k}}(\mathbf{x}) = \theta^*(\mathbf{x})$ for all $k \geq U(\mathbf{x})$, and therefore it suffices to show that for any discount factor $\phi \in (0, 1)$, the discount-optimal policy θ^*_ϕ forbids joining facility i at any state $\mathbf{x}$ with $x_i = \tilde{B}_i$.

For a contradiction, suppose $x_i = \tilde{B}_i$ and $\theta^*_\phi(\mathbf{x}) = i$ for some state $\mathbf{x} \in S$ and discount factor $\phi \in (0, 1)$. Then the discount optimality equations in (4.2.4) imply:

$$\hat{r}(\mathbf{x}, i) + \phi\lambda\Delta\hat{v}_\phi(\mathbf{x}^{i+}) \geq \phi\lambda\Delta\hat{v}_\phi(\mathbf{x}), \tag{4.2.7}$$

i.e. joining i is preferable to balking at state $\mathbf{x}$. Given that $x_i = \tilde{B}_i$, it follows that $\hat{r}(\mathbf{x}, i) < 0$ and therefore (4.2.7) implies $\hat{v}_\phi(\mathbf{x}^{i+}) > \hat{v}_\phi(\mathbf{x})$, but this contradicts Lemma 4.2.1. □

Theorem 4.2.4 may be regarded as a generalisation of the famous result of Naor [131] discussed in Chapter 2. Naor showed, in the context of a single $M/M/1$ queue, that the selfishly optimal and socially optimal strategies are threshold strategies with thresholds n_s and n_o respectively, and that $n_o \leq n_s$. This is the $M/M/1$ version of the containment property which Theorem 4.2.4 proves

for multiple, heterogeneous facilities (each with multiple service channels allowed). The theorem also implies that a socially optimal policy may be found by searching within the class of stationary policies 'contained' in the finite set $\tilde{S}$. Thus, it is possible to apply the techniques of dynamic programming discussed in Section 3.7 (e.g. value iteration, policy improvement) by restricting the state space so that it consists only of states in $\tilde{S}$; any policy that would cause a state outside $\tilde{S}$ to become positive recurrent may be ignored, since it is not necessary to consider such policies in order to find an optimal policy. For example, value iteration may be performed by looping over all states in $\tilde{S}$ on each iteration and simply restricting the set of actions so that joining facility i is not allowed at any state $\mathbf{x} \in \tilde{S}$ with $x_i = \tilde{B}_i$. This 'capping' technique avoids the use of alternative techniques which have been proposed in the literature for searching for optimal policies on infinite state spaces, such as the method of 'approximating sequences' proposed by Sennott in [158], or Ha's method of approximating the limiting behaviour of the value function in [73].

As discussed in Section 3.7, value iteration is useful for theoretical as well as practical purposes. The technique of proving structural properties of optimal policies using induction on the finite-stage value functions obtained using dynamic programming is well-established in the literature; for example, Hajek [74] used this technique to demonstrate the optimality of a 'switching-curve' policy in a model consisting of two interacting service stations, while Stidham [169] discussed the use of inductive analysis to establish the optimality of monotone policies in various types of problems involving control of admissions to a queueing system. Koole [105] (see also [106]) developed a more general approach for proving monotonicity properties of optimal policies based on analysing the form of the value function, rather than the specification of the function itself. Structural properties of optimal policies will be discussed in much greater depth in Chapter 5 of this thesis.

The proof of Theorem 4.2.4 makes use of a structural property of the value function $\hat{v}_\phi$ proved earlier (specifically, the monotonicity property proved in Lemma 4.2.1) in order to establish the existence of a socially optimal policy with the required 'containment' property. In fact, it is possible to use an alternative method of proof to establish an even stronger result. The proof of the next result uses a stochastic coupling argument; please refer to Section 3.8 for details of this approach, including definitions of the terms 'stochastic coupling' and 'sample path'.

Theorem 4.2.5. *Any stationary policy θ^* which maximises the expected long-run average reward*

for the process Υ induces an ergodic Markov chain on some set of states contained in $\tilde{S}$.

Proof. Let θ be a stationary policy which maximises (4.1.4). Evidently, there must exist a stationary distribution $\{\pi_\theta(\mathbf{x})\}_{\mathbf{x}\in S}$, where $\pi_\theta(\mathbf{x})$ is the steady-state probability of the process being in state $\mathbf{x} \in S$ under θ and $\sum_{\mathbf{x}\in S} \pi_\theta(\mathbf{x}) = 1$; if this were not the case, then the system would be unstable under θ and the expected long-run average reward would be (negatively) infinite.

Suppose, for a contradiction, that there exists a state $\mathbf{x} \notin \tilde{S}$ which is positive recurrent under θ. Clearly, such a state must be accessible from within $\tilde{S}$ and hence there exists $\mathbf{z} \in \tilde{S}$ with $z_i = \tilde{B}_i$ for some $i \in \{1, 2, ..., N\}$, $\pi_\theta(\mathbf{z}) > 0$ and $\theta(\mathbf{z}) = i$; that is, $\mathbf{z}$ is on the boundary of the selfish state space $\tilde{S}$ and is positive recurrent, and the decision taken at $\mathbf{z}$ causes the system to pass outside $\tilde{S}$. The aim of the proof is to show, using a sample path argument, that θ is inferior (in terms of average reward) to a particular *non-stationary* policy ψ, to be defined shortly.

Let Υ_1 and Υ_2 be discrete-time MDPs with the same formulation (i.e. the same rewards, transition probabilities etc.) as Υ. It will also be convenient to assume that both Υ_1 and Υ_2 are uniformised with parameter $\Delta = (\lambda + \sum_i c_i\mu_i)^{-1}$. The processes Υ_1 and Υ_2 evolve as follows:

- Υ_1 is initialised in an arbitrary state $\mathbf{x}_0 \in S$ and follows the policy θ at all times;
- Υ_2 is also initialised in state $\mathbf{x}_0$ and follows the policy ψ at all times.

Here, ψ is a non-stationary policy which chooses the same actions as θ at all times, *unless* Υ_1 is in state $\mathbf{z}$, in which case ψ chooses to balk instead of joining facility i. One might think of ψ as a 'copying' policy, which copies the actions chosen by θ unless Υ_1 is in state $\mathbf{z}$. Let $(\mathbf{x}_n)_{n\in\mathbb{N}_0}$ be the sequence of states visited by the first process Υ_1, and let $(\mathbf{y}_n)_{n\in\mathbb{N}_0}$ be the corresponding sequence for the second process Υ_2. In notation, ψ operates as follows (for all $n \in \mathbb{N}_0$):

$$\psi(\mathbf{y}_n) = \begin{cases} \theta(\mathbf{x}_n), & \text{if } \mathbf{x}_n \neq \mathbf{z}, \\ 0, & \text{if } \mathbf{x}_n = \mathbf{z}. \end{cases} \tag{4.2.8}$$

Assume that the processes $(\mathbf{x}_n)$ and $(\mathbf{y}_n)$ evolve according to identical sequences of random events; that is, the two processes follow the same *sample path* $(\omega_0, \omega_1, \omega_2, ...)$. The justification for this assumption is provided in Section 3.8 and depends on the implicit construction of a stochastic

coupling between the processes $(\mathbf{x}_n)$ and $(\mathbf{y}_n)$; please refer to the discussion beginning on page 119 for further explanation. At any time step n, the random event ω_n satisfies:

$$\omega_n \in \{\Lambda\} \cup \bigcup_{i=1}^{N} \left\{ M_i^{(k)} \right\}_{k=1}^{c_i}.$$

The random events Λ and $M_i^{(k)}$ (for $i \in \{1, 2, ..., N\}$ and $k \in \{1, 2, ..., c_i\}$) are defined below. In order to reduce the amount of unnecessary mathematical detail, the consequences of these events are described 'physically' rather than mathematically; for example, when it is said that the process $(\mathbf{x}_n)$ 'gains a customer at facility i', this naturally means that the i^{th} component of $\mathbf{x}_n$ increases by one, so that $\mathbf{x}_{n+1} \geq \mathbf{x}_n$ with strict inequality in only the i^{th} component.

- The event Λ occurs with probability $\lambda\Delta$ and causes an arrival to be seen by both of the marginal processes $(\mathbf{x}_n)$ and $(\mathbf{y}_n)$. In this case, $(\mathbf{x}_n)$ gains a customer at the facility corresponding to the action $\theta(\mathbf{x}_n)$ if this action is non-zero; similarly, $(\mathbf{y}_n)$ gains a customer at the facility corresponding to action $\psi(\mathbf{y}_n)$ if this action is non-zero.

- For $i = 1, 2, ..., N$ and $k = 1, 2, ..., c_i$, the event $M_i^{(k)}$ occurs with probability $\mu_i\Delta$. If this event occurs at the n^{th} time step, then the process $(\mathbf{x}_n)$ (respectively, $(\mathbf{y}_n)$) sees a service completion at facility i if and only if there are at least k customers present at i under state $\mathbf{x}_n$ (respectively, $\mathbf{y}_n$). Recall that any event that goes unseen by a particular process causes that process to remain in the same state at the next time step.

At any time step n, the number of possibilities for ω_n is $1 + \sum_{i=1}^{N} c_i$. Since it is assumed that $\Delta = (\lambda + \sum_i c_i\mu_i)^{-1}$, the probabilities of these events sum to 1 as required.

The general strategy of this proof is to show that the long-run average reward earned by Υ_2 along some arbitrary sample path is greater than the average reward earned by Υ_1 on the same path. Clearly, if this is the case for *any* sample path, then the policy ψ followed by Υ_2 must be superior to the policy θ followed by Υ_1 in terms of *expected* long-run average reward.

For all $n \in \mathbb{N}_0$, $\mathbf{x}_n$ dominates $\mathbf{y}_n$ in the sense that each facility i has at least as many customers present under $\mathbf{x}_n$ as under $\mathbf{y}_n$. This can be shown using an inductive argument:

- The (componentwise) inequality $\mathbf{x}_0 \geq \mathbf{y}_0$ holds trivially since the two processes Υ_1 and Υ_2 are initialised in the same state.

- Suppose $\mathbf{x}_n \geq \mathbf{y}_n$ for some $n \in \mathbb{N}_0$. Let $(\mathbf{x}_n)_j$ and $(\mathbf{y}_n)_j$ denote the j^{th} components of $\mathbf{x}_n$ and $\mathbf{y}_n$ respectively (for $j = 1, 2, ..., N$). If $\omega_n = \Lambda$ then, for some $j \in \{1, 2, ..., N\}$, $(\mathbf{y}_{n+1})_j$ may be one greater than $(\mathbf{y}_n)_j$; however, due to the definition of ψ in (4.2.8), this can only be the case if $(\mathbf{x}_{n+1})_j$ is also one greater than $(\mathbf{x}_n)_j$. Also, any event $M_j^{(k)}$ which causes $(\mathbf{x}_{n+1})_j$ to be *smaller* than $(\mathbf{x}_n)_j$ must also cause the same discrepancy to occur between $(\mathbf{y}_{n+1})_j$ and $(\mathbf{y}_n)_j$, *unless* $(\mathbf{y}_n)_j$ is strictly smaller than $(\mathbf{x}_n)_j$ (and also smaller than k), in which case $(\mathbf{y}_{n+1})_j$ is guaranteed to be less than or equal to $(\mathbf{x}_{n+1})_j$ because $(\mathbf{x}_{n+1})_j$ can differ from $(\mathbf{x}_n)_j$ by at most one. By these arguments, it follows that $\mathbf{x}_{n+1} \geq \mathbf{y}_{n+1}$.

The argument in the second of the bullet points above may be expressed more succinctly (albeit informally) as follows: any customer who joins facility j at a certain point during the evolution of Υ_2 also does so at the same point during the evolution of Υ_1. Any service completion which is seen by Υ_1 is also seen by Υ_2, unless the service completion in question takes place at some facility j which has more customers present under Υ_1 than under Υ_2, in which case there must still be at least as many customers present under Υ_1 as there are under Υ_2 after the service completion takes place. This establishes the required inequality, $\mathbf{x}_n \geq \mathbf{y}_n$ for $n \in \mathbb{N}_0$. Moreover, note that due to the definition of ψ, the only component for which this inequality may be strict is the i^{th} component, which corresponds to the action chosen by θ at state $\mathbf{z}$; in other words, facility i is the only facility at which there may be more customers present under Υ_1 than under Υ_2.

Consider the process Υ_1 initialised in state $\mathbf{x}_0$ and evolving along an arbitrary sample path $\boldsymbol{\omega} = (\omega_0, \omega_1, \omega_2, ...)$. Since the state $\mathbf{z}$ is positive recurrent under θ, $\mathbf{z}$ is visited infinitely many times by Υ_1 with probability one. Let $t_1 \in \mathbb{N}_0$ denote the first discrete time epoch at which Υ_1 is in state $\mathbf{z}$ and an arrival occurs (obviously, t_1 has a dependence on the path $\boldsymbol{\omega}$). That is:

$$t_1 := \inf \{n \in \mathbb{N}_0 : \mathbf{x}_n = \mathbf{z} \text{ and } \omega_n = \Lambda\}.$$

Assuming that the process Υ_2 is also initialised in state $\mathbf{x}_0$ and follows the same path $\boldsymbol{\omega}$, it must also be in state $\mathbf{z}$ at time t_1, since the 'copying' policy ψ chooses exactly the same actions as θ in the interval $[0, t_1)$ and the same pattern of arrivals and service completions occurs.

Let u_1 denote the next time, after t_1, that Υ_1 returns to the regenerative state $\mathbf{0}$. At time step t_1, the first process Υ_1 earns a *negative* reward $\hat{r}(\mathbf{z}, i)$ by choosing to join facility i when there are $\tilde{B}_i$ customers present, whereas Υ_2 earns a reward of zero by choosing to balk. Moreover, for

all $n \in (t_1, u_1]$ it may be seen from the formulation of $\hat{r}$ in (3.5.15) that the single-step reward $\hat{r}(\mathbf{x}_n, \theta(\mathbf{x}_n))$ earned by Υ_1 cannot possibly exceed the corresponding reward $\hat{r}(\mathbf{y}_n, \psi(\mathbf{y}_n))$ earned by Υ_2; this is because the presence of extra customers at facility i results in either a smaller reward being earned (if facility i is chosen) or an equal reward (if a different facility, or balking, is chosen). Therefore the total reward earned by Υ_1 over the interval $[t_1, u_1]$ must be strictly smaller than that earned by Υ_2. At time u_1, since Υ_1 has no customers present, neither does Υ_2, so the two processes are in the same state (this is referred to as a 'joining' of the two processes; in fact, depending on the path $\boldsymbol{\omega}$ being followed, the joining may occur at some point *before* u_1, but it must happen by time u_1 at the latest, given that $\mathbf{x}_n \geq \mathbf{y}_n$ for all $n \in \mathbb{N}_0$ and hence $\mathbf{x}_{u_1} = \mathbf{0} \geq \mathbf{y}_{u_1}$).

Using similar arguments, it can be said that if t_2 denotes the time of the next visit (after u_1) of Υ_1 to state $\mathbf{z}$, then Υ_2 must earn a strictly greater total reward than Υ_1 over the interval $[t_2, u_2]$, where u_2 is the time of the next visit (after t_2) of Υ_1 to state $\mathbf{0}$. As noted previously, the state $\mathbf{z}$ is visited infinitely often by Υ_1 with probability one. Hence, by repetition of this argument, it is easy to see that θ is strictly inferior to the non-stationary policy ψ in terms of expected long-run average reward. Furthermore, Theorem 4.2.3 states that an optimal stationary policy exists for Υ, so there must exist some other stationary policy θ^* which is superior to θ. $\square$

In the proof of Theorem 4.2.5, ψ is described as a 'copying policy' applied to a parallel process Υ_2, which simply copies the action chosen by θ on the process Υ_1, unless Υ_1 is in state $\mathbf{z}$. Given the nature of the dependence of ψ on θ, it is reasonable to question whether ψ is actually a *permissible* policy, in the sense that it fits within the classification of policies for MDPs introduced in Section 3.6. After all, what does it actually mean for a policy to 'copy' another policy? Does it mean that the policy ψ cannot be implemented in its own right, since it needs to 'copy' the actions of θ, and therefore it requires θ to be running side-by-side on some parallel process?

In fact, ψ is a permissible policy because it is an example of a *history-dependent* policy. The history of the process Υ, as defined in Section 3.6, includes a record of all states visited, actions chosen and random events occurring since the process began. At any point in time, given the initial state $\mathbf{x}_0$ and the history of random events, it is possible to determine what the latest system state *would* have been if the policy θ had been followed since initialisation; moreover, since θ is a stationary policy, the action $\theta(\mathbf{x}_n)$ is determined by the state $\mathbf{x}_n$ only. Therefore, at any discrete time step n,

the action $\psi(\mathbf{y}_n)$ chosen by ψ can be determined uniquely according to the history of the process and the definition of ψ itself; that is, ψ is an admissible history-dependent policy.

Finally, it is worthwhile to note that the proof of Theorem 4.2.5 assumed that the anticipatory reward formulation (3.5.15) was used. However, the lemma was intentionally stated without making reference to this assumption. Indeed, the equivalence of the reward formulations $\hat{r}(\mathbf{x}, a)$ and $r(\mathbf{x})$ with respect to average rewards under fixed stationary policies (see Theorem 3.5.3) ensures that it is sufficient to prove the result under only one of the two reward formulations.

The results in this section have shown that it is necessary for socially optimal policies to be 'contained' on the finite set $\tilde{S}$. The next section will establish that socially optimal policies should also possess another intuitively sensible property, referred to as a *non-idling* property.

4.3 Non-idling policies

The results in Section 4.2 have established that there exists a stationary socially optimal policy satisfying the optimality equations in (4.2.6), and that the set of states which are positive recurrent under this policy must be a subset of the *selfishly* optimal state space, $\tilde{S}$. It follows that a policy of this form may be found by searching within the finite set $\tilde{S}$, using the techniques of dynamic programming described in Section 3.7. In this section, θ^* will be used to denote the particular policy found using the Relative Value Iteration Algorithm (RVIA), which (due to Theorem 3.7.4) is average reward optimal. For each state $\mathbf{x} \in \tilde{S}$, the action $\theta^*(\mathbf{x})$ satisfies:

$$\theta^*(\mathbf{x}) \in \arg\max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h(\mathbf{y}) \right\}, \tag{4.3.1}$$

where the values $h(\mathbf{x})$ may be computed using relative value iteration. Note that (4.3.1) assumes that the real-time reward formulation (3.5.4) is used; under this formulation, the single-step reward $r(\mathbf{x})$ is independent of a. Moreover, the transition probabilities $p(\mathbf{x}, a, \mathbf{y})$ depend on a only to the extent that there is a probability $\lambda\Delta$ of transferring from state $\mathbf{x}$ to $\mathbf{x}^{a+}$ in a single time step (recall the notational convention that $\mathbf{x}^{0+} = \mathbf{x}$). Therefore, (4.3.1) may be simplified to:

$$\theta^*(\mathbf{x}) \in \arg\max_{a \in A_{\mathbf{x}}} h(\mathbf{x}^{a+}).$$

Hence, if there exists an action a^* such that $h(\mathbf{x}^{a^*+}) > h(\mathbf{x}^{a+})$ for all $a \in \{0, 1, 2, ..., N\} \setminus \{a^*\}$, then a^* is the action chosen by θ^* at state $\mathbf{x}$. This creates the possibility of using inductive arguments based on the finite-stage functions $h_n(\cdot)$ defined in Theorem 3.7.4 to show that certain actions *cannot* be chosen by θ^* at particular states. In particular, if it can be shown that $h_n(\mathbf{x}^{a_1+}) - h_n(\mathbf{x}^{a_2+}) \geq \epsilon$ for some actions $a_1, a_2 \in \{0, 1, 2, ..., N\}$ and constant $\epsilon > 0$ at all stages $n \in \mathbb{N}_0$, then since $h(\mathbf{x}) = \lim_{n\to\infty} h_n(\mathbf{x})$, it follows that a_2 cannot be the action chosen by θ^* at $\mathbf{x}$.

The result to be proved shortly utilises these arguments in order to establish further properties of the policy θ^* which will be useful in later chapters. First, it is necessary to introduce a definition for a particular type of stationary policy, referred to as a *non-idling* policy.

Definition 4.3.1. ***(Non-idling policies)***

Let θ be a stationary policy, and suppose $\theta(\mathbf{x}) \neq 0$ for all states $\mathbf{x} \in S$ with $x_i < c_i$ for at least one facility $i \in \{1, 2, ..., N\}$; that is, θ never chooses to balk when there is at least one facility with one or more idle service channels. Then θ is said to be a non-idling *policy.*

In this section it will be shown that the policy θ^* obtained by relative value iteration is a non-idling policy. First, a property of the relative value function $h(\mathbf{x})$ must be proved.

Lemma 4.3.2. *Let $\mathbf{x} \in \tilde{S}$ be a state with $x_i < c_i$ for some $i \in \{1, 2, ..., N\}$. Then:*

$$h(\mathbf{x}^{i+}) > h(\mathbf{x}),$$

where h is the relative value function satisfying the equations (4.2.1).

Proof. The proof is accomplished using induction on the functions $h_n(\cdot)$ defined in (3.7.7). Indeed, it can be shown that for all states $\mathbf{x} \in \tilde{S}$ with $x_i < c_i$ for some $i \in \{1, 2, ..., N\}$:

$$h_n(\mathbf{x}^{i+}) - h_n(\mathbf{x}) \geq \alpha_i \mu_i - \beta_i \qquad \forall\, n \in \mathbb{N}_0,$$

which is sufficient to establish the result. For details, refer to Appendix A.4, page 425.

Lemma 4.3.2 directly implies the next theorem.

Theorem 4.3.3. *There exists a socially optimal policy θ^* which is a non-idling policy.*

Proof. Consider the stationary policy θ^* obtained by relative value iteration on the finite state space $\tilde{S}$, using the real-time reward formulation defined in (3.5.4). As discussed earlier, the action $\theta^*(\mathbf{x})$ chosen by policy θ^* at any state $\mathbf{x} \in \tilde{S}$ satisfies:

$$\theta^*(\mathbf{x}) \in \arg\max_{a \in A_{\mathbf{x}}} h(\mathbf{x}^{a+}). \tag{4.3.2}$$

For any state $\mathbf{x} \in \tilde{S}$ with $x_i < c_i$ for some $i \in \{1, 2, ..., N\}$, Lemma 4.3.2 states that $h(\mathbf{x}^{i+}) > h(\mathbf{x})$, and hence the action of balking fails to attain the maximum in (4.3.2). Moreover, one can assume that the actions chosen by θ^* at states $\mathbf{x} \notin \tilde{S}$ satisfy the requirement that balking is not chosen at any state with $x_i < c_i$ for some $i \in \{1, 2, ..., N\}$, since these decisions do not affect the average reward earned by θ^*. Hence, θ^* is a non-idling policy which also attains social optimality due to Theorem 3.7.4 and the results proved in Section 4.2, which imply that a socially optimal policy can be found by restricting attention to the process $\tilde{\Upsilon}$ with finite state space $\tilde{S}$. □

It is worth noting that the definition of non-idling policies (Definition 4.3.1) does not make it possible to say that *any* stationary socially optimal policy must be a non-idling policy. Indeed, as noted in the proof of Theorem 4.3.3, the decisions chosen at states $\mathbf{x} \notin \tilde{S}$ by the policy θ^* cannot affect its optimality. Since the set $S \setminus \tilde{S}$ includes infinitely many states with $x_i < c_i$ for at least one $i \in \{1, 2, ..., N\}$, one can simply consider a stationary policy θ which matches θ^* at all states $\mathbf{x} \in \tilde{S}$ but chooses to balk at some transient state $\mathbf{x} \notin \tilde{S}$ with $x_i < c_i$ for some $i \in \{1, 2, ..., N\}$. The resulting policy θ is socially optimal, but obviously is not a non-idling policy. However, it is possible to prove a slightly weaker property which will prove to be useful later.

Theorem 4.3.4. *Let θ^* be any stationary socially optimal policy, and let S_{θ^*} be the set of states which are positive recurrent under θ^*. Let the set $S^\circ \subseteq \tilde{S}$ be defined as follows:*

$$S^\circ = \left\{ \mathbf{x} \in \tilde{S} : x_i \leq c_i \; \forall \; i \in \{1, 2, ..., N\} \right\}. \tag{4.3.3}$$

That is, S° is the set of states at which no customers are waiting in queues. Then:

$$S^\circ \subseteq S_{\theta^*}.$$

Proof. Let θ^* be a stationary socially optimal policy. Suppose the process is in the regenerative state $\mathbf{0}$ at some discrete time epoch $n_0 \in \mathbb{N}_0$ and then a sequence of $\tilde{B} + 1$ consecutive customer arrivals occurs (without any service completions), where $\tilde{B} = \sum_{i=1}^{N} \tilde{B}_i$ and $\tilde{B}_i$ is the selfish threshold

for facility i defined in (4.1.2). Recall that, by Theorem 4.2.5, it must be the case that $S_{\theta^*} \subseteq \tilde{S}$ and therefore the policy θ^* must choose to balk at least once during this sequence of $\tilde{B}+1$ arrivals, otherwise the process would pass outside the set $\tilde{S}$. Let $n_1 \geq n_0$ denote the first time epoch after n_0 at which balking is chosen. Consider the following two possibilities:

1. At time n_1, θ^* chooses to balk at some $\mathbf{x} \in \tilde{S}$ with $x_i \geq c_i$ for all $i \in \{1, 2, ..., N\}$.
2. At time n_1, θ^* chooses to balk at some $\mathbf{x} \in \tilde{S}$ with $x_i < c_i$ for some $i \in \{1, 2, ..., N\}$.

In the first case, the state $\mathbf{x}$ (with $x_i \geq c_i$ for all i) must be positive recurrent under θ^*, since it is accessible from $\mathbf{0}$. Furthermore, all states $\mathbf{y} \in \tilde{S}$ which satisfy the componentwise inequality $y_i \leq x_i$ for all $i \in \{1, 2, ..., N\}$ are also positive recurrent under θ^*, since they are accessible from $\mathbf{x}$ via an appropriate sequence of service completions. This implies that all states belonging to S° are positive recurrent under θ^*, so in this case the claim of the theorem is proved.

In the second case, one may again note that the state $\mathbf{x}$ is positive recurrent under θ^*, since it is accessible from $\mathbf{0}$. The proof will proceed that by showing that, in this case, a contradiction is obtained with the fact that θ^* is an optimal policy. This will be accomplished using a sample path argument, somewhat similar to the proof of Theorem 4.2.5 except that in this case it will be necessary to make use of the real-time reward formulation (used throughout this section) rather than the anticipatory formulation. Let ψ denote a *non-stationary* policy which operates by 'copying' the actions of policy θ^* at all times, *unless* either of the following scenarios apply:

1. θ^* chooses to balk at the state $\mathbf{x}$ (with $x_i < c_i$), in which case ψ chooses to join facility i if the process following policy ψ is also in state $\mathbf{x}$, and otherwise chooses to balk.
2. θ^* chooses to join facility i at some point in time at which the process following policy ψ has more customers at i than the process following θ^*, in which case ψ chooses to balk.

Let $(\mathbf{x}_n)_{n\in\mathbb{N}_0}$ and $(\mathbf{y}_n)_{n\in\mathbb{N}_0}$ denote the state evolutions of two processes Υ_1 and Υ_2 which follow policies θ^* and ψ respectively. Then, in notation, ψ operates as follows:

$$\psi(\mathbf{y}_n) = \begin{cases} i, & \text{if } \mathbf{x}_n = \mathbf{y}_n = \mathbf{x}, \\ 0, & \text{if } \theta^*(\mathbf{x}_n) = i \text{ and } (\mathbf{x}_n)_i < (\mathbf{y}_n)_i, \\ \theta^*(\mathbf{x}_n), & \text{otherwise,} \end{cases}$$

where $(\mathbf{x}_n)_i$ and $(\mathbf{y}_n)_i$ denote the i^{th} components of $\mathbf{x}_n$ and $\mathbf{y}_n$ respectively. Suppose that both of the processes Υ_1 and Υ_2 are initialised in state $\mathbf{0}$ and evolve according to the same sequence of random events $\boldsymbol{\omega} = (\omega_0, \omega_1, ...)$ (again, this relies upon the implicit idea of a stochastic coupling between the two processes, as explained in Section 3.8). Let $t_0 \geq 0$ be the first discrete time epoch at which both processes are in state $\mathbf{x}$ and an arrival occurs. Then, $t_0 + 1$ will be the first point in time at which the two processes are not in the same state. Noting that $x_i < c_i$, it follows from the reward formulation (3.5.4) that the process Υ_2 following ψ will earn a single-stage reward at time epoch $t_0 + 1$ which is strictly greater than that of Υ_1, since it has an extra customer being served at facility i. Furthermore, let u_0 denote the next time epoch (after t_0) at which the processes Υ_1 are Υ_2 are once again in the same state (referred to as a *joining* of the two processes); this may occur as a result of a service completion at facility i being seen by Υ_2 but not by Υ_1, or as a result of θ^* choosing to join facility i while the process Υ_2 already has an extra customer present at i. In either case, it can easily be checked using the definition of the policy ψ that:

$$(\mathbf{x}_n)_i < (\mathbf{y}_n)_i \leq c_i \quad \forall\, n \in [t_0 + 1, u_0 - 1]. \tag{4.3.4}$$

Noting that $u_0 \geq t_0 + 2$ (since the two processes must first 'un-join' before joining again), it follows from (4.3.4) and the reward formulation (3.5.4) that Υ_2 earns a strictly greater total reward than Υ_1 over the interval $[t_0, u_0]$. Since $\mathbf{x}$ is positive recurrent under θ^*, the two processes make infinitely many visits to state $\mathbf{x}$ with probability one. Hence, one may define t_n as the n^{th} time that an arrival occurs when both processes are in state $\mathbf{x}$, and u_n as the next time (after t_n) that the two processes 'join' again, with the result that Υ_2 earns a strictly greater total reward than Υ_1 over the interval $[t_n, u_n]$ for each $n \in \mathbb{N}$. It then follows that the policy ψ is superior to θ^* with respect to the *average* reward criterion, which contradicts the fact that θ^* is a socially optimal policy.

These arguments confirm that if θ^* is a stationary socially optimal policy, then there must be a positive recurrent state $\mathbf{x} \in \tilde{S}$ with $x_i \geq c_i$ for all $i \in \{1, 2, ..., N\}$, which in turn implies that all states in S° are positive recurrent under θ^*. This completes the proof. $\square$

Theorem 4.3.4 may be regarded as a complement of Theorem 4.2.5, since it effectively establishes a lower bound for the set S_{θ^*} associated with a socially optimal policy θ^*, whereas Theorem 4.2.5 established an upper bound. By combining these two results, one can conclude:

$$S^\circ \subseteq S_{\theta^*} \subseteq \tilde{S},$$

for any stationary socially optimal policy θ^*. In Section 4.5, the 'containment' property will be generalised to a problem involving heterogeneous customers. However, the topic of the next section will be the effect of varying the demand rate λ on socially optimal policies.

4.4 Relationships with demand

Section 4.1 introduced the definitions of *selfishly optimal* and *socially optimal* policies. It is of interest to investigate how optimal policies are affected by varying the system demand rate λ. In particular, the sub-optimality of the selfish policy $\tilde{\theta}$ in the context of overall social welfare has important implications in applications where customers may or may not be given freedom of control over their own outcomes. If it can be shown that this sub-optimality increases or decreases as λ becomes larger, the insights gained can potentially be useful (see [101] for applications in healthcare systems). It will also be of interest to examine light-traffic and heavy-traffic limits ($\lambda \to 0$ and $\lambda \to \infty$ respectively) and the characterisation of optimal policies in these limits.

It was already noted in Section 4.1 that the selfish policy $\tilde{\theta}$ is independent of the demand rate λ. As such, the focus in this section will be on socially optimal policies. It should be noted that throughout this section, 'optimal' means 'socially optimal' unless stated otherwise.

The next result shows that if socially optimal policies are followed, then increasing the demand rate cannot be harmful in terms of the expected long-run average reward earned.

Theorem 4.4.1. *The optimal expected long-run average reward, g^* which satisfies the optimality equations in (4.2.6) is monotonically increasing with the demand rate λ.*

Proof. Consider two demand rates λ_1 and λ_2, with $\lambda_1 < \lambda_2$. Let θ_1^* denote an average reward optimal stationary policy under demand rate λ_1; such a policy is known to exist by Theorem 4.2.3. Let θ_2^* be a *randomised* policy which, when in state $\mathbf{x} \in S$, operates as follows:

- With probability $(\lambda_2 - \lambda_1)/\lambda_2$, θ_2^* chooses to balk, regardless of the state $\mathbf{x}$.
- With probability λ_1/λ_2, θ_2^* chooses the same action $\theta_1^*(\mathbf{x})$ that would be chosen by the stationary policy θ_1^* at the current state $\mathbf{x} \in S$.

Consider the system operating under policy θ_2^* with demand rate λ_2. This policy operates by *splitting* the original Poisson process by which customers arrive, so that there are two separate Poisson processes with demand rates $\lambda_2 - \lambda_1$ and λ_1 respectively. Customers arriving via the first process (with demand rate $\lambda_2 - \lambda_1$) invariably balk, and therefore make no contribution to the aggregate reward and have no effect on the state of the system. Thus, the only customers who *may* join a facility and receive service are those who arrive via the second Poisson process (with demand rate λ_1), and these customers are subject to the same state-dependent routing decisions as those who follow the stationary policy θ_1^*. It follows that the policy θ_2^* earns exactly the same expected long-run average reward under demand rate λ_2 as θ_1^* earns under demand rate λ_1.

An *optimal* policy under demand rate λ_2 must perform at least as well as the policy θ_2^*; that is, the optimal expected long-run average reward under λ_2 must be at least as great as the average reward earned by θ_1^* under the smaller demand rate λ_1, which (by definition of θ_1^*) is also the *optimal* average reward under λ_1. This completes the proof of the theorem. □

The proof of Theorem 4.4.1 essentially argues that when the demand rate is increased from λ_1 to λ_2, it is always possible to 'match' the performance of an optimal stationary policy under λ_1 by simply 'forcing the extra traffic to balk'. It should also be noted that if a *randomised* policy is able to achieve a certain expected long-run average reward, then (by Theorem 4.2.3) there must also exist a *stationary* policy which yields at least the same value. Hence, the statement of Theorem 4.4.1 is true whether or not one restricts attention to stationary policies.

As discussed earlier in this chapter, it is reasonable to regard a *selfish* policy $\tilde{\theta}$ as a policy which is followed when customers make decisions as individuals, whereas a *socially* optimal policy θ^* is a consequence of exercising central control in such a way as to maximise the collective welfare of all customers. A natural extension of this idea is to consider a scenario in which *some* customers, but not all, are subject to central control. That is, a certain proportion $q \in [0, 1]$ of customers are 'selfish' and follow decisions which maximise their individual expected net reward as defined in (4.1.1), whereas the remaining proportion $(1 - q)$ act in order to maximise the expected long-run average reward, taking due account of the actions of selfish customers. The next result captures the natural idea that the optimal average reward g^* is monotonically decreasing with q, the proportion of selfish customers. The proof is somewhat similar to that of Theorem 4.4.1.

Theorem 4.4.2. *Consider an MDP which is identical to* Υ *except that there are two Poisson arrival streams: a 'selfish' stream with demand rate* $q\lambda$ *and a 'social' stream with demand rate* $(1-q)\lambda$*, where* $q \in [0,1]$ *is the proportion of selfish customers. Customers arriving via the selfish stream when the system is in state* $\mathbf{x} \in S$ *choose an action* $a \in \arg\max_a w(\mathbf{x}, a)$*, where:*

$$w(\mathbf{x},a) = \begin{cases} \alpha_i - \dfrac{\beta_i}{\mu_i}, & \text{if } a = i \text{ for some } i \in \{1,2,...,N\} \text{ and } x_i < c_i, \\ \alpha_i - \dfrac{\beta_i(x_i+1)}{c_i\mu_i}, & \text{if } a = i \text{ for some } i \in \{1,2,...,N\} \text{ and } x_i \geq c_i, \\ 0, & \text{otherwise.} \end{cases}$$

Meanwhile, customers arriving via the social stream follow a 'sub-policy' ψ *which may or may not be stationary. Let* $g_\psi(\mathbf{x})$ *denote the average reward given an initial state* $\mathbf{x} \in S$ *when the sub-policy* ψ *is followed, and let* $g^*(\mathbf{x}) = \sup_\psi g_\psi(\mathbf{x})$ *denote the optimal average reward over all permissible sub-policies. Then, for all* $\mathbf{x} \in S$*,* $g^*(\mathbf{x})$ *is monotonically decreasing with* q*.*

Proof. It will be somewhat easier from a mathematical point of view to consider a system with a single Poisson arrival stream λ operating under a randomised policy θ, which selects 'selfish' actions with probability q and other (possibly non-selfish) actions with probability $1-q$.

Consider two different values $q_1, q_2 \in [0,1]$, with $q_1 < q_2$. It is sufficient to show that, given any policy θ_2 which earns an average reward $g_{\theta_2}(\mathbf{x})$ when $q = q_2$, there exists a policy θ_1 which earns the same average reward $g_{\theta_2}(\mathbf{x})$ when $q = q_1$. Note that the only restriction on the policy θ_1 is that selfish decisions must be chosen with a probability of at least q_1, which is smaller than q_2. Hence, one can simply define θ_1 in such a way that selfish decisions are chosen with probability q_2 and decisions are taken otherwise (with probability $1-q_2$) which are identical to those which would be chosen by θ_2 given the same current state and process history, with the result that θ_1 is exactly equivalent to θ_2 and earns the same average reward. This completes the proof. □

The statement of Theorem 4.4.2 assumes the existence of an optimal 'sub-policy' ψ, for customers arriving via the social stream. As the proof of the theorem makes clear, an optimal sub-policy followed by the social customers *only* is equivalent to an optimal policy followed by *all* customers which operates subject to the constraint that *selfishly* optimal decisions are chosen with probability q or greater. One might argue, therefore, that the results of this thesis thus far are not sufficient

to guarantee the existence of ψ, since they do not address the existence of constrained, randomised policies which are optimal within the class of similarly-constrained policies.

In considering how previous results might be adapted to deal with the scenario where a certain proportion q of customers make selfish decisions, it is useful to observe that, given any current state $\mathbf{x} \in S$, there is a minimum probability $q\lambda\Delta$ of transferring to the state $\mathbf{x}^{\tilde{\theta}(\mathbf{x})+}$ in a single time step, where $\tilde{\theta}(\mathbf{x})$ is the selfishly optimal decision under state $\mathbf{x}$. There still exists a Poisson stream of arrivals whose destinations are fully controllable, but these customers arrive at a rate $(1-q)\lambda$ rather than λ. The finite-stage discount optimality equations from (A.4.2) can be modified in the following way to take into account the behaviour of selfish customers:

$$\hat{v}_\phi^{(n+1)}(\mathbf{x}) = \max_a \left\{ (1-q)\lambda w(\mathbf{x},a) + \phi(1-q)\lambda\Delta\hat{v}_\phi^{(n)}(\mathbf{x}^{a+}) \right\} + \phi q\lambda\Delta\hat{v}_\phi^{(n)}(\mathbf{x}^{\tilde{\theta}(\mathbf{x})+})$$
$$+ \phi\sum_{i=1}^{N}\min(x_i,c_i)\mu_i\Delta\hat{v}_\phi^{(n)}(\mathbf{x}^{i-}) + \phi\left(1-\lambda\Delta-\sum_{i=1}^{N}\min(x_i,c_i)\mu_i\Delta\right)\hat{v}_\phi^{(n)}(\mathbf{x}), \quad (4.4.1)$$

where $w(\mathbf{x},a)$ is as defined in the statement of Theorem 4.4.2. One may then proceed to verify that the conditions of Theorem 4.2.3 (which guarantee the existence of an average reward optimal stationary policy) hold when the value functions $\hat{v}_\phi^{(n)}$ are defined as in (4.4.1); however, the proofs of Lemmas 4.2.1 and 4.2.2 will require some modifications. Details of these modifications will not be provided here; instead, it suffices to note that an alternative approach would involve using an argument very similar to the sample path argument given in the proof of Theorem 4.2.5 to show that a policy which is not 'contained' on the selfish state space $\tilde{S}$ cannot be optimal. This provides a justification for searching for an optimal policy within the finite set $\tilde{S}$, and it then follows by Theorem 3.7.4 that an optimal stationary policy exists which is contained on $\tilde{S}$.

The next example illustrates how the proportion of selfish customers $q \in [0,1]$ affects the optimal expected long-run average reward for various different values of the demand rate λ.

Example 4.4.3. ***(Mixtures of selfish and non-selfish customers)***

Consider a system with 3 facilities, with the parameters shown below:

$$c_1 = 2, \qquad \mu_1 = 2, \qquad \beta_1 = 4, \qquad \alpha_1 = 10,$$

$$c_2 = 2, \qquad \mu_2 = 3, \qquad \beta_2 = 3, \qquad \alpha_2 = 6,$$
$$c_3 = 1, \qquad \mu_3 = 4, \qquad \beta_3 = 7, \qquad \alpha_3 = 20.$$

Figure 4.1 illustrates the relationship between the optimal expected long-run average reward and the proportion of selfish customers, $q \in [0, 1]$, for various different values of the demand rate $\lambda > 0$. To clarify, it is assumed (as in Theorem 4.4.2) that *non-selfish* customers follow an optimal 'sub-policy' which takes due account of the actions of selfish customers; they do not simply follow the same policy that they would follow if *all* customers acted to maximise the average reward. The optimal average rewards shown in the figure have been computed using relative value iteration as described in Section 3.7, with the recursive equations (3.7.7) modified in an appropriate way in order to take into account transitions caused by the actions of selfish customers.

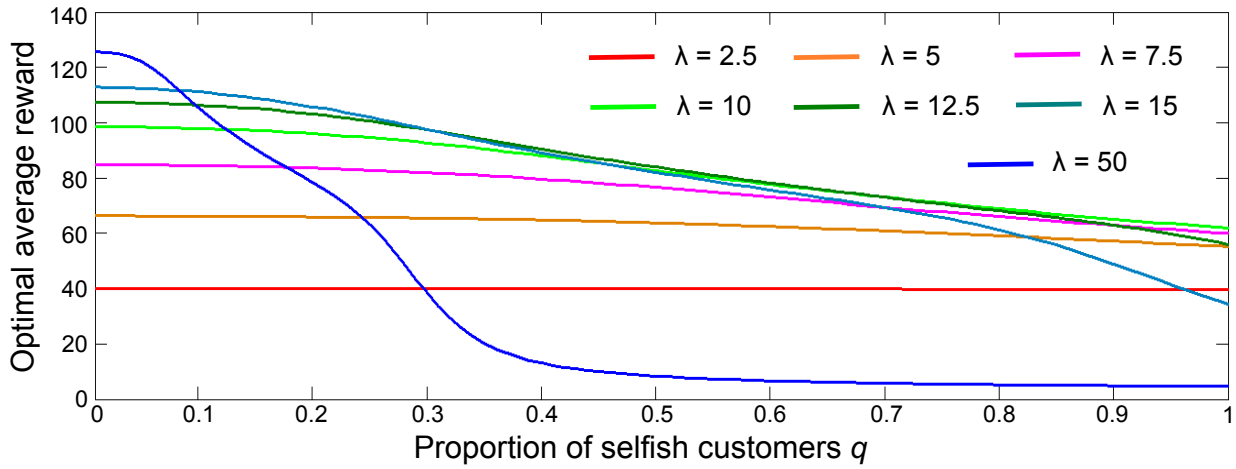


Figure 4.1: The relationship between the proportion of selfish customers $q \in [0, 1]$ and the optimal expected long-run average reward for various different values of the demand rate λ.

Figure 4.1 shows that, as expected, the optimal expected long-run average reward is monotonically decreasing with q for all of the demand rates considered. It is somewhat interesting to note that the effect of selfish decision-making appears to become more dramatic as the demand rate increases; indeed, the line in the figure representing $\lambda = 2.5$ is almost horizontal, indicating that the average reward is almost unaffected by whether or not customers make selfish decisions, whereas the average rewards for larger values of λ appear to decrease much more steeply with q. The case $\lambda = 50$ is an extreme case, in which the value of λ is much greater than the system's maximum service completion rate, $\sum_{i=1}^{N} c_i \mu_i$. The fact that selfish decision-making appears to be almost socially

optimal for small values of λ will be explained by the next result in this section.

It may also be noted that Figure 4.1 is consistent with the result of Theorem 4.4.1, which implies that when *all* customers are non-selfish (i.e. $q = 0$), the optimal average reward is monotonically increasing with the demand rate λ. However, the figure also shows that when a non-zero proportion of customers make selfish decisions, the optimal average reward (given by non-selfish customers following an optimal sub-policy) may *not* be monotonically increasing with λ. ⊠

The next results in this section address the characterisation of average reward optimal (i.e. socially optimal) policies in light-traffic and heavy-traffic limits. As discussed in Section 4.1, a *selfishly* optimal policy may be regarded as a simple heuristic rule which maximises *immediate* rewards, i.e. the expected net rewards of newly-arriving customers, without taking due account of longer-term consequences. Specifically, the 'longer-term consequences' of a selfish customer's decision are the potential increases in the waiting times of customers who arrive *after* their decision has been made, and the possible re-routing of future customers (with implications for the aggregate rewards and holding costs incurred by the system). It is clear that, although a selfish customer's action may have undesirable effects on the state of the system for a certain period of time, this period of time will only be *finite*; indeed, under any stationary policy (including the selfishly optimal policy) the state $\mathbf{0}$ is positive recurrent and so the system must eventually return to the state of being empty, at which point the process regenerates. Thus, it is natural to suppose that if the demand rate λ is *small*, then the 'long-term' consequences of a selfish customer's decision will be minimal, for the intuitively simple reason that only a small number of customers will arrive during the finite period of time in which their outcomes stand to be affected by an earlier selfish decision.

Building upon these arguments, the next result shows that in fact, the selfishly optimal policy is *asymptotically* average reward optimal in a light-traffic limit, i.e. as the demand rate λ tends to zero. The concept of 'asymptotic optimality' requires some explanation. It can trivially be shown that for any permissible policy θ, the expected long-run average reward $g_\theta(\mathbf{x})$ (given any initial state $\mathbf{x} \in S$) tends to zero as $\lambda \to 0$. It follows that, for any arbitrary policy θ:

$$\lim_{\lambda \to 0} \left(g_\theta(\mathbf{x}) - g^* \right) = 0 \qquad \forall\, \mathbf{x} \in S,$$

where g^* is the optimal value for the average reward which satisfies the equations (4.2.6). Therefore, in order to show that the selfishly optimal policy $\tilde{\theta}$ is light-traffic optimal in a sense that is actually

meaningful, it is necessary to show that the *relative* sub-optimality (or percentage sub-optimality) of $\tilde{\theta}$ tends to zero as $\lambda \to 0$. The next theorem formalises this result.

Theorem 4.4.4. *Given a demand rate $\lambda > 0$, let $\tilde{g}(\lambda)$ denote the expected long-run average reward attained by the selfishly optimal policy $\tilde{\theta}$, and let $g^*(\lambda)$ denote the optimal value for the average reward which satisfies the equations (4.2.6). Then:*

$$\lim_{\lambda \to 0} \left(\frac{g^*(\lambda) - \tilde{g}(\lambda)}{g^*(\lambda)} \right) = 0. \tag{4.4.2}$$

That is, the policy $\tilde{\theta}$ is asymptotically (socially) optimal in a light-traffic limit.

Proof. Throughout this proof, it will be assumed that single-stage rewards for the MDP Υ are given by the *anticipatory* formulation (3.5.15). Recall that $\alpha_i > \beta_i/\mu_i$ for all $i \in \{1, 2, ..., N\}$; as discussed in Section 3.1, this is an assumption made in order to avoid degenerate cases where at least one facility fails to offer any reasonable incentive for joining (even when it is empty). It follows that, given any $\lambda > 0$, $g^*(\lambda)$ must be strictly positive. This can be shown by observing that it is possible to construct a stationary policy which earns a strictly positive average reward. Indeed, consider the stationary policy θ with $\theta(\mathbf{0}) = i \neq 0$ and $\theta(\mathbf{x}) = 0$ for all $\mathbf{x} \neq \mathbf{0}$, i.e. joining facility i is chosen when the system is empty and balking is chosen otherwise. Since $\alpha_i > \beta_i/\mu_i$, it follows that $\hat{r}(\mathbf{0}, i) > 0$. Then, since $\mathbf{0}$ and $\mathbf{0}^{i+}$ are the only positive recurrent states under θ, the average reward under θ is given by $\pi_\theta(\mathbf{0})\,\hat{r}(\mathbf{0}, i) + \pi_\theta(\mathbf{0}^{i+})\,\hat{r}(\mathbf{0}^{i+}, 0)$, where $\pi_\theta(\mathbf{0})\,\hat{r}(\mathbf{0}, i)$ is positive (since the stationary probability $\pi_\theta(\mathbf{0})$ is non-zero) and $\pi_\theta(\mathbf{0}^{i+})\,\hat{r}(\mathbf{0}^{i+}, 0) = 0$. Since $g^*(\lambda)$ must be at least as great as the average reward under θ, it then follows that $g^*(\lambda) > 0$ for all $\lambda > 0$.

Next, consider a fixed demand rate $\lambda > 0$. The selfishly optimal policy $\tilde{\theta}$, as defined in Section 4.1, is independent of λ and induces an irreducible, ergodic Markov chain defined on the finite set $\tilde{S}$. By Theorem 4.2.4, there exists a stationary policy θ^*_λ (with a dependence on λ) which attains the optimal average reward $g^*(\lambda)$, and this policy induces an irreducible, ergodic Markov chain defined on some finite set $S_{\theta^*_\lambda}$ which is contained in $\tilde{S}$. Hence, $\tilde{g}(\lambda)$ and $g^*(\lambda)$ can be expressed in terms of the stationary distributions under policies $\tilde{\theta}$ and θ^*_λ respectively as follows:

$$\begin{aligned} \tilde{g}(\lambda) &= \sum_{\mathbf{x} \in \tilde{S}} \pi_{\tilde{\theta}}(\mathbf{x})\,\hat{r}\big(\mathbf{x}, \tilde{\theta}(\mathbf{x})\big), \\ g^*(\lambda) &= \sum_{\mathbf{x} \in \tilde{S}} \pi_{\theta^*_\lambda}(\mathbf{x})\,\hat{r}\big(\mathbf{x}, \theta^*_\lambda(\mathbf{x})\big), \end{aligned} \tag{4.4.3}$$

where the dependence of the stationary distributions $\{\pi_{\tilde{\theta}}(\mathbf{x})\}$ and $\{\pi_{\theta^*_\lambda}(\mathbf{x})\}$ on λ has been suppressed for notational convenience. Note that $\pi_{\tilde{\theta}}(\mathbf{x}) = \pi_{\theta^*_\lambda}(\mathbf{x}) = 0$ for all $\mathbf{x} \notin \tilde{S}$ due to Theorem 4.2.5, which is why the summations in (4.4.3) can be taken over $\tilde{S}$ rather than S. Recall that $\hat{r}(\mathbf{x}, a) = \lambda w(\mathbf{x}, a)$ for any state-action pair $(\mathbf{x}, a) \in S \times A$, where $w(\mathbf{x}, a)$ is the expected net reward for an individual customer defined in (4.1.1). Hence, following cancellation of λ, one can write:

$$\frac{g^*(\lambda) - \tilde{g}(\lambda)}{g^*(\lambda)} = \frac{\sum_{\mathbf{x}\in\tilde{S}} \pi_{\theta^*_\lambda}(\mathbf{x})\, w\big(\mathbf{x}, \theta^*_\lambda(\mathbf{x})\big) - \sum_{\mathbf{x}\in\tilde{S}} \pi_{\tilde{\theta}}(\mathbf{x})\, w\big(\mathbf{x}, \tilde{\theta}(\mathbf{x})\big)}{\sum_{\mathbf{x}\in\tilde{S}} \pi_{\theta^*_\lambda}(\mathbf{x})\, w\big(\mathbf{x}, \theta^*_\lambda(\mathbf{x})\big)}. \tag{4.4.4}$$

The advantage of using the formulation in (4.4.4) is that, for all pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$, $w(\mathbf{x}, a)$ (unlike $\hat{r}(\mathbf{x}, a)$) is independent of λ. Next, it will be useful to characterise the stationary distributions $\{\pi_{\tilde{\theta}}(\mathbf{x})\}$ and $\{\pi_{\theta^*_\lambda}(\mathbf{x})\}$ in the light-traffic limit. Naturally, one ought to have:

$$\lim_{\lambda\to 0} \pi_{\tilde{\theta}}(\mathbf{x}) = \lim_{\lambda\to 0} \pi_{\theta^*_\lambda}(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} = \mathbf{0}, \\ 0, & \text{otherwise.} \end{cases} \tag{4.4.5}$$

Indeed, the limits in (4.4.5) can be proved using stochastic bounding arguments. For convenience, define $\tilde{B} := \sum_i \tilde{B}_i$, with $\tilde{B}_i$ as defined in (4.1.2), and $\mu_{\min} := \min(\mu_1, \mu_2, ..., \mu_N)$. Suppose the system is initialised in state $\mathbf{0}$. Then, under the selfish policy $\tilde{\theta}$, customers balk at a particular state $\mathbf{x} \in S$ if and only if $\sum_i x_i = \tilde{B}$. Hence, given any $\lambda > 0$, the total number of customers present in the system is stochastically bounded above by the number of customers present in an $M/M/1/\tilde{B}$ queue with demand rate λ, service rate $\mu_{\min}$ and finite capacity $\tilde{B}$.

To show this in greater detail, first note that the $M/M/1/\tilde{B}$ queue can be discretised using the technique of uniformisation described in Section 3.3 and choosing the discrete step size Δ to be any positive value not exceeding $(\lambda + \mu_{\min})^{-1}$. In particular, $\Delta = (\lambda + \sum_i c_i\mu_i)^{-1}$ is a valid choice; so the $M/M/1/\tilde{B}$ queue and the MDP Υ can both be discretised using the same value of Δ. Let $y_n \in \mathbb{N}_0$ be the state of the $M/M/1/\tilde{B}$ queue at the n^{th} time step, i.e. the number of customers present. One can construct a coupled process $(\hat{\mathbf{x}}_n, \hat{y}_n)_{n\in\mathbb{N}_0}$ with state space $S \times \mathbb{N}_0$, initialised in state $(\mathbf{0}, 0)$, which evolves according to the following transition probabilities:

- With probability $\lambda\Delta$, both of the marginal processes $(\hat{\mathbf{x}}_n)$ and $(\hat{y}_n)$ see a new customer arrival. In the case of $(\hat{\mathbf{x}}_n)$, the new customer takes action $\tilde{\theta}(\hat{\mathbf{x}}_n)$. In the case of $(\hat{y}_n)$, the new customer joins the queue if $\hat{y}_n < \tilde{B}$, and balks otherwise.

- With probability $\mu_{\min}\Delta$, the process $(\hat{\mathbf{x}}_n)$ sees a service completion at some arbitrary non-empty facility j (say, for example, $j = \min\{i \in \{1, 2, ..., N\} : x_i \geq 1\}$) if and only if $\hat{\mathbf{x}}_n \neq \mathbf{0}$, and the process $(\hat{y}_n)$ sees a service completion if and only if $\hat{y}_n \geq 1$.

- If $\hat{\mathbf{x}}_n \neq \mathbf{0}$, then there is an additional probability $(\min(x_j, c_j)\mu_j - \mu_{\min})\Delta$ that the process $(\hat{\mathbf{x}}_n)$ sees a service completion at facility j and the process $(\hat{y}_n)$ sees no event. Here, j is the non-empty facility referred to in the previous bullet point.

- For all facilities $i \neq j$, there is a probability $\min(x_i, c_i)\mu_i\Delta$ that the process $(\hat{\mathbf{x}}_n)$ sees a service completion at facility i and the process $(\hat{y}_n)$ sees no event.

- With probability $1 - \lambda\Delta - \sum_{i=1}^{N} \min(x_i, c_i)\mu_i\Delta$, both processes see no event.

Let $(\hat{\mathbf{x}}_n)_i$ denote the i^{th} component of state $\hat{\mathbf{x}}_n \in S$. It can easily be checked from the above that if $\sum_i(\hat{\mathbf{x}}_n)_i \leq \hat{y}_n$ for some $n \in \mathbb{N}_0$, then $\sum_i(\hat{\mathbf{x}}_{n+1})_i \leq \hat{y}_{n+1}$ also. Given that $(\hat{\mathbf{x}}_0, \hat{y}_0) = (\mathbf{0}, 0)$ by assumption, it follows that $\sum_i(\hat{\mathbf{x}}_n)_i \leq \hat{y}_n$ for all $n \in \mathbb{N}_0$. Due to the coupling construction, the n^{th} 'marginal' state $\hat{y}_n$ has a distribution identical to that of y_n, the n^{th} state of the $M/M/1/\tilde{B}$ queue (see Section 3.8 for further explanation of the stochastic coupling technique). Utilising standard formulae for $M/M/1$ queues with finite buffers (see [67], p. 74):

$$\lim_{\lambda\to 0}\left(\lim_{n\to\infty} P(y_n = 0)\right) = 1.$$

Hence, since $\sum_i(\hat{\mathbf{x}}_n)_i \leq \hat{y}_n$ for all $n \in \mathbb{N}_0$:

$$\lim_{\lambda\to 0}\left(\lim_{n\to\infty} P\left(\sum_{i=1}^{N}(\hat{\mathbf{x}}_n)_i = 0\right)\right) = 1,$$

and due to the equivalence between $(\hat{\mathbf{x}}_n)$ and the process Υ following policy $\tilde{\theta}$:

$$\lim_{\lambda\to 0} \pi_{\tilde{\theta}}(\mathbf{0}) = 1,$$

which obviously implies $\lim_{\lambda\to 0} \pi_{\tilde{\theta}}(\mathbf{x}) = 0$ for all $\mathbf{x} \neq \mathbf{0}$. The argument to show that $\lim_{\lambda\to 0} \pi_{\theta^*_\lambda}(\mathbf{0}) = 1$ is similar, except that it must be remembered that the policy θ^*_λ is dependent on λ. Nevertheless, using the 'containment' principle (Theorem 4.2.5), one may observe that given any $\lambda > 0$ (and assuming that the system is initialised in state $\mathbf{0}$), the total number of customers in the system will never exceed $\tilde{B}$ under θ^*_λ. Hence, this total may be bounded above by the number of customers

present in an $M/M/1/\tilde{B}$ queue with the same demand rate and a service rate $\mu_{\min}$, in exactly the same way as for $\tilde{\theta}$. Given any $\lambda > 0$, one may therefore write the following:

$$\lim_{n\to\infty} P(y_n = 0) \leq \lim_{n\to\infty} P\left(\sum_{i=1}^{N} (\mathbf{x}_n)_i = 0\right) \leq 1, \tag{4.4.6}$$

where, in this case, $\mathbf{x}_n$ denotes the n^{th} state of the process Υ following a policy θ^*_λ which is socially optimal under demand rate λ, and y_n is the n^{th} state of an $M/M/1/\tilde{B}$ queue which has been uniformised in the same way as described previously. Since $\lim_{n\to\infty} P(y_n = 0) \to 1$ as $\lambda \to 0$, the 'squeeze theorem' from analysis (see [166], p. 909) may be applied to (4.4.6) in order to show that $\lim_{\lambda\to 0} \pi_{\theta^*_\lambda}(\mathbf{0}) = 1$. These arguments confirm that the limits in (4.4.5) hold.

Consider the 'non-zero' states $\mathbf{x} \in \tilde{S} \setminus \{\mathbf{0}\}$. Given any $\lambda > 0$, the individual expected net rewards $w(\mathbf{x}, \tilde{\theta}(\mathbf{x}))$ and $w(\mathbf{x}, \theta^*_\lambda(\mathbf{x}))$ are finite for all such states; indeed, both quantities may be bounded above by $\max(\alpha_1, \alpha_2, ..., \alpha_N)$, and bounded below by zero. Hence:

$$\lim_{\lambda\to 0} \left(\pi_{\theta^*_\lambda}(\mathbf{x})\, w\big(\mathbf{x}, \theta^*_\lambda(\mathbf{x})\big)\right) = \lim_{\lambda\to 0} \left(\pi_{\tilde{\theta}}(\mathbf{x})\, w\big(\mathbf{x}, \tilde{\theta}(\mathbf{x})\big)\right) = 0 \qquad \forall\, \mathbf{x} \in \tilde{S} \setminus \{\mathbf{0}\}.$$

Recalling that $\hat{r}(\mathbf{x}, a) = \lambda w(\mathbf{x}, a)$ for all $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$, it then follows from (4.4.3) that:

$$\lim_{\lambda\to 0} \frac{\tilde{g}(\lambda)}{\lambda} = \lim_{\lambda\to 0} \left(\pi_{\tilde{\theta}}(\mathbf{0})\, w\big(\mathbf{0}, \tilde{\theta}(\mathbf{0})\big)\right),$$

$$\lim_{\lambda\to 0} \frac{g^*(\lambda)}{\lambda} = \lim_{\lambda\to 0} \left(\pi_{\theta^*_\lambda}(\mathbf{0})\, w\big(\mathbf{0}, \theta^*_\lambda(\mathbf{0})\big)\right), \tag{4.4.7}$$

assuming, of course, that these limits exist. The remainder of the proof will confirm this. It has already been shown that $\lim_{\lambda\to 0} \pi_{\tilde{\theta}}(\mathbf{0}) = 1$. Also, by definition of the selfishly optimal policy $\tilde{\theta}$, $w\big(\mathbf{0}, \tilde{\theta}(\mathbf{0})\big) = \max_i \{\alpha_i - \beta_i/\mu_i\}$ for all $\lambda > 0$. Hence:

$$\lim_{\lambda\to 0} \frac{\tilde{g}(\lambda)}{\lambda} = \lim_{\lambda\to 0} \left(\pi_{\tilde{\theta}}(\mathbf{0})\, w\big(\mathbf{0}, \tilde{\theta}(\mathbf{0})\big)\right) = \max_i \left\{\alpha_i - \frac{\beta_i}{\mu_i}\right\} > 0. \tag{4.4.8}$$

Also, for any demand rate $\lambda > 0$, one must have:

$$\frac{\tilde{g}(\lambda)}{\lambda} \leq \frac{g^*(\lambda)}{\lambda} \leq \max_i \left\{\alpha_i - \frac{\beta_i}{\mu_i}\right\},$$

where the left-hand inequality is due to the fact that $g^*(\lambda) \geq \tilde{g}(\lambda)$ by definition of the optimal average reward, and the right-hand inequality is due to the fact that $\lambda \max_i\{\alpha_i - \beta_i/\mu_i\}$ is the maximum value of $\hat{r}(\mathbf{x}, a)$ over all state-action pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$, and hence it can be seen

from (4.4.3) that $g^*(\lambda) \leq \lambda \max_i\{\alpha_i - \beta_i/\mu_i\}$. Thus, using (4.4.8) and again applying the 'squeeze theorem' ([166], p. 909), one obtains the same limit for $g^*(\lambda)/\lambda$ as for $\tilde{g}(\lambda)/\lambda$:

$$\lim_{\lambda \to 0} \frac{g^*(\lambda)}{\lambda} = \max_i \left\{ \alpha_i - \frac{\beta_i}{\mu_i} \right\} > 0. \tag{4.4.9}$$

Finally, by combining (4.4.8) and (4.4.9):

$$\lim_{\lambda \to 0} \left(\frac{g^*(\lambda) - \tilde{g}(\lambda)}{g^*(\lambda)} \right) = \frac{\max_i\{\alpha_i - \beta_i/\mu_i\} - \max_i\{\alpha_i - \beta_i/\mu_i\}}{\max_i\{\alpha_i - \beta_i/\mu_i\}} = 0,$$

which completes the proof of the theorem. □

In fact, the proof of Theorem 4.4.4 shows that *any* stationary policy which chooses an action a at state $\mathbf{0}$ maximising $w(\mathbf{0}, a)$ is asymptotically optimal in a light-traffic limit; the selfishly optimal policy $\tilde{\theta}$ is merely a special case. It will be useful to state this as a corollary.

Corollary 4.4.5. *Let θ be any stationary policy satisfying the following criterion:*

$$\theta(\mathbf{0}) \in \operatorname*{arg\,max}_{i \in \{1,2,\ldots,N\}} \left\{ \alpha_i - \frac{\beta_i}{\mu_i} \right\},$$

and let $g_\theta(\lambda)$ be the expected long-run average reward earned by θ given a demand rate $\lambda > 0$. Then θ is asymptotically optimal in a light-traffic limit. That is:

$$\lim_{\lambda \to 0} \left(\frac{g^*(\lambda) - g_\theta(\lambda)}{g^*(\lambda)} \right) = 0.$$

Proof. The proof follows the same arguments as that of Theorem 4.4.4. The important principle is that as $\lambda \to 0$, the decisions made by θ at states $\mathbf{x} \neq \mathbf{0}$ become irrelevant.

It should be noted that if θ is a policy which does *not* induce a Markov chain on some finite set of states contained in $\tilde{S}$, then the stochastic bounding argument given in the proof of Theorem 4.4.4 will need to be modified in order to show that $\pi_\theta(\mathbf{0}) \to 1$ as $\lambda \to 0$. This should not be problematic; indeed, the stationary probability $\pi_\theta(\mathbf{0})$ associated with *any* stationary policy θ may be shown to be stochastically bounded below by the steady-state probability of an infinite-capacity $M/M/1$ queue with demand rate λ and service rate $\mu_{\min} = \min_i \mu_i$ being empty. □

The next result in this section concerns the characterisation of socially optimal policies in the *heavy-traffic limit*, where $\lambda \to \infty$. First, it will be convenient to introduce a definition for a particular type of stationary policy, which will be referred to as a *vacancy policy*.

Definition 4.4.6. ***(Vacancy policy)***

Let θ be a stationary policy which satisfies the following conditions:

- *If $x_i < c_i$ for at least one $i \in \{1, 2, ..., N\}$, then $\theta(\mathbf{x}) \in \{i \in \{1, 2, ..., N\} : x_i < c_i\}$.*
- *If $x_i \geq c_i$ for all $i \in \{1, 2, ..., N\}$, then $\theta(\mathbf{x}) = 0$.*

Then θ is referred to as a vacancy policy*.*

In words, a vacancy policy always chooses to send a customer to a facility with an idle service channel if such a facility is available; otherwise, it chooses to balk. Naturally, no queues are formed under such a policy, since all customers either balk or begin service immediately. The next result shows that vacancy policies are optimal in a heavy-traffic limit, i.e. as $\lambda \to \infty$.

Theorem 4.4.7. *Let θ be a vacancy policy, and let $g_\theta(\lambda)$ denote the expected long-run average reward attained under θ given a demand rate $\lambda > 0$. Then:*

$$\lim_{\lambda \to \infty} \left(g^*(\lambda) - g_\theta(\lambda) \right) = 0.$$

That is, θ is socially optimal in the heavy-traffic limit.

Proof. This proof will assume the real-time formulation (3.5.4) for the single-stage rewards earned by Υ. Given any demand rate $\lambda > 0$, Theorem 4.2.4 implies that there exists a stationary policy θ^*_λ which attains the optimal average reward $g^*(\lambda)$ and also induces an irreducible, ergodic Markov chain on some set of states $S_{\theta^*_\lambda}$ contained in $\tilde{S}$. Hence, for any $\lambda > 0$:

$$g^*(\lambda) = \sum_{\mathbf{x} \in \tilde{S}} \pi_{\theta^*_\lambda}(\mathbf{x})\, r(\mathbf{x}).$$

By definition of $r(\mathbf{x})$ and the assumption that $\alpha_i \mu_i - \beta_i > 0$ for all $i \in \{1, 2, ..., N\}$, it can be seen that for all states $\mathbf{x} \in S$ the following upper bound holds:

$$r(\mathbf{x}) \leq \sum_{i=1}^{N} c_i \left(\alpha_i \mu_i - \beta_i \right). \tag{4.4.10}$$

Moreover, equality holds in (4.4.10) if and only if $\mathbf{x} = \mathbf{c}$, where $\mathbf{c} \in S$ is the unique state with $x_i = c_i$ for all $i \in \{1, 2, ..., N\}$. Let θ be an arbitrary vacancy policy, and (as in Section 4.3) let S° denote the set of states in S with no customers queueing. That is:

$$S^\circ := \{\mathbf{x} \in S : x_i \leq c_i \; \forall \; i \in \{1, 2, ..., N\}\}.$$

Clearly, S° is identical to the set of states S_θ which are positive recurrent in the Markov chain induced by θ. Hence, the average reward $g_\theta(\lambda)$ can be written as:

$$g_\theta(\lambda) = \sum_{\mathbf{x} \in S^\circ} \pi_\theta(\mathbf{x})\, r(\mathbf{x}), \tag{4.4.11}$$

where $\{\pi_\theta(\mathbf{x})\}_{\mathbf{x} \in S}$ is the stationary distribution under the vacancy policy θ. The aim of this proof is to show that $g_\theta(\lambda)$ tends to $\sum_i c_i\,(\alpha_i \mu_i - \beta_i)$ as $\lambda \to \infty$. In view of (4.4.11), this can be accomplished by showing that the stationary probabilities $\pi_\theta(\mathbf{x})$ satisfy:

$$\lim_{\lambda \to \infty} \pi_\theta(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{x} = \mathbf{c}, \\ 0, & \text{otherwise.} \end{cases} \tag{4.4.12}$$

The limits in (4.4.12) can be proved using a stochastic bounding argument, similar to that used to establish the limits (4.4.5) in the proof of Theorem 4.4.4. Indeed, consider the process Υ initialised in state $\mathbf{0}$ and operating under the vacancy policy θ. A customer arriving under state $\mathbf{x} \in S$ joins some facility $i \in \{1, 2, ..., N\}$ if and only if $\sum_j x_j < K$, where $K = \sum_i c_i$ is the total number of service channels across all facilities. It follows that, given any $\lambda > 0$, the total number of customers in the system under θ is stochastically bounded *below* by the number of customers in an $M/M/1/K$ queue with finite capacity K, demand rate λ and service rate μ_T, where:

$$\mu_T := \sum_{i=1}^{N} c_i \mu_i.$$

Let the $M/M/1/K$ queue and the process Υ both be *uniformised* with the same parameter, $\Delta = (\lambda + \sum_i c_i \mu_i)^{-1} = (\lambda + \mu_T)^{-1}$. Let $y_n \in \mathbb{N}_0$ denote the state of the $M/M/1/K$ queue at the n^{th} discrete time step. It is possible to construct a coupled process $(\hat{\mathbf{x}}_n, \hat{y}_n)$ with state space $S \times \mathbb{N}_0$ similar to the coupled process used in the proof of Theorem 4.4.4. In this case, one can show that $\sum_i (\hat{\mathbf{x}}_n)_i \geq \hat{y}_n$ for all $n \in \mathbb{N}_0$ (where $(\hat{\mathbf{x}}_n)_i$ is the i^{th} component of $\hat{\mathbf{x}}_n$), assuming that $(\hat{\mathbf{x}}_0, \hat{y}_0) = (\mathbf{0}, 0)$. The transition rules for $(\hat{\mathbf{x}}_n, \hat{y}_n)$ may be defined as follows:

- With probability $\lambda\Delta$, both of the marginal processes $(\hat{\mathbf{x}}_n)$ and $(\hat{y}_n)$ see a new customer arrival. In the case of $(\hat{\mathbf{x}}_n)$, the new customer takes action $\theta(\hat{\mathbf{x}}_n)$. In the case of $(\hat{y}_n)$, the new customer joins the queue if $\hat{y}_n < K$, and balks otherwise.

- With probability $\min(x_i, c_i)\mu_i\Delta$ (for $i = 1, 2, ..., N$), the process $(\hat{\mathbf{x}}_n)$ sees a service completion at facility i and the process $(\hat{y}_n)$ also sees a service completion if and only if $\hat{y}_n \geq 1$.

- With probability $(\mu_T - \sum_i \min(x_i, c_i)\mu_i)\,\Delta$, the process $(\hat{y}_n)$ sees a service completion if and only if $\hat{y}_n \geq 1$, and the process $(\hat{\mathbf{x}}_n)$ sees no event.
- With probability $1 - \lambda\Delta - \sum_i c_i\mu_i\Delta$, both processes see no event.

It can be verified that if $\sum_i(\hat{\mathbf{x}}_n)_i \geq \hat{y}_n$ for some $n \in \mathbb{N}_0$ then one must have $\sum_i(\hat{\mathbf{x}}_{n+1})_i \geq \hat{y}_{n+1}$ also. Given that $(\hat{\mathbf{x}}_0, \hat{y}_0) = (\mathbf{0}, 0)$ by assumption, it follows that $\sum_i(\hat{\mathbf{x}}_n)_i \geq \hat{y}_n$ for all $n \in \mathbb{N}_0$. Due to the coupling construction, each 'marginal' state $\hat{y}_n$ has a distribution identical to that of y_n. Hence, standard formulae for finite-buffer $M/M/1$ queues (see, e.g. [67]) imply:

$$\lim_{\lambda\to\infty}\left(\lim_{n\to\infty} P(\hat{y}_n = K)\right) = 1.$$

Therefore, since $\hat{y}_n \leq \sum_i(\hat{\mathbf{x}}_n)_i \leq K$ for all $n \in \mathbb{N}_0$:

$$\lim_{\lambda\to\infty}\left(\lim_{n\to\infty} P\left(\sum_{i=1}^{N}(\hat{\mathbf{x}}_n)_i = K\right)\right) = 1.$$

Of course, by construction of the policy θ and the fact that $\hat{\mathbf{x}}_0 = \mathbf{0}$, $\sum_{i=1}^{N}(\hat{\mathbf{x}}_n)_i = K$ if and only if $\hat{\mathbf{x}}_n = \mathbf{c}$. Hence, by the equivalence of $(\hat{\mathbf{x}}_n)$ and the process Υ following θ:

$$\lim_{\lambda\to\infty} \pi_\theta(\mathbf{c}) = 1,$$

and $\lim_{\lambda\to\infty} \pi_\theta(\mathbf{x}) = 0$ for all $\mathbf{x} \neq \mathbf{c}$, which establishes (4.4.12). Then, since the rewards $r(\mathbf{x})$ are independent of λ, taking limits in (4.4.11) immediately yields:

$$\lim_{\lambda\to\infty} g_\theta(\lambda) = r(\mathbf{c}) = \sum_{i=1}^{N} c_i\left(\alpha_i\mu_i - \beta_i\right). \tag{4.4.13}$$

Recall that, by (4.4.10), the single-step rewards $r(\mathbf{x}_n)$ (for $n = 0, 1, 2, ...$) are bounded above by $\sum_{i=1}^{N} c_i\left(\alpha_i\mu_i - \beta_i\right)$. Hence, given any $\lambda > 0$, the expected long-run average reward $g^*(\lambda)$ must also be bounded above by the same quantity. This implies, for all $\lambda > 0$:

$$g_\theta(\lambda) \leq g^*(\lambda) \leq \sum_{i=1}^{N} c_i\left(\alpha_i\mu_i - \beta_i\right).$$

Hence, again relying on the 'squeeze theorem' from analysis ([166], p. 909):

$$\lim_{\lambda\to\infty} g^*(\lambda) = \sum_{i=1}^{N} c_i\left(\alpha_i\mu_i - \beta_i\right). \tag{4.4.14}$$

Finally, in view of (4.4.13) and (4.4.14):

$$\lim_{\lambda\to\infty}\big(g^*(\lambda)-g_\theta(\lambda)\big)=\lim_{\lambda\to\infty}g^*(\lambda)-\lim_{\lambda\to\infty}g_\theta(\lambda)=0,$$

which completes the proof that the vacancy policy θ is optimal in heavy traffic. $\square$

The proof of Theorem 4.4.7 offers some insight into the behaviour of optimal policies in general. In particular, the fact that $r(\mathbf{c}) > r(\mathbf{x})$ for all $\mathbf{x} \neq \mathbf{c}$ is insightful. The state $\mathbf{c}$, with all service channels occupied and no customers queueing, may be perceived as a 'Utopian' state; clearly, if it were possible for the system to remain permanently in state $\mathbf{c}$, then this would be the ideal scenario in order to maximise the long-run average reward. As $\lambda \to \infty$, it becomes possible for the system to spend (almost) all its time in state $\mathbf{c}$, since any service completion is followed almost immediately by a new customer arrival, so the system can almost immediately 'refill' any service channel that becomes idle. Of course, customers who arrive when all service channels are occupied can simply be rejected. This explains the optimality of vacancy policies as $\lambda \to \infty$. On the other hand, when λ is small, it may be optimal to admit customers to the system under state $\mathbf{c}$ in order to compensate for the negative drift (due to an increase in the frequency of service completions relative to arrivals) which moves the system towards sparsely-populated states.

It is also interesting to note that the 'anticipatory' reward formulation with reward function $\hat{r}(\mathbf{x}, a)$ was used to prove Theorem 4.4.4, whereas the 'real-time' formulation with reward function $r(\mathbf{x})$ was used in the proof of Theorem 4.4.7. These formulations were not chosen arbitrarily; in fact, the anticipatory formulation was used to prove the light-traffic result due to the fact that under this formulation, $\hat{r}\big(\mathbf{0}, \tilde{\theta}(\mathbf{0})\big) \geq \hat{r}(\mathbf{0}, a)$ for any action a, where $\tilde{\theta}(\mathbf{0})$ is the action chosen by the selfish policy at state $\mathbf{0}$. No such convenient property holds under the real-time reward formulation; in fact, $r(\mathbf{0}) = 0$ regardless of the action chosen. On the other hand, the real-time formulation was used to prove the heavy-traffic result due to the fact that $r(\mathbf{c}) > r(\mathbf{x})$ for all $\mathbf{x} \neq \mathbf{c}$, whereas the rewards under the anticipatory formulation do not attain their largest values at $\mathbf{x} = \mathbf{c}$. The contrasting properties of the two reward formulations and their respective uses highlight the practical advantages to be gained from proving the equivalence result in Theorem 3.5.3.

The following corollary summarises the relationship between $g^*(\lambda)$ and λ.

Corollary 4.4.8. *As the demand rate λ increases, the optimal expected long-run average reward*

$g^*(\lambda)$ converges monotonically to the following limit:

$$\lim_{\lambda\to\infty} g^*(\lambda) = \sum_{i=1}^{N} c_i \left(\alpha_i\mu_i - \beta_i\right).$$

Proof. The limit itself has been shown in the proof of Theorem 4.4.7. The fact that the convergence is monotonic follows from Theorem 4.4.1. □

It is interesting to note that 'real-time' reward formulations such as (3.5.4) may be used to prove the optimality of vacancy policies in more general types of problems. The reader is invited to refer to Appendix A.7 for an example involving a *Jackson network*, which differs from the queueing system described in Section 3.1 in that customers who complete service at any facility may be re-routed to another facility (or the same facility) as opposed to departing from the system. The example describes a particular type of Jackson network problem in which optimal policies are extremely simple to characterise, despite the complexity of the underlying MDP formulation.

The results proved in this section will be used to examine the light-traffic and heavy-traffic optimality (or sub-optimality) of certain heuristic policies considered in Chapter 6. This chapter will conclude by discussing an extension involving heterogeneous customers.

4.5 Extension: Heterogeneous customers

This section will consider an extension of the queueing system described in Section 3.1 in which customers belong to heterogeneous groups, or *classes*. Let $M \geq 2$ be the number of classes, so that each customer belongs to some class in $\{1, 2, ..., M\}$. Each class has its own independent Poisson arrival stream, set of holding costs and set of fixed rewards. To be precise:

- Customers of class $i \in \{1, 2, ..., M\}$ arrive via their own independent, time-homogeneous Poisson process with demand rate $\lambda_i > 0$.
- Suppose a customer of class $i \in \{1, 2, ..., M\}$ goes to be served at facility $j \in \{1, 2, ..., N\}$. Then a holding cost $\beta_{ij} > 0$ is incurred per unit time until they exit the system, and a reward $\alpha_{ij} > 0$ is earned after they complete service.

Note that, at this stage, the service rates μ_j and service capacities c_j at the various facilities remain independent of customer class. Incorporating class-dependent service rates within an MDP framework causes a significant amount of extra complexity due to the effect on the transition probabilities and the complexity of the state space required; for this reason, an alternative approach to dealing with class-dependent service rates will be discussed in Chapter 7.

The system may be modelled using an MDP which is *uniformised* as described in Section 3.3, so that it evolves in discrete time steps of size Δ, where $0 < \Delta \leq \left(\sum_i \lambda_i + \sum_j c_j \mu_j\right)^{-1}$. In Section 4.2, the anticipatory reward formulation (3.5.15) was used in order to prove certain properties of the associated discounted-reward value function $\hat{v}_\phi(\cdot)$ which are sufficient to imply the existence of an average reward optimal stationary policy satisfying the equations (4.2.6).

A further advantage of the anticipatory reward formulation is that it enables the results in Sections 4.2 to be extended to a scenario involving heterogeneous customers without a re-description of the state space S being required. Suppose one wanted to use a 'real-time' reward formulation, similar to (3.5.4), for the reward $r(\cdot)$ in the heterogeneous customers problem. Then the system state would need to include information about the classes of customers in service at each facility, and also the classes of all customers waiting in queues. On the other hand, using an 'anticipatory' reward formulation, it is possible to allow the state space representation to be the same as before; that is, one may define $S = \left\{\mathbf{x} = (x_1, x_2, ..., x_N) : x_1, x_2, ..., x_N \in \mathbb{N}_0\right\}$, where x_j is simply the number of customers present (irrespective of class) at facility j, for $j = 1, 2, ..., N$. On the other hand, the set of actions A available at each state $\mathbf{x} \in S$ now requires a more complicated representation in order to allow actions to depend upon customer class. Let the set A be given by:

$$A = \left\{\mathbf{a} = (a_1, a_2, ..., a_M) : a_1, a_2, ..., a_M \in \{0, 1, ..., N\}\right\}. \tag{4.5.1}$$

That is, the action $\mathbf{a}$ is a vector which prescribes, for each customer class $i \in \{1, 2, ..., M\}$, the destination a_i of any customer of class i who arrives at the present discrete epoch of time. The reward $\hat{r}(\mathbf{x}, \mathbf{a})$ for choosing action $\mathbf{a} \in A$ at state $\mathbf{x}$ is then given by:

$$\hat{r}(\mathbf{x}, \mathbf{a}) = \sum_{i=1}^{M} \hat{r}_i(\mathbf{x}, a_i), \tag{4.5.2}$$

where a_i is the i^{th} component of $\mathbf{a}$, and $\hat{r}_i(\mathbf{x}, a_i)$ is defined for $i \in \{1, 2, ..., M\}$ by:

$$\hat{r}_i(\mathbf{x}, a_i) := \begin{cases} \lambda_i \left(\alpha_{ij} - \dfrac{\beta_{ij}}{\mu_j} \right), & \text{if } a_i = j \text{ for some } j \in \{1, 2, ..., N\} \text{ and } x_j < c_j, \\ \lambda_i \left(\alpha_{ij} - \dfrac{\beta_{ij}(x_j + 1)}{c_j \mu_j} \right), & \text{if } a_i = j \text{ for some } j \in \{1, 2, ..., N\} \text{ and } x_j \geq c_j, \\ 0, & \text{otherwise.} \end{cases} \tag{4.5.3}$$

In fact, expanding the action set in this manner is not the only possible way of formulating the extended model (with heterogeneous customers) as an MDP, but it is the natural extension of the MDP Υ formulated in Section 3.5. An alternative approach would be to augment the state space so that information about the class of the most recent customer to arrive is included in the state description; this would involve using a state space formulation similar to (3.2.7), with the random event Λ replaced by a set of events $\{\Lambda_i\}_{i=1}^{M}$ in order to indicate the class of the latest customer to arrive. Actions would then need to be chosen only at arrival epochs, and these actions would simply be integers in the set $\{0, 1, ..., N\}$ as opposed to vectors. By keeping the state space S unchanged, however, it is possible to show that the results of Section 4.2 can be generalised quite easily. Under the new MDP formulation, the discount optimality equations (using the anticipatory reward functions $\hat{r}_i$ in (4.5.3)) are given for states $\mathbf{x} \in S$ and $\phi \in (0, 1)$ by:

$$\begin{aligned} \hat{v}_\phi(\mathbf{x}) = &\sum_{i=1}^{M} \max_{a_i \in A} \left\{ \hat{r}_i(\mathbf{x}, a_i) + \phi \lambda_i \Delta \hat{v}_\phi(\mathbf{x}^{a_i+}) \right\} + \phi \sum_{j=1}^{N} \min(x_j, c_j) \mu_j \Delta \hat{v}_\phi(\mathbf{x}^{j-}) \\ &+ \phi \left(1 - \sum_{i=1}^{M} \lambda_i \Delta - \sum_{j=1}^{N} \min(x_j, c_j) \mu_j \Delta \right) \hat{v}_\phi(\mathbf{x}). \end{aligned} \tag{4.5.4}$$

From a practical perspective, it is worthwhile to note that the maximisation in (4.5.4) is carried out in a componentwise fashion. This means that instead of having to find the maximiser among *all* vectors $\mathbf{a}$ in A (of which the total number is $(N+1)^M$), it is sufficient to find, for each customer class $i \in \{1, 2, ..., M\}$, the 'marginal' action a_i which maximises $\hat{r}_i(\mathbf{x}, a_i) + \phi \lambda_i \Delta \hat{v}_\phi(\mathbf{x}^{a_i+})$. This can be exploited in the implementation of dynamic programming algorithms (e.g. value iteration), so that computation times increase only in proportion to the number of classes M.

As before, the *selfishly optimal policy* is defined as the policy under which the action chosen for each customer arriving in the system is the action which maximises $\hat{r}_i(\mathbf{x}, a_i)$ (obviously this action is dependent upon the class i). A selfish customer of class i accepts service at facility j if and only if, prior to joining, the number x_j of customers at facility j, satisfies $x_j \leq \tilde{B}_{ij}$, where:

$$\tilde{B}_{ij} := \left\lfloor \frac{\alpha_{ij} c_j \mu_j}{\beta_{ij}} \right\rfloor . \tag{4.5.5}$$

Consequently, under steady-state conditions, the number of customers present at facility j is bounded above by $\max_i \tilde{B}_{ij}$. The selfishly optimal state space $\tilde{S}$ is then given by:

$$\tilde{S} := \left\{ (x_1, x_2, ..., x_N) : x_j \leq \max_i \tilde{B}_{ij},\ j = 1, 2, ..., N \right\} . \tag{4.5.6}$$

Example 4.5.1. ***(Containment in two dimensions with two customer classes)***

Consider a system with two facilities, both with two service channels ($c_1 = c_2 = 2$) and service rates $\mu_1 = 5$ and $\mu_2 = 1$. It will be useful to begin by examining the selfishly and socially optimal policies in the case where all customers belong to the *same* class. Suppose customers arrive at a Poisson rate $\lambda_1 = 12$ and earn fixed rewards $\alpha_{11} = 1$ and $\alpha_{12} = 3$ at facilities 1 and 2 respectively, while the corresponding holding costs β_{11} and β_{12} are both equal to 3. Figure 4.2 shows that the selfishly optimal policy for these customers induces a Markov chain with 12 positive recurrent states, while the socially optimal policy induces a Markov chain with only 6 recurrent states.

Selfish Policy

	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	1	1	1
$x_1 = 1$	1	1	1
$x_1 = 2$	1	1	1
$x_1 = 3$	2	2	0

Social Policy

	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	1	1	1
$x_1 = 1$	1	1	1
$x_1 = 2$	2	0	0
$x_1 = 3$	2	0	0

Figure 4.2: Selfishly and socially optimal policies in Example 4.5.1, with only one customer class. For each state $\mathbf{x} = (x_1, x_2) \in \tilde{S}$, the corresponding decisions under the respective policies are shown.

Now suppose that a *second* stream of customers also arrives with demand rate $\lambda_2 = 10$, so that the system receives two independent arrival streams. Customers in the second class have steeper

holding costs and a much greater value of service at the second facility: $\beta_{21} = \beta_{22} = 5$, $\alpha_{21} = 1$, $\alpha_{22} = 12$. The service rates $\mu_1 = 5$ and $\mu_2 = 1$ remain independent of customer class. The system may be uniformised with step-size parameter $\Delta = \left(\sum_i \lambda_i + \sum_i c_i\mu_i\right)^{-1} = 1/34$.

Figure 4.3 shows the selfishly and socially optimal policies for the system with two classes of customer arriving. It is clear that the incorporation of a second class of customer has no effect on the selfish decisions made by the first class of customer, so (as shown by the figure) these decisions remain the same as shown in Figure 4.2 previously. The first table in Figure 4.3 shows that selfish customers of the second class are unwilling to join the first facility when $x_1 \geq 2$; however, under certain states they will choose to join the second facility when $x_2 = 3$ (but never when $x_2 > 3$). As a result, the new selfish space $\tilde{S}$ as defined in (4.5.6) is expanded from 12 states to 20.

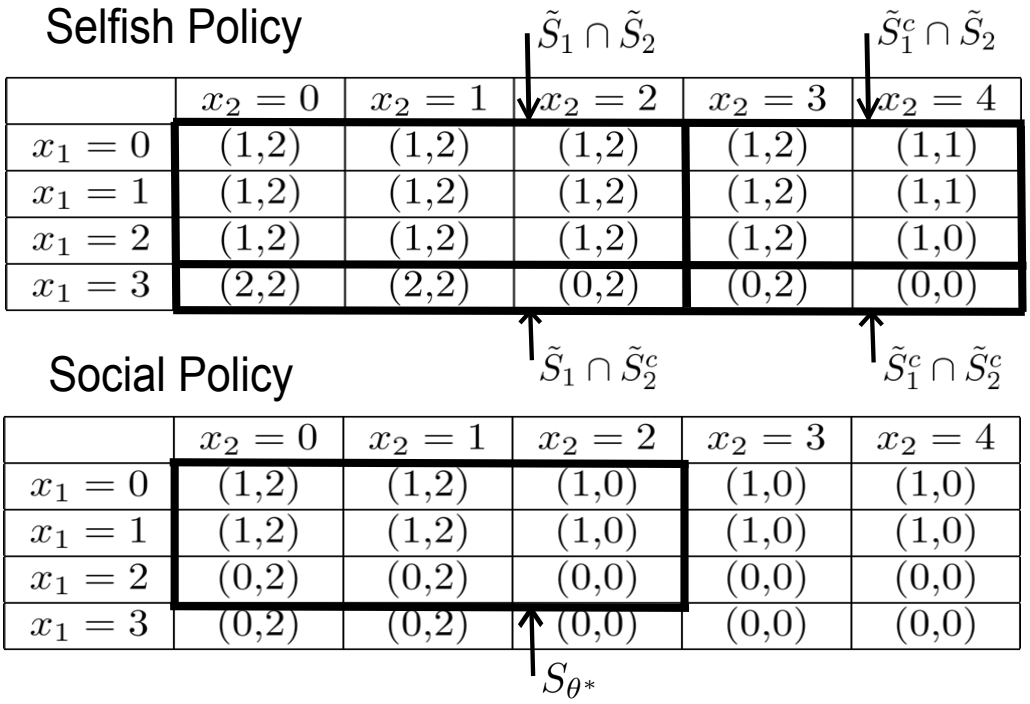
Figure 4.3 shows that the new selfish state space $\tilde{S}$ may be represented diagrammatically as the smallest rectangle which encompasses both $\tilde{S}_1$ and $\tilde{S}_2$, where (for $i = 1, 2$):

$$\tilde{S}_i := \Big\{(x_1, x_2, ..., x_N) : x_j \leq \tilde{B}_{ij},\ j = 1, 2, ..., N\Big\}.$$

It is somewhat interesting to observe that the selfish state space $\tilde{S}$ includes states in the 'intersection of complements' $\tilde{S}_1^c \cap \tilde{S}_2^c$, consisting of the states $(3, 3)$ and $(3, 4)$. These states would not occur (under steady-state conditions) if the system operated with only a single (selfish) class of customer of either type, but they *do* occur with both classes of customer in attendance.

The policy θ^* depicted in the second table in Figure 4.3 has been obtained using relative value iteration, and illustrates the containment property for systems with heterogeneous customer classes. It may be seen that the socially optimal state space S_{θ^*} consists of only 9 states; under steady-state conditions, the system remains within this set of 9 states. It is also apparent that, unlike the selfish decisions, the socially optimal decisions for a particular class of customer *are* affected by the decisions made by the other class of customer (as can be seen, in the case of the first customer class, by direct comparison with Figure 4.2). Indeed, under θ^*, customers of the first class never join facility 2, while customers of the second class never join facility 1. ⊠

It can be verified that the results of Lemmas 4.2.1 and 4.2.2, Theorems 4.2.3 and 4.2.4 and Theorem 4.2.5 apply to the extended model with heterogeneous customers, with only small modifications required to the proofs. For example, in Lemma 4.2.1 the inequality $\hat{v}_\phi(\mathbf{x}^{j+}) \leq \hat{v}_\phi(\mathbf{x})$ may be proved



	$x_2=0$	$x_2=1$	$x_2=2$	$x_2=3$	$x_2=4$
$x_1=0$	(1,2)	(1,2)	(1,2)	(1,2)	(1,1)
$x_1=1$	(1,2)	(1,2)	(1,2)	(1,2)	(1,1)
$x_1=2$	(1,2)	(1,2)	(1,2)	(1,2)	(1,0)
$x_1=3$	(2,2)	(2,2)	(0,2)	(0,2)	(0,0)

	$x_2=0$	$x_2=1$	$x_2=2$	$x_2=3$	$x_2=4$
$x_1=0$	(1,2)	(1,2)	(1,0)	(1,0)	(1,0)
$x_1=1$	(1,2)	(1,2)	(1,0)	(1,0)	(1,0)
$x_1=2$	(0,2)	(0,2)	(0,0)	(0,0)	(0,0)
$x_1=3$	(0,2)	(0,2)	(0,0)	(0,0)	(0,0)

Figure 4.3: Selfishly and socially optimal policies for the system in Example 4.5.1, with two customer classes. For each state $\mathbf{x} = (x_1, x_2)$, the corresponding decision vector $\mathbf{a} = (a_1, a_2)$ is shown.

by showing that, *for all classes* $i \in \{1, 2, ..., M\}$ and facilities $j \in \{1, 2, ..., N\}$:

$$\max_{a_i} \left\{ \hat{r}_i(\mathbf{x}^{j+}, a_i) + \phi\lambda_i \Delta\hat{v}_\phi^{(k)}\big((\mathbf{x}^{j+})^{a_i+}\big) \right\} \leq \max_{b_i} \left\{ \hat{r}_i(\mathbf{x}, b_i) + \phi\lambda_i \Delta\hat{v}_\phi^{(k)}\big(\mathbf{x}^{b_i+}\big) \right\},$$

for all $k \in \mathbb{N}_0$. In Lemma 4.2.2, one may define $\alpha^* = \max_{i,j} \alpha_{ij}$ as the maximum value of service across all facilities *and* customer classes, and establish a lower bound for $\hat{v}_\phi(\mathbf{0}^{j+}) - \hat{v}_\phi(\mathbf{0})$ similar to (A.4.9) by writing $\hat{r}(\mathbf{0}^{j+}, \mathbf{0})$ and $p(\mathbf{0}^{j+}, \mathbf{0}, \mathbf{y})$ instead of $\hat{r}(\mathbf{0}^{j+}, 0)$ and $p(\mathbf{0}^{j+}, 0, \mathbf{y})$ respectively (so that the action at state $\mathbf{0}^{j+}$ is the zero vector $\mathbf{0}$, i.e. all customer classes balk); the rest of the inductive proof goes through using similar adjustments. Theorem 4.2.4 holds because if S_{θ^*} was *not* contained in $\tilde{S}$, then the discount optimality equations would imply:

$$\hat{r}_i(\mathbf{x}, j) + \phi\lambda_i \Delta\hat{v}_\phi(\mathbf{x}^{j+}) \geq \phi\lambda_i \Delta\hat{v}_\phi(\mathbf{x}),$$

for some class $i \in \{1, 2, ..., M\}$ and facility $j \in \{1, 2, ..., N\}$ with $\hat{r}_i(\mathbf{x}, j) < 0$, thus contradicting the result of (the modified) Lemma 4.2.1. The sample path argument used to prove Theorem 4.2.5 can be applied to a customer of any class, with only trivial adjustments needed.

This section has shown that it is a straightforward task to generalise the results of Section 4.2 to a problem involving heterogeneous customers. However, this generalisation of the problem is still somewhat restricted, since it must be assumed that the service rates μ_j and service capacities c_j at the various facilities remain independent of customers' classes in order for the complexity of the resulting MDP formulation to be manageable. One can easily imagine that the assumption of non-discriminatory service rates would be unsatisfactory in many applications; for example, the length of stay of a patient in a hospital would typically depend upon many factors, including their age and medical history. For this reason, the heterogeneous customers model described in this section will not be considered again until Chapter 7, at which point it will become possible (using techniques to be discussed later) to incorporate class-dependent service rates.

4.6 Conclusions

Section 4.1 explained how the themes of selfish and social optimisation discussed in Chapter 2 may be generalised to a problem involving N multiple-server facilities. The principle of socially optimal policies being 'contained' in the selfishly optimal state space $\tilde{S}$, as discussed in Section 4.2, can be exploited by truncating the infinite state space and restricting attention to the finite set of states $\tilde{S}$ when searching for a socially optimal policy. The results of Section 4.3 are also useful in order to establish the inequality $S^\circ \subseteq S_{\theta^*} \subseteq \tilde{S}$ for any socially optimal policy θ^*. Unfortunately, it is difficult to prove further characteristics of socially optimal policies which hold in complete generality for the system introduced in Section 3.1, as the next chapter will show.

The results in Section 4.4 will prove useful later in order to determine whether certain types of stationary policies achieve light-traffic and/or heavy-traffic optimality. In particular, when heuristic policies are considered in Chapter 6, it will be possible to determine whether or not these policies are optimal in light or heavy-traffic limits by referring to the aforementioned results.

Section 4.5 considered a more intricate problem involving heterogeneous customers. While it was shown that certain results from earlier sections could be applied successfully to this more general problem under suitable conditions, the assumptions required (in particular, the lack of dependence of service rates on customer classes) were also observed to be somewhat restrictive. Problems involving heterogeneous customers will be revisited later, using a different approach.

5 Monotonicity and structure

The results in Chapter 4 established certain properties of socially optimal policies by drawing comparisons with selfishly optimal policies; in particular, an important result is Theorem 4.2.4, which states that it is possible to find a stationary socially optimal policy which induces an ergodic, irreducible Markov chain on a set of states contained in the (finite) selfishly optimal state space $\tilde{S}$. The results in Section 4.4 also provide some insight into the effect of varying the demand rate λ on socially optimal policies. The results in this chapter are concerned with the *structure* of socially optimal policies given an arbitrary and (usually) fixed set of input parameters. More specifically, it is of interest to determine whether the decision made at a particular state $\mathbf{x} \in S$ by some socially optimal policy θ^* can be inferred from the decision chosen at another state in S.

Certain structural properties of the *selfishly* optimal policy $\tilde{\theta}$ (as defined in Section 4.1) can be shown trivially. For example, if $\tilde{\theta}$ chooses to balk at some state $\mathbf{x} \in S$, then it must also choose to balk at state $\mathbf{x}^{j+}$, for any $j \in \{1, 2, ..., N\}$. This is referred to as a *monotonicity* property, in the sense that if balking is chosen at state $\mathbf{x}$, then balking is also chosen at any state $\mathbf{y} \in S$ which satisfies the componentwise inequality $\mathbf{x} \leq \mathbf{y}$ (that is, $x_i \leq y_i$ for all $i \in \{1, 2, ..., N\}$). It is of considerable interest to establish whether or not socially optimal policies exist which satisfy similar monotonicity properties. If such policies exist, then it may be possible to construct specialised algorithms (as alternatives to conventional dynamic programming algorithms) in order to find them efficiently. Unfortunately, establishing monotonicity properties of an N-facility queueing system with heterogeneous servers is far from an easy task; indeed, counter-examples can be found to disprove many reasonable theories (see Appendix A.8). For this reason, most of the results given in this chapter apply only to systems which are significantly reduced in complexity.

For clarity, all of the results in this chapter assume a single Poisson arrival stream of customers, *without* the extension to heterogeneous customers discussed in Section 4.5; that is, the rewards α_i and holding costs β_i at the various facilities are assumed to be the same for all customers. The MDP Υ formulated in Section 3.5 will be considered throughout, with the real-time reward formulation (3.5.4) used in some results and the anticipatory formulation (3.5.15) used in others. Section 5.1 will consider single-facility systems. Section 5.2 will discuss '$2DSS$' systems, consisting of two facilities with a single server each. Section 5.3 will briefly consider N-facility systems where all

facilities are homogeneous (i.e. they share the same set of parameters). Section 5.4 will introduce some computational algorithms which are based on results proved in the preceding sections. In addition, Appendix A.8 provides various counter-examples to show that many of the results proved for smaller systems cannot be generalised easily in systems of greater complexity.

5.1 A single facility

This section considers the special case $N = 1$ in the queueing system described in Section 3.1. As such, it will be convenient to drop the facility index i, so that the number of channels, service rate, fixed reward and holding cost per unit time are given by c, μ, α and β (all > 0) respectively. Customers arrive at a rate $\lambda > 0$ and may either *join* or *balk* (denoted by actions 1 and 0 respectively), so that essentially the system consists of a single $M/M/c$ queue with admission control. Given that the assumption of multiple, heterogeneous facilities is fundamental to much of the work in this thesis, it is reasonable to question whether a problem involving a single $M/M/c$ queue is actually worthy of consideration at all. In fact, there are several justifications for examining this single-facility problem before moving on to larger-scale problems. For example:

- A number of interesting results involving contrasts between selfishly and socially optimal customer behaviour can be proved in single-facility problems, which are difficult (or, unfortunately, impossible) to generalise for multiple-facility problems.
- Structural properties of optimal policies in single-facility problems offer clues regarding the nature of policies which may perform extremely well in multiple-facility problems, even if they do not necessarily achieve optimality; this will be explored later on.
- Deriving the *heuristic* policies which are the subject of Chapter 6 in this thesis depends, to some extent, on analysing individual facilities as if they were operating in isolation. The subsequent analysis of these heuristic policies can be greatly simplified if certain properties of socially optimal policies in single-facility systems are already known.

Unsurprisingly, the control of single-facility queueing systems has already been treated extensively in the literature. Naor [131] considered admission control in an observable $M/M/1$ queue, and Knudsen [103] extended some of Naor's results to observable $M/M/c$ queues. Other significant

early contributions include those of Yechiali [203, 204], Littlechild [121] and Lippman and Stidham [120]. Hassin and Haviv [76] have provided a comprehensive survey of the literature. In view of these existing works, it must be acknowledged that some of the results in this section are already 'known'; this will be indicated where appropriate. However, several of the proofs presented will be original ones based on dynamic programming arguments. It will be useful to demonstrate the use of inductive arguments based on relative value iteration to prove structural properties in the one-facility case, since this technique will be relied upon throughout this chapter.

All of the results in this section assume $N = 1$ (a single facility), so this assumption will be omitted from the statements of lemmas, theorems etc. in order to avoid unnecessary repetition. Since there is a single facility, let the system state be denoted by a scalar $x \in \mathbb{N}_0$. As mentioned previously, there are two possible actions at each state: a customer may join ($a = 1$) or balk ($a = 0$). Before proceeding, it will be useful to recall results from previous chapters which will simplify the analysis of the single-facility system. Firstly, it may be observed that a customer's individual expected net reward $w(x, a)$ for taking action $a \in \{0, 1\}$ under some state $x \in \mathbb{N}_0$ is given by:

$$w(x,a) := \begin{cases} \alpha - \dfrac{\beta}{\mu}, & \text{if } a = 1 \text{ and } x < c, \\ \alpha - \dfrac{\beta(x+1)}{c\mu}, & \text{if } a = 1 \text{ and } x \geq c, \\ 0, & \text{if } a = 0. \end{cases} \tag{5.1.1}$$

According to the definition of the selfishly optimal policy $\tilde{\theta}$ in Section 4.1, $\tilde{\theta}(x) = 1$ if and only if $w(x, 1) \geq 0$. Using (5.1.1), it then follows that $\tilde{\theta}(x) = 1$ if and only if:

$$x < \left\lfloor \frac{\alpha c \mu}{\beta} \right\rfloor =: \tilde{T}.$$

So, in the single-facility case, the selfishly optimal state space $\tilde{S}$ is the set of integers $\{0, 1, 2, ..., \tilde{T}\}$. Theorem 4.2.4, which holds for arbitrary $N \in \mathbb{N}$, implies that there exists a stationary socially optimal policy θ^* which induces a Markov chain with a positive recurrent state space contained in $\{0, 1, 2, ..., \tilde{T}\}$. Moreover, at any state $x \in \mathbb{N}_0$, one may assume that θ^* chooses the action $a \in \{0, 1\}$ which attains the maximum in the average reward optimality equations:

$$g^* + h(x) = \max_{a \in \{0,1\}} \left\{ r(x) + \sum_{y \in \mathbb{N}_0} p(x, a, y) h(y) \right\} \qquad (x \in \mathbb{N}_0), \tag{5.1.2}$$

where g^* is the optimal expected long-run average reward, $h(\cdot)$ is a relative value function, and $r(\cdot)$ is the real-time reward function defined in (3.5.4). Specifically:

$$r(x) = \min(x, c)\alpha\mu - \beta x. \tag{5.1.3}$$

The anticipatory reward function $\hat{r}$ is defined for state-action pairs $(x, a) \in \mathbb{N}_0 \times \{0, 1\}$ as:

$$\hat{r}(x, a) = \lambda w(x, a). \tag{5.1.4}$$

Using the technique of uniformisation (see Section 3.3), the transition probabilities $p(x, a, y)$ are as defined in (3.5.3) (after appropriate translation to the $N = 1$ case), with $\Delta \in (0, (\lambda + c\mu)^{-1}]$ as the discrete-time step size. Hence, the optimality equations (5.1.2) may be written:

$$\begin{aligned} g^* + h(x) = r(x) &+ \lambda\Delta \max(h(x), h(x+1)) \\ &+ \min(x, c)\mu\Delta\, h(x-1) \\ &+ (1 - \lambda\Delta - \min(x, c)\mu\Delta)\, h(x) \qquad (x \in \mathbb{N}_0), \end{aligned} \tag{5.1.5}$$

where it should be noted that $\min(x, c)\mu\Delta = 0$ if $x = 0$, so that the non-existence of $h(x-1)$ in this case constitutes only a minor abuse of notation rather than an invalidation of the equation. *Throughout this chapter, extensive use will be made of the principle (justified by Theorem 4.2.4) that a socially optimal policy θ^* can be found by applying relative value iteration on the finite state space $\tilde{S}$.* Accordingly, in this section, the following conventions are assumed:

- The values $h(x)$ satisfying the equations (5.1.2) for states $x \in \{0, 1, ..., \tilde{T}\}$ are assumed to be found by relative value iteration on $\{0, 1, ..., \tilde{T}\}$, with $x = 0$ chosen as a reference state and the real-time reward formulation (5.1.3) used. Thus, these values can be found by considering a problem in which balking ($a = 0$) is the only action permitted at state $\tilde{T}$.

- Similarly, the notation $\hat{h}(\cdot)$ is used to represent the function satisfying the equations which are obtained by replacing h with $\hat{h}$ and $r(x)$ with $\hat{r}(x, a)$ in (5.1.2), and the values $\hat{h}(x)$ for states $x \in \{0, 1, ..., \tilde{T}\}$ are obtained by relative value iteration on $\{0, 1, ..., \tilde{T}\}$.

- θ^* denotes the socially optimal policy which chooses the action attaining the maximum in (5.1.2) for each state $x \in \{0, 1, ..., \tilde{T} - 1\}$, with joining ($a = 1$) chosen in the event of a tie. It may be assumed that balking is chosen by θ^* at all states $x \geq \tilde{T}$.

These conventions will help to avoid some ambiguity, since socially optimal policies are generally non-unique, even when attention is restricted to the stationary class (see, for example, [78]). Furthermore, they enable inductive proofs to be constructed in a straightforward way. Many of the results in this section are proved using somewhat repetitive arguments, so it will be desirable to keep these proofs as brief as possible. Accordingly, it will be useful to introduce some shorthand notation for the first-order difference $h(x+1) - h(x)$. Given any state $x \in \mathbb{N}_0$ and an arbitrary real-valued function $f : \mathbb{N}_0 \to \mathbb{R}$, let the operator $D(x, f)$ be defined as follows:

$$D(x, f) := f(x+1) - f(x).$$

Also, in order to shorten some of the proofs that follow later, it will be useful to state a general principle which will be relied upon frequently. The result is stated in the context of a general N-facility system, since it will be useful to refer to it in later sections.

Lemma 5.1.1. ***(Action Selection Principle (ASP))***

Let $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ and $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n$ be arbitrary states in S, for some $n \in \mathbb{N}$. Suppose there exist actions $a_1, a_2, ..., a_n$ permissible at states $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ respectively, such that:

$$\sum_{i=1}^{n} f(\mathbf{x}_i^{a_i+}) - \sum_{i=1}^{n} \max_{a \in A_{\mathbf{y}_i}} f(\mathbf{y}_i^{a+}) \geq 0, \tag{5.1.6}$$

where $f(\cdot)$ is an arbitrary real-valued function. Then the following also holds:

$$\sum_{i=1}^{n} \max_{a \in A_{\mathbf{x}_i}} f(\mathbf{x}_i^{a+}) - \sum_{i=1}^{n} \max_{a \in A_{\mathbf{y}_i}} f(\mathbf{y}_i^{a+}) \geq 0, \tag{5.1.7}$$

Similarly, suppose that for some actions $b_1, b_2, ..., b_n$ permissible at $\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n$ respectively:

$$\sum_{i=1}^{n} \max_{a \in A_{\mathbf{x}_i}} f(\mathbf{x}_i^{a+}) - \sum_{i=1}^{n} f(\mathbf{y}_i^{b_i+}) \leq 0. \tag{5.1.8}$$

Then the following also holds:

$$\sum_{i=1}^{n} \max_{a \in A_{\mathbf{x}_i}} f(\mathbf{x}_i^{a+}) - \sum_{i=1}^{n} \max_{a \in A_{\mathbf{y}_i}} f(\mathbf{y}_i^{a+}) \leq 0. \tag{5.1.9}$$

Proof. The proof is immediate by definition of the maximisation operator; specifically, $\max_{a \in A_{\mathbf{x}}} f(\mathbf{x}^{a+}) \geq f(\mathbf{x}^{b+})$ for any state $\mathbf{x} \in S$ and action $b \in A_{\mathbf{x}}$. $\square$

The implication of Lemma 5.1.1 is that, in order to show that an inequality of the form (5.1.7) holds, it is sufficient to select some 'convenient' actions $a_1, a_2, ..., a_n$ for which it can be shown that (5.1.6) holds (with a similar procedure for establishing an inequality of the form (5.1.9)). In the proofs that follow, it will often be possible to show that the same actions $a_1, a_2, ..., a_n$ which maximise $f(\mathbf{y}_1^{a+}), f(\mathbf{y}_2^{a+}), ..., f(\mathbf{y}_n^{a+})$ can also be chosen at states $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$, with the result that (5.1.6) holds. This will be an important strategy throughout this chapter.

The next result establishes important structural properties of the relative value functions h and $\hat{h}$. Specifically, the properties of uniform upper boundedness and monotonicity for the first-order differences $D(x,h) = h(x+1) - h(x)$ and $D(x,\hat{h}) = \hat{h}(x+1) - \hat{h}(x)$ will be proved.

Lemma 5.1.2. *The function h satisfies the properties:*

$$D(x,h) \leq (\alpha - \beta/\mu)/\Delta \qquad \forall\, x \in \{0,1,...,\tilde{T}-1\}, \tag{5.1.10}$$

$$D(x+1,h) - D(x,h) \leq 0 \qquad \forall\, x \in \{0,1,...,\tilde{T}-2\}. \tag{5.1.11}$$

On the other hand, the function $\hat{h}$ satisfies:

$$D(x,\hat{h}) \leq 0 \qquad \forall\, x \in \{0,1,...,\tilde{T}-1\}, \tag{5.1.12}$$

$$\hat{r}(x+1,1) - \hat{r}(x,1) + \lambda\Delta\big(D(x+1,\hat{h}) - D(x,\hat{h})\big) \leq 0 \qquad \forall\, x \in \{0,1,...,\tilde{T}-2\}. \tag{5.1.13}$$

Proof. By Theorem 3.7.4, it is sufficient to consider the finite-stage functions h_n and $\hat{h}_n$ obtained during relative value iteration and show that for all integers $n \geq 0$:

$$D(x,h_n) \leq (\alpha - \beta/\mu)/\Delta \qquad \forall\, x \in \{0,1,...,\tilde{T}-1\}, \tag{5.1.14}$$

$$D(x+1,h_n) \leq D(x,h_n) \qquad \forall\, x \in \{0,1,...,\tilde{T}-2\}, \tag{5.1.15}$$

$$D(x,\hat{h}_n) \leq 0 \qquad \forall\, x \in \{0,1,...,\tilde{T}-1\}, \tag{5.1.16}$$

$$\hat{r}(x+1,1) + \lambda\Delta D(x+1,\hat{h}_n) \leq \hat{r}(x,1) + \lambda\Delta D(x,\hat{h}_n) \qquad \forall\, x \in \{0,1,...,\tilde{T}-2\}. \tag{5.1.17}$$

The proof can be accomplished using induction. For details, see Appendix A.5, page 426. □

The results of Lemma 5.1.2 can be used to establish the optimality of *threshold policies* in single-facility systems. Threshold policies for $M/M/1$ queues have already been discussed in Chapter 2, and they have an identical definition in the $M/M/c$ case, as stated below.

Definition 5.1.3. ***(Threshold policy)***

Let θ be a stationary policy. If there exists an integer $T \in \mathbb{N}_0$ such that $\theta(x) = 0$ if and only if $x \geq T$; then θ is said to be a threshold policy *with threshold T.*

The next result introduces the concept of a monotone control policy by showing that it is possible to find selfishly and socially optimal policies for the system which are *threshold* policies. This has also been shown by Knudsen [103] using a different approach. Recall that θ^* denotes the stationary policy for which $\theta^*(x)$ attains the maximum in (5.1.2) for all $x \in \{0, 1, ..., \tilde{T}\}$, with joining chosen over balking in the event of a tie and balking chosen at all states $x \geq \tilde{T}$.

Theorem 5.1.4. *The selfishly optimal and socially optimal policies $\tilde{\theta}$ and θ^* are both threshold policies with thresholds $\tilde{T} \geq 1$ and $T^* \geq 1$ respectively. Moreover:*

$$T^* \leq \tilde{T}.$$

Proof. First, consider the selfishly optimal case. As discussed earlier, $\tilde{\theta}(x) = 1$ if and only if:

$$x < \left\lfloor \frac{\alpha c \mu}{\beta} \right\rfloor,$$

so $\tilde{T} = \lfloor \alpha c \mu / \beta \rfloor$ is the selfishly optimal threshold. The fact that $\tilde{T} \geq 1$ follows from the assumption (made in Section 3.1) that $\alpha - \beta/\mu > 0$, and hence $\alpha\mu/\beta > 1$. Next, consider the socially optimal case. The policy θ^* chooses to balk at state $x \in \{0, 1, ..., \tilde{T} - 1\}$ if and only if:

$$h(x + 1) < h(x).$$

In order to show that θ^* is a threshold policy, it suffices to show that if balking is chosen by θ^* at some state $x \in \{0, 1, ..., \tilde{T} - 2\}$, then it must also be chosen at state $x + 1$. That is:

$$\theta^*(x) = 0 \;\; \Rightarrow \;\; \theta^*(x + 1) = 0.$$

Equivalently, since $\theta^*(x) = 0$ if and only if $h(x + 1) < h(x)$:

$$h(x + 1) < h(x) \;\; \Rightarrow \;\; h(x + 2) < h(x + 1).$$

This follows directly from the property $D(x + 1, h) \leq D(x, h)$ proved in Lemma 5.1.2, so θ^* is a socially optimal policy which is also a threshold policy with some threshold $T^* \in \mathbb{N}$. Finally, the

fact that $T^* \leq \tilde{T}$ is a direct consequence of the containment property for socially optimal policies established (in greater generality) by Theorem 4.2.4 and Theorem 4.2.5. □

The next lemma addresses the progression of the first-order differences $D(x, h_n)$ and $D(x, \hat{h}_n)$ obtained during relative value iteration, and in doing so highlights an interesting contrast between the real-time and anticipatory reward formulations in (5.1.3) and (5.1.4) respectively.

Lemma 5.1.5. *The finite-stage relative value functions h_n and $\hat{h}_n$ corresponding to the reward formulations (5.1.3) and (5.1.4) respectively satisfy, for all integers $n \in \mathbb{N}_0$:*

$$D(x, \hat{h}_{n+1}) \leq D(x, \hat{h}_n) \qquad \forall\, x \in \{0, 1, ..., \tilde{T} - 1\}, \tag{5.1.18}$$

$$D(x, h_{n+1}) \geq 0 \;\Rightarrow\; D(x, h_{n+2}) \geq 0 \qquad \forall\, x \in \{0, 1, ..., \tilde{T} - 1\}. \tag{5.1.19}$$

Proof. Again, the results can be proved by induction. See Appendix A.5, page 433.

An implication of Lemma 5.1.5 is that if the anticipatory reward formulation is used and $D(x, \hat{h}_n) < 0$ for some $x \in \{0, 1, ..., \tilde{T} - 1\}$ on iteration $n \geq 0$ of relative value iteration, then $D(x, \hat{h}_m) \leq D(x, \hat{h}_n) < 0$ for all $m > n$ and (by taking limits) $D(x, \hat{h}) < 0$. Therefore if one considers a *finite horizon* problem with n stages, it may be said that balking becomes a 'more attractive' option at x as the finite horizon n increases in length. On the other hand, under the real-time formulation, the converse result is true. That is, if $D(x, h_n) \geq 0$ on iteration $n \geq 1$ of relative value iteration, then $D(x, h_m) \geq 0$ for all $m > n$ and hence $D(x, h) \geq 0$. So, under this formulation, joining becomes *more* attractive as n increases. These insights into the nature of optimal finite-horizon policies under the respective reward formulations will prove to be useful in Section 5.4.

Naturally, by definition of the socially optimal policy, the expected long-run average reward attained under the optimal threshold T^* must be greater than or equal to that attained under the selfish policy with threshold $\tilde{T}$. The next result shows that in fact, any of the 'intermediate' thresholds T satisfying $T^* < T < \tilde{T}$ yield a better performance than the selfish threshold $\tilde{T}$.

Lemma 5.1.6. *Let g_T denote the expected long-run average reward attained under a threshold policy with threshold $T \in \mathbb{N}_0$. Then:*

$$g_{T^*} \geq g_{T^*+1} \geq ... \geq g_{\tilde{T}-1} \geq g_{\tilde{T}}, \tag{5.1.20}$$

where $\tilde{T}$ and T^* are, respectively, the thresholds corresponding to the selfishly optimal policy $\tilde{\theta}$ and the socially optimal policy θ^* found by relative value iteration.

Proof. If $T^* = \tilde{T}$ or $T^* = \tilde{T} - 1$ then the result is immediate since $g_{T^*} \geq g_{\tilde{T}}$ by definition of the policy θ^*. Suppose $T^* \leq \tilde{T} - 2$. The aim of this proof is to show that $g_T \geq g_{T+1}$ for any threshold T satisfying $T^* \leq T < \tilde{T}$. Consider the evolution of the Relative Value Iteration Algorithm (RVIA), assuming that the *anticipatory* reward formulation (5.1.4) is used. Since the RVIA converges to the policy with threshold T^*, there exists an integer k such that for all $n \geq k$:

$$\hat{r}(x,1) + \lambda \Delta D(x, \hat{h}_n) \geq 0 \quad \forall\, x \in \{0, 1, ..., T^* - 1\}, \tag{5.1.21}$$

$$\hat{r}(x,1) + \lambda \Delta D(x, \hat{h}_n) < 0 \quad \forall\, x \in \{T^*, T^* + 1, ..., \tilde{T} - 1\}. \tag{5.1.22}$$

Importantly, this implies that (5.1.21) holds for all integers $n \geq 0$, since if the left-hand side of (5.1.21) was negative at some stage $n = n_0$, then Lemma 5.1.5 would imply that it would remain negative at all subsequent stages $n > n_0$. Let U_n be defined for $n \geq 1$ as follows:

$$U_n := \min \left\{ x \in \{0, 1, ..., \tilde{T} - 1\} : \hat{r}(x,1) + \lambda \Delta D(x, \hat{h}_{n-1}) < 0 \right\}. \tag{5.1.23}$$

In cases where $\{x \in \{0, 1, ..., \tilde{T} - 1\} : \hat{r}(x,1) + \lambda \Delta D(x, \hat{h}_{n-1}) < 0\}$ is an empty set, $U_n = \tilde{T}$ will be assumed. Hence, U_n may be referred to as the 'optimal finite-stage threshold' at stage n of the RVIA. Clearly, $U_1 = \tilde{T}$ since $\hat{h}_0(x) = 0$ for all $x \in \{0, 1, ..., \tilde{T}\}$ and $\hat{r}(x,1) < 0$ if and only if $x \geq \tilde{T}$. The sequence $(U_1, U_2, U_3, ...)$ must be monotonically decreasing due to Lemma 5.1.5. Furthermore, $\lim_{n\to\infty} U_n = T^*$ since the RVIA eventually converges to the T^*-threshold policy.

Given some threshold $T \in \mathbb{N}_0$ satisfying $T^* < T < \tilde{T}$, consider a re-defined problem where balking is *not allowed* at any state $x < T$; that is, the action set at any state $x < T$ consists only of the single action $a = 1$. Suppose the RVIA is applied to this problem, again assuming that the anticipatory reward formulation is used. It can easily be verified that the inequality (5.1.18) in Lemma 5.1.5 still holds when the problem is re-defined in this way. Indeed, this can be done without having to rely upon Lemma 5.1.2, since Lemma 5.1.2 was used in the proof of Lemma 5.1.5 only in order to rule out the decision-making case $a_1 = 1$ and $b_0 = 0$ in (A.5.20), but even if this case is allowed then one can simply choose $a_0 = 0$ and $b_1 = 1$ in order to show that (A.5.20) holds.

Let U_n be defined as in (5.1.23), and also let $n_T \in \mathbb{N}_0$ be defined as follows:

$$n_T := \min \{ n \in \mathbb{N}_0 : U_n \leq T \}.$$

That is, n_T is the earliest iteration of the RVIA at which the threshold 'drops' to T. It must be the case that n_T is *finite*, i.e. the algorithm will reach a threshold of T at some point. This is because,

for $n < n_T$, the iterates $\hat{h}_n(\cdot)$ take exactly the same values in the re-defined problem as they do in the original problem (with balking allowed at all states). The fact that balking is forbidden at states $x < T$ makes no difference during the early iterations $n < n_T$, since balking would not be chosen at any state $x < T$ on these iterations in the original problem. So, given that the RVIA converges to the threshold $T^* < T$ in the original problem, there must be a stage n_T during its evolution at which $\hat{r}(T,1) + \lambda\Delta D(T, \hat{h}_{n_T}) < 0$, and therefore the same inequality holds at the same stage in the re-defined problem. Due to (5.1.18), this implies that for all $n \geq n_T$:

$$\hat{r}(x,1) + \lambda\Delta D(x, \hat{h}_n) < 0 \quad \forall\, x \geq T.$$

Hence, the RVIA converges to the threshold policy with threshold T in the re-defined problem, which implies that the T-threshold policy is superior (or at least equal) to any larger-threshold policy in terms of expected long-run average reward. In particular, it is superior to the policy with threshold $T+1$; that is, $g_T \geq g_{T+1}$ as required. Since this argument can be applied to any threshold T satisfying $T^* < T < \tilde{T}$, this completes the proof of the lemma. □

Incidentally, it is also possible to use an argument very similar to that in the previous proof which uses the real-time reward formulation (5.1.3) and the property (5.1.19) proved in Lemma 5.1.5 in order to show that the average rewards $g_0, g_1, ..., g_{T^*}$ (where g_T again denotes the average reward earned under the threshold policy with threshold T) satisfy the relationship:

$$g_0 \leq g_1 \leq ... \leq g_{T^*}. \tag{5.1.24}$$

Combining (5.1.24) with the result (5.1.20) proved by Lemma 5.1.6 then shows that the average reward g_T is *unimodal* in T; that is, a local maximum is also a global maximum. This is a result also proved by Knudsen [103] (p. 520), and is illustrated in Figure 5.1. This unimodality property will not be needed in subsequent results, but Lemma 5.1.6 will prove to be useful.

The next result concerns the effect of varying the demand rate λ on the relative values $h(x)$ and the first-order differences $D(x,h)$. The result is given for the real-time reward formulation (5.1.3) only, although it would be possible to obtain a similar result under the alternative formulation (5.1.4). For each $x \in \{0,1,...,\tilde{T}\}$, let $h(x,\lambda)$ denote the relative value $h(x)$ (as determined by relative value iteration) given a demand rate $\lambda > 0$. Also, for each $x \in \{0,1,...,\tilde{T}-1\}$, define:

$$D(x,\lambda,h) := h(x+1,\lambda) - h(x,\lambda),$$

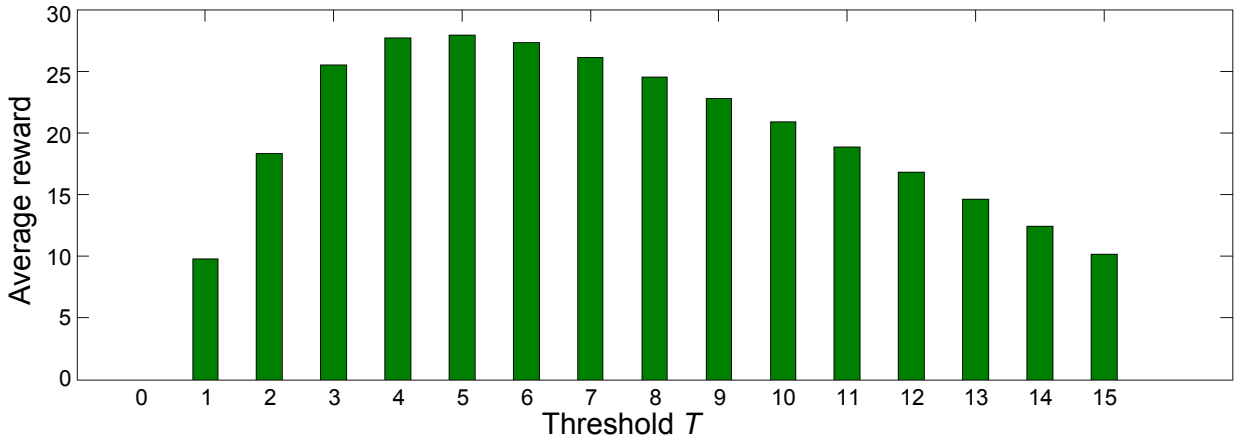


Figure 5.1: Expected long-run average rewards under various different threshold policies in a system with $\lambda = 6$, $c = 3$, $\mu = 2$, $\alpha = 9$, $\beta = 5$. The selfishly and socially optimal thresholds are $\tilde{T} = 10$ and $T^* = 5$.

$$D(x, \lambda, h_n) := h_n(x+1, \lambda) - h_n(x, \lambda) \quad \forall\, n \in \mathbb{N}_0.$$

Lemma 5.1.7. *Suppose that, for some demand rate $\lambda_0 > 0$ and state $x \in \{0, 1, ..., \tilde{T} - 1\}$:*

$$D(x, \lambda_0, h) < 0.$$

Then, for all *demand rates $\lambda \in (\lambda_0, \infty)$:*

$$D(x, \lambda, h) < 0. \tag{5.1.25}$$

Proof. The proof again relies upon induction. See Appendix A.5, page 436.

From Lemma 5.1.7 one obtains the following important result.

Theorem 5.1.8. *Let $T^*(\lambda)$ denote the threshold associated with the socially optimal policy θ^*_λ obtained using relative value iteration, given a demand rate $\lambda > 0$. Then:*

- *$T^*(\lambda)$ is monotonically decreasing with λ;*
- $\lim_{\lambda \to \infty} T^*(\lambda) = c$.

Proof. Note that the fact that θ^*_λ is a threshold policy is already known from Theorem 5.1.4. Given a demand rate $\lambda > 0$, the policy θ^*_λ chooses actions in the following way:

$$\theta^*_\lambda(x) = \begin{cases} 1, & \text{if } D(x, \lambda, h) \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

The socially optimal threshold $T^*(\lambda)$ may therefore be defined as:

$$T^*(\lambda) = \min\left\{x \in \{0, 1, ..., \tilde{T} - 1\} : D(x, \lambda, h) < 0\right\}, \tag{5.1.26}$$

with $T^*(\lambda) = \tilde{T}$ in the case where $\{x \in \{0, 1, ..., \tilde{T} - 1\} : D(x, \lambda, h) < 0\}$ is an empty set. Lemma 5.1.7 states that if $D(x, \lambda_0, h) < 0$ for some demand rate $\lambda_0 > 0$, then $D(x, \lambda_1, h) < 0$ for all demand rates $\lambda_1 > \lambda_0$. Therefore, given $\lambda_0 < \lambda_1$, (5.1.26) implies that:

$$T^*(\lambda_1) \leq T^*(\lambda_0).$$

The fact that $\lim_{\lambda\to\infty} T^*(\lambda) = c$ can be argued using the fact that, for sufficiently large values of λ, the policy with threshold $T^*(\lambda) = c$ is the only threshold policy which attains average reward optimality. Note that the fact that the c-threshold policy is optimal in a heavy-traffic limit is already known from Theorem 4.4.7, since it is an example of a vacancy policy (see Definition 4.4.6). However, it is desirable to show that the c-threshold policy is the *only* socially optimal threshold policy in heavy traffic, in order to establish that it is not possible for relative value iteration to converge to any other policy. Let $\sigma > 0$ be defined as follows:

$$\sigma := \max\left(r(c-1),\ r(c+1)\right).$$

Due to the fact that the reward $r(x)$ is strictly increasing with x when $x < c$ and strictly decreasing when $x > c$, it follows that $\sigma \geq r(x)$ for any $x \neq c$. Let θ be a policy with threshold $T \neq c$. Using standard results for finite-buffer $M/M/c$ queues (see, for example, [67], p. 75), the steady-state probability $\pi_\theta(x)$ of the system being in state $x \in \mathbb{N}_0$ satisfies the following:

$$\lim_{\lambda\to\infty} \pi_\theta(x) = \begin{cases} 1, & \text{if } x = T, \\ 0, & \text{otherwise.} \end{cases} \tag{5.1.27}$$

Let ϵ be a positive quantity whose value is given by:

$$\epsilon := \frac{r(c) - \sigma}{r(c) - \sigma + \alpha c\mu} > 0. \tag{5.1.28}$$

Note that ϵ depends only on the parameters α, β, μ and c, and not on the demand rate λ. Due to (5.1.27), there exists a value $\delta(\epsilon) > 0$ such that, for all $\lambda > \delta(\epsilon)$:

$$g_\theta(\lambda) = \pi_\theta(T)\, r(T) + \sum_{x\neq T} \pi_\theta(x)\, r(x) < (1 - \epsilon)\sigma + \epsilon\, \alpha c\mu, \tag{5.1.29}$$

where $g_\theta(\lambda)$ is the long-run average reward under policy θ given demand rate λ, and the inequality is due to the fact that $r(x) < \alpha c\mu$ for all $x \in \mathbb{N}_0$ and (since $T \neq c$), $r(T) \leq \sigma$. Meanwhile, let θ^* be the threshold policy with threshold c. Then, for all demand rates $\lambda > \delta(\epsilon)$, using the fact that only states $x \leq c$ are positive recurrent under θ^*, one may write:

$$g_{\theta^*}(\lambda) = \pi_\theta(c)\, r(c) + \sum_{x<c} \pi_\theta(x)\, r(x) > (1-\epsilon)\, r(c), \tag{5.1.30}$$

where the inequality is due to the fact that $r(x) \geq 0$ for all $x \leq c$. Suppose, for a contradiction, that $g_\theta(\lambda) \geq g_{\theta^*}(\lambda)$ for some $\lambda > \delta(\epsilon)$. Then, in view of (5.1.29) and (5.1.30):

$$(1-\epsilon)\sigma + \epsilon\, \alpha c\mu > (1-\epsilon) r(c).$$

However, this is equivalent to:

$$\epsilon > \frac{r(c) - \sigma}{r(c) - \sigma + \alpha c\mu},$$

which contradicts the definition of ϵ in (5.1.28). The conclusion is that, for sufficiently large demand rates, the policy θ^* with threshold c attains a strictly greater average reward than any other threshold policy. This completes the proof of the theorem. □

The next lemma is required in order to prove the final result of this section.

Lemma 5.1.9. *Let $\tilde{g}(\lambda)$ be the expected long-run average reward attained under the selfish policy $\tilde{\theta}$ given a demand rate $\lambda > 0$, and let $g^*(\lambda)$ be the optimal expected long-run average reward under the same demand rate. Then $\tilde{g}(\lambda)$ and $g^*(\lambda)$ are both continuous in λ.*

Proof. In the case of the selfish policy $\tilde{\theta}$, it suffices to show that the expected long-run average reward under a fixed threshold policy (where the threshold is independent of λ) is continuous in λ. In the case of the optimal value $g^*(\lambda)$, one may consider the threshold $T^*(\lambda)$ associated with the policy θ^*_λ found by relative value iteration, and show that $g^*(\lambda)$ is continuous at the critical values of λ where $T^*(\lambda)$ changes. For details, see Appendix A.5, page 438. □

The next result brings together all of the previous results in order to prove an interesting result for $M/M/1$ queues. In the special case $c = 1$, it is possible to show that the (absolute) sub-optimality of the selfishly optimal policy $\tilde{\theta}$ increases monotonically with the demand rate λ. The proof relies

on the construction of a stochastic coupling (see Section 3.8); however, in this particular case it is possible to obtain the stationary distribution for the coupling explicitly and then prove the result by examining the derivatives of the stationary probabilities with respect to λ.

Theorem 5.1.10. *Consider an $M/M/1$ system ($c = 1$). Given a demand rate $\lambda > 0$, let $L(\lambda)$ be the 'loss per unit time' due to selfish decision-making, defined as follows:*

$$L(\lambda) = g^*(\lambda) - \tilde{g}(\lambda),$$

where $g^*(\lambda)$ and $\tilde{g}(\lambda)$ are (respectively) the optimal expected long-run average reward and the average reward under the selfish policy $\tilde{\theta}$. Then $L(\lambda)$ is monotonically increasing with λ.

Proof. Naturally, all of the previous results in this section apply to the $M/M/1$ queue since it is merely a special case of an $M/M/c$ queue. For any demand rate $\lambda > 0$, Theorem 5.1.4 implies that the optimal expected long-run average reward $g^*(\lambda)$ is attained by a policy θ^*_λ with a threshold $T^*(\lambda)$ satisfying $T^*(\lambda) \leq \tilde{T}$, where $\tilde{T} = \lfloor \alpha\mu/\beta \rfloor$ is the selfishly optimal threshold. By Theorem 5.1.8, $T^*(\lambda)$ is monotonically decreasing with λ and $\lim_{\lambda\to\infty} T^*(\lambda) = c = 1$, assuming that $T^*(\lambda)$ corresponds to the policy found by relative value iteration. As stated in the proof of Lemma 5.1.9, it follows that there exist m demand rates $\lambda_1 < \lambda_2 < ... < \lambda_m$ and $m + 1$ thresholds $T^*_0 > T^*_1 > ... > T^*_{m-1} > T^*_m = 1$, such that, for $i = 0, 1, ..., m - 1$:

$$T^*(\lambda) = T^*_i \quad \forall\, \lambda \in (\lambda_i, \lambda_{i+1}], \tag{5.1.31}$$

where $\lambda_0 = 0$ and $T^*(\lambda) = 1$ for $\lambda > \lambda_m$. Let $g_T(\lambda)$ denote the average reward under the policy with threshold $T \in \mathbb{N}_0$ given a demand rate $\lambda > 0$. For any fixed $\lambda > 0$, one has $L(\lambda) = g_{T^*_i}(\lambda) - \tilde{g}(\lambda)$ for some $i \in \{0, 1, .., m\}$. Due to the continuity of $\tilde{g}(\lambda)$ and $g^*(\lambda)$ proved by Lemma 5.1.9, $L(\lambda)$ is continuous at the points λ_i where the optimal threshold changes. Therefore it remains only to show that $L(\lambda)$ is monotonically increasing in each of the intervals where the optimal threshold remains constant. More specifically, it is sufficient to show that $L(\lambda)$ is monotonically increasing with λ in the intervals $(\lambda_i, \lambda_{i+1}]$ (for $i \in \{0, 1, .., m - 1\}$) and also (λ_m, ∞).

Given any $\lambda > 0$, one has $L(\lambda) = g_{T^*_i}(\lambda) - \tilde{g}(\lambda)$ for some $i \in \{0, 1, ..., m\}$. Equivalently:

$$L(\lambda) = \left(g_{T^*_i}(\lambda) - g_{T^*_i+1}(\lambda)\right) + \left(g_{T^*_i+1}(\lambda) - g_{T^*_i+2}(\lambda)\right) + ... + \left(g_{\tilde{T}-1}(\lambda) - g_{\tilde{T}}(\lambda)\right), \tag{5.1.32}$$

where the bracketed expressions are all non-negative due to Lemma 5.1.6. Let $i \in \{0, 1, ..., m\}$ be fixed. Due to (5.1.32), it will be sufficient to consider two threshold policies with thresholds T and $T+1$ respectively, where it is assumed that $T_i^* \leq T < \tilde{T}$, and show that the difference $g_T(\lambda) - g_{T+1}(\lambda)$ is monotonically increasing with λ in the interval $(\lambda_i, \lambda_{i+1}]$. Given a fixed service rate μ, holding cost β and reward for service α, let $(x_n)_{n\in\mathbb{N}_0}$ and $(y_n)_{n\in\mathbb{N}_0}$ denote the state-time evolutions of two processes following threshold policies with thresholds T and $T+1$ respectively. Furthermore, let $(\hat{x}_n, \hat{y}_n)_{n\in\mathbb{N}_0}$ be a coupled process, constructed in such a way that the marginal processes $(\hat{x}_n)$ and $(\hat{y}_n)$ have distributions identical to those of (x_n) and (y_n) respectively (refer to Section 3.8 for some background details on the coupling technique). Specifically, let it be assumed that the random transitions of the coupling $(\hat{x}_n, \hat{y}_n)$ obey the following rules:

- At any time step n, there is a probability $\lambda\Delta$ that both of the marginal processes $(\hat{x}_n)$ and $(\hat{y}_n)$ see an arrival. In the case of $(\hat{x}_n)$, the new customer joins if and only if $\hat{x}_n < T$; in the case of $(\hat{y}_n)$, the customer joins if and only if $\hat{y}_n < T+1$.
- With the remaining probability $\mu\Delta = 1 - \lambda\Delta$, a service completion occurs; this event is 'seen by' (i.e. affects) the marginal process $(\hat{x}_n)$ if and only if $\hat{x}_n \geq 1$, and similarly it is seen by the other marginal process $(\hat{y}_n)$ if and only if $\hat{y}_n \geq 1$.

It is easy to see that these transition rules ensure that, given any $n \in \mathbb{N}_0$, the distributions of $\hat{x}_n$ and $\hat{y}_n$ are identical to those of x_n and y_n respectively. Furthermore, it is guaranteed that one must have either $\hat{y}_n = \hat{x}_n$ or $\hat{y}_n = \hat{x}_n + 1$ at all times steps n. Assuming that the coupling is initialised in state $(0, 0)$, the set of attainable states for $(\hat{x}_n)$ is $\{0, 1, ..., T\}$, and it follows that the number of attainable states for the coupling $(\hat{x}_n, \hat{y}_n)$ during its evolution is $2(T+1)$. This may be seen from Figure 5.2, which illustrates the state transitions for the coupling. Let R denote the set of states in the coupled process with $\hat{x}_n = \hat{y}_n \leq T$ (i.e. the bottom row of states in Figure 5.2), and let G denote the complementary set of states with $\hat{x}_n = \hat{y}_n - 1 \leq T$. Here, '$R$' and '$G$' stand for 'Regular' and 'Ghost' respectively; one might imagine that the states belonging to the set G are those which represent the smaller-threshold process (x_n) carrying the 'ghost' of a customer who would have been present if the policy with threshold $T+1$ had been followed. Suppose the coupled

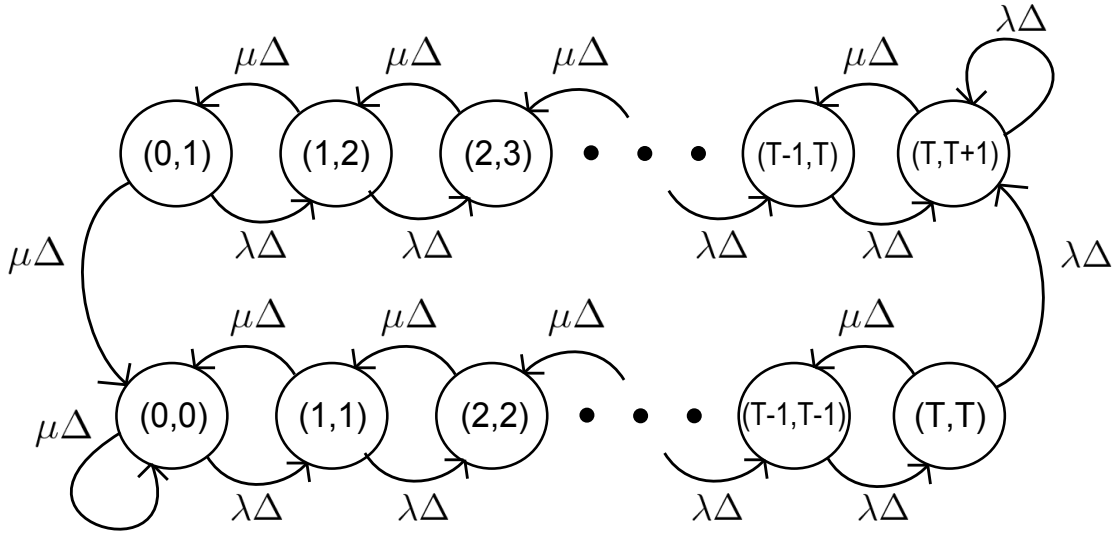


Figure 5.2: Transition diagram for the coupled process $(\hat{x}_n, \hat{y}_n)$.

process $(\hat{x}_n, \hat{y}_n)$ earns state-dependent rewards $r_c((\hat{x}_n, \hat{y}_n))$, defined as follows:

$$r_c((\hat{x}_n, \hat{y}_n)) := \begin{cases} \beta - \alpha\mu, & \text{if}(\hat{x}_n, \hat{y}_n) = (0,1), \\ \beta, & \text{if } (\hat{x}_n, \hat{y}_n) \in G \setminus \{(0,1)\}, \\ 0, & \text{if } (\hat{x}_n, \hat{y}_n) \in R. \end{cases} \tag{5.1.33}$$

Let $\pi((\hat{x}_n, \hat{y}_n))$ denote the steady-state probability of the coupled process being in state $(\hat{x}_n, \hat{y}_n)$, with the dependence of $\pi((\hat{x}_n, \hat{y}_n))$ on λ suppressed for notational convenience. It may be seen that, given any $\lambda > 0$, the expected long-run average reward $g_c(\lambda) = \sum_{(\hat{x}_n, \hat{y}_n)} \pi((\hat{x}_n, \hat{y}_n)) r_c((\hat{x}_n, \hat{y}_n))$ earned by the coupled process corresponds exactly to $g_T(\lambda) - g_{T+1}(\lambda)$, the (non-negative) difference between the average rewards under the policies with thresholds T and $T+1$. Indeed, the state $(0,1)$ represents the scenario where the process (y_n) has one customer in service, whereas (x_n) has no customers present. By the definition of the rewards in (5.1.3), (y_n) earns a positive reward $\alpha\mu - \beta$ under this scenario, whereas (x_n) earns a zero reward; in fact, this represents the only scenario in which the larger-threshold process (y_n) is 'winning' against (x_n). If $(\hat{x}_n, \hat{y}_n) \in G \setminus \{(0,1)\}$, this represents a scenario where both of the processes (x_n) and (y_n) have one customer in service, but (y_n) has one extra customer waiting in the queue and thereby incurs an extra holding cost; hence, (x_n) is at an advantage of β over (y_n) in terms of the single-step reward at that instance. Finally, the case $(\hat{x}_n, \hat{y}_n) \in R$ represents a scenario where (x_n) and (y_n) have the same number of customers

present; hence, $r_c((\hat{x}_n, \hat{y}_n)) = 0$ since there is no advantage earned by either process.

Let $\pi(R) = \sum_{(\hat{x}_n,\hat{y}_n)\in R} \pi((\hat{x}_n, \hat{y}_n))$ be the stationary probability of the coupled process being in one of the 'regular' states, and let $\pi(G) = 1 - \pi(R)$ denote the corresponding probability for the 'ghost' states. By the previous arguments, the following convenient formula relates the stationary probabilities of the coupled process to the average rewards $g_T(\lambda)$ and $g_{T+1}(\lambda)$:

$$g_T(\lambda) - g_{T+1}(\lambda) = \beta\pi(G) - \alpha\mu\pi((0,1)).$$

This may equivalently be written as:

$$g_T(\lambda) - g_{T+1}(\lambda) = \pi(G)\left(\beta - \frac{\alpha\mu\pi((0,1))}{\pi(G)}\right), \tag{5.1.34}$$

where $\pi((0,1))/\pi(G)$ (naturally) represents the conditional probability of the coupled process being in state $(0,1)$, given that it is in the set G. Note that $\beta - \alpha\mu\pi((0,1))/\pi(G)$ must be non-negative, because it is known (from Lemma 5.1.6) that $g_T(\lambda) \geq g_{T+1}(\lambda)$. In order to show that $g_T(\lambda) - g_{T+1}(\lambda)$ is monotonically increasing with λ, it therefore suffices to show:

1. $\pi(G)$ is monotonically increasing with λ,
2. $\pi((0,1))/\pi(G)$ is monotonically decreasing with λ.

One may proceed by deriving the stationary probabilities for the coupled process explicitly. First, consider the states belonging to the set $R = \{(0,0),(1,1),...,(T,T)\}$. Given a demand rate $\lambda > 0$, let $\rho = \lambda/\mu$. Obviously, showing that a certain quantity is increasing (decreasing) with λ is equivalent to showing that it is increasing (decreasing) with ρ. The steady-state balance equations (see, for example, [167]) may be derived for states $(\hat{x}_n, \hat{y}_n) \in R$ as follows:

$$\begin{aligned}
\rho\pi((0,0)) &= \pi((1,1)) + \pi((0,1)) \\
(1+\rho)\pi((1,1)) &= \pi((2,2)) + \rho\pi((0,0)) \\
(1+\rho)\pi((2,2)) &= \pi((3,3)) + \rho\pi((1,1)) \\
&\vdots \\
(1+\rho)\pi((T-1,T-1)) &= \pi((T,T)) + \rho\pi((T-2,T-2)) \\
(1+\rho)\pi((T,T)) &= \rho\pi((T-1,T-1)).
\end{aligned} \tag{5.1.35}$$

Also, using the detailed balance equations for ergodic Markov chains (see, e.g. [167]), the 'rate of flow' into the set G must equal the 'rate of flow' out of G. Hence:

$$\rho\pi((T,T)) = \pi((0,1)). \tag{5.1.36}$$

Therefore, by replacing $\pi((0,1))$ with $\rho\pi((T,T))$ in the first equation in (5.1.35), one obtains a system of $T+1$ linear equations in the $T+1$ unknowns $\pi((0,0)), \pi((1,1)), ..., \pi((T,T))$ only. Using recursive substitution, it can then be shown that the stationary probabilities $\pi((n,n))$ (for $n = 0, 1, ..., T$) may be expressed in terms of the probability $\pi((T,T))$ as follows:

$$\pi((n,n)) = \left(\frac{1+\rho+\rho^2+...+\rho^{T-n}}{\rho^{T-n}}\right)\pi((T,T)).$$

Equivalently, for $n \in \{0, 1, ..., T\}$:

$$\pi((n.n)) = \begin{cases} \dfrac{1-\rho^{T-n+1}}{\rho^{T-n}(1-\rho)}\,\pi((T,T)), & \rho \neq 1, \\ (T-n+1)\,\pi((T,T)), & \rho = 1. \end{cases} \tag{5.1.37}$$

Now consider the complementary set $G = \{(0,1), (1,2), ..., (T,T+1)\}$. The steady-state balance equations for states in G may be derived in a similar fashion:

$$\begin{gathered} (1+\rho)\pi((0,1)) = \pi((1,2)) \\ (1+\rho)\pi((1,2)) = \pi((2,3)) + \rho\pi((0,1)) \\ (1+\rho)\pi((2,3)) = \pi((3,4)) + \rho\pi((1,2)) \\ \vdots \\ (1+\rho)\pi((T-1,T)) = \pi((T,T+1)) + \rho\pi((T-2,T-1)) \\ \pi((T,T+1)) = \rho\pi((T-1,T)) + \rho\pi((T,0)). \end{gathered} \tag{5.1.38}$$

Recalling that $\rho\pi((T,T)) = \pi((0,1))$, the last equation in (5.1.38) can be re-written as:

$$\pi((T,T+1)) = \rho\pi((T-1,T)) + \pi((0,1)),$$

which yields a closed system of $T+1$ linear equations in the $T+1$ unknowns $\pi((0,1)), \pi((1,2)), ...,$ $\pi((T,T+1))$. Then, again by recursive substitution, the stationary probabilities $\pi((n,n+1))$ (for

$n = 0, 1, ..., T$) may be written in terms of $\pi((0,1))$ as follows:

$$\pi((n, n+1)) = (1 + \rho + ... + \rho^n)\ \pi((0,1)) = \begin{cases} \dfrac{1-\rho^{n+1}}{1-\rho}\,\pi((0,1)), & \rho \neq 1, \\ (n+1)\,\pi((0,1)), & \rho = 1. \end{cases}$$

Equivalently, for $n = 0, 1, ..., T$, due to (5.1.36):

$$\pi((n, n+1)) = \begin{cases} \dfrac{\rho(1-\rho^{n+1})}{1-\rho}\,\pi((T,T)), & \rho \neq 1, \\ (n+1)\,\pi((T,T)), & \rho = 1. \end{cases} \tag{5.1.39}$$

The expressions for the stationary probabilities in (5.1.37) and (5.1.39) are given in terms of the unknown $\pi((T,T))$, so it is obviously desirable to obtain an expression for $\pi((T,T))$ in terms of ρ only. This can be done using the standard normalisation condition that $\sum_{n=0}^{T}(\pi((n,n)) + \pi((n,n+1))) = 1$. After some simplifications, one obtains the following:

$$\pi((T,T)) = \begin{cases} \left(\displaystyle\sum_{n=0}^{T} \frac{1-\rho^{T+2}}{\rho^{T-n}(1-\rho)}\right)^{-1}, & \rho \neq 1, \\ \dfrac{1}{(T+1)(T+2)}, & \rho = 1. \end{cases} \tag{5.1.40}$$

Then, using (5.1.37) and (5.1.39), the expressions for $\pi(R)$ and $\pi(G)$ are:

$$\pi(R) = \begin{cases} \dfrac{1-\rho^{T+1}\,(T(1-\rho)-\rho+2)}{1-\rho^{T+1}\,(1+\rho-\rho^{T+2})}, & \rho \neq 1, \\ 1/2, & \rho = 1. \end{cases} \tag{5.1.41}$$

$$\pi(G) = \begin{cases} \dfrac{\rho^{T+1}\left(T(1-\rho)+1-2\rho+\rho^{T+2}\right)}{1-\rho^{T+1}\,(1+\rho-\rho^{T+2})}, & \rho \neq 1, \\ 1/2, & \rho = 1. \end{cases} \tag{5.1.42}$$

In order to show that $\pi(R)$ is monotonically decreasing with λ (equivalently, $\pi(G) = 1 - \pi(R)$ is increasing with λ), it will be useful to put (5.1.41) and (5.1.42) into an alternative form. First, note that the expression for $\pi((T,T))$ in (5.1.40) may be re-written as:

$$\pi((T,T)) = \left(\sum_{n=0}^{T} \frac{1+\rho+...+\rho^{T+1}}{\rho^{T-n}}\right)^{-1} = \left((1+\rho+...+\rho^{T+1})\left(1+\frac{1}{\rho}+...+\frac{1}{\rho^T}\right)\right)^{-1}$$

$$= \frac{\rho^T}{(1+\rho+...+\rho^{T+1})(1+\rho+...+\rho^T)} = \frac{\rho^T}{\displaystyle\sum_{n=0}^{T}(n+1)\rho^n + \rho^{T+1}\sum_{n=0}^{T}(T+1-n)\rho^n}. \tag{5.1.43}$$

Then, after substituting (5.1.43) into (5.1.37), further manipulations yield:

$$\pi(R) = \frac{\sum_{n=0}^{T}(n+1)\rho^n}{\sum_{n=0}^{T}(n+1)\rho^n + \rho^{T+1}\sum_{n=0}^{T}(T+1-n)\rho^n} = \left(1 + \frac{\rho^{T+1}\sum_{n=0}^{T}(T+1-n)\rho^n}{\sum_{n=0}^{T}(n+1)\rho^n}\right)^{-1}, \quad (5.1.44)$$

$$\pi(G) = \frac{\rho^{T+1}\sum_{n=0}^{T}(T+1-n)\rho^n}{\sum_{n=0}^{T}(n+1)\rho^n + \rho^{T+1}\sum_{n=0}^{T}(T+1-n)\rho^n} = \left(1 + \frac{\sum_{n=0}^{T}(n+1)\rho^n}{\rho^{T+1}\sum_{n=0}^{T}(T+1-n)\rho^n}\right)^{-1}. \quad (5.1.45)$$

For $\rho > 0$, let the function $f(\rho)$ be defined as:

$$f(\rho) := \frac{\rho^{T+1}\sum_{n=0}^{T}(T+1-n)\rho^n}{\sum_{n=0}^{T}(n+1)\rho^n}. \quad (5.1.46)$$

Then, referring to (5.1.44), it suffices to show (in order to show that $\pi(R)$ is monotonically decreasing with ρ) that $f(\rho)$ is increasing with ρ. This can be done by splitting up the numerator in (5.1.46) and then differentiating term-by-term. Firstly, one may write:

$$f(\rho) = \sum_{n=0}^{T}\frac{(T+1-n)\rho^{T+n+1}}{\sum_{j=0}^{T}(j+1)\rho^j}. \quad (5.1.47)$$

Then, differentiating term-by-term yields the following:

$$\frac{\partial f}{\partial \rho} = \sum_{n=0}^{T}\frac{(T+1-n)\rho^{T+n}\left((T+1+n)\sum_{j=0}^{T}(j+1)\rho^j - \sum_{j=0}^{T}j(j+1)\rho^j\right)}{\left(\sum_{j=0}^{T}(j+1)\rho^j\right)^2}. \quad (5.1.48)$$

Therefore in order to show $\partial f/\partial\rho \geq 0$, it suffices to show that for $n \in \{0, 1, ..., T\}$:

$$(T+1+n)\sum_{j=0}^{T}(j+1)\rho^j - \sum_{j=0}^{T}j(j+1)\rho^j \geq 0. \quad (5.1.49)$$

This is equivalent to showing, for $n \in \{0, 1, ..., T\}$:

$$\sum_{j=0}^{T}(T+1+n-j)(j+1)\rho^j \geq 0, \quad (5.1.50)$$

and this is clearly true since $(T+1+n-j)$ is positive for $j \in \{0,1,...,T\}$. The remaining task, in order to show that the difference $g_T(\lambda) - g_{T+1}(\lambda)$ is increasing with λ, is to show that the conditional probability $\pi((0,1))/\pi(G)$ is monotonically *decreasing* with λ. This can be done very easily using the previous results. Indeed, using (5.1.36) and (5.1.43):

$$\pi((0,1)) = \rho\pi((T,T)) = \frac{\rho^{T+1}}{\sum_{n=0}^{T}(n+1)\rho^n + \rho^{T+1}\sum_{n=0}^{T}(T+1-n)\rho^n}. \tag{5.1.51}$$

Due to (5.1.45), this implies the following:

$$\frac{\pi((0,1))}{\pi(G)} = \left(\sum_{n=0}^{T}(T+1-n)\rho^n\right)^{-1}. \tag{5.1.52}$$

This immediately implies that $\pi((0,1))/\pi(G)$ is monotonically decreasing with ρ as required. In view of (5.1.34), this is sufficient to show that the difference $g_T(\lambda) - g_{T+1}(\lambda)$ is monotonically increasing in the interval $\lambda \in (\lambda_i, \lambda_{i+1}]$, for any threshold T satisfying $T_i^* \leq T < \tilde{T}$. Hence, due to (5.1.32), the same must be true for $L(\lambda) = g^*(\lambda) - \tilde{g}(\lambda)$. Since this argument can be repeated for any fixed $i \in \{0,1,...,m\}$, this completes the proof of the theorem. □

Figure 5.3 illustrates the result of Theorem 5.1.10 by showing the relationship between $L(\lambda)$ and λ in the case of an $M/M/1$ system with parameters $\mu = 5$, $\alpha = 20$ and $\beta = 10$.

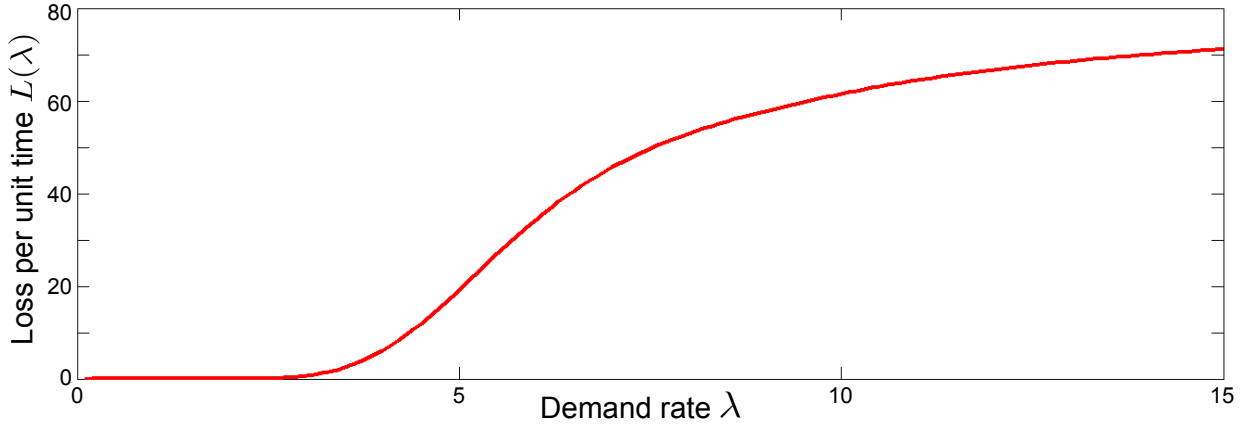


Figure 5.3: The relationship between $L(\lambda)$ and λ in an $M/M/1$ system with $\mu = 5$, $\alpha = 20$, $\beta = 10$.

The 'loss per unit time' $L(\lambda)$ referred to in Theorem 5.1.10 is only one means of quantifying the sub-optimality of the selfish policy $\tilde{\theta}$. An alternative would be to consider the ratio $g^*(\lambda)/\tilde{g}(\lambda)$ of

the average rewards under the socially optimal and selfishly optimal policies. In problems involving routing games, where the objective is to minimise long-run average costs, there exists a well-studied measure known as the *Price of Anarchy (PoA)* which quantifies the inefficiency of a system due to selfish decision-making; see, for example, Roughgarden [147]. The PoA is sometimes studied in problems involving routing of customers in queueing systems (see [54, 79, 101]) although the context is usually slightly different, as *static* routing policies (in which decisions are made in a randomised fashion, without any dependence on the state of the system) are usually considered. In the context of the queueing systems considered in this thesis, let $\mathrm{PoA}(\lambda)$ be defined by:

$$\mathrm{PoA}(\lambda) = \frac{g^*(\lambda)}{\tilde{g}(\lambda)},$$

where $g^*(\lambda)$ and $\tilde{g}(\lambda)$ are as defined in Theorem 5.1.10. Thus, $\mathrm{PoA}(\lambda) \geq 1$ for all $\lambda > 0$ due to the definition of $g^*(\lambda)$, and large values of $\mathrm{PoA}(\lambda)$ indicate a dramatic difference between the performances of the selfish and socially optimal policies. In the present context, the term "Price of Anarchy" may be regarded as something of a misnomer, since $\mathrm{PoA}(\lambda)$ in fact quantifies the relative extra benefit (or reward) of following a socially optimal policy as opposed to a selfish policy; the term makes more sense in the context of cost minimisation problems, in which it is defined as the ratio of the selfish cost to the optimal cost and thereby represents the extra relative cost of following a selfish policy. Obviously, this is only a minor point and the PoA has essentially the same meaning regardless of whether one considers cost minimisation or reward maximisation.

Theorem 5.1.10 states that the loss per unit time $L(\lambda)$ is monotonically increasing with λ. Numerical experiments involving $M/M/1$ queues appear to indicate that $\mathrm{PoA}(\lambda)$ is also monotonically increasing with λ, but this appears to be much more difficult to prove than the corresponding result for $L(\lambda)$ and therefore no proof will be given here. Figure 5.4 shows the relationship between $\mathrm{PoA}(\lambda)$ and λ in a system with $\mu = 5$, $\alpha = 20$ and $\beta = 10$ (as used in Figure 5.3).

An interesting way of converting the reward maximisation problem considered in this section (which is solved by a socially optimal policy) into a *cost minimisation* problem is to re-interpret the parameter α so that it becomes a *cost for balking*, as opposed to a reward for service. This can be done by altering the problem formulation so that the system incurs a penalty $\alpha > 0$ for every customer who balks, and customers do not earn rewards for service but are still charged holding

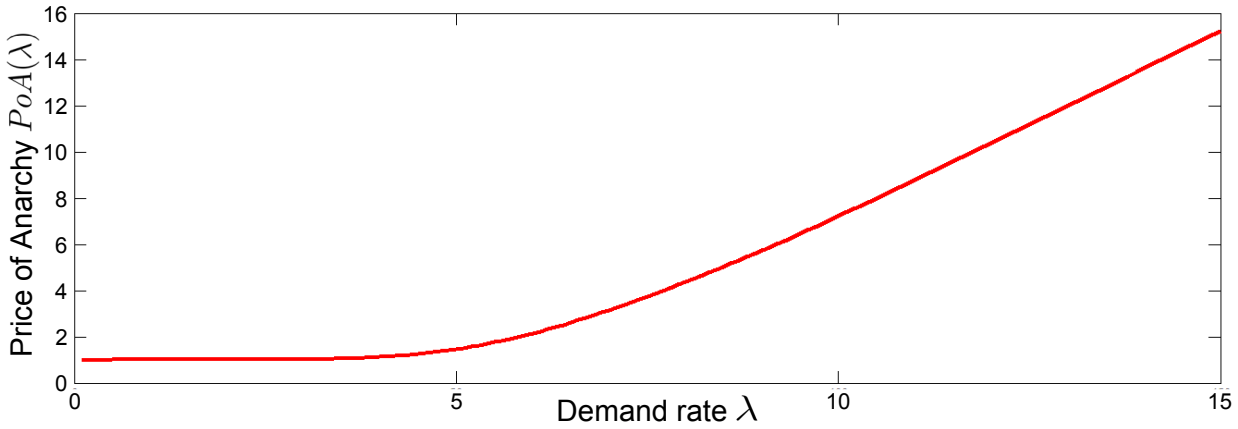


Figure 5.4: The relationship between PoA(λ) and λ in an $M/M/1$ system with $\mu = 5$, $\alpha = 20$, $\beta = 10$.

costs in the same way as before. Effectively, this simply means that the value α is deducted from each customer's individual expected net reward (whether or not they balk), and it follows that selfishly and socially optimal policies are unaffected by the change. The problem then becomes a reward maximisation problem involving non-positive rewards, which can equivalently be treated as a *cost minimisation* problem in which all costs are non-negative. The interpretation of α as a cost for balking rather than a reward for service not only ensures comparability with routing game problems (in which costs are generally assumed non-negative), but also makes sense in healthcare applications and other public service settings; refer to [101] for further discussion.

It is easy to see that, given any set of system parameters, altering the problem formulation as described above will not change the value of $|g^*(\lambda) - \tilde{g}(\lambda)|$. As such, the loss per unit time $L(\lambda)$ is unaffected (except that it becomes a cost *increase* per unit time, rather than a loss in reward). However, PoA(λ) *is* affected, due to the contribution made to the aggregate cost made by customers who balk. Indeed, large values of λ will generally result in values of PoA(λ) close to 1, since (under either of the policies $\tilde{\theta}$ and θ^*) the majority of customers will tend to balk, and therefore balking costs will dominate over customers' holding costs. As a result, PoA(λ) is no longer monotonically increasing with λ in the new problem formulation; instead, it appears to attain a peak value at some finite value of λ, and then decrease towards 1 as λ becomes large. Figure 5.5 illustrates the shape of the PoA(λ) curve when the new problem formulation (with balking costs) is used. It may be observed that the shape is similar to those of the PoA curves found in [101].

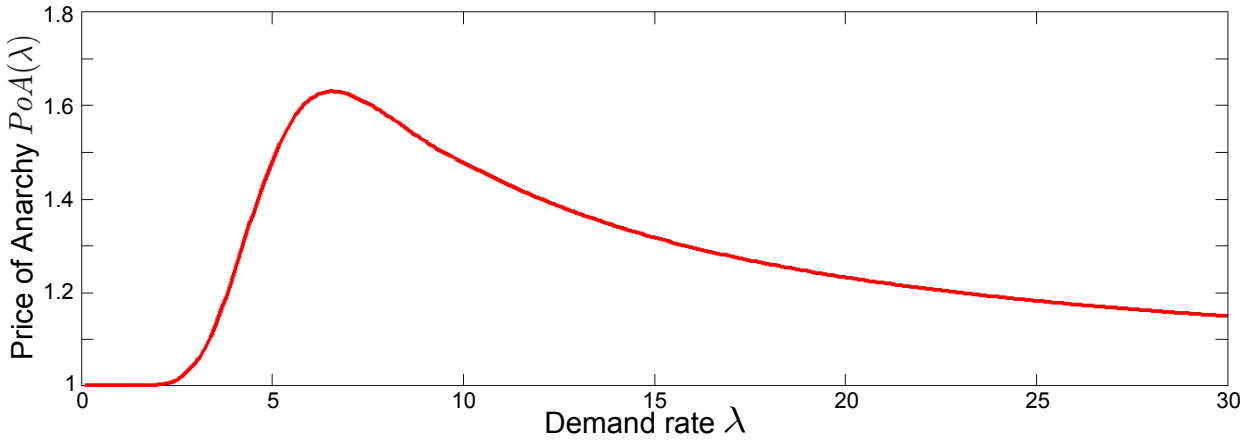


Figure 5.5: The relationship between PoA(λ) and λ in an $M/M/1$ system with $\mu = 5$, $\alpha = 20$, $\beta = 10$, after re-formulation of the problem so that α is a cost for balking as opposed to a reward for service.

The next section will consider the possible extension of the structural properties proved in this section to a problem involving two facilities with only one service channel each.

5.2 Two facilities, one server each

In Section 5.1, various properties of socially optimal policies for systems consisting of a single, multiple-server queue were established by appealing to structural properties of the relative value functions $h(x)$ and $\hat{h}(x)$. In general, inductive proofs based on value iteration are quite thematic in the literature; unfortunately, however, such proofs may not necessarily be achievable in systems of greater complexity. Indeed, it appears to be the case that optimal policies for larger-scale systems may exhibit quite counter-intuitive characteristics; this is shown by the examples in Appendix A.8. This section, like the previous one, considers a special case of the general N-facility, multiple-server queueing system introduced in Section 3.1. In this case, it is assumed that there are two facilities ($N = 2$), both of which have only a single service channel available ($c_1 = c_2 = 1$).

Before proceeding, it will be useful to define certain properties of real-valued functions defined on a *lattice*, i.e. a partially-ordered set which contains both the least upper bound and the greatest lower bound of any two constituent elements. As discussed in earlier sections, the N-dimensional state space S defined in (3.5.1) is a partially ordered set according to the ordering relation where, for any two states $\mathbf{x}, \mathbf{y} \in S$, $\mathbf{x} \leq \mathbf{y}$ if and only if $x_i \leq y_i$ for all components $i \in \{1, 2, ..., N\}$. The

following terms are commonly used in lattice theory (see, for example, [183] p. 13):

- The least upper bound of two elements $\mathbf{x}, \mathbf{y} \in S$ is called their *join* and denoted $\mathbf{x} \vee \mathbf{y}$.
- The greatest lower bound of two elements $\mathbf{x}, \mathbf{y} \in S$ is called their *meet* and denoted $\mathbf{x} \wedge \mathbf{y}$.

For example, given the partial ordering of states described above, one may write $(2, 5) \vee (3, 4) = (3, 5)$ or (in a higher-dimensional state space) $(4, 1, 7, 5, 3) \wedge (6, 8, 3, 9, 2) = (4, 1, 3, 5, 2)$. Using this vocabulary, one may define the property of *submodularity* as follows:

Definition 5.2.1. ***(Submodularity)***

Let S be a lattice and let $f : S \to \mathbb{R}$ be an arbitrary real-valued function. Then f is said to be submodular *on S if, for all pairs of elements $\mathbf{x}, \mathbf{y} \in S$:*

$$f(\mathbf{x} \vee \mathbf{y}) + f(\mathbf{x} \wedge \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y}). \tag{5.2.1}$$

If $(-f)$ is submodular on S, then f is said to be supermodular *on S.*

The following lemma implies that an inductive method can be used to check a function for submodularity in the case where the state space S is two-dimensional.

Lemma 5.2.2. *A function $f : S \to \mathbb{R}$ is submodular on the two-dimensional state space $S = \{(x_1, x_2) : x_1, x_2 \in \mathbb{N}_0\}$ if and only if, for all elements $\mathbf{x} \in S$ and $i, j \in \{1, 2\}$ with $i \neq j$:*

$$f((\mathbf{x}^{i+})^{j+}) + f(\mathbf{x}) \leq f(\mathbf{x}^{i+}) + f(\mathbf{x}^{j+}). \tag{5.2.2}$$

Proof. If f is submodular on S, then (5.2.2) follows immediately from Definition 5.2.1, since $(\mathbf{x}^{i+})^{j+}$ and $\mathbf{x}$ are (respectively) the join and the meet of the states $\mathbf{x}^{i+}$ and $\mathbf{x}^{j+}$. It must also be shown that (5.2.2) implies (5.2.1). Note that if $\mathbf{x} \leq \mathbf{y}$ (or $\mathbf{x} \geq \mathbf{y}$) with respect to the partial ordering, then (5.2.1) holds trivially, since the terms on the left-hand side are identical to those on the right-hand side. Hence, in order to eliminate trivial cases, one may assume that $\mathbf{x}$ and $\mathbf{y}$ are *incomparable*; that is, $x_i > y_i$ and $x_j < y_j$ for some permutation $(i, j) \in \{1, 2\}^2$ with $i \neq j$. For convenience, let $\mathbf{z} = \mathbf{x} \wedge \mathbf{y} \in S$. If $\mathbf{x}$ and $\mathbf{y}$ are incomparable, then (without loss of generality) one may write $\mathbf{x} = (z_1 + m, z_2)$ and $\mathbf{y} = (z_1, z_2 + n)$ for some strictly positive integers m and n. Hence, it suffices to show that the following inequality holds for arbitrary $\mathbf{z} \in S$ and $m, n \in \mathbb{N}$:

$$f(z_1 + m, z_2 + n) + f(z_1, z_2) \leq f(z_1 + m, z_2) + f(z_1, z_2 + n). \tag{5.2.3}$$

Due to (5.2.2), this inequality holds when $m = n = 1$. As an inductive assumption, let $(k_1, k_2) \in \mathbb{N}^2$ be an arbitrary pair of positive integers and assume that (5.2.3) holds for any state $\mathbf{z} \in S$ and positive integers m, n satisfying $m \leq k_1$ and $n \leq k_2$. This implies:

$$f(z_1 + k_1, z_2 + k_2) + f(z_1, z_2) \leq f(z_1 + k_1, z_2) + f(z_1, z_2 + k_2),$$
$$f(z_1 + k_1 + 1, z_2 + k_2) + f(z_1 + k_1, z_2) \leq f(z_1 + k_1 + 1, z_2) + f(z_1 + k_1, z_2 + k_2).$$

Combining these two inequalities, one obtains:

$$f(z_1 + k_1 + 1, z_2 + k_2) + f(z_1, z_2) \leq f(z_1 + k_1 + 1, z_2) + f(z_1, z_2 + k_2),$$

which states that (5.2.3) holds when $m = k_1 + 1$ and $n = k_2$. Analogously, one can show that it also holds when $m = k_1$ and $n = k_2 + 1$. This completes the inductive proof that (5.2.3) holds for all $\mathbf{z} \in S$ and $m, n \in \mathbb{N}$, which confirms that (5.2.2) implies (5.2.1). □

Example 5.2.3. ***(Submodularity on*** $\mathbb{N}^2$***)***

Let $f : \mathbb{N}^2 \to \mathbb{R}$ be defined by $f(x_1, x_2) = x_1 x_2$ for pairs $(x_1, x_2) \in \mathbb{N}^2$. In order to show that f is *supermodular* on $\mathbb{N}^2$, it suffices to show that for arbitrary $(x_1, x_2) \in \mathbb{N}^2$ and $m, n \in \mathbb{N}$:

$$f(x_1 + m, x_2 + n) + f(x_1, x_2) \geq f(x_1 + m, x_2) + f(x_1, x_2 + n). \tag{5.2.4}$$

Indeed, given that $f(x_1, x_2) = x_1 x_2$, the inequality (5.2.4) is equivalent to:

$$2x_1 x_2 + nx_1 + mx_2 + mn \geq 2x_1 x_2 + nx_1 + mx_2,$$

which holds trivially for all $m, n \in \mathbb{N}$. Since f is supermodular on $\mathbb{N}^2$, it follows that the function $g : \mathbb{N}^2 \to \mathbb{R}$ defined by $g(x_1, x_2) := -x_1 x_2$ is *submodular* on $\mathbb{N}^2$. ⊠

Two further properties of real-valued functions defined on the N-dimensional integer lattice $S = \{(x_1, x_2, ..., x_N) : x_1, x_2, ..., x_N \in \mathbb{N}_0\}$ are defined below.

Definition 5.2.4. ***(Concavity)***

Let $f : S \to \mathbb{R}$ be an arbitrary real-valued function, with S as defined in (3.5.1). Then f is said to be concave *on S if, for all $\mathbf{x} \in S$ and $j \in \{1, 2, ..., N\}$:*

$$f((\mathbf{x}^{j+})^{j+}) - f(\mathbf{x}^{j+}) \leq f(\mathbf{x}^{j+}) - f(\mathbf{x}). \tag{5.2.5}$$

If $(-f)$ is concave on S, then f is said to be convex *on S.*

Definition 5.2.5. ***(Diagonal submissiveness)***

Let $f : S \to \mathbb{R}$ be an arbitrary real-valued function, with S as defined in (3.5.1). Then f is said to be diagonally submissive *on S if, for all $\mathbf{x} \in S$ and $i, j \in \{1, 2, ..., N\}$ with $i \neq j$:*

$$f((\mathbf{x}^{j+})^{j+}) - f((\mathbf{x}^{i+})^{j+}) \leq f(\mathbf{x}^{j+}) - f(\mathbf{x}^{i+}). \tag{5.2.6}$$

If $(-f)$ is diagonally submissive on S, then f is said to be diagonally dominant *on S.*

Examples are given below of functions which possess the properties defined above.

Example 5.2.6. ***(Concavity on $\mathbb{N}^2$)***

Let $f : \mathbb{N}^2 \to \mathbb{R}$ be defined by $f(x_1, x_2) = (x_1 + x_2)^2$ for pairs $(x_1, x_2) \in \mathbb{N}^2$. In order to show that f is *convex* on $\mathbb{N}^2$, it suffices to show that for an arbitrary pair $(x_1, x_2) \in \mathbb{N}^2$:

$$f(x_1 + 2, x_2) - f(x_1 + 1, x_2) \geq f(x_1 + 1, x_2) - f(x_1, x_2). \tag{5.2.7}$$

Indeed, given that $f(x_1, x_2) = (x_1 + x_2)^2$, the inequality (5.2.7) is equivalent to:

$$2x + 2y + 3 \geq 2x + 2y + 1$$

which holds trivially for all $(x_1, x_2) \in \mathbb{N}^2$. Since f is convex on $\mathbb{N}^2$, it then follows that the function $g : \mathbb{N}^2 \to \mathbb{R}$ defined by $g(x_1, x_2) := -(x_1 + x_2)^2$ is *concave* on $\mathbb{N}^2$. ⊠

Example 5.2.7. ***(Diagonal submissiveness on $\mathbb{N}^2$)***

Let $f : \mathbb{N}^2 \to \mathbb{R}$ be defined as in Example 5.2.3; that is, $f(x_1, x_2) = x_1 x_2$ for $(x_1, x_2) \in \mathbb{N}^2$. In order to show that f is diagonally submissive on $\mathbb{N}^2$ it suffices to show, for $(x_1, x_2) \in \mathbb{N}^2$:

$$f(x_1 + 2, x_2) - f(x_1 + 1, x_2 + 1) \leq f(x_1 + 1, x_2) - f(x_1, x_2 + 1). \tag{5.2.8}$$

Indeed, given that $f(x_1, x_2) = x_1 x_2$, the inequality (5.2.8) is equivalent to:

$$x_2 - x_1 - 1 \leq x_2 - x_1$$

which holds trivially for all $(x_1, x_2) \in \mathbb{N}^2$. So, f is diagonally submissive on $\mathbb{N}^2$. ⊠

It will be convenient to introduce some operator notation, similar to the notation $D(x, f)$ used in Section 5.1 to represent the first-order difference $f(x+1) - f(x)$. For an arbitrary function f and state $\mathbf{x} \in S$, let the first-order difference with respect to component j be defined as:

$$D_j(\mathbf{x}, f) := f(\mathbf{x}^{j+}) - f(\mathbf{x}). \tag{5.2.9}$$

In addition, let the *second-order* differences $D_{jj}(\mathbf{x}, f)$ and $D_{ij}(\mathbf{x}, f)$ be defined as:

$$D_{jj}(\mathbf{x}, f) := D_j(\mathbf{x}, D_j(\mathbf{x}, f)) = f((\mathbf{x}^{j+})^{j+}) - f(\mathbf{x}^{j+}) - f(\mathbf{x}^{j+}) + f(\mathbf{x}), \tag{5.2.10}$$

$$D_{ij}(\mathbf{x}, f) := D_i(\mathbf{x}, D_j(\mathbf{x}, f)) = f((\mathbf{x}^{i+})^{j+}) - f(\mathbf{x}^{i+}) - f(\mathbf{x}^{j+}) + f(\mathbf{x}). \tag{5.2.11}$$

From (5.2.10) and (5.2.11) one obtains the following characterisations of submodularity, concavity and diagonal submissiveness as defined in Definitions (5.2.1), (5.2.4) and (5.2.5):

- A real-valued function $f : S \to \mathbb{R}$ is submodular if and only if $D_{ij}(\mathbf{x}, f) \leq 0$ for all states $\mathbf{x} \in S$ and components $i, j \in \{1, 2, ..., N\}$ with $i \neq j$.
- A real-valued function $f : S \to \mathbb{R}$ is concave if and only if $D_{jj}(\mathbf{x}, f) \leq 0$ for all states $\mathbf{x} \in S$ and components $j \in \{1, 2, ..., N\}$.
- A real-valued function $f : S \to \mathbb{R}$ is diagonally submissive if and only if $D_{jj}(\mathbf{x}, f) \leq D_{ij}(\mathbf{x}, f)$ for all states $\mathbf{x} \in S$ and components $i, j \in \{1, 2, ..., N\}$ with $i \neq j$.

Thus, it is immediate that if f is both submodular and diagonally submissive on S, then it must also be concave on S. It is easy to show that the 'real-time' reward formulation (3.5.4) possesses all three of these structural properties. This is stated below as a lemma.

Lemma 5.2.8. *The reward function r defined in (3.5.4) is submodular, concave and diagonally submissive on the N-dimensional state space $S = \{(x_1, x_2, ..., x_N) : x_1, x_2, ..., x_N \in \mathbb{N}_0\}$.*

Proof. Using (3.5.4) one may write, for any $\mathbf{x} \in S$ and $i, j \in \{1, 2, ..., N\}$ with $i \neq j$:

$$D_{ij}(\mathbf{x}, r) = 0,$$

$$D_{jj}(\mathbf{x}, r) = D_{jj}(\mathbf{x}, r) - D_{ij}(\mathbf{x}, r) = \begin{cases} -\alpha_j \mu_j, & \text{if } x_j = c_j - 1, \\ 0, & \text{otherwise.} \end{cases}$$

So $D_{ij}(\mathbf{x}, r)$, $D_{jj}(\mathbf{x}, r)$ and $D_{jj}(\mathbf{x}, r) - D_{ij}(\mathbf{x}, r)$ are all non-positive as required. □

Another result which will be required later is that, when the real-time reward function r is used, the first-order differences $D_j(\mathbf{x}, h)$ are uniformly bounded above. Here, h denotes the relative value function which, together with the optimal average reward g^*, satisfies the optimality equations (4.2.1). As in the previous section, the assumption will be made that $h(\mathbf{0}) = 0$ in order to determine the $h(\mathbf{x})$ values uniquely. The result of the next lemma actually applies to a system with an arbitrary number of facilities, each with multiple servers allowed (i.e. without any restriction on the parameters N or $\{c_i\}_{i=1}^N$) and therefore the proof will be given in the general context of the MDP Υ described in Section 3.5, so that the result can be used in later sections.

Recall that $\tilde{S}$ denotes the selfishly optimal state space, defined in (4.1.3).

Lemma 5.2.9. *Assume that the reward function $r(\mathbf{x})$ defined in (3.5.4) is used. Then, for all $\mathbf{x} \in \tilde{S}$ and $j \in \{1, 2, ..., N\}$ such that $\mathbf{x}^{j+} \in \tilde{S}$, the relative value function h satisfies:*

$$D_j(\mathbf{x}, h) \leq \frac{\alpha_j \mu_j - \beta_j}{\mu_j \Delta}. \tag{5.2.12}$$

Proof. Like many of the earlier proofs in this chapter, the result is proved using induction on the iterates $h_n(\mathbf{x})$ obtained by relative value iteration. See Appendix A.5, page 440.

The definitions provided earlier for submodularity, concavity and diagonal submissiveness may be applied to value functions (or any other functions of interest) related to queueing systems with an arbitrary number of facilities $N \in \mathbb{N}$. However, throughout the remainder of this section, attention will be restricted to systems with only two facilities, each with a single server. The abbreviation $2DSS$ (meaning '2 Dimensions, Single Server') will henceforth be used to describe any queueing system with these properties; i.e. a '$2DSS$ system' is a queueing system which satisfies the assumptions of Section 3.1 and also has $N = 2$ and $c_1 = c_2 = 1$.

The next lemma establishes structural properties of the relative value function h in a $2DSS$ system, assuming (once again) that the real-time reward formulation is used.

Lemma 5.2.10. *Assume that the system is $2DSS$ and the reward function $r(\mathbf{x})$ defined in (3.5.4) is used. Then the relative value function h is submodular, concave and diagonally submissive on the two-dimensional selfish state space $\tilde{S} = \{(x_1, x_2) : x_1, x_2 \in \mathbb{N}_0, x_1 \leq \tilde{B}_1, x_2 \leq \tilde{B}_2\}$.*

Proof. Again, the proof relies upon induction on the finite-stage iterates $h_n(\mathbf{x})$. The three structural properties (submodularity, concavity and diagonal submissiveness) actually depend upon each other to some extent, and therefore the inductive hypothesis must be that all three properties hold at an arbitrary stage $n = k$. For details of the proof, refer to Appendix A.5, page 441.

Naturally, there is a purpose for showing that the function h possesses the three structural properties discussed in Lemma 5.2.10. Indeed, the next result exploits these properties to show that, in a $2DSS$ system, the optimal policy θ^* found by relative value iteration on the finite state space $\tilde{S}$ is *monotonic* with respect to the actions prescribed at the various states $\mathbf{x} \in \tilde{S}$.

Theorem 5.2.11. ***(Optimality of monotone policies)***

Assume that the system is $2DSS$ and let θ^* be the optimal policy found by relative value iteration on the selfish state space $\tilde{S}$. Then, for all $\mathbf{x} \in \tilde{S}$ and $j \in \{1, 2\}$ such that $\mathbf{x}^{j+} \in \tilde{S}$:

- *If balking is chosen by θ^* at $\mathbf{x}$, then it is also chosen at $\mathbf{x}^{j+}$. That is:*

$$\theta^*(\mathbf{x}) = 0 \Rightarrow \theta^*(\mathbf{x}^{j+}) = 0.$$

- *If joining facility j is chosen by θ^* at $\mathbf{x}^{j+}$, then it is also chosen at $\mathbf{x}$. That is:*

$$\theta^*(\mathbf{x}^{j+}) = j \Rightarrow \theta^*(\mathbf{x}) = j.$$

Proof. Assume that the real-time reward function as defined in (3.5.4) is used. Let $\mathbf{x} \in \tilde{S}$ be an arbitrary state and let $i, j \in \{1, 2\}$ be distinct ($i \neq j$). Then, using the structural properties of the relative value function h proved by Lemma 5.2.10, one may write:

$$h(\mathbf{x}) > h(\mathbf{x}^{j+}) \;\; \Rightarrow \;\; h(\mathbf{x}^{i+}) > h((\mathbf{x}^{i+})^{j+}), \tag{5.2.13}$$

$$h(\mathbf{x}) > h(\mathbf{x}^{j+}) \;\; \Rightarrow \;\; h(\mathbf{x}^{j+}) > h((\mathbf{x}^{j+})^{j+}), \tag{5.2.14}$$

$$h((\mathbf{x}^{j+})^{j+}) \geq h((\mathbf{x}^{i+})^{j+}) \;\; \Rightarrow \;\; h(\mathbf{x}^{j+}) \geq h(\mathbf{x}^{i+}), \tag{5.2.15}$$

assuming, of course, that all of the states referred to in (5.2.13)-(5.2.15) are elements of $\tilde{S}$. Furthermore, the actions $\theta^*(\mathbf{x})$ are related to the values $h(\mathbf{x})$ as follows:

$$\theta^*(\mathbf{x}) = \begin{cases} 0, & \text{if } h(\mathbf{x}^{j+}) < h(\mathbf{x}) \text{ and } h(\mathbf{x}^{i+}) < h(\mathbf{x}), \\ j, & \text{if } h(\mathbf{x}^{j+}) \geq h(\mathbf{x}) \text{ and } h(\mathbf{x}^{j+}) \geq h(\mathbf{x}^{i+}), \\ i, & \text{otherwise.} \end{cases} \tag{5.2.16}$$

If $h(\mathbf{x}^{j+}) = h(\mathbf{x}^{i+})$, it is assumed that the stationary policy θ^* uses an arbitrary tie-breaking rule to choose between facilities j and i at state $\mathbf{x}$ (not a randomised rule, since this would make the policy non-stationary); accordingly, (5.2.16) assumes without loss of generality that j is preferred over i in such cases. Using (5.2.13)-(5.2.15) and (5.2.16), the results follow:

$$\begin{aligned}
\theta^*(\mathbf{x}) = 0 &\Rightarrow h(\mathbf{x}) > \max\big(h(\mathbf{x}^{i+}),\ h(\mathbf{x}^{j+})\big) \\
&\Rightarrow h(\mathbf{x}^{j+}) > \max\big(h((\mathbf{x}^{i+})^{j+}),\ h((\mathbf{x}^{j+})^{j+})\big) \Rightarrow \theta^*(\mathbf{x}^{j+}) = 0, \\
\theta^*(\mathbf{x}^{j+}) = j &\Rightarrow h((\mathbf{x}^{j+})^{j+}) \geq \max\big(h(\mathbf{x}^{j+}),\ h((\mathbf{x}^{i+})^{j+})\big) \\
&\Rightarrow h(\mathbf{x}^{j+}) \geq \max\big(h(\mathbf{x}),\ h(\mathbf{x}^{i+})\big) \Rightarrow \theta^*(\mathbf{x}) = j, \\
\theta^*(\mathbf{x}^{i+}) = i &\Rightarrow h((\mathbf{x}^{i+})^{i+}) \geq h(\mathbf{x}^{i+}) \text{ and } h((\mathbf{x}^{i+})^{i+}) > h((\mathbf{x}^{i+})^{j+}) \\
&\Rightarrow h(\mathbf{x}^{i+}) \geq h(\mathbf{x}) \text{ and } h(\mathbf{x}^{i+}) > h(\mathbf{x}^{j+}) \Rightarrow \theta^*(\mathbf{x}) = i.
\end{aligned}$$

So θ^* possesses the monotonicity properties stated by the theorem. □

So far, all of the results in this section have assumed that the real-time reward formulation (3.5.4) is used. It is worth noting that if the alternative (anticipatory) reward formulation (3.5.15) is adopted instead, it is possible to prove structural properties of the relative value function which are similar (but not identical) to those in (A.5.42)-(A.5.44). Indeed, let $\hat{h}$ denote the relative value function under the formulation (3.5.15). Then, using similar arguments to those in the proof of Lemma 5.2.10, one can show that for states $\mathbf{x} \in \tilde{S}$ and $i, j \in \{1, 2\}$ with $i \neq j$:

$$\hat{h}((\mathbf{x}^{i+})^{j+}) - \hat{h}(\mathbf{x}^{i+}) \leq \hat{h}(\mathbf{x}^{j+}) - \hat{h}(\mathbf{x}), \tag{5.2.17}$$

$$\hat{r}(\mathbf{x}^{j+}, j) + \lambda\Delta\big(\hat{h}((\mathbf{x}^{j+})^{j+}) - \hat{h}(\mathbf{x}^{j+})\big) \leq \hat{r}(\mathbf{x}, j) + \lambda\Delta\big(\hat{h}(\mathbf{x}^{j+}) - \hat{h}(\mathbf{x})\big), \tag{5.2.18}$$

$$\hat{r}(\mathbf{x}^{j+}, j) + \lambda\Delta\big(\hat{h}((\mathbf{x}^{j+})^{j+}) - \hat{h}((\mathbf{x}^{i+})^{j+})\big) \leq \hat{r}(\mathbf{x}, j) + \lambda\Delta\big(\hat{h}(\mathbf{x}^{j+}) - \hat{h}(\mathbf{x}^{i+})\big), \tag{5.2.19}$$

where it is again assumed that all of the states referred to in (5.2.17)-(5.2.19) are elements of $\tilde{S}$. Thus, $\hat{h}$ possesses the submodularity property, but (in general) it does not possess the concavity or diagonal submissiveness properties. However, (5.2.17)-(5.2.19) are sufficient to establish the monotonicity properties of θ^* stated by Theorem 5.2.11. That is, the optimality of monotone policies in a $2DSS$ system can be established using either of the two reward formulations.

Recall that, by Theorem 4.2.4, the set S_{θ^*} of states which are positive recurrent under θ^* is a subset of the selfish state space $\tilde{S}$. Let $\tilde{B}_1$ and $\tilde{B}_2$ be the boundaries of $\tilde{S}$, defined in (4.1.2).

Lemma 5.2.10 implies that there exists a set of values $\{C^*(0), C^*(1), ..., C^*(\tilde{B}_1)\}$ such that, for $x_1 \in \{0, 1, ..., \tilde{B}_1\}$, $h(\mathbf{x}^{1+}) \geq h(\mathbf{x}^{2+})$ if and only if $x_2 \geq C^*(x_1)$. Similarly, there exists a set of values $\{B^*(0), B^*(1), ..., B^*(\tilde{B}_1)\}$ such that, for $x_1 \in \{0, 1, ..., \tilde{B}_1\}$, $h(\mathbf{x}) > \max\left(h(\mathbf{x}^{1+}),\ h(\mathbf{x}^{2+})\right)$ if and only if $x_2 \geq B^*(x_1)$. To be precise, for $x_1 \in \{0, 1, ..., \tilde{B}_1\}$ one may define:

$$C^*(x_1) := \min\left\{x_2 \in \{0, 1, ..., \tilde{B}_2\} : D_1(\mathbf{x}, h) \geq D_2(\mathbf{x}, h)\right\}, \tag{5.2.20}$$

$$B^*(x_1) := \min\left\{x_2 \in \{0, 1, ..., \tilde{B}_2\} : D_1(\mathbf{x}, h) < 0 \text{ and } D_2(\mathbf{x}, h) < 0\right\}. \tag{5.2.21}$$

At this stage, there is no theoretical guarantee that $\{x_2 \in \{0, 1, ..., \tilde{B}_2\} : D_1(\mathbf{x}, h) \geq D_2(\mathbf{x}, h)\}$ and $\{x_2 \in \{0, 1, ..., \tilde{B}_2\} : D_1(\mathbf{x}, h) < 0 \text{ and } D_2(\mathbf{x}, h) < 0\}$ are non-empty sets, so a convention will be adopted whereby $C^*(x_1) = +\infty$ when the former set is empty, and $B^*(x_1) = +\infty$ when the latter set is empty. In words, $C^*(x_1)$ is the smallest number of customers at facility 2 which would cause the socially optimal policy θ^* to 'prefer' joining facility 1 over facility 2 when there are x_1 customers at facility 1. Similarly, $B^*(x_1)$ is the smallest number of customers at facility 2 which would cause the policy θ^* to choose to balk when there are x_1 customers at facility 1. The set of values $\{C^*(0), C^*(1), ..., C^*(\tilde{B}_1)\}$ may be referred to as a *switching curve*, since each value $C^*(x_1)$ is the critical value of x_2 for which the policy θ^* 'switches' from choosing facility 2 to facility 1; similarly, the set $\{B^*(0), B^*(1), ..., B^*(\tilde{B}_1)\}$ may be referred to as a *balking curve*. The use of switching and balking curves to characterise optimal policies is a theme which has received considerable attention in the MDP literature; see, for example, [56, 73, 74, 105, 106].

The next result establishes monotonicity properties of the switching and balking curves.

Lemma 5.2.12. *For $x_1 \in \{0, 1, ..., \tilde{B}_1\}$, let $C^*(x_1)$ and $B^*(x_1)$ be as defined in (5.2.20) and (5.2.21) respectively. Then $C^*(x_1)$ and $B^*(x_1)$ have the following properties:*

- *$C^*(x_1)$ is monotonically increasing with x_1;*
- *$B^*(x_1)$ is monotonically decreasing with x_1.*

Proof. The monotonicity of $C^*(x_1)$ will be shown first. Consider an arbitrary $x_1 \in \{0, 1, ..., \tilde{B}_1 - 1\}$. By definition, $C^*(x_1)$ is the smallest value of x_2 such that $h((x_1 + 1, x_2)) \geq h((x_1, x_2 + 1))$. Hence, for any non-negative integer $k < C^*(x_1)$, it is the case that $h((x_1 + 1, k)) < h((x_1, k + 1))$. By the diagonal submissiveness property of h proved in Lemma 5.2.10, $h((x_1 + 2, k)) - h((x_1 + 1, k +$

$1)) \leq h((x_1+1,k)) - h((x_1,k+1))$ (equivalently, $D_{11}((x_1,k),h) \leq D_{12}((x_1,k),h)$) and hence $h((x_1+2,k)) < h((x_1+1,k+1))$, which implies that the value of $C^*(x_1+1)$ must be strictly greater than k. Hence, $C^*(x_1)$ is monotonically increasing with x_1 as required.

Meanwhile, by definition $B^*(x_1)$ is the smallest value of x_2 such that $h((x_1,x_2))$ is strictly greater than both $h((x_1+1,x_2))$ and $h((x_1,x_2+1))$. Let $x_1 \in \{1,2,...,\tilde{B}_1\}$ be arbitrary. Then, for any non-negative integer $k < B^*(x_1)$, it is the case that either $h((x_1,k+1)) \geq h((x_1,k))$ or $h((x_1+1,k)) \geq h((x_1,k))$. If $h((x_1,k+1)) \geq h((x_1,k))$, then due to the submodularity property of h proved in Lemma 5.2.10, $h((x_1-1,k+1)) \geq h((x_1-1,k))$. On the other hand, if $h((x_1+1,k)) \geq h((x_1,k))$ then the concavity of h implies $h((x_1,k)) \geq h((x_1-1,k))$. So in either case, $h((x_1-1,k))$ is bounded above by $\max\big(h((x_1-1,k+1)),\ h((x_1,k))\big)$, implying that $B^*(x_1-1)$ is strictly greater than k. It follows that $B^*(x_1)$ is monotonically decreasing with x_1 as required. $\square$

Naturally, the *selfishly* optimal policy $\tilde{\theta}$ in a $2DSS$ system may also be described using a switching curve, since for each integer $x_1 \in \{0,1,...,\tilde{B}_1\}$ there will be some value $x_2 \in \{0,1,...,\tilde{B}_2\}$ such that $w((x_1,x_2),1) \geq w((x_1,x_2),2)$ (with $w(\mathbf{x},a)$ as defined in (4.1.1)), and hence $w((x_1,k),1) \geq w((x_1,k),2)$ for all $k \geq x_2$. That is, for each $x_1 \in \{0,1,...,\tilde{B}_1\}$, one may define:

$$\tilde{C}(x_1) := \min\Big\{x_2 \in \{0,1,...,\tilde{B}_2\} : w((x_1,x_2),1) \geq w((x_1,x_2),2)\Big\}, \tag{5.2.22}$$

$$\tilde{B}(x_1) := \begin{cases} \infty, & \text{if } x_1 < \tilde{B}_1, \\ \tilde{B}_2, & \text{if } x_1 = \tilde{B}_1, \end{cases} \tag{5.2.23}$$

with the result that the selfish policy $\tilde{\theta}$ is characterised for $(x_1,x_2) \in \tilde{S}$ as follows:

$$\tilde{\theta}((x_1,x_2)) = \begin{cases} 1, & \text{if } x_2 < \tilde{B}(x_1) \text{ and } x_2 \geq \tilde{C}(x_1), \\ 2, & \text{if } x_2 < \tilde{B}(x_1) \text{ and } x_2 < \tilde{C}(x_1), \\ 0, & \text{if } x_2 \geq \tilde{B}(x_1). \end{cases}$$

The next example shows how the switching and balking curves (for both the optimal policy θ^* and the selfish policy $\tilde{\theta}$) may be represented graphically in a $2DSS$ system.

Example 5.2.13. ***(Switching and balking curves)***

Consider a system with two single-server facilities ($N=2$, $c_1=c_2=1$) and a demand rate $\lambda=4$.

The parameters for the two facilities are given as follows:

$$\mu_1 = 5, \qquad \beta_1 = 9, \qquad \alpha_1 = 14,$$
$$\mu_2 = 2, \qquad \beta_2 = 3, \qquad \alpha_2 = 11.$$

Table 5.1 shows the selfishly and socially optimal policies for the system, with the latter having been found using relative value iteration. The selfish state space for the system is $\tilde{S} = \{(x_1, x_2) \in \mathbb{N}_0^2 : x_1 \leq 7 \text{ and } x_2 \leq 7\}$; that is, $\tilde{B}_1 = 7$ and $\tilde{B}_2 = 7$. For each state $(x_1, x_2) \in \tilde{S}$, the selfishly optimal decision $\tilde{\theta}((x_1, x_2)) \in \{0, 1, 2\}$ is shown first, followed by the socially optimal decision $\theta^*((x_1, x_2))$ (in brackets). States at which the two policies differ are highlighted in bold.

	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$	$x_2 = 3$	$x_2 = 4$	$x_2 = 5$	$x_2 = 6$	$x_2 = 7$
$x_1 = 0$	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
$x_1 = 1$	**1 (2)**	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
$x_1 = 2$	2 (2)	**1 (2)**	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)
$x_1 = 3$	2 (2)	2 (2)	**1 (2)**	**1 (2)**	1 (1)	1 (1)	**1 (0)**	**1 (0)**
$x_1 = 4$	2 (2)	2 (2)	2 (2)	**1 (2)**	**1 (0)**	**1 (0)**	**1 (0)**	**1 (0)**
$x_1 = 5$	2 (2)	2 (2)	2 (2)	2 (2)	**2 (0)**	**1 (0)**	**1 (0)**	**1 (0)**
$x_1 = 6$	2 (2)	2 (2)	2 (2)	**2 (0)**	**2 (0)**	**2 (0)**	**1 (0)**	**1 (0)**
$x_1 = 7$	2 (2)	2 (2)	2 (2)	**2 (0)**	**2 (0)**	**2 (0)**	**2 (0)**	0 (0)

Table 5.1: Selfishly (socially) optimal decisions at states $\mathbf{x} = (x_1, x_2)$ with $x_1 \leq 7$, $x_2 \leq 7$.

Table 5.1 confirms that both of the policies $\tilde{\theta}$ and θ^* possess the monotonicity properties discussed earlier; for example, if joining facility 1 is chosen at some state $\mathbf{x} \in \tilde{S}$ with $x_1 \geq 1$, then this implies that facility 1 will be chosen at the state $\mathbf{x}^{1-}$. For each $x_1 \in \{0, 1, ..., 7\}$, the values $C^*(x_1)$, $B^*(x_1)$ and $\tilde{C}(x_1)$ as defined in (5.2.20)-(5.2.21) and (5.2.22) are plotted in Figure 5.6.

It may be observed from Figure 5.6 that the switching curve for the socially optimal policy θ^* is somewhat steeper than the switching curve for the selfish policy $\tilde{\theta}$, which indicates that facility 1 is preferred to facility 2 at a greater number of states under $\tilde{\theta}$ than under θ^*. ⊠

It is possible to use Lemma 5.2.10 to make further inferences about the structure of the socially optimal policy θ^* in a $2DSS$ system. First, another definition is required.

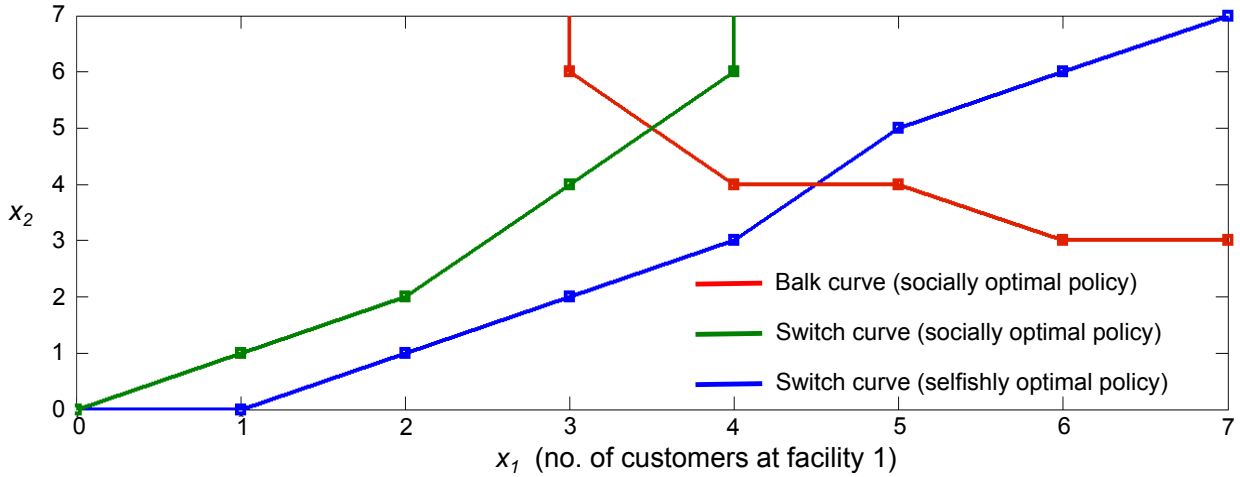


Figure 5.6: Switching curves for the policies $\tilde{\theta}$ and θ^*, and the balking curve for θ^*, in Example 5.2.13.

Definition 5.2.14. ***(Sink policy)***

Let θ be a stationary policy, and let S_θ denote the (possibly empty) set of states in S which are positive recurrent under θ. Then θ is said to be a sink policy *if and only if:*

1. *S_θ is non-empty and finite;*

2. *There exists* exactly one *state $\mathbf{z} \in S_\theta$ for which $\theta(\mathbf{z}) = 0$.*

Moreover, the unique state $\mathbf{z} \in S_\theta$ with $\theta(\mathbf{z}) = 0$ is referred to as the 'sink state'.

It is easy to see that the sink state of any sink policy θ must be located at the 'corner' of the recurrent region S_θ, since otherwise S_θ would not be finite; see Table 5.2.

Sink policies can be constructed for systems with an arbitrary number of facilities; indeed, given any system with N facilities, the *selfish* policy $\tilde{\theta}$ defined in Section 4.1 is a sink policy with sink state $(\tilde{B}_1, \tilde{B}_2, ..., \tilde{B}_N)$. It is reasonable to suppose that in many types of system, the socially optimal policy θ^* found by relative value iteration should be a sink policy, but this may not be easy to prove. The next result addresses the special case of a $2DSS$ system. The proof is similar to Ha's proof of the optimality of 'base stock policies' in make-to-stock production systems which follow similar dynamics to those of the $2DSS$ queueing system under consideration; see [73].

Policy θ_1			
	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	1	1	1
$x_1 = 1$	2	1	1
$x_1 = 2$	2	2	0

Policy θ_2			
	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	1	1	1
$x_1 = 1$	2	1	0
$x_1 = 2$	2	2	0

Table 5.2: An example of two different policies, θ_1 and θ_2, for a $2DSS$ system. θ_1 is a sink policy, with a sink state $(2, 2)$ located at the 'corner' of the positive recurrent set S_{θ_1}. θ_2 has the same set of positive recurrent states, but is not a sink policy because balking is chosen at both of the states $(1, 2)$ and $(2, 2)$.

Theorem 5.2.15. *Assume that the system is $2DSS$ and let θ^* be the socially optimal policy found by relative value iteration on the finite state space $\tilde{S}$. Then θ^* is a sink policy.*

Before proceeding with the proof, it is useful to observe that the policy θ_2 shown in Table 5.2 cannot possibly be an average reward optimal policy found by relative value iteration, assuming that the usual conventions for tie-breaking are followed (so that balking is never preferred to joining a facility in the case of a tie). Indeed, suppose (for a contradiction) that θ_2 *is* an optimal policy which always chooses actions attaining the maximum in the optimality equations (4.2.1). Then, given that facility 2 is preferred to balking at state $(2, 1)$, it must be the case that $h((2, 2)) \geq h((2, 1)$. Also, given that balking is preferred to facility 1 at state $(1, 2)$, it must be the case that $h((1, 2)) > h((2, 2))$. These two inequalities together imply $h((1, 2)) > h((2, 1))$. However, given that facility 1 is preferred to facility 2 at state $(1, 1)$, it is also the case that $h((2, 1)) \geq h((1, 2))$, which gives a contradiction. If the decision at state $(1, 1)$ was changed from 1 to 2 in order to ensure consistency with the previous inequalities, then the states $(2, 0)$, $(2, 1)$ and $(2, 2)$ would no longer be part of the positive recurrent region S_{θ_2} and the policy θ_2 would then become a sink policy with sink state $(1, 2)$. The proof of the theorem will essentially rely upon arguments similar to these (and also the monotonicity properties proved by Theorem 5.2.11) in order to show that θ^* must be a sink policy.

Proof. First, note that due to Theorem 4.2.4, the set S_{θ^*} of states which are positive recurrent under θ^* is contained within the selfish state space $\tilde{S}$ defined in (4.1.3), so S_{θ^*} is finite and non-empty. Let the thresholds T_1^* and T_2^* associated with θ^* be defined as follows:

$$T_1^* := \min\left\{x_1 \in \mathbb{N}_0 : \ B^*(x_1) < \infty \text{ and } C^*(x_1) \geq B^*(x_1)\right\},$$

$$T_2^* := B^*(T_1^*),$$

where $C^*(x_1)$ and $B^*(x_1)$ are as defined in (5.2.20) and (5.2.21) respectively. It must be the case that T_1^* and T_2^* are both finite; indeed, one can show that $T_1^* \leq \tilde{B}_1$ and $T_2^* \leq \tilde{B}_2$, where $\tilde{B}_1$ and $\tilde{B}_2$ are the boundaries of $\tilde{S}$ defined in (4.1.2). To see this, recall that it was established in the proof of Theorem 4.2.4 that the policy θ^* never chooses to join facility $j \in \{1, 2\}$ at any state $\mathbf{x} \in \tilde{S}$ with $x_j = \tilde{B}_j$. It follows that $\theta^*((\tilde{B}_1, \tilde{B}_2)) = 0$, and $\theta^*((\tilde{B}_1, k)) \neq 1$ for all $k \in \{0, 1, ..., \tilde{B}_2\}$, which in turn implies $B^*(\tilde{B}_1) \leq \tilde{B}_2$ and $C^*(\tilde{B}_1) \geq \tilde{B}_2$ and hence $C^*(\tilde{B}_1) \geq B^*(\tilde{B}_1)$ and $T_1^* \leq \tilde{B}_1$. Then $B^*(T_1^*) < \infty$ by definition of T_1^*, implying $B^*(T_1^*) \leq \tilde{B}_2$ by definition of $B^*(x_1)$.

The aim of this proof is to show that θ^* is a sink policy with sink state (T_1^*, T_2^*). Since T_2^* is the smallest value of x_2 such that θ^* chooses to balk at state (T_1^*, x_2), it follows that $\theta^*((T_1^*, T_2^*)) = 0$ and $\theta^*((T_1^*, T_2^* - 1)) \neq 0$. In addition, $C^*(T_1^*) \geq B^*(T_1 *) = T_2^*$ by definitions of T_1^* and T_2^*, which implies that at least T_2^* customers must be present at facility 2 in order for θ^* to prefer facility 1 to facility 2 when there are T_1^* customers present at facility 1. Hence, $\theta^*((T_1^*, T_2^* - 1)) \neq 1$. So the only remaining option at state $(T_1^*, T_2^* - 1)$ is facility 2, i.e. $\theta^*((T_1^*, T_2^* - 1)) = 2$.

The next stage of the proof is to show that $\theta^*((T_1^* - 1, T_2^*)) = 1$, and this is achieved by contradiction. First suppose $\theta^*((T_1^* - 1, T_2^*)) = 0$. Then $B^*(T_1^* - 1) \leq T_2^*$ by definition of B^*, i.e. the minimum value of x_2 required for θ^* to choose to balk at state $(T_1^* - 1, x_2)$ is at most T_2^*. In addition, $C^*(T_1^* - 1) < B^*(T_1^* - 1)$ by definition of T_1^*, so $C^*(T_1^* - 1) < T_2^*$. This states that, with $T_1^* - 1$ customers at facility 1, the number of customers required at facility 2 in order for θ^* to prefer facility 1 over facility 2 is smaller than T_2^*, which implies $\theta^*((T_1^* - 1, T_2^* - 1)) \neq 2$ (otherwise $C^*(T_1^* - 1)$ would be at least T_2^*). It has already been established that $\theta^*((T_1^*, T_2^* - 1)) = 2$, so it cannot be the case that $\theta^*((T_1^* - 1, T_2^* - 1)) = 0$, because this would contradict the first monotonicity property of θ^* stated by Theorem 5.2.11. The only remaining possibility at state $(T_1^* - 1, T_2^* - 1)$ (with $\theta^*((T_1^* - 1, T_2^*)) = 0$ assumed) is $\theta^*((T_1^* - 1, T_2^* - 1)) = 1$, but it may be shown that this is not possible either. Indeed, if $\theta^*((T_1^* - 1, T_2^* - 1)) = 1$ then one has:

$$\begin{aligned}
h((T_1^*, T_2^* - 1)) &\leq h((T_1^*, T_2^*)) && \text{(because } \theta^*((T_1^*, T_2^* - 1)) = 2) \\
&< h((T_1^* - 1, T_2^*)) && \text{(implied by } \theta^*((T_1^* - 1, T_2^*)) = 0) \\
&\leq h((T_1^*, T_2^* - 1)) && \text{(implied by } \theta^*((T_1^* - 1, T_2^* - 1)) = 1),
\end{aligned}$$

which states that $h((T_1^*, T_2^* - 1)) < h((T_1^*, T_2^* - 1))$ and thereby yields a contradiction. Thus, it

is impossible to have $\theta^*((T_1^*-1,T_2^*))=0$, since this does not allow any feasible choices of action at state (T_1^*-1,T_2^*-1). Next, suppose (again for a contradiction) that $\theta^*((T_1^*-1,T_2^*))=2$. As stated previously, $C^*(T_1^*-1)<B^*(T_1^*-1)$ by definition of T_1^*. Hence, there exists some integer $k>T_2^*$ such that $\theta^*((T_1^*-1,k))=1$ and $\theta^*((T_1^*-1,k-1))=2$. Given that $\theta^*((T_1^*,T_2^*))=0$ and $k>T_2^*$, it must be the case that $\theta^*((T_1^*,k-1))=0$ due to the first monotonicity property of θ^* stated in Theorem 5.2.11. As a result, one may write the following:

$$\begin{aligned} h((T_1^*-1,k)) &\leq h((T_1^*,k)) && \text{(because } \theta^*((T_1^*-1,k))=1\text{)} \\ &< h((T_1^*,k-1)) && \text{(because } \theta^*((T_1^*,k-1))=0\text{)} \\ &\leq h((T_1^*-1,k)) && \text{(because } \theta^*((T_1^*-1,k-1))=2\text{),} \end{aligned}$$

which states that $h((T_1^*-1,k))<h((T_1^*-1,k))$ and thereby yields another contradiction. In conclusion, it is impossible to have $\theta^*((T_1^*-1,T_2^*))=2$. Since it has already been shown that $\theta^*((T_1^*-1,T_2^*))\neq 0$, the only remaining possibility is $\theta^*((T_1^*-1,T_2^*))=1$.

In summary, it has been shown that $\theta^*((T_1^*,T_2^*))=0$, $\theta^*((T_1^*-1,T_2^*))=1$ and $\theta^*((T_1^*,T_2^*-1))=2$. Given that $\theta^*((T_1^*-1,T_2^*))=1$, it follows directly from the monotonicity properties of θ^* stated by Theorem 5.2.11 that $\theta^*((x_1,T_2^*))=1$ for all $x_1\leq T_1^*-1$. Similarly, given that $\theta^*((T_1^*,T_2^*-1))=2$, the same monotonicity properties imply that $\theta^*((T_1^*,x_2))=2$ for all $x_2\leq T_2^*-1$. Thus, at this point the actions chosen by θ^* at all states along the upper boundaries of the region $\{(x_1,x_2):x_1\leq T_1^*,x_2\leq T_2^*\}$ have been fully established, and importantly it has been found that $\theta^*((x_1,T_2^*))\neq 0$ for all $x_1<T_1^*$ and similarly $\theta^*((T_1^*,x_2))\neq 0$ for all $x_2<T_2^*$. Hence, by the monotonicity of θ^*, one cannot have $\theta^*((x_1,x_2))=0$ for any state (x_1,x_2) satisfying $x_1\leq T_1^*$ and $x_2\leq T_2^*$, since this would imply $\theta^*((x_1,T_2^*))=0$ and $\theta^*((T_1^*,x_2))=0$, thus yielding a contradiction.

Given that $\theta^*((x_1,T_2^*))\neq 2$ for all $x_1\leq T_1^*$ and $\theta^*((T_1^*,x_2))\neq 1$ for all $x_2\leq T_2^*$, it is clear that the set of states which are positive recurrent under θ^* is given by $S_{\theta^*}=\{(x_1,x_2):x_1\leq T_1^*,x_2\leq T_2^*\}$. Moreover, since it has also been shown that (T_1^*,T_2^*) is the only state in this set at which θ^* chooses to balk, it follows that θ^* is a sink policy with sink state (T_1^*,T_2^*). □

Example 5.2.16. ***(Sink policies in a*** $2DSS$ ***system)***

This example revisits the $2DSS$ system considered in Example 5.2.13. Table 5.1 shows the decisions made by the policies $\tilde{\theta}$ and θ^* at each state in $\tilde{S}$. Naturally, the selfish policy $\tilde{\theta}$ is a sink policy

with sink state $(7,7)$. At first glance, it might appear that the socially optimal policy θ^* is not a sink policy, as there are several states within $\tilde{S}$ at which balking is chosen by θ^*. However, upon closer inspection, it becomes clear that states in $\tilde{S}$ with $x_1 > 4$ or $x_2 > 4$ (or both) are not positive recurrent under θ^*; that is, the set of states which are positive recurrent under θ^* is $S_{\theta^*} = \big\{(x_1, x_2) \in \mathbb{N}_0^2 : x_1 \leq 4 \text{ and } x_2 \leq 4\big\}$. Since $(4,4)$ is the only state in S_{θ^*} at which balking is chosen by θ^*. it follows that θ^* is indeed a sink policy, with sink state $(4,4)$. ⊠

Of course, the thresholds T_1^* and T_2^* in Theorem 5.2.15 satisfy $T_1^* \leq \tilde{B}_1$ and $T_2^* \leq \tilde{B}_2$. This leads to the intuitively appealing concept of the socially optimal state space S_{θ^*} being a 'rectangle contained inside another rectangle', with the latter being the selfishly optimal state space $\tilde{S}$. It would be nice to be able to generalise this idea and say that, in a system with N facilities and multiple servers allowed at each facility, the socially optimal and selfishly optimal state spaces are both N-dimensional sink policies (or alternatively, N-dimensional cuboids), with the former being contained inside the latter. However, it transpires that this is not always true (see Appendix A.8). Of course, it is already known from Theorem 4.2.4 that $S_{\theta^*} \subseteq \tilde{S}$ holds when N and $\{c_i\}_{i=1}^N$ are arbitrary, but this does not imply that S_{θ^*} is cuboid in shape. Nevertheless, experiments have shown that sink policies can very often be found which achieve social optimality in systems which are not $2DSS$; see Section 5.4 for further discussion, including numerical results.

In the next section, another special case of the queueing system described in Section 3.1 will be considered; this time, it will be assumed that all facilities share the same parameters.

5.3 Homogeneous facilities

As discussed earlier in this chapter, structural properties of socially optimal policies in queueing systems with N heterogeneous facilities are not easy to obtain in complete generality; indeed, the results in Sections 5.1 and 5.2 apply to systems severely reduced in dimensionality. In this section, the number of facilities N is arbitrary and multiple servers are allowed at each facility, but it is assumed that service facilities are *homogeneous*. To be specific, each facility i has $c \geq 1$ servers available, a service rate $\mu > 0$ at each server, a holding cost $\beta > 0$ per customer per unit time and a reward for service $\alpha > 0$. As one might imagine, it is possible to characterise socially optimal policies to a much greater extent in the homogeneous facilities case than in general.

The results in this section will again make use of the principle that, due to Theorem 4.2.4, a socially (average reward) optimal policy for the system can be found by searching within the finite state space $\tilde{S} = \{(x_1, x_2, ..., x_N) \in S : x_i \leq \tilde{B} \ \ \forall\, i \in \{1, 2, ..., N\}\}$, where $\tilde{B} = \lfloor \alpha c \mu / \beta \rfloor$. This makes it possible to construct proofs based on the application of relative value iteration.

The first result in this section relates to a pair of states $\mathbf{x}, \mathbf{y} \in \tilde{S}$ which are *similar*, in the sense that $(x_1, x_2, ..., x_N)$ and $(y_1, y_2, ..., y_N)$ are two different permutations of the same set of N integers, or equivalently $(y_1, y_2, ..., y_N)$ is simply a re-ordering of $(x_1, x_2, ..., x_N)$. One might expect that, in the homogeneous facilities case, a socially optimal policy should be indifferent between two states $\mathbf{x}$ and $\mathbf{y}$ which are similar in this respect, and the next result confirms that this is true by comparing the values $h(\mathbf{x})$ and $h(\mathbf{y})$. Note that throughout this section, it is assumed that facilities are homogeneous (with the parameters α, β etc. stated previously), so this assumption is omitted from the statements of lemmas, theorems, etc. in order to avoid repetition. Also, as in the previous section, $h(\cdot)$ will denote the relative value function satisfying the average reward optimality equations (4.2.1) when the real-time reward formulation (3.5.4) is used.

Lemma 5.3.1. *Let $\mathbf{x}, \mathbf{y} \in \tilde{S}$ be two states which are 'similar', in the sense that $(y_1, y_2, ..., y_N)$ is a re-ordering of $(x_1, x_2, ..., x_N)$. Then $h(\mathbf{x}) = h(\mathbf{y})$.*

Proof. The proof can be accomplished by induction. See Appendix A.5, page 445.

Another lemma is required in order to prove the main result of this section.

Lemma 5.3.2. *For all $\mathbf{x} \in \tilde{S}$ and $i, j \in \{1, 2, ..., N\}$ such that $\mathbf{x}^{i+}, \mathbf{x}^{j+} \in \tilde{S}$:*

$$x_i \leq x_j \quad \Rightarrow \quad h(\mathbf{x}^{i+}) \geq h(\mathbf{x}^{j+}).$$

Proof. Again, the proof is based on induction. See Appendix A.5, page 447.

Using Lemma 5.3.2 it is possible to establish the average reward optimality of an intuitively simple type of policy known as a 'Join the Shortest Queue' (JSQ) policy.

Theorem 5.3.3. *For the N-facility system with homogeneous facilities, there exists a stationary average reward optimal policy θ^* such that, for all $\mathbf{x} \in \tilde{S}$:*

$$\theta^*(\mathbf{x}) \in \arg\min_i x_i \cup \{0\}.$$

More simply, it may be said that the stationary policy θ^* described in Theorem 5.3.3 always prefers facility i to facility j when $x_i < x_j$; that is, at any given state, θ^* always chooses either to balk or to join the shortest queue. This is why it is called a JSQ policy.

Proof. Assume that the reward formulation (3.5.4) is used. The average reward optimality equations (4.2.1) imply that a sufficient condition for θ^* to be optimal is that, for all $\mathbf{x} \in \tilde{S}$:

$$\theta^*(\mathbf{x}) \in \arg\max_{a \in A_{\mathbf{x}}} h(\mathbf{x}^{a+}) \tag{5.3.1}$$

where $h(\mathbf{x}^{0+}) = h(\mathbf{x})$ by definition, and it may be assumed that the values $h(\mathbf{x})$ are found by relative value iteration on the finite state space $\tilde{S}$. Consider an arbitrary state $\mathbf{x} \in \tilde{S}$, and let $i \in \{1, 2, ..., N\}$ be a facility satisfying $x_i < \tilde{B}$ (so that joining i is a permissible action at $\mathbf{x}$) and $x_i \leq x_j$ for all $j \neq i$. Then, by Lemma 5.3.2, $h(\mathbf{x}^{i+}) \geq h(\mathbf{x}^{j+})$ for any facility j such that $x_j < \tilde{B}$. Hence, the only way action i can fail to attain the maximum in (5.3.1) is if $h(\mathbf{x}^{i+}) < h(\mathbf{x})$; that is, at least one of the two actions i and 0 must attain the maximum in (5.3.1). It follows that one can define a stationary policy θ^* in such a way that, at any given state $\mathbf{x} \in \tilde{S}$:

$$\theta^*(\mathbf{x}) \in \begin{cases} \arg\min_{i \in \{1,2,...,N\}} x_i, & \text{if } \exists\ i \in \{1, 2, ..., N\} \text{ such that } x_i < \tilde{B} \text{ and } h(\mathbf{x}^{i+}) \geq h(\mathbf{x}), \\ \{0\}, & \text{otherwise,} \end{cases}$$

with the result that θ^* is an average reward optimal JSQ policy. □

The optimality of JSQ policies in queueing systems has previously been studied in a variety of contexts. Winston [201], Weber [192], Hordijk and Koole [87], Menich and Serfozo [129] and Koole et. al. [107] have all considered JSQ policies in queueing systems with homogeneous servers or facilities, although all of these authors have considered objectives which are somewhat different from the maximisation of the average net reward considered throughout this thesis. For example, Winston and Weber both discussed queueing systems with identical servers in which the objective is to maximise the number of service completions occurring in a finite time interval.

Interestingly, it may be shown that Lemmas 5.3.1 and 5.3.2 also hold in systems with *no balking allowed*, in which all customers arriving in the system must be routed to one of the N facilities (it is appropriate to make the assumption that $\lambda < c\mu$ in such systems in order to ensure system stability). Theorem 5.3.3 then implies that, in systems with no balking allowed, socially optimal

policies may be characterised very easily, since the decision at any state should simply be to join the shortest queue (or one of the shortest queues, if there is a tie). When balking is allowed (as is the case throughout this thesis), the characterisation of an optimal policy is essentially a matter of deciding which states should be the 'balk' states, since the JSQ rule provides the decision at any state where balking is not preferred. However, numerical experiments have shown that the distribution of 'balk states' over the finite space $\tilde{S}$ may be somewhat contrary to what one would expect, even when facilities are homogeneous; see Appendix A.8 for an example.

The next section will explore methods of searching for socially optimal policies which involve restricting attention to stationary policies with particular structural properties.

5.4 Computational algorithms

There are certain practical advantages to be gained from proving the existence of socially optimal policies with particular structural properties. For example, if it is known that an optimal policy exists which is *monotone* in some respect, then the search for an optimal policy may be confined to the set of policies which possess the relevant monotonicity property. Unfortunately, in a system with an arbitrary number of heterogeneous facilities, it is not realistic to suppose that optimal policies will necessarily possess structural properties such as monotonicity; indeed, the examples in Appendix A.8 show that the policy θ^* found by relative value iteration may exhibit quite surprising characteristics. Nevertheless, by devising computational algorithms which restrict attention to policies with ostensibly 'logical' properties, one might very often be able to find policies which perform extremely well, even if their optimality is not theoretically guaranteed.

Two algorithms will be presented in this section; broadly speaking, their objectives will be to obtain optimal or near-optimal policies in a more efficient way than the conventional dynamic programming algorithms (e.g. RVIA, PIA) discussed in Section 3.7; in other words, they will expend less computational effort than conventional DP algorithms, but they will also rely upon assumptions which may not hold in complete generality. Both algorithms are applicable to systems with any number of heterogeneous facilities N, with multiple servers permitted at all facilities. Experimental results (to be given later) indicate that these algorithms are capable of performing extremely well in systems where the selfish state space $\tilde{S}$ is of a manageable size.

In Section 5.1, certain results were proved involving the progression of finite-stage optimal policies in a system with only one multiple-server facility (essentially, an $M/M/c$ queue). Recall that, by Theorem 5.1.4, the optimal policy θ^* found by relative value iteration in a single-facility system is a threshold policy with some threshold $T^* \leq \tilde{T}$ (where $\tilde{T} = \lfloor \alpha c\mu/\beta \rfloor$ is the selfishly optimal threshold); that is, θ^* chooses to join at state $x \in \{0, 1, ..., \tilde{T}\}$ if and only if $x < T^*$. Let the finite-stage optimal thresholds $\hat{T}_n^*$ and T_n^* be defined for $n \in \mathbb{N}_0$ as follows:

$$\hat{T}_n^* := \min\big\{x \in \{0, 1, ..., \tilde{T} - 1\} : \hat{r}(x, 1) + \lambda\Delta\hat{h}_n(x + 1) < \lambda\Delta\hat{h}_n(x)\big\},$$
$$T_n^* := \min\big\{x \in \{0, 1, ..., \tilde{T} - 1\} : h_n(x + 1) < h_n(x)\big\},$$

where $\hat{h}_n(\cdot)$ is the finite-stage relative value function given that the anticipatory reward formulation (3.5.15) is used, and $h_n(\cdot)$ is the corresponding function when the real-time formulation (3.5.4) is used. One may define $\hat{T}_n^* = \tilde{T}$ in the case where $\{x \in \{0, 1, ..., \tilde{T} - 1\} : \hat{r}(x, 1) + \lambda\Delta\hat{h}_n(x + 1) < \lambda\Delta\hat{h}_n(x)\}$ is an empty set; similarly, $T_n^* = \tilde{T}$ in the case where $\{x \in \{0, 1, ..., \tilde{T} - 1\} : h_n(x + 1) < h_n(x)\}$ is empty. By Lemma 5.1.5, the following may be stated about $\hat{T}_n^*$ and T_n^*:

- $\hat{T}_n^*$ is monotonically decreasing with n;
- T_n^* is monotonically increasing with n.

Accordingly, it may be said that when the reward formulation (3.5.15) is used, the finite-stage optimal policies become increasingly conservative as the length of the horizon n increases, while the opposite is true when the formulation (3.5.4) is used. This is an intuitively appealing concept, but unfortunately Example A.8.3 in Appendix A.8 shows that it cannot be generalised to systems with an arbitrary number of facilities. However, experiments show that in systems where N is relatively small (so that optimal policies can be computed in a reasonable amount of time), there is a *general trend* for finite-stage optimal policies to gradually become either more conservative or less conservative as the length of the horizon increases, depending on which reward formulation is used; in other words, although counter-examples exist to show that Lemma 5.1.5 cannot be generalised completely, these counter-examples involve isolating particular iterations of the relative value iteration procedure which go very much against the prevailing trend. Another useful observation is that finite-stage optimal policies tend to be *sink policies* (see Definition 5.2.14), regardless of which reward formulation is used. Example 5.4.1, which follows shortly, illustrates this.

By the results in Section 4.2, the policy θ^* found by relative value iteration on the finite state space $\tilde{S}$ is average reward optimal. Of course, the set of recurrent states S_{θ^*} satisfies:

$$S_{\theta^*} \subseteq \tilde{S}. \tag{5.4.1}$$

Indeed, by Theorem 4.2.5, any average reward optimal policy must induce an irreducible, ergodic Markov chain which remains 'contained' within $\tilde{S}$. Moreover, due to Theorem 4.4.4, the selfishly optimal policy $\tilde{\theta}$ (whose recurrent state space is $\tilde{S}$) is asymptotically optimal in a light-traffic limit, so it does not appear possible to obtain a stronger result than (5.4.1) (i.e. a smaller containing set for S_{θ^*}) for a general system with N heterogeneous facilities, unless further information is known about the system parameters. This is somewhat of a drawback, since one might very conceivably have a system where S_{θ^*} is much smaller than $\tilde{S}$; indeed, Theorem 4.4.7 implies that if the demand rate λ is large, then S_{θ^*} may be of a similar size to the set S° defined in (4.3.3), in which case relative value iteration applied to the selfish state space $\tilde{S}$ is likely to be inefficient due to the large number of states outside S_{θ^*} evaluated by the algorithm on each of its iterations.

It would be useful to have a DP algorithm with the ability to intelligently adjust the bounds of the finite state space during its evolution according to the progression of the finite-stage optimal policies, in order to avoid repeatedly sweeping through a state space loaded with states which would become transient (and therefore redundant) under the infinite-horizon policy θ^*.

The two algorithms discussed in the remainder of this section operate by progressively altering the bounds of the finite state space within which an optimal policy is assumed to exist, in the manner described above. Essentially, both algorithms are variants on the same theme. It will be convenient to introduce some extra notation at this point. Let $\hat{\theta}_n$ denote the finite-stage optimal policy obtained on the n^{th} iteration of relative value iteration when the reward formulation (3.5.15) is used, and let θ_n denote the corresponding finite-stage policy when the formulation (3.5.4) is used. That is, for all $\mathbf{x} \in \tilde{S}$ and $n \in \mathbb{N}_0$, the actions $\hat{\theta}_{n+1}(\mathbf{x})$ and $\theta_{n+1}(\mathbf{x})$ satisfy:

$$\begin{aligned} \hat{\theta}_{n+1}(\mathbf{x}) &\in \underset{a \in A_{\mathbf{x}}}{\arg\max} \left(\hat{r}(\mathbf{x}, a) + \lambda \Delta \hat{h}_n(\mathbf{x}^{a+}) \right), \\ \theta_{n+1}(\mathbf{x}) &\in \underset{a \in A_{\mathbf{x}}}{\arg\max}\, h_n(\mathbf{x}^{a+}). \end{aligned} \tag{5.4.2}$$

Also, let $\hat{S}_n$ and S_n denote the sets of states which are positive recurrent under $\hat{\theta}_n$ and θ_n respectively. The next example illustrates the progressions of $\hat{S}_n$ and S_n as n increases.

Example 5.4.1. ***(Finite-horizon sink policies)***

This example illustrates the progression of optimal finite-horizon policies in a system with 2 facilities. Suppose the demand rate is $\lambda = 20$ and the parameters for the facilities are:

$$\begin{aligned} c_1 &= 4, \qquad \mu_1 = 6, \qquad \beta_1 = 11, \qquad \alpha_1 = 7, \\ c_2 &= 4, \qquad \mu_2 = 4, \qquad \beta_2 = 6, \qquad \alpha_2 = 4. \end{aligned}$$

Suppose the *anticipatory* reward formulation (3.5.15) is adopted. Experiments show that, for this particular system, the finite-horizon optimal policies $\hat{\theta}_n$ obtained during the Relative Value Iteration Algorithm (RVIA) are invariably sink policies, and that these policies tend to become more conservative as the number of iterations increases. The selfishly optimal state space is $\tilde{S} = \big\{(x_1, x_2) \in \mathbb{N}_0^2 : x_1 \leq 15 \text{ and } x_2 \leq 10\big\}$, and the algorithm converges to a policy θ^* whose positive recurrent state space is the smaller region $S_{\theta^*} = \big\{(x_1, x_2) \in \mathbb{N}_0^2 : x_1 \leq 8 \text{ and } x_2 \leq 8\big\}$.

Figure 5.7 provides a somewhat crude graphical representation of the progression of finite-horizon optimal policies during the RVIA. Six values of n (the iteration counter) have been chosen arbitrarily, and for each value of n an array of small boxes has been drawn to represent the individual states in $\tilde{S}$. The state in the upper-left corner of each grid is $(0, 0)$, and the state in the lower-right corner of each grid is $(15, 10)$. Each small box is coloured either green or white depending on whether the corresponding state is positive recurrent or transient under the policy $\hat{\theta}_n$ obtained after the stated number of iterations. The figure shows that all states in $\tilde{S}$ are positive recurrent under the initial policy $\hat{\theta}_1$, but the regions $\hat{S}_n$ gradually become smaller as n increases, until eventually (for all $n \geq 41$) only states belonging to S_{θ^*} are positive recurrent under the policy $\hat{\theta}_n$.

On the other hand, when the *real-time* reward formulation (3.5.4) is used, the finite-horizon policies θ_n progress in a different manner. Figure 5.8 shows that, while it remains true that each policy θ_n is a sink policy, these policies become *less* conservative as the number of iterations n increases. The first policy depicted in Figure 5.8, θ_2, is in fact a *vacancy* policy (see Definition 4.4.6), but the policies θ_8, θ_{15} etc. are obviously somewhat less stringent. Eventually, as expected, the algorithm converges to the same policy θ^* arrived at under the anticipatory formulation. ⊠

The first of the two algorithms to be presented next is applicable when the anticipatory reward formulation (3.5.15) is used, and is based on the premise that on each iteration n, it is reasonable

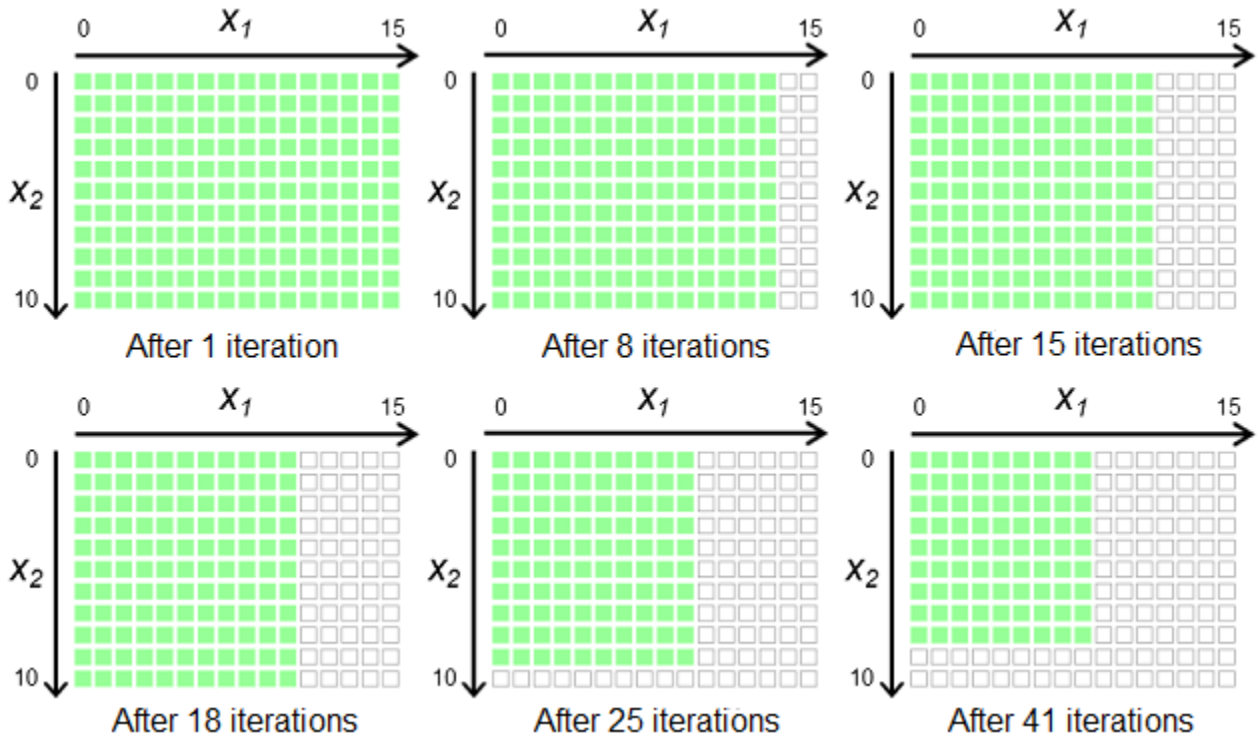


Figure 5.7: The progression of finite-horizon optimal policies for the system in Example 5.4.1, assuming the reward formulation (3.5.15). Green (white) boxes represent positive recurrent (transient) states.

(albeit not completely sound theoretically) to suppose that an optimal policy θ^* exists whose recurrent state space S_{θ^*} is contained within the finite set $\hat{S}_n$. This is because, as illustrated by Example 5.4.1, the finite-stage optimal policies tend to become *more* conservative as the horizon length n increases when the formulation (3.5.15) is used; so if a particular state is excluded from the recurrent state space of $\hat{\theta}_n$, it is reasonable to suppose that it will also be excluded from the recurrent state space of the limiting policy θ^*. Consequently, one may proceed by applying relative value iteration with an initial finite state space $\tilde{S}$ and, on each iteration n, permanently 'deleting' any state not included in $\hat{S}_n$ from the finite state space, so that the set of states evaluated on iteration $n+1$ consists only of states which are positive recurrent under $\hat{\theta}_n$.

The procedure can also be made somewhat simpler by assuming that each finite-stage optimal policy $\hat{\theta}_n$ is a *sink policy*. For each $j \in \{1, 2, ..., N\}$, define $\hat{B}_j^{(n)}$ as follows:

$$\hat{B}_j^{(n)} = \max\big\{k \in \mathbb{N}_0 : \exists \mathbf{x} \in \hat{S}_n \text{ such that } x_j = k\big\}.$$

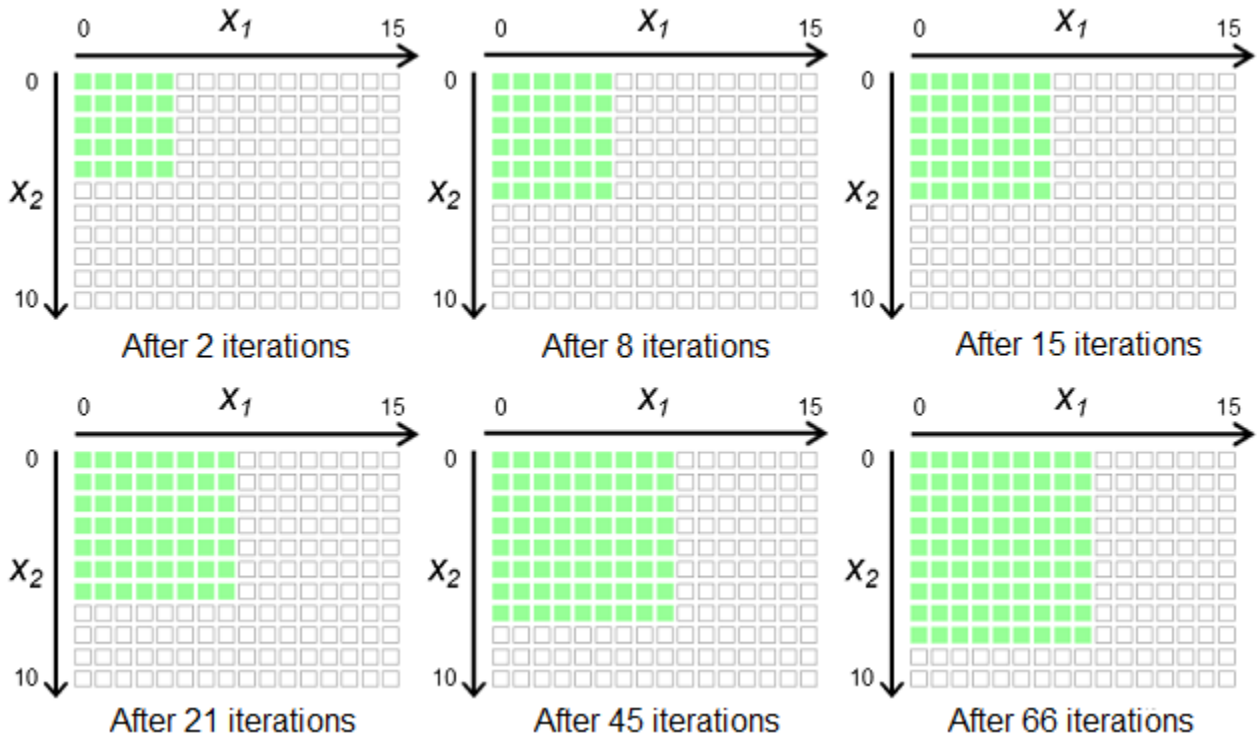


Figure 5.8: The progression of finite-horizon optimal policies for the system in Example 5.4.1, assuming the reward formulation (3.5.4). Green (white) boxes represent positive recurrent (transient) states.

Also, let the set $\hat{R}_n$ (for $n = 0, 1, 2, ...$) be defined as follows:

$$\hat{R}_n = \{\mathbf{x} \in \tilde{S} : x_j \leq \hat{B}_j^{(n)} \ \forall \ j \in \{1, 2, ..., N\}\}. \tag{5.4.3}$$

If $\hat{\theta}_n$ is a sink policy, then clearly $\hat{R}_n$ is identical to the set $\hat{S}_n$; that is, $\hat{R}_n$ is equal to the set of states which are positive recurrent under $\hat{\theta}_n$. On the other hand, if $\hat{\theta}_n$ is *not* a sink policy, then clearly $\hat{R}_n$ contains all states in $\hat{S}_n$ due to the definition of $\hat{B}_j^{(n)}$, plus (possibly) some extra states which are not included in $\hat{S}_n$. To put it another way, $\hat{R}_n$ is the smallest N-dimensional cuboid which contains $\hat{S}_n$. The notion of a sequence of N-dimensional cuboids which become progressively smaller being used to represent the progression of finite-horizon optimal policies during relative value iteration provides the inspiration for the 'Shrinking Box' algorithm below.

Shrinking Box Algorithm

1. Set $n = 0$, $\hat{h}_0(\mathbf{x}) = 0$ and $\hat{w}_0(\mathbf{x}) = 0$ for all $\mathbf{x} \in \tilde{S}$. Also set $\hat{B}_j^{(0)} = \tilde{B}_j$ for all $j \in \{1, 2, ..., N\}$, where $\tilde{B}_j$ is the selfish threshold for facility j defined in (4.1.2).

2. Let $\hat{R}_n = \big\{\mathbf{x} \in \tilde{S} : x_j \leq \hat{B}_j^{(n)} \ \ \forall\ j \in \{1, 2, ..., N\}\big\}$, as in (5.4.3). For each $\mathbf{x} \in \hat{R}_n$, define the set of actions $A_{\mathbf{x}}^{(n)}$ available at state $\mathbf{x}$ as follows:

$$A_{\mathbf{x}}^{(n)} = \big\{0, 1, ..., N\big\} \setminus \big\{j \in \{1, 2, ..., N\} : x_j \geq \hat{B}_j^{(n)}\big\}.$$

That is, joining facility j at state $\mathbf{x}$ is not permitted if $x_j \geq \hat{B}_j^{(n)}$.

3. For each $\mathbf{x} \in \hat{R}_n$, let $\hat{w}_{n+1}(\mathbf{x}) = \max\limits_{a \in A_{\mathbf{x}}^{(n)}} \left\{ \hat{r}(\mathbf{x}, a) + \sum\limits_{\mathbf{y} \in \hat{R}_n} p(\mathbf{x}, a, \mathbf{y}) \hat{h}_n(\mathbf{y}) \right\}$.

4. Set $\hat{h}_{n+1}(\mathbf{x}) = \hat{w}_{n+1}(\mathbf{x}) - \hat{w}_{n+1}(\mathbf{0})$ for all $\mathbf{x} \in \hat{R}_n$.

5. If $n = 0$, set $\delta = 1$; otherwise set $\delta = \max\limits_{\mathbf{x} \in \hat{R}_n} \big|\hat{h}_{n+1}(\mathbf{x}) - \hat{h}_n(\mathbf{x})\big|$. If $\delta < \epsilon$, where ϵ is a small positive number, go to step 7; otherwise, proceed to step 6.

6. Let $\hat{\theta}_{n+1}$ be a stationary policy which satisfies, for all $\mathbf{x} \in \hat{R}_n$:

$$\hat{\theta}_{n+1}(\mathbf{x}) \in \arg\max_{a \in A_{\mathbf{x}}^{(n)}} \left(\hat{r}(\mathbf{x}, a) + \lambda \Delta \hat{h}_n(\mathbf{x}^{a+}) \right).$$

For each $j \in \{1, 2, ..., N\}$, let $\hat{B}_j^{(n+1)}$ be defined as follows:

$$\hat{B}_j^{(n+1)} = 1 + \max\big\{k \in \mathbb{N}_0 : \exists \mathbf{x} \in \hat{R}_n \text{ such that } \hat{\theta}_{n+1}(\mathbf{x}) = j \text{ and } x_j = k\big\}.$$

Then increment n by 1 and return to step 2.

7. Output $\hat{h}_{n+1}(\mathbf{x})$ for all $\mathbf{x} \in \hat{R}_n$ and a stationary policy θ^* such that, for each $\mathbf{x} \in \hat{R}_n$, $\theta^*(\mathbf{x})$ maximises $\left\{ \hat{r}(\mathbf{x}, a) + \sum\limits_{\mathbf{y} \in \hat{R}_n} p(\mathbf{x}, a, \mathbf{y}) \hat{h}_{n+1}(\mathbf{y}) \right\}$ over all actions $a \in A_{\mathbf{x}}^{(n)}$.

Note that in step 4 of the algorithm, the state $\mathbf{0}$ is used as a fixed 'reference state' for ensuring that the iterates $\hat{h}_n(\mathbf{x})$ do not become unboundedly large (refer to the discussion in Section 3.7). In the standard version of the Relative Value Iteration Algorithm on page 101, this reference state is allowed to be an arbitrary state in $\tilde{S}$, but in the Shrinking Box algorithm it is necessary to choose a reference state which is not at risk of being permanently deleted from the finite state space at some stage of the procedure. The state $\mathbf{0}$ obviously satisfies this requirement.

It is important to emphasise that the Shrinking Box algorithm assumes that the reward formulation (3.5.15) is used. As alluded to earlier, there exists an alternative variant of this algorithm which

utilises the 'real-time' reward formulation, (3.5.4). When the real-time formulation is used, the general trend is for the finite-stage optimal policies to become *less* conservative as the horizon length n increases. However, it may be seen from the proof of Lemma 4.3.2 that if $\mathbf{x}$ is a state with $x_j < c_j$ for some $j \in \{1, 2, ..., N\}$, then the inequality $h_n(\mathbf{x}^{j+}) > h_n(\mathbf{x})$ holds for all $n \in \mathbb{N}_0$. As such, all of the finite-horizon optimal policies θ_n satisfying (5.4.2) must be *non-idling* policies; that is, none of the finite-stage optimal policies θ_n choose to balk at any state with at least one idle server. Recall that, for $n \in \mathbb{N}_0$, S_n denotes the set of states which are positive recurrent under θ_n. By the non-idling principle, it follows that each set S_n satisfies the following:

$$S^{\circ} = \left\{\mathbf{x} \in \tilde{S} : x_j \leq c_j \ \ \forall\, j \in \{1, 2, ..., N\}\right\} \subseteq S_n. \tag{5.4.4}$$

When the real-time reward formulation is used, it is logical to use an algorithm which takes the opposite approach from that of the Shrinking Box algorithm, by starting with a *small* subset of $\tilde{S}$ as the finite state space and gradually *enlarging* the set of visitable states according to the progression of the finite-stage optimal policies. Roughly speaking, the aim of the algorithm should be to ensure that the finite set of states evaluated on some iteration $n \in \mathbb{N}_0$ includes not only the set of all states accessible from $\mathbf{0}$ under the policy θ_n, but also a certain number of 'outlying' states which lie slightly beyond the boundaries of S_n, so that there is some 'leeway' for the finite-stage optimal policy to expand the set of states that it wishes to visit when the length of the horizon increases by one. As the iterations progress and the finite-stage policies 'expand' (i.e. become less conservative), the set of states evaluated should also expand in size, so that there is always scope for further expansion. This idea will be made clear by the full specification of the new algorithm, referred to as the 'Expanding Box algorithm', and the example that follows later.

Expanding Box Algorithm

1. Set $n = 1$ and $h_1(\mathbf{x}) = r(\mathbf{x})$ for all $\mathbf{x} \in \tilde{S}$. Also set $B_j^{(1)} = \min(c_j + 1, \tilde{B}_j)$ for all $j \in \{1, 2, ..., N\}$, where $\tilde{B}_j$ is the selfish threshold for facility j defined in (4.1.2).

2. Let $R_n = \left\{\mathbf{x} \in \tilde{S} : x_j \leq B_j^{(n)} \ \ \forall\ j \in \{1, 2, ..., N\}\right\}$. If $n \geq 2$ and $R_n \neq R_{n-1}$, then for each newly-added state $\mathbf{x} \in R_n \setminus R_{n-1}$ initialise the value $h_n(\mathbf{x})$ as follows:

$$h_n(\mathbf{x}) = \min_{\mathbf{y} \in R_{n-1}} h_n(\mathbf{y}).$$

For each $\mathbf{x} \in R_n$, define the set of actions $A_{\mathbf{x}}^{(n)}$ available at state $\mathbf{x}$ as follows:

$$A_{\mathbf{x}}^{(n)} = \{0, 1, ..., N\} \setminus \{j \in \{1, 2, ..., N\} : x_j \geq B_j^{(n)}\}.$$

That is, joining facility j at state $\mathbf{x}$ is not permitted if $x_j \geq B_j^{(n)}$.

3. For each $\mathbf{x} \in R_n$, let $w_{n+1}(\mathbf{x}) = \max\limits_{a \in A_{\mathbf{x}}^{(n)}} \left\{ r(\mathbf{x}) + \sum\limits_{\mathbf{y} \in R_n} p(\mathbf{x}, a, \mathbf{y}) h_n(\mathbf{y}) \right\}$.

4. Set $h_{n+1}(\mathbf{x}) = w_{n+1}(\mathbf{x}) - w_{n+1}(\mathbf{0})$ for all $\mathbf{x} \in R_n$.

5. If $n = 0$, set $\delta = 1$; otherwise set $\delta = \max\limits_{\mathbf{x} \in R_n} \left| h_{n+1}(\mathbf{x}) - h_n(\mathbf{x}) \right|$. If $\delta < \epsilon$, where ϵ is a small positive number, go to step 7; otherwise, proceed to step 6.

6. Let θ_{n+1} be a stationary policy which satisfies, for all $\mathbf{x} \in R_n$:

$$\theta_{n+1}(\mathbf{x}) \in \arg\max_{a \in A_{\mathbf{x}}^{(n)}} h_n(\mathbf{x}^{a+}).$$

For each facility $j \in \{1, 2, ..., N\}$, let $B_j^{(n+1)}$ be defined as follows:

$$B_j^{(n+1)} = \begin{cases} \min(B_j^{(n)} + 1,\ \tilde{B}_j), & \text{if } \exists\, \mathbf{x} \in R_n \text{ such that } \theta_{n+1}(\mathbf{x}) = j \text{ and } x_j = B_j^{(n)} - 1, \\ B_j^{(n)}, & \text{otherwise.} \end{cases}$$

Then increment n by 1 and return to step 2.

7. Output $h_{n+1}(\mathbf{x})$ for all $\mathbf{x} \in R_n$ and a stationary policy θ^* such that, for each $\mathbf{x} \in R_n$, $\theta^*(\mathbf{x})$ maximises $\left\{ r(\mathbf{x}) + \sum\limits_{\mathbf{y} \in R_n} p(\mathbf{x}, a, \mathbf{y}) h_{n+1}(\mathbf{y}) \right\}$ over all actions $a \in A_{\mathbf{x}}^{(n)}$.

Steps 1 and 2 of the Expanding Box algorithm define the initial finite state space R_1 as the set $\{\mathbf{x} \in \tilde{S} : x_j \leq c_j + 1 \text{ and } x_j \leq \tilde{B}_j\}$. On the first iteration, the two-stage optimal policy θ_2 is derived and this policy is forced to be 'contained' within the set R_1 due to the definition of the action sets $A_{\mathbf{x}}^{(1)}$ in step 2. For each $j \in \{1, 2, ..., N\}$, it may or may not be the case that the policy θ_1 'reaches the boundary' of R_1 by choosing to join facility j at some state $\mathbf{x} \in R_1$ with $x_j = B_j^{(1)} - 1$ and thereby causing the state $\mathbf{y} = \mathbf{x}^{j+}$ (with $y_j = B_j^{(1)}$) to be included in the set of positive recurrent states S_1. If this occurs, then the boundary $B_j^{(1)}$ is deemed to be too restrictive, and so (in step 6) this boundary is increased by one, *unless* $B_j^{(1)}$ is already equal to the selfish threshold $\tilde{B}_j$ defined in (4.1.2), in which case no change is made. A similar procedure is followed on subsequent iterations,

so that on each iteration n there is a possibility of new states being 'added' to the finite state space R_n, but (in contrast to the Shrinking Box algorithm) no states are ever removed.

The strategy of 'adding new states' to the state space at various stages of the algorithm causes a slight technical dilemma, which is addressed in step 2. Obviously, step 3 requires a value $h_n(\mathbf{x})$ to be associated with each state $\mathbf{x} \in R_n$. States which have been 'retained' from the previous iteration will already possess a $h_n(\mathbf{x})$ value, but for any newly-added state $\mathbf{x} \in R_n \setminus R_{n-1}$ it becomes necessary to define a value $h_n(\mathbf{x})$ artificially. The simple solution of setting $h_n(\mathbf{x}) = \min_{\mathbf{y} \in R_{n-1}} h_n(\mathbf{y})$ for all such states works sufficiently well in practice, since the important principle is *not* to set these values too high. If the value $h_n(\mathbf{x})$ for some newly-added state $\mathbf{x} \in R_n \setminus R_{n-1}$ is set too high, this may cause unnecessary enlargement of the finite state space R_{n+1} due to $\mathbf{x}$ being incorrectly identified as a 'desirable' state by the policy θ_{n+1}. While unnecessary enlargement of the state space R_{n+1} does not jeopardise the ability of the algorithm to find an optimal policy, it does somewhat defeat the purpose of the algorithm by slowing down computation times. It is therefore desirable to avoid this problem by setting the value of $h_n(\mathbf{x})$ so that it is bounded above by $h_n(\mathbf{y})$ for any $\mathbf{y} \in R_{n-1}$. The next example demonstrates both of the algorithms introduced in this section.

Example 5.4.2. ***(Shrinking Box and Expanding Box algorithms)***

This brief example demonstrates how the Shrinking Box and Expanding Box algorithms operate in the case of a small system with only two dual-server facilities. Suppose the demand rate is $\lambda = 12$ and the parameters for the two facilities are given as follows:

$$\begin{aligned} c_1 &= 2, \qquad \mu_1 = 6, \qquad \beta_1 = 11, \qquad \alpha_1 = 4, \\ c_2 &= 2, \qquad \mu_2 = 4, \qquad \beta_2 = 6, \qquad \alpha_2 = 5. \end{aligned}$$

The selfish thresholds for this system, as defined in (4.1.2), are $\tilde{B}_1 = 4$ and $\tilde{B}_2 = 6$. Suppose the Shrinking Box algorithm is applied, which requires the reward formulation (3.5.15) to be assumed. The policy $\hat{\theta}_1$ obtained on the first iteration of the algorithm is shown in Table 5.3.

Clearly, the values $\hat{B}_1^{(1)}$ and $\hat{B}_2^{(1)}$ as defined in step 6 of the Shrinking Box algorithm are equal to $\tilde{B}_1$ and $\tilde{B}_2$ respectively; this is as expected, since $\hat{\theta}_1$ must be identical to the selfishly optimal policy $\tilde{\theta}$. As a result, $\hat{R}_1$ is equal to $\hat{R}_0$; that is, the state space does not 'shrink' after the first iteration. On the next two iterations, it transpires that $\hat{R}_2$ and $\hat{R}_3$ are also equal to $\hat{R}_0$. However,

on the *fourth* iteration of the algorithm, the policy shown in Table 5.4 is obtained.

	$x_2=0$	$x_2=1$	$x_2=2$	$x_2=3$	$x_2=4$	$x_2=5$	$x_2=6$
$x_1=0$	2	2	2	1	1	1	1
$x_1=1$	2	2	2	1	1	1	1
$x_1=2$	2	2	2	2	1	1	1
$x_1=3$	2	2	2	2	2	1	1
$x_1=4$	2	2	2	2	2	2	0

Table 5.3: The policy $\hat{\theta}_1$ obtained after 1 iteration of the Shrinking Box algorithm.

	$x_2=0$	$x_2=1$	$x_2=2$	$x_2=3$	$x_2=4$	$x_2=5$	$x_2=6$
$x_1=0$	2	2	2	1	1	1	1
$x_1=1$	2	2	2	1	1	1	1
$x_1=2$	2	2	2	2	2	1	1
$x_1=3$	2	2	2	2	2	2	0
$x_1=4$	2	2	2	2	2	2	0

Table 5.4: The policy $\hat{\theta}_4$ obtained after 4 iterations of the Shrinking Box algorithm.

The non-highlighted states $(4,0)$, $(4,1)$ etc. are *transient* states under the policy $\hat{\theta}_4$ and are thus excluded from the set $\hat{R}_4$. That is, these seven states are permanently deleted (along with their relative values) and the action sets $A_{\mathbf{x}}^{(4)}$ for states $\mathbf{x} \in \hat{R}_4$ with $x_1 = 3$ are amended so that joining facility 1 at these states is no longer permitted. The algorithm then continues with a 'shrunken' state space. The policy $\hat{\theta}_5$ obtained on the next iteration is shown in Table 5.5.

	$x_2=0$	$x_2=1$	$x_2=2$	$x_2=3$	$x_2=4$	$x_2=5$	$x_2=6$
$x_1=0$	2	2	2	1	1	1	1
$x_1=1$	2	2	2	1	1	1	1
$x_1=2$	2	2	2	2	2	1	1
$x_1=3$	2	2	2	2	2	2	0

Table 5.5: The policy $\hat{\theta}_5$ obtained after 5 iterations of the Shrinking Box algorithm.

Naturally, all iterations from iteration 5 onwards will be faster than the first four iterations, since

the number of states to be evaluated is smaller. Now suppose that the Expanding Box algorithm is applied to the same system; of course, this requires the real-time formulation (3.5.4) to be assumed. Since both facilities have two servers, the size of the initial state space R_1 is $(c_1+2)(c_2+2)=16$. The policy θ_2 obtained on the first iteration of the algorithm is shown in Table 5.6.

	$x_2=0$	$x_2=1$	$x_2=2$	$x_2=3$
$x_1=0$	2	2	1	1
$x_1=1$	2	2	1	1
$x_1=2$	2	2	0	0
$x_1=3$	2	2	0	0

Table 5.6: The policy θ_2 obtained after 1 iteration of the Expanding Box algorithm.

The non-highlighted states in Table 5.6 are transient under the policy θ_2, but these states are *not* deleted from the finite state space, since the Expanding Box algorithm always includes an extra 'layer' of transient states in order to allow the finite-horizon policies room to 'expand' (indeed, no states are ever deleted at any stage of the algorithm). However, the fact that the highlighted states are transient implies that the state space does not expand after the first iteration; that is, R_2 is equal to R_1. It transpires that the sets R_3, R_4, R_5 and R_6 obtained after the next four iterations are also equal to R_1. On the sixth iteration, the policy shown in Table 5.7 is found.

	$x_2=0$	$x_2=1$	$x_2=2$	$x_2=3$
$x_1=0$	2	2	1	1
$x_1=1$	2	2	1	1
$x_1=2$	2	2	2	0
$x_1=3$	2	2	2	0

Table 5.7: The policy θ_7 obtained after 6 iterations of the Expanding Box algorithm.

The states $(0,3)$, $(1,3)$ and $(2,3)$ become positive recurrent under θ_7, which causes the finite state space to expand in the x_2-direction. This means that the action sets $A_{\mathbf{x}}^{(7)}$ for states $\mathbf{x}\in R_6$ with $x_2=3$ are defined in such a way that joining facility 2 is permitted, and the new states $(0,4)$, $(1,4)$, $(2,4)$ and $(3,4)$ are included in the set R_7. Hence, the set R_7 consists of 20 states as opposed to 16. The policy θ_8 obtained on the next iteration is shown in Table 5.8.

	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$	$x_2 = 3$	$x_2 = 4$
$x_1 = 0$	2	2	1	1	0
$x_1 = 1$	2	2	1	1	0
$x_1 = 2$	2	2	2	0	0
$x_1 = 3$	2	2	2	0	0

Table 5.8: The policy θ_8 obtained after 7 iterations of the Expanding Box algorithm.

Of course, since the expansion of the state space is permanent, the sets R_n for all $n \geq 7$ will consist of at least 20 states. In this particular example, both the Shrinking Box algorithm and the Expanding Box algorithm successfully converge to the same stationary policy θ^* as the conventional RVIA given in Section 3.7; that is, both algorithms yield an optimal policy. ⊠

The next example presents the results of numerical experiments, designed to evaluate the performances of the Shrinking Box and Expanding Box algorithms in systems with $N \leq 5$.

Example 5.4.3. ***(Numerical results)***

The Shrinking Box and Expanding Box algorithms discussed in this section thus far are intended to attain optimal or near-optimal policies without performing as many computation steps as conventional DP algorithms. As discussed earlier, this is done by making assumptions about the progression of finite-horizon optimal policies. This example describes the results of a series of experiments involving 15,096 randomly-generated sets of parameters. For each set of parameters, three algorithms were executed: the RVIA (as described on page 101), the Shrinking Box algorithm, and the Expanding Box algorithm. All three algorithms used the value $\epsilon = 10^{-6}$ as a stopping parameter. For each system, the random parameters were generated as follows:

- The number of facilities, N, was sampled unbiasedly from the set $\{2, 3, 4, 5\}$.
- Each service rate μ_i was sampled from a uniform distribution between 5 and 25.
- Each service capacity c_i was sampled unbiasedly from the set $\{2, 3, 4, 5\}$.
- Each holding cost β_i was sampled from a uniform distribution between 5 and 25.

- Each fixed reward α_i was sampled from a uniform distribution between 2 and 12.
- The demand rate λ was sampled from a uniform distribution between $0.1 \times \sum_{i=1}^{N} c_i\mu_i$ and $1.1 \times \sum_{i=1}^{N} c_i\mu_i$.

In order to allow a large number of experiments to be performed in a reasonable amount of time, parameter sets were accepted only if they gave a value of $|\tilde{S}|$ (where, as usual, $\tilde{S}$ denotes the selfish state space) between 1000 and 100,000. In addition, all facilities i were required to satisfy the condition $\alpha_i - \beta_i/\mu_i > 0$ in order to avoid degeneracy. Parameter sets which did not satisfy the criteria were rejected, and in these cases all parameter values except for the number of facilities N were re-sampled until the criteria were satisfied. The Shrinking Box and Expanding Box algorithms both demonstrated an extremely strong performance in all of the 15,096 experiments performed in terms of their ability to find an optimal or near-optimal policy. In summary:

- In all of the 15,096 experiments performed, the Expanding Box algorithm converged to a policy which was identical to the RVIA policy θ^* in terms of the decisions chosen at positive recurrent states $\mathbf{x} \in S_{\theta^*}$ (and hence earned the same average reward).
- In 13,994 of the 15,096 experiments performed (92.7%), the Shrinking Box algorithm also converged to a policy which was identical to the RVIA policy θ^* in terms of the decisions chosen at positive recurrent states $\mathbf{x} \in S_{\theta^*}$. In cases where the Shrinking Box policy was *not* identical to the RVIA policy, the average sub-optimality of the Shrinking Box policy was approximately 0.0012%. Its maximum sub-optimality (over all trials) was 0.11%.

The next objective in this example is to examine the computational savings that the Expanding Box and Shrinking Box algorithms offer in comparison to the RVIA (in other words, their ability to improve upon the running time required by the RVIA). For this purpose, the *number of value function updates* performed by the three algorithms during their respective running times will be considered. For clarity, a *value function update* is said to occur when the finite-stage relative value function $h_n(\mathbf{x})$ is updated for some state $\mathbf{x} \in \tilde{S}$ on some iteration n. The RVIA performs $|\tilde{S}|$ value function updates on each iteration. The Shrinking Box algorithm begins by performing $|\tilde{S}|$ updates per iteration, but the number of updates per iteration decreases as the algorithm progresses.

Conversely, the Expanding Box algorithm may begin by performing a relatively small number of updates per iteration, but this number will increase as the algorithm progresses.

Figure 5.9 shows a percentile plot, illustrating the distribution of the total number of value function updates performed by each of the three algorithms (RVIA, Expanding Box and Shrinking Box) over 15,096 trials. The plot shows (for each of the three algorithms) the percentage of trials in which the total number of value function updates performed was K or less, for various different values of K. For example, the plot indicates that the percentage of trials which required fewer than 10 million value function updates in total was about 47.8% in the case of the RVIA, and 86.9% and 92.8% in the cases of the Expanding Box and Shrinking Box algorithms respectively.

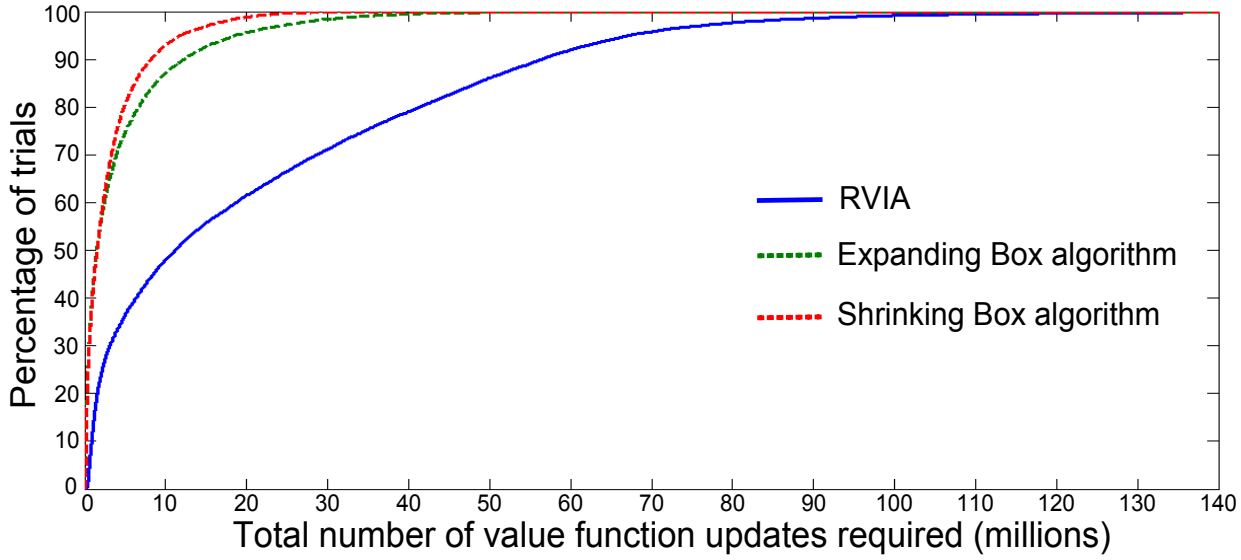


Figure 5.9: A percentile plot illustrating the distributions of the total number of value function updates required for the RVIA, Shrinking Box algorithm and Expanding Box algorithm.

Figure 5.10 illustrates the distributions of the percentage savings made by the Expanding Box and Shrinking Box algorithms over all 15,096 trials. These savings are shown in terms of the number of value function updates performed; for example, the rightmost two columns in the figure indicate that the number of trials in which the Expanding Box algorithm improved by more than 90% on the total number of value function updates made by the RVIA was approximately 4900, whereas the corresponding figure for the Shrinking Box algorithm was approximately 4700.

The leftmost bar in Figure 5.10 indicates that in approximately 1% of the trials performed, the

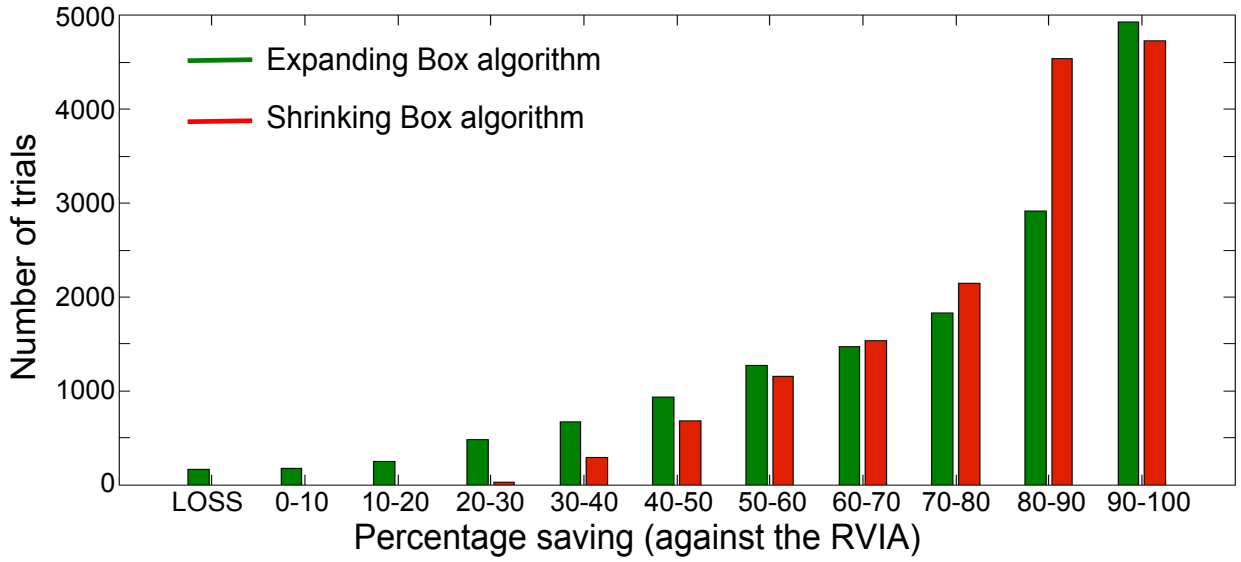


Figure 5.10: The percentage savings (in terms of the total number of value function updates required) made by the Expanding Box and Shrinking Box algorithms against the RVIA, over 15,096 trials.

number of value function updates made by the Expanding Box algorithm actually exceeded the corresponding number performed by the RVIA. Of course, the number of updates performed by the Expanding Box algorithm *on a single iteration* cannot exceed $|\tilde{S}|$, so it cannot perform more updates than the RVIA on a single iteration; however, it may require a larger number of iterations than the RVIA in total before convergence is reached. The reason for this is that the Expanding Box algorithm operates in such a way that it *adds new states* to the finite state space R_n each time the state space is expanded; when these new states are added, the corresponding relative values $h_n(\mathbf{x})$ must be given arbitrary initial values (see step 2). These initial values may be some distance from the limiting values $h(\mathbf{x})$, in which case convergence may be somewhat slower than one would attain by updating the values for *all* states $\mathbf{x} \in \tilde{S}$ on each iteration (as per the RVIA).

With a little thought, one realises that certain results from earlier chapters offer clues as to the conditions on the system parameters which will cause the Expanding Box algorithm to converge more quickly (in terms of the number of value function updates required) than the Shrinking Box algorithm, and vice versa. In particular, the value of the demand rate λ is critical. Theorem 4.4.4 suggests that when λ is small, the recurrent state space S_{θ^*} of the policy found by the RVIA should be of roughly the same size as the selfish state space $\tilde{S}$. The Shrinking Box algorithm begins with

a state space equal to $\tilde{S}$, so one would expect this state space to 'shrink' only a small number of times during the evolution of the algorithm when λ is small; on the other hand, the Expanding Box algorithm may have to expand the size of its state space many times in such circumstances. Hence, the Shrinking Box algorithm appears to have an advantage over the Expanding Box when the demand rate λ is small. Conversely, if λ is large, then Theorem 4.4.7 suggests that the recurrent state space S_{θ^*} will have dimensions similar to those of the set S° defined in (4.3.3). One might suppose that the Expanding Box algorithm, which begins with a state space of similar size to S°, should have an advantage over the Shrinking Box algorithm in such circumstances.

Figures 5.11, 5.12 and 5.13 show the distributions of the savings made by the Expanding Box and Shrinking Box algorithms against the RVIA (represented in the same way as in Figure 5.10), with attention restricted to particular subsets of the data collected over the 15,096 trials. Let $\rho := \lambda / \sum_{i=1}^{N} c_i \mu_i$ be a measure of the relative traffic intensity in a particular system. Figure 5.11 shows the savings made in systems with 'low' traffic intensity, in which ρ is between 0.1 and 0.5 (approximately 40% of the 15,096 systems tested satisfied this criterion). Figure 5.12 shows the savings made in systems with $0.5 < \rho < 1$ (comprising about 50% of the systems tested), and Figure 5.13 represents systems with $\rho > 1$ (comprising about 10% of systems tested).

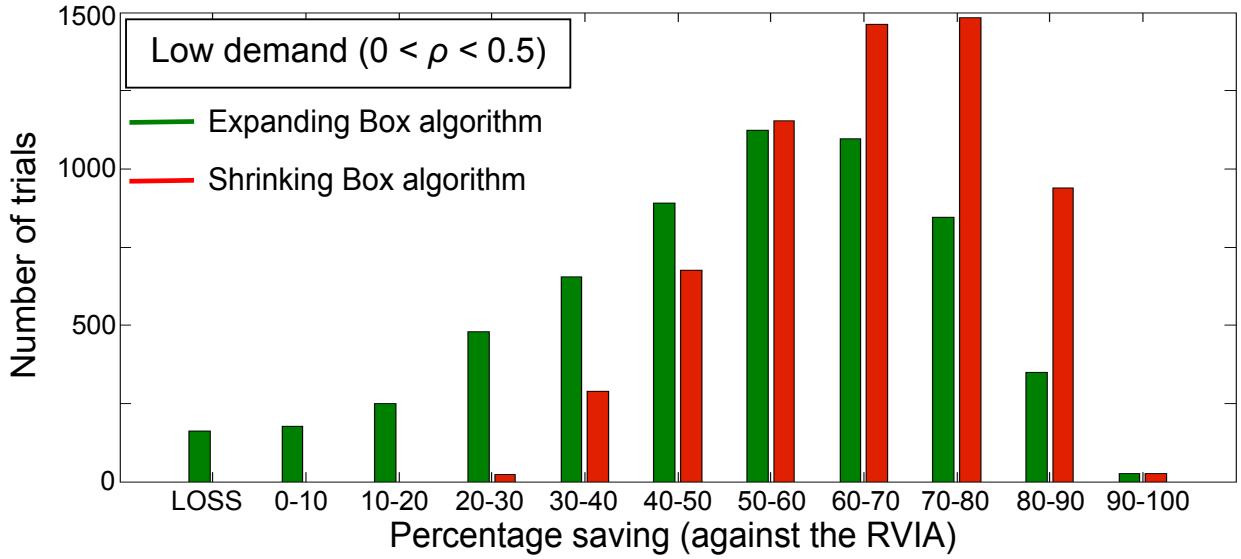


Figure 5.11: The percentage savings (in terms of the total number of value function updates required) made by the Expanding Box and Shrinking Box algorithms against the RVIA, in systems with $0.1 < \rho < 0.5$.

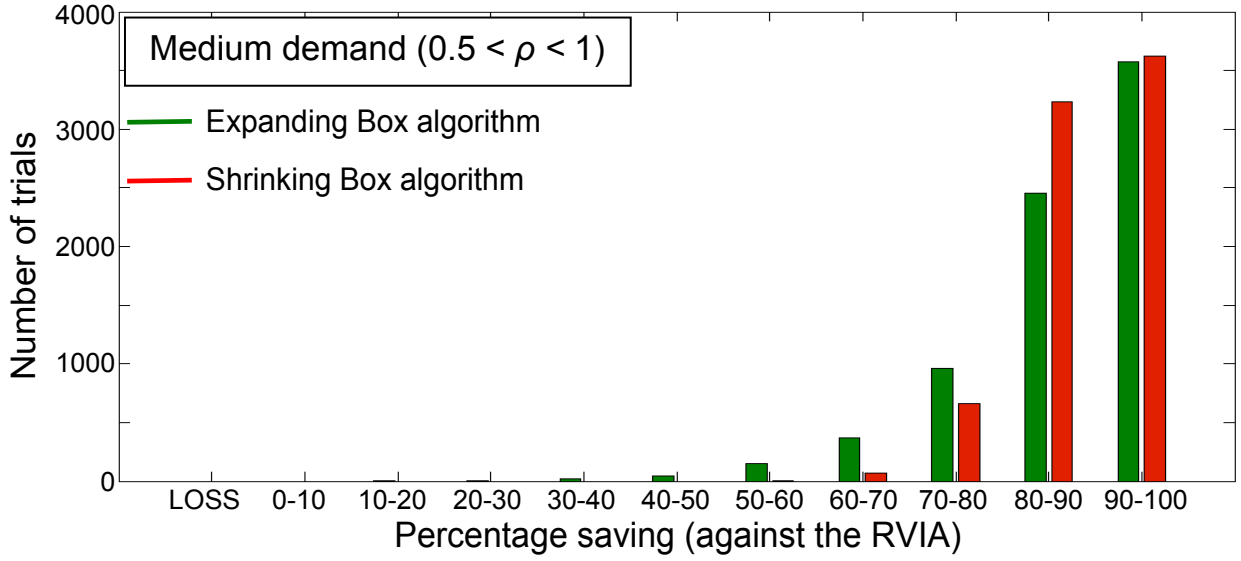


Figure 5.12: The percentage savings (in terms of the total number of value function updates required) made by the Expanding Box and Shrinking Box algorithms against the RVIA, in systems with $0.5 < \rho < 1$.

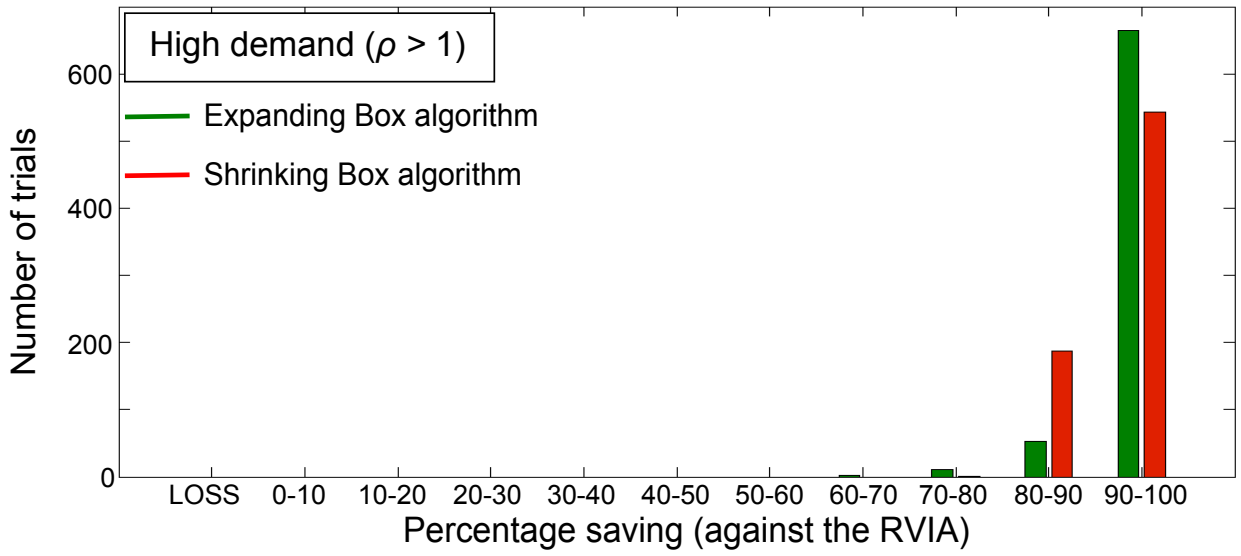


Figure 5.13: The percentage savings (in terms of the total number of value function updates required) made by the Expanding Box and Shrinking Box algorithms against the RVIA, in systems with $\rho > 1$.

Figure 5.11 suggests that, as one would expect, the Shrinking Box algorithm tends to achieve greater savings than the Expanding Box algorithm when the demand rate λ is small. Figure 5.13 shows that when λ is greater 1, both algorithms are able to achieve very large computational savings indeed against the RVIA. Indeed, when ρ is large, all three algorithms will often converge to a policy θ^* which has a recurrent state space S_{θ^*} much smaller than $\tilde{S}$. The RVIA will continue to update the value functions for $|\tilde{S}|$ states on each iteration, whereas the other two algorithms will restrict attention to small subsets of $\tilde{S}$ towards the end of their running times. Figure 5.13 indicates that, as expected, the Expanding Box algorithm tends to be more consistent than the Shrinking Box algorithm in achieving savings greater than 90% against the RVIA when $\rho > 1$.

In conclusion, the Expanding Box and Shrinking Box algorithms are both capable of finding optimal or near-optimal policies at only a fraction of the computational expense required by the conventional RVIA. The Expanding Box algorithm appears to be somewhat more reliable than the Shrinking Box algorithm in converging to policies which *exactly* correspond to the policy θ^* found by the RVIA. However, in many cases the Expanding Box algorithm may be more computationally expensive than the Shrinking Box algorithm, particularly if the demand rate λ is small. ⊠

Example 5.4.3 brings this section to an end. However, please note that Appendix A.8 is also relevant to the topics discussed in this chapter. Appendix A.8 presents various examples in order to show that average reward optimal policies may possess unexpected and counter-intuitive properties. As such, structural properties of these optimal policies are difficult to prove in general.

5.5 Conclusions

It is natural to suppose that socially optimal (i.e. average reward optimal) policies should have a logical structure, regardless of the number of facilities N or the values of the other system parameters. The *selfishly* optimal policy $\tilde{\theta}$, as defined in Section 4.1, can easily be shown to possess intuitively 'sensible' properties; for example, it is monotonic with respect to the decisions made at the various states, and the selfish state space $\tilde{S}$ is always 'cuboid' in shape. The results in Sections 5.1, 5.2 and 5.3 have shown that if one restricts attention to small values of N (e.g. $N = 1$ or $N = 2$), or makes additional restrictive assumptions about the system parameters (such as the assumption of homogeneous facilities in Section 5.3), it is sometimes possible to prove structural

properties of the socially optimal policy θ^* obtained using relative value iteration.

There are certain properties of socially optimal policies that appear to hold in *most* cases, but not necessarily all; for example, the Shrinking Box and Expanding Box algorithms discussed in Section 5.4 rely on the assumption of a monotonic progression of the finite-stage optimal policies obtained during relative value iteration. When these algorithms are applied to *single-facility* systems, they are both guaranteed to converge to an average reward optimal policy due to the result of Lemma 5.1.5. However, Example A.8.3 has shown that the same guarantee is not possible in systems with $N \geq 2$. Moreover, the counter-examples in Appendix A.8 show that several properties which might (if they were true) simplify the characterisation of the optimal policy θ^* do not hold in general. Indeed, counter-examples can frequently be found even in systems where N is small.

In conclusion, if one considers a general N-facility system without any restrictions on the system parameters, it may not be possible to prove any properties of the RVIA policy θ^* beyond those which were already proved in Chapter 4 (see, for example, Theorems 4.2.4 and 4.3.3). This illustrates the challenging nature of average reward optimisation in general. The next two chapters of this thesis will investigate methods for finding *near-optimal* policies in systems where the size of the finite state space $\tilde{S}$ prevents the efficient computation of an average reward optimal policy.

6 Heuristic policies

A fundamental concept in this thesis thus far has been that of a *socially optimal policy*, which routes customers to service facilities in such a way that the expected long-run average reward earned by the system is maximised. Section 3.7 introduced methods for computing socially optimal policies in systems with a finite number of states. Section 4.2 established that a socially optimal policy can always be found by searching within the finite set of states $\tilde{S}$, where $\tilde{S}$ is the *selfishly* optimal state space defined in Section 4.1. This result implies that, from a purely theoretical point of view, it is always possible to find a socially optimal policy for any N-facility queueing system of the type described in Section 3.1, by simply applying a suitable dynamic programming algorithm. One might suggest that, in view of the results in earlier chapters, the mathematical problem of finding a socially optimal policy for any given set of system parameters has been 'solved'.

Unfortunately, to make such a suggestion would be to ignore practical considerations. Specifically, the amount of time required by a dynamic programming algorithm to converge to an optimal policy may be prohibitive. Even when the number of facilities N is relatively small (for example, less than 10), the selfishly optimal state space $\tilde{S}$ may comprise billions or trillions of states. Even a state-of-the-art computer may be unable to carry out the task of finding an optimal policy using the RVIA or PIA (see pages 101 and 109) in a reasonable amount of time; in fact, since these algorithms require a unique value $h(\mathbf{x})$ to be stored for every state $\mathbf{x} \in \tilde{S}$, memory constraints alone may cause a computer program to break down, before it has been able to complete its first 'sweep' of the state space $\tilde{S}$. For larger values of N (e.g. $N \geq 100$), the reality is that the amount of time required to find an optimal policy using a DP algorithm would reach astronomical proportions.

The "curse of dimensionality" in dynamic programming has been discussed extensively by Powell [140]. In fact, Powell argues that there are three different curses of dimensionality which hinder DP algorithms. Arguably the most obvious problem, as alluded to in the previous paragraph, is that of an extremely large state space. The other two problems considered by Powell are somewhat more relevant in a more general MDP context; these are related to the action sets $A_{\mathbf{x}}$ available at the various states $\mathbf{x} \in \tilde{S}$, and the number of possible outcomes that may arise from choosing a particular action $a \in A_{\mathbf{x}}$ at a state $\mathbf{x} \in \tilde{S}$. In the MDP Υ formulated in Section 3.5, the number of actions permissible at any state $\mathbf{x} \in \tilde{S}$ (excluding actions which might cause a transition to a state

$\mathbf{y} \notin \tilde{S}$) is always bounded above by $N+1$, and the number of states that can be 'reached' from any state $\mathbf{x} \in \tilde{S}$ in a single discrete-time transition is at most $N+2$ (see Figure 3.5); of course, generalisations of the problem are possible (such as the incorporation of heterogeneous customers discussed in Section 4.5), but without considering such extensions, it is clear that the dimensionality of the state space will be the main obstacle as far as DP algorithms are concerned.

The counter-examples in Appendix A.8 show that, in general, socially optimal policies may not possess intuitively logical properties such as monotonicity, even when attention is restricted to optimal policies found by DP algorithms such as the RVIA. Consequently, it would seem that there are no 'shortcuts' available which would enable these policies to be found without exhaustive computation. The 'Shrinking Box' and 'Expanding Box' algorithms presented in Section 5.4 offer the possibility of tremendous time savings in comparison to the conventional RVIA, but these algorithms still suffer (albeit to a lesser extent) from some of the same problems as the RVIA, such as the requirement for an array of values $h(\mathbf{x})$ to be stored in the system memory; furthermore, there is no theoretical guarantee that the policies found by these algorithms will be optimal (although their extremely strong performance in systems with $N \leq 5$ has been demonstrated).

Unfortunately, there is no alternative but to acknowledge that, from a practical point of view, the computation of socially optimal policies in systems where $\tilde{S}$ is very large is infeasible. This chapter marks a change in emphasis from previous chapters in that the objective is now to find, and analyse the properties of, *approximately* optimal policies in systems which are assumed to be too large (in the sense that $\tilde{S}$ is too large) for DP algorithms to be applied. This will involve devising *heuristic policies*, and testing their performance. Roughly speaking, a heuristic policy is a *near-optimal* policy which can be determined in a 'reasonable' amount of time and implemented easily in practice. In general, the main objectives of a heuristic policy should be:

1. To generate an expected long-run average reward for the system which (at worst) is only a small distance away from the average reward earned by an optimal policy;

2. To require a relatively small amount of computational effort in its execution, regardless of the number of facilities N and the size of the selfish state space $\tilde{S}$.

An inherent difficulty lies in the fact that the success (or otherwise) of a particular heuristic approach depends on whether or not the policy obtained is 'near-optimal', and this may not be easy

to determine. Ideally, one would wish to evaluate the expected average reward g_θ earned by the heuristic policy θ in question and compare this with the optimal value g^*. However, evaluating g_θ exactly requires a set of values $h(\mathbf{x})$ (for $\mathbf{x} \in \tilde{S}$) to be found which satisfy the policy evaluation equations (3.7.5). If $\tilde{S}$ is large, then this task may be infeasible for the same reasons that DP algorithms are infeasible. This problem may be circumvented to some extent by using discrete-event simulation to estimate the value of g_θ; if the simulation program is well-designed and allowed to run for a moderate amount of time, then it should be able to produce a robust estimate of the average reward earned by the fixed policy θ (this is discussed further in Chapter 7). However, even in this case, the optimal value g^* will generally be unknown. An approach used in this chapter will be to show that the heuristic policies under consideration perform strongly in moderate-sized systems for which it is possible to evaluate g^* exactly using dynamic programming.

It is worthwhile to note that the selfish policy $\tilde{\theta}$ may itself be regarded as a heuristic policy, since it implements a simple decision rule which always directs customers to take the action which maximises their expected individual net reward. The selfish policy may therefore be interpreted as a *greedy heuristic*, since only the welfare of the latest customer to arrive is taken into account when making a routing decision, without any consideration of future consequences. Clearly, $\tilde{\theta}$ is a simple policy to implement in any system, regardless of the number of facilities N and the size of the state space $\tilde{S}$; in this respect it qualifies as a heuristic policy, although the examples from earlier chapters indicate that its performance may be poor (especially if the demand rate λ is large). It would obviously be desirable to find more sophisticated heuristic policies which are capable of outperforming the selfish policy $\tilde{\theta}$ in terms of average reward. Two heuristic policies will be considered in this chapter, both of which are based on approaches found in the literature.

Section 6.1 will describe the *Whittle index heuristic*, which has its origins in the work of Gittins [55] and Whittle [198] on deriving 'dynamic allocation indices' for multi-armed bandit processes. Section 6.2 will present an alternative heuristic approach, which involves applying a single step of policy improvement to a *static routing* policy; the development in this section will be similar to that in [6]. Results from numerical experiments will be presented in Section 6.3.

6.1 Whittle index heuristic

The first of the two heuristic methods to be considered in this chapter is motivated by the work of Glazebrook et al. [59], who demonstrated the strong performance of an index-based heuristic in a problem involving routing to parallel queues (see also Argon et al. [6]). Informally speaking, an *index-based* policy is a policy which associates a certain, easily-computable score or *index* to the various possible decision options in any given system state, and then chooses the option with the highest index. A precise definition of *indexability* will be given later in this section.

In [59], the authors consider a problem which differs somewhat from the assumptions made in this thesis; most importantly, customers are *impatient* and may leave the system before receiving service (after having initially been 'routed' to one of the N service facilities), in which case the system incurs a non-negative penalty. Balking also incurs a penalty, but holding costs are not included explicitly. In addition, the authors allow for state-dependent service rates by using $\mu_{i,n}$ to represent the rate at which service completions occur at facility $i \in \{1, 2, ..., N\}$ when there are $n \in \mathbb{N}_0$ customers present there; so, for example, this rate would be given by $\mu_{i,n} = \min(n, c_i)\mu_i$ under the assumptions made in Section 3.1. On the other hand, the system in [59] is comparable to that described in Section 3.1 with respect to its formulation as a continuous-time Markov process, the complete observability of the system at all times and the fact that the optimisation problem of interest involves routing customers to parallel service facilities in such a way that the expected long-run average return (after deduction of balking and reneging costs) is maximised.

Throughout this chapter, the queueing system described in Section 3.1 will be considered. The objectives in this section are to show empirically that a heuristic policy derived in an analogous way to that in [59] is able to attain an expected long-run average reward close to the optimal value in many problem instances, and to analyse the properties of this heuristic policy. In particular, since the heuristic policy will be index-based, it will be interesting to compare the indices that it uses for decision-making with those of the *selfishly* optimal policy $\tilde{\theta}$ defined in Section 4.1, which may also be regarded as an index policy. The resulting observations may offer some insight into why the selfish policy $\tilde{\theta}$ is generally sub-optimal, and how its sub-optimality may be affected by the values of the system parameters. It will also be useful to verify that the heuristic policy retains certain structural properties which were discussed in Chapter 5, such as monotonicity.

In order to derive the heuristic policy, it will be useful to introduce some notation. Let Θ denote the class of *stationary* policies under which the system is stable and has a stationary distribution; that is, if $\theta \in \Theta$ then the distribution $\{\pi_\theta(\mathbf{x})\}_{\mathbf{x}\in S}$ exists, where $\pi_\theta(\mathbf{x})$ is the steady-state probability of the system being in state $\mathbf{x} \in S$ under θ and $\sum_{\mathbf{x}\in S} \pi_\theta(\mathbf{x}) = 1$. Clearly Θ is non-empty, since it includes the trivial policy which chooses to balk at all states in S. For each policy $\theta \in \Theta$ and facility $i \in \{1, 2, ..., N\}$, let $\eta_i(\theta)$ denote the *effective* queue-joining rate per unit time at facility i under θ (i.e. the long-run average number of customers joining facility i per unit time), and let $L_i(\theta)$ denote the long-run expected number of customers present at facility i under θ. Then the expected long-run average reward g_θ under policy θ may be expressed in the form:

$$g_\theta = \sum_{i=1}^{N} \left(\alpha_i \eta_i(\theta) - \beta_i L_i(\theta)\right). \tag{6.1.1}$$

Naturally, closed-form expressions for $\eta_i(\theta)$ and $L_i(\theta)$ are unattainable in general when $N \geq 2$, but (6.1.1) will nevertheless prove to be useful since the heuristic policy to be discussed in this section is derived by using a set of N independent *single*-facility admission control problems to approximate a solution to the original N-facility routing problem. This heuristic policy will be referred to as the *Whittle index policy* (or, more simply, the 'Whittle heuristic') since it was first proposed, in a rather more general context, by Whittle [198]. In order to motivate the derivation of the Whittle heuristic, note first that under any policy $\theta \in \Theta$, the sum of the effective queue-joining rates $\eta_i(\theta)$ at the various facilities must be bounded above by the system demand rate. That is:

$$\sum_{i=1}^{N} \eta_i(\theta) \leq \lambda. \tag{6.1.2}$$

It will be convenient to assume that the system is *uniformised* as described in Section 3.3, so that it evolves in discrete time steps of size Δ, where $\Delta \in \left(0, \left(\lambda + \sum_{i=1}^{N} c_i\mu_i\right)^{-1}\right]$. In the discretised system, the number of arrivals occurring at any discrete time step is at most one, and hence it is not possible for two or more facilities to receive a new customer at the same time step. Let U_n denote the number of facilities chosen to receive a new customer at time step $n \in \mathbb{N}_0$, irrespective of whether or not a new customer actually arrives at step n. Then, obviously:

$$U_n \leq 1 \quad \forall\, n \in \mathbb{N}_0. \tag{6.1.3}$$

Following in the spirit of Whittle [198], suppose that the mechanism for allocating customers to

service facilities is altered so that (6.1.3) is replaced by a *weaker* constraint:

$$E\left[U_n\right] \leq 1, \tag{6.1.4}$$

where the expectation is with respect to the long-term average; that is, the average number of facilities chosen to receive a new customer (over all time steps) should be at most 1. Clearly, the requirement (6.1.4) would be met not only by all policies in Θ, but also by some policies with the ability to choose several different destinations *for the same customer*. Consider a relaxed version of the original N-facility optimisation problem involving an expanded class of stationary policies Θ' which are at liberty to 'break' the natural physical restrictions of the system, in the sense that any new customer who arrives can be sent to any *subset* of the set of facilities $\{1, 2, ..., N\}$. That is, for each state $\mathbf{x} \in S$, the action $\theta(\mathbf{x})$ chosen by a policy $\theta \in \Theta'$ satisfies:

$$\theta(\mathbf{x}) \in \mathcal{P}\left(\{1, 2, ..., N\}\right),$$

where $\mathcal{P}\left(\{1, 2, ..., N\}\right)$ is the *power set* (i.e. the set of all subsets, including the empty set) of $\{1, 2, ..., N\}$. Conceptually, one now considers a new optimisation problem in which the option is available to produce 'copies' of each customer who arrives, and send these copies to any number of facilities (at most one copy per facility). For each state $\mathbf{x} \in S$, $\theta(\mathbf{x})$ is the set of facilities which, under the policy $\theta \in \Theta'$, receive (a copy of) a new customer if an arrival occurs under state $\mathbf{x}$. If $\theta(\mathbf{x}) = \emptyset$ then no facility receives a new customer; this is akin to balking in the original problem. Suppose also that the constraint (6.1.2) is imposed on the new optimisation problem. Following [59] and [198], this constraint may be incorporated in a Lagrangian fashion by letting $g^{\dagger}(W)$ denote the optimal expected long-run average reward for the new problem, defined as:

$$g^{\dagger}(W) := \sup_{\theta \in \Theta'} \left(\sum_{i=1}^{N} \left(\alpha_i \eta_i(\theta) - \beta_i L_i(\theta)\right) + W\left(\lambda - \sum_{i=1}^{N} \eta_i(\theta)\right)\right), \tag{6.1.5}$$

where $W \in \mathbb{R}$ is a Lagrange multiplier which, as discussed later, may be interpreted economically as a *charge* for granting entry to a new customer. Clearly, any policy θ belonging to the class of policies Θ for the original problem may be represented by a policy θ' in the new class Θ' for which the cardinality of $\theta'(\mathbf{x})$ is either 1 or 0 at all states $\mathbf{x} \in S$. Hence, for $W \geq 0$:

$$g^* \leq g^{\dagger}(W),$$

where $g^* = \sup_{\theta \in \Theta} g_\theta$ is the optimal expected long-run average reward in the original problem. At this point, it will be useful to re-write (6.1.5) in the equivalent form:

$$g^\dagger(W) = \sup_{\theta \in \Theta'} \left(\sum_{i=1}^{N} \Big((\alpha_i - W)\, \eta_i(\theta) - \beta_i L_i(\theta) \Big) \right) + \lambda W. \tag{6.1.6}$$

Then, as in [59] (p. 979), one obtains a *facility-wise decomposition*:

$$g^\dagger(W) = \sum_{i=1}^{N} g_i^\dagger(W) + \lambda W, \tag{6.1.7}$$

where, for each facility $i \in \{1, 2, ..., N\}$:

$$g_i^\dagger(W) = \sup_{\theta \in \Theta_i{}'} \Big((\alpha_i - W)\, \eta_i(\theta) - \beta_i L_i(\theta) \Big).$$

Here, $\Theta_i{}'$ (for $i = 1, 2, ..., N$) is a class of stationary policies which choose either to accept a customer (denoted by 1) or reject (denoted by 0) at any given state. To clarify, recall that in the relaxed version of the N-facility problem under consideration, each newly-arrived customer can be sent to any subset of the N facilities. As such, the decision of whether or not to admit a customer at some facility $i \in \{1, 2, ..., N\}$ can be made independently of the decisions made in regard to the other facilities $j \neq i$. It follows that an optimal solution to the relaxed N-facility problem can be found by solving N independent *single-facility* problems. For each facility $i \in \{1, 2, ..., N\}$, the corresponding single-facility problem involves customers arriving according to a Poisson process with a demand rate λ (the same demand rate as for the N-facility problem), c_i service channels, and exponentially-distributed service times with mean μ_i^{-1}. The holding cost is β_i per customer per unit time, but importantly the reward for service is now $\alpha_i - W$ as opposed to α_i. Hence, it is natural to interpret W as an extra charge for admitting a customer; see Figure 6.1.

The single-facility problem described above exactly fits the formulation of Section 3.1 (with $N = 1$), except that the reward $\alpha_i - W$ may not be positive. However, if $\alpha_i - W \leq 0$ then there is no way for the system to earn a positive expected average reward, and hence the stationary policy which chooses to balk at every state (thus achieving an average reward of zero) trivially attains average reward optimality. On the other hand, assuming that $\alpha_i - W > 0$, results from previous chapters imply that for the single-facility optimisation problem under consideration:

- There exists an average reward optimal stationary policy (Theorem 4.2.3);

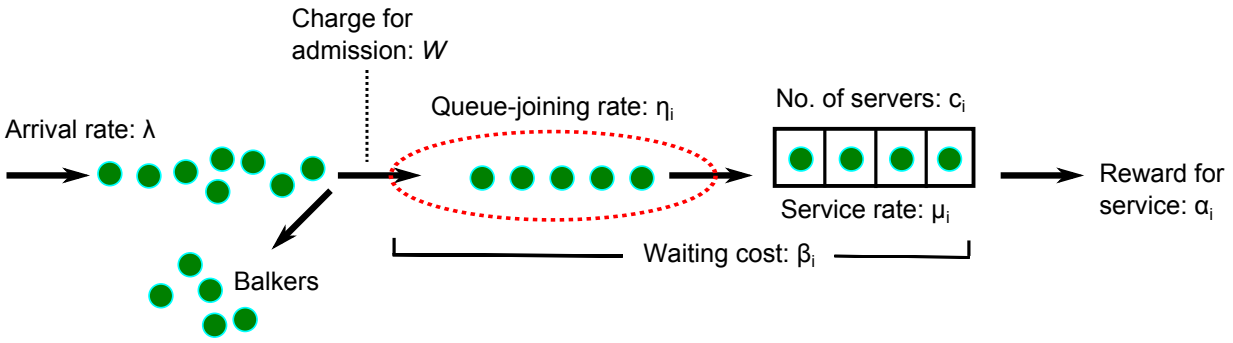


Figure 6.1: An $M/M/c_i$ queue with an extra admission charge W.

- There exists an average reward optimal *threshold* policy (Theorem 5.1.4).

Accordingly, let θ_i^* denote an optimal threshold policy. This means that θ_i^* is a map from $\mathbb{N}_0$ to $\{0,1\}$, where $\mathbb{N}_0$ is the state space for the single-facility problem and $\{0,1\}$ is the set of actions available at any state. Re-interpreting the summary measures $\eta_i(\cdot)$ and $L_i(\cdot)$ so that they are now functions of policies θ_i belonging to the set Θ_i' associated with the *single-facility* problem, it follows that $(\alpha_i - W)\eta_i(\theta_i^*) - \beta_i L_i(\theta_i^*)$ is a valid expression for $g_i^\dagger(W)$, the optimal average reward for facility i. Now, recall that the facility $i \in \{1,2,...,N\}$ was arbitrary in this discussion and let $\theta_1^*, \theta_2^*, ..., \theta_N^*$ be optimal threshold policies at the various facilities. Also, let θ^* be a stationary policy belonging to the expanded class Θ' which operates in such a way that, for each state $\mathbf{x} \in S$:

$$\theta^*(\mathbf{x}) = \big\{ i \in \{1,2,...,N\} : \theta_i^*(x_i) = 1 \big\}. \tag{6.1.8}$$

That is, each time a new customer arrives, they are sent to *all* of the facilities $i \in \{1,2,...,N\}$ at which the optimal threshold policy θ_i^* (taking into account only the number of customers x_i at facility i) would choose to accept a customer. By the previous arguments, the expected long-run average reward earned by operating policy θ^* in the relaxed N-facility system is maximised by operating the optimal threshold policies θ_i^* at the individual facilities i, and therefore it follows that θ^* attains average reward optimality in the relaxed version of the problem.

One may conclude from the above that the task of finding an optimal routing policy in the relaxed version of the original N-facility problem is not a difficult one, since it can be accomplished by finding optimal admission control policies for the N facilities individually. More to the point, this solution method avoids the need to trawl through a vast N-dimensional state space, and thereby

avoids suffering from the 'curse of dimensionality'. In order to derive the Whittle heuristic for the *original* N-facility problem, it remains to establish the connection between this heuristic and the optimal solutions for the relaxed version of the problem discussed thus far. The Whittle heuristic relies upon the notion of *indexability* of a service facility. It will therefore be appropriate to provide a definition of indexability, similar to the definition in [198] (see also [57, 58, 59]).

Definition 6.1.1. ***(Indexability)***

For each facility $i \in \{1, 2, ..., N\}$, let $\theta_i^*(W)$ be the optimal threshold policy found by relative value iteration in a single-facility problem with demand rate λ, c_i service channels, service rate μ_i, holding cost β_i, reward for service $\alpha_i - W$ and finite state space $\{0, 1, ..., \tilde{T}_i(W)\}$, where:

$$\tilde{T}_i(W) = \max\left(\left\lfloor \frac{(\alpha_i - W)c_i\mu_i}{\beta_i} \right\rfloor, 0\right).$$

It is assumed that, in cases where the actions $a = 0$ and $a = 1$ both attain the maximum in the optimality equations (5.1.2) for some state $x \in \{0, 1, ..., \tilde{T}_i(W)\}$, the policy $\theta_i^*(W)$ chooses to join at state x. Also define $T_i^*(W)$ as the threshold associated with $\theta_i^*(W)$. That is:

$$T_i^*(W) = \min\left\{x \in \{0, 1, ..., \tilde{T}_i(W)\} : \theta_i^*(W) \text{ chooses to balk at state } x\right\}. \tag{6.1.9}$$

Then facility i is said to be indexable *if $T_i^*(W)$ satisfies the following properties:*

1. *$T_i^*(W)$ is monotonically decreasing with W. That is, for $W_1, W_2 \in \mathbb{R}$:*

$$W_1 > W_2 \;\Rightarrow\; T_i^*(W_1) \leq T_i^*(W_2).$$

2. *For any $x \in \mathbb{N}_0$, there exists $W_i(x) \in \mathbb{R}$ such that $T_i^*(W) > x$ if and only if $W \leq W_i(x)$.*

Defining each policy $\theta_i^*(W)$ as the optimal policy found by relative value iteration is convenient in order to avoid any ambiguity that would be caused by the non-uniqueness of optimal policies. Note that the approach of finding $\theta_i^*(W)$ by searching within the *finite* (selfishly optimal) state space $\{0, 1, ..., \tilde{T}_i(W)\}$ is justified by the result of Theorem 4.2.4. The next result establishes that the service facilities $i \in \{1, 2, ..., N\}$ satisfy the conditions in Definition 6.1.1.

Theorem 6.1.2. *Each facility $i \in \{1, 2, ..., N\}$ is indexable.*

Proof. The proof follows arguments similar to those found in [59]. Consider an arbitrary facility $i \in \{1, 2, ..., N\}$, and let $g_i(T, W)$ denote the expected long-run average reward under a threshold policy with threshold $T \in \mathbb{N}_0$ and admission charge $W \in \mathbb{R}$. Then:

$$g_i(T, W) = \lambda(1 - \pi_i(T, T))(\alpha_i - W) - \beta_i \sum_{y=0}^{T} y\pi_i(y, T), \tag{6.1.10}$$

where $\pi_i(y, T)$ denotes the steady-state probability of the facility being in state $y \in \mathbb{N}_0$, given that a threshold of T is applied. From (6.1.10) it follows that, given any fixed $T \in \mathbb{N}_0$, the average reward $g_i(T, W)$ is a linear, strictly decreasing function of W, and its gradient is:

$$\frac{\partial g_i}{\partial W} = -\lambda(1 - \pi_i(T, T)). \tag{6.1.11}$$

Given some admission charge $W \in \mathbb{R}$, let $T_i^*(W)$ be the optimal threshold defined in (6.1.9); hence, $g_i(T_i^*(W), W) \geq g_i(T, W)$ for all $T \in \mathbb{N}_0$. It can be verified using standard formulae for finite-capacity $M/M/c$ queues (see [67], p. 74) that the steady-state probability $\pi_i(T, T)$ is strictly decreasing with T; hence, the gradient in (6.1.11) is also strictly decreasing with T. Given that $g_i(T_i^*(W), W) \geq g_i(T_i^*(W) + n, W)$ for arbitrary $n \geq 1$, it must therefore be the case that $g_i(T_i^*(W), \tilde{W}) > g_i(T_i^*(W) + n, \tilde{W})$ for any $\tilde{W} > W$, and therefore the policy with threshold $T + n$ cannot be the optimal policy found by relative value iteration under an admission charge $\tilde{W}$, since it does not maximise the average reward. It follows that $T_i^*(\tilde{W}) \leq T_i^*(W)$ for any two admission charges $W, \tilde{W} \in \mathbb{R}$ with $W < \tilde{W}$, which verifies the first of the required conditions.

Using similar arguments, for any state $x \in \mathbb{N}$ there must exist some value $W_i(x) \in \mathbb{R}$ such that $g_i(x, W) \geq g_i(x - n, W)$ for all $n \in \{1, 2, ..., x\}$ if and only if $W \leq W_i(x)$. This is due to the fact that the linear functions $g_i(x - n, W)$ have larger gradients than that of $g_i(x, W)$. Hence, it must be the case that $T_i^*(W) > x$ if and only if $W \leq W_i(x)$, since this is the only scenario in which a threshold policy with threshold greater than x maximises the average reward. This completes the proof that the indexability conditions stated in Definition 6.1.1 are both satisfied. □

Given that the service facilities are indexable, it follows that for each facility $i \in \{1, 2, ..., N\}$ and integer $x \in \mathbb{N}_0$ there exists some value $W_i(x)$ such that joining is chosen by the optimal threshold policy $\theta_i^*(W)$ if and only if $W \leq W_i(x)$. The critical value $W_i(x)$ will be referred to in this section as the *Whittle index* for facility i and state x. The next definition states this formally.

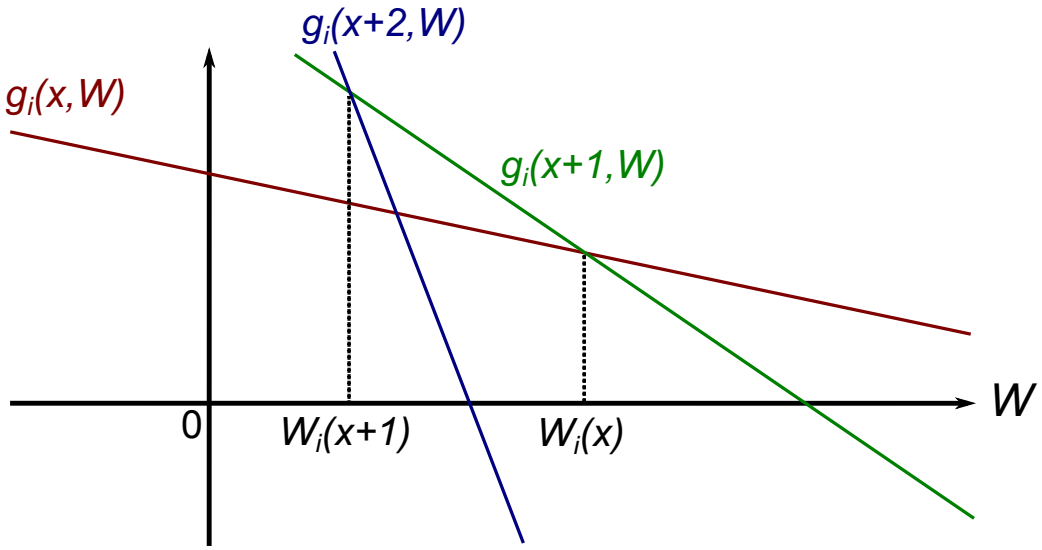


Figure 6.2: The linear dependence of the functions $g_i(x, W)$ on the admission charge W.

Definition 6.1.3. ***(Whittle index)***

Suppose facility i is indexable. The Whittle index *for facility i and state $x \in \mathbb{N}_0$ is given by:*

$$W_i(x) := \sup\big\{W \in \mathbb{R} : \theta_i^*(W) \text{ chooses to join at state } x\big\}, \tag{6.1.12}$$

where $\theta_i^*(W)$ is the optimal policy for facility i described in Definition 6.1.1.

As discussed previously, the relaxed version of the N-facility routing problem is solved by a policy θ^* which, upon any new customer's arrival, sends the customer *only* to facilities i at which the number of customers present x_i is smaller than the threshold $T_i^*(W)$ associated with the optimal policy $\theta_i^*(W)$ found by applying relative value iteration to a single-facility problem involving only facility i. Clearly, in view of Definition 6.1.3, this is equivalent to sending the new customer to all facilities i for which $W_i(x_i) \geq W$, but *not* to any facility j for which $W_j(x_j) < W$. In fact, one might replace the description of the policy θ^* given in (6.1.8) with the following:

$$\theta^*(\mathbf{x}) = \big\{i \in \{1, 2, ..., N\} : W_i(x_i) \geq W\big\}.$$

As observed by Glazebrook et al. [59] and Argon et al. [6], the fact that an optimal solution to the relaxed problem may be described using the Whittle indices makes it logical to propose a *heuristic* policy for the original N-facility problem, which involves sending any new customer who arrives

under state $\mathbf{x} \in S$ to a facility i which maximises $W_i(x_i)$, or choosing to balk if all of the $W_i(x_i)$ values are negative. The optimality of such a policy cannot be theoretically guaranteed, but its intuitive justification lies in the fact that $W_i(x_i)$, when positive, is a measure of the amount by which the 'charge for admission' W would need to be increased before the optimal policy θ^* for the *relaxed* problem would choose not to admit a customer to facility i. Thus, $W_i(x_i)$ may be regarded somewhat crudely as a measure of the margin by which one would be 'in favour' of having an extra customer present at facility i. A similar interpretation, as discussed in [59] and [6], is that $W_i(x_i)$ is a 'fair charge' for admitting a customer to facility i when there are x_i customers already present; thus, the decision made by θ^* concerning whether or not to admit a new customer to facility i depends on whether or not the 'real' charge W is in excess of the fair charge $W_i(x_i)$.

As stated at the beginning of this section, the general mode of operation of an *index-based* policy involves choosing actions by associating indices (scores) with all of the possible decision options at any given state, and then choosing the action with the largest score. Moreover, these indices should be *easily computable*. If calculation of the indices were to require a time-consuming algorithmic procedure, then obviously the index-based policy in question would not be as easy to implement in practice as one might desire. In the case of the N-facility queueing system under consideration, the objective is to find a policy which, while being only *near*-optimal, is also much faster to obtain than the optimal policy that one would find using a dynamic programming algorithm. In order to construct a heuristic policy based on the Whittle indices $W_i(x_i)$, it is therefore highly important to ensure that the indices $W_i(x_i)$ themselves are relatively simple to calculate.

Fortunately, the fact that the indices $W_i(x_i)$ can be easily calculated for any facility $i \in \{1, 2, ..., N\}$ is ensured by their independence from the indices for the other facilities $j \neq i$. In other words, one may derive the index $W_i(x_i)$ for any facility i and head count x_i by considering a single-facility problem involving only facility i. Moreover, it is possible to obtain a convenient formula for $W_i(x_i)$. This can be shown using the arguments below, which make use of Theorem 6.1.2.

Consider an arbitrary $W_0 \in \mathbb{R}$ and let $T_i^*(W_0)$ be the corresponding optimal threshold. Recall from the proof of Theorem 6.1.2 that the average rewards $g_i(T_i^*(W_0), W)$ and $g_i(T_i^*(W_0) + 1, W)$ are both linear functions of W and their gradients are not equal. Hence, there must exist a unique intersection point $W_1 < W_0$ such that $g_i(T_i^*(W_0), W_1) = g_i(T_i^*(W_0) + 1, W_1)$ (refer to Figure

6.2). Moreover, since the optimal threshold $T_i^*(W)$ is monotonically decreasing with W (and it is assumed that higher-threshold policies are preferred when average rewards are tied), it must be the case that $T_i^*(W_1) > T_i^*(W_0)$. Hence, given the unique intersection point W_1 satisfying $g_i(T_i^*(W_0), W_1) = g_i(T_i^*(W_0) + 1, W_1)$, one can say that the threshold $T_i^*(W)$ is greater than $T_i^*(W_0)$ if and only if $W \leq W_1$. Hence, W_1 is equal to the *Whittle index* $W_i^*(T_i^*(W_0))$, since joining is chosen at state $T_i^*(W_0)$ by the optimal threshold policy if and only if $W \leq W_1$.

To summarise the preceding arguments, the Whittle index $W_i(x)$ for any facility i and head count $x \in \mathbb{N}_0$ may be obtained as the unique value of W which results in the thresholds $T = x$ and $T = x + 1$ both yielding the same expected long-run average reward $g_i(T, W)$ in a single-facility problem. Equivalently, one might say that an admission charge of $W_i(x)$ results in the optimal policy $\theta_i^*(W_i(x))$ being *indifferent* between admitting and rejecting a customer at facility i when the head count is x. Recalling (6.1.10), it follows that $W_i(x)$ satisfies the equation:

$$\begin{aligned}&\lambda(1 - \pi_i(x, x))(\alpha_i - W_i(x)) - \beta_i \sum_{y=0}^{x} y\pi_i(y, x) \\ &= \lambda(1 - \pi_i(x + 1, x + 1))(\alpha_i - W_i(x)) - \beta_i \sum_{y=0}^{x+1} y\pi_i(y, x + 1).\end{aligned}$$

Solving this equation directly for $W_i(x)$ yields:

$$W_i(x) = \alpha_i - \frac{\beta_i \left(\sum_{y=0}^{x+1} y\pi_i(y, x + 1) - \sum_{y=0}^{x} y\pi_i(y, x) \right)}{\lambda \left(\pi_i(x, x) - \pi_i(x + 1, x + 1) \right)}. \tag{6.1.13}$$

Given that relatively simply formulae are available for the steady-state probabilities $\pi_i(y, x)$, (6.1.13) provides the required convenient formula for the Whittle indices $W_i(x)$. The *Whittle index heuristic policy* $\theta^{[W]}$ (hereafter referred to as the *Whittle policy*) is defined below.

Definition 6.1.4. ***(Whittle index policy)***

For all $\mathbf{x} \in S$, *the* Whittle index policy $\theta^{[W]}$ *chooses an action as follows:*

$$\theta^{[W]}(\mathbf{x}) \in \begin{cases} \underset{i \in \{1,2,...,N\}}{\arg\max}\ W_i(x_i), & \text{if } \exists\ i \in \{1, 2, ..., N\} \text{ such that } W_i(x_i) \geq 0, \\ \{0\}, & \text{otherwise,} \end{cases} \tag{6.1.14}$$

where, for all facilities $i \in \{1, 2, ..., N\}$ and integers $x \geq 0$:

$$W_i(x) = \alpha_i - \frac{\beta_i \left(\sum_{y=0}^{x+1} y\pi_i(y, x+1) - \sum_{y=0}^{x} y\pi_i(y, x) \right)}{\lambda \left(\pi_i(x, x) - \pi_i(x+1, x+1) \right)}.$$

In cases where two or more facilities attain the maximum in (6.1.14), it will be assumed that a decision is made according to some fixed ranking order of the N facilities.

As discussed earlier, it is possible (in theory) to evaluate the average reward $g_{\theta^{[W]}}$ earned by the Whittle policy using the Policy Evaluation Algorithm in Section 3.7. However, this requires the use of dynamic programming, which may be impractical. A much quicker method for estimating $g_{\theta^{[W]}}$ involves *simulating* the system operating under the Whittle policy. In doing so, it is possible to make use of the 'containment' principle proved in Section 4.2; that is, due to Theorem 4.2.4, any N-facility system may be associated with a socially optimal policy whose recurrent state space is contained within the selfishly optimal state space $\tilde{S}$ defined in Section 4.1. As a result, one may infer that only the Whittle indices at states $\mathbf{x} \in \tilde{S}$ should be required, and therefore it is efficient to compute the indices $W_i(x_i)$ for all $i \in \{1, 2, ..., N\}$ and $x_i \in \{0, 1, ..., \tilde{B}_i\}$ (where $\tilde{B}_i$ is the selfish threshold defined in (4.1.2)) in the initial phase of the simulation program and store these indices inside an array, so that they can be 'looked up' during the simulation phase itself as and when required. Note that the size of the array used for this purpose would be only $\sum_i (\tilde{B}_i + 1)$, which in general will be much smaller than $\left|\tilde{S}\right|$. In fact, this pre-computation approach is theoretically justified, since it can easily be shown that one must have $W_i(x_i) < 0$ for any state $\mathbf{x} \in \tilde{S}$ with $x_i = \tilde{B}_i$ (this is confirmed by Theorem 6.1.7 later in this section). This implies that an index-based policy based on the Whittle indices will never choose to join facility i at a state $\mathbf{x} \in S$ with $x_i = \tilde{B}_i$, and thus the recurrent state space of the Whittle policy must be contained in $\tilde{S}$.

As a further note, the fact that balking is chosen by $\theta^{[W]}$ at any state $\mathbf{x} \in S$ with $W_i(x_i) < 0$ for all $i \in \{1, 2, ..., N\}$ is not just an arbitrary convention, but in fact is entirely consistent with the general heuristic approach of choosing the facility with the largest index value at any given state. To see this, one may observe that balking is essentially equivalent to joining a 'degenerate' facility (which one may regard as 'facility zero', given that 0 is used throughout this thesis to represent the action of balking) at which services are always completed immediately, so that holding costs

are irrelevant, and the reward for service is zero. If one regards this extra degenerate facility as being present in the relaxed version of the problem discussed earlier (as a conceptual alternative to balking), then it is obvious that at any given state $\mathbf{x} \in S$, an optimal policy for the relaxed problem will decide to send a (copy of a) customer to the degenerate facility if and only if $W \leq 0$, since only the admission charge W is relevant to such a decision. Hence, by Definition 6.1.3, one has $W_0(x) = 0$ for all $x \in \mathbb{N}_0$. In effect, this states that the Whittle index associated with the action of balking is always zero (regardless of the state), and therefore if one has $W_i(x_i) < 0$ for all $i \in \{1, 2, ..., N\}$ then balking is indeed the action with the largest Whittle index.

As discussed earlier, formulae for the steady-state probabilities $\pi_i(y, x)$ in (6.1.13) are available from the general theory for $M/M/c/k$ queues; that is, $M/M/c$ queueing systems with a finite system capacity $k \in \mathbb{N}_0$. These formulae, while being somewhat messy, should not significantly hinder a computer program. In the special case of a single-server queue ($c_i = 1$), the Whittle indices $W_i(x)$ have a particularly simple form which enables an interesting comparison to be made with the results for $M/M/1$ queues from Chapter 2. This is shown by the next example.

Example 6.1.5. ***(Whittle indices in a single-server queue)***

This example concerns only the derivation of the Whittle indices $W_i(x)$ for a facility i with $c_i = 1$. While facility i may be thought of as being part of a larger N-facility system, the indices $W_i(x)$ are obtained independently of the characteristics of the other facilities $j \neq i$ and therefore it will be convenient to drop the facility identifier i. That is, for $x \in \mathbb{N}_0$:

$$W(x) = \alpha - \frac{\beta \left(\sum_{y=0}^{x+1} y\pi(y, x+1) - \sum_{y=0}^{x} y\pi(y, x) \right)}{\lambda \left(\pi(x, x) - \pi(x+1, x+1) \right)}. \tag{6.1.15}$$

Using standard results (see [67], p. 74) one has, for $x, y \in \mathbb{N}$ with $y \leq x$:

$$\pi(y, x) = \begin{cases} \dfrac{\rho^y(1-\rho)}{1-\rho^{x+1}}, & \text{if } \rho \neq 1, \\ 1/(x+1), & \text{if } \rho = 1, \end{cases} \tag{6.1.16}$$

where $\rho = \lambda/\mu$. Substituting (6.1.16) into (6.1.15) yields, after some manipulations:

$$W(x) = \begin{cases} \alpha - \dfrac{\beta\left((x+1)(1-\rho) - \rho(1-\rho^{x+1})\right)}{\mu(1-\rho)^2}, & \text{if } \rho \neq 1, \\ \alpha - \dfrac{\beta(x+1)(x+2)}{2\mu}, & \text{if } \rho = 1. \end{cases} \tag{6.1.17}$$

Recall that under the Whittle policy $\theta^{[W]}$, joining the facility in question is preferred to balking if and only if $W(x) \geq 0$. Equivalently, using (6.1.17), this condition states:

$$\begin{cases} \dfrac{\alpha\mu}{\beta} \geq \dfrac{(x+1)(1-\rho) - \rho(1-\rho^{x+1})}{(1-\rho)^2}, & \text{if } \rho \neq 1, \\ \dfrac{\alpha\mu}{\beta} \geq \dfrac{(x+1)(x+2)}{2}, & \text{if } \rho = 1. \end{cases} \tag{6.1.18}$$

Recall that under the *selfishly optimal* policy $\tilde{\theta}$ (in an $M/M/1$ queue), joining is preferred to balking at state $x \in \mathbb{N}_0$ if and only if $\alpha - \beta(x+1)/\mu \geq 0$, or equivalently:

$$\frac{\alpha\mu}{\beta} \geq x + 1. \tag{6.1.19}$$

The conditions in (6.1.18) may be written equivalently as follows:

$$\begin{cases} \dfrac{\alpha\mu}{\beta} \geq x + 1 + \dfrac{x\rho(1-\rho) - \rho^2(1-\rho^x)}{(1-\rho)^2}, & \text{if } \rho \neq 1, \\ \dfrac{\alpha\mu}{\beta} \geq x + 1 + \dfrac{x(x+1)}{2}, & \text{if } \rho = 1. \end{cases} \tag{6.1.20}$$

It may be shown that, for any fixed $x \in \mathbb{N}$, the function $f(\rho) := \left(x\rho(1-\rho) - \rho^2(1-\rho^x)\right)/(1-\rho)^2$ is positive and strictly increasing with ρ. Indeed, by differentiating, one finds:

$$f'(\rho) = \frac{x(1-\rho)(1+\rho^{x+1}) - 2\rho(1-\rho^x)}{(1-\rho)^3}. \tag{6.1.21}$$

The fact that the derivative in (6.1.21) is positive for all $x \in \mathbb{N}$ and $\rho > 0$ can be verified using a proof by induction over x, treating the cases $\rho \in (0,1)$ and $\rho > 1$ separately. Since $f(0) = 0$ for any fixed $x \in \mathbb{N}$, it then follows that the function is positive for all $\rho > 0$ (since it is strictly increasing). Hence, by comparing (6.1.19) and (6.1.20), it may be seen that the condition for joining to be preferred to balking by the Whittle policy is *stronger* than the corresponding condition for the selfish policy $\tilde{\theta}$; that is, the Whittle policy is more conservative than the selfish policy. This result is established in proper generality (i.e. not restricted to the case of a single-server queue) by

Theorem 6.1.7 later in this section. Moreover, since $f(\rho)$ is increasing with ρ, the Whittle policy for the single-server queue actually becomes more conservative as the demand rate increases. This is consistent with results from previous chapters (in particular, Theorem 5.1.8).

The condition in (6.1.18) may also be compared with the classical results of Naor [131] which were discussed in Chapter 2. Naor showed that the socially optimal threshold in an observable $M/M/1$ queue, denoted by n_o, must satisfy the following pair of inequalities:

$$\begin{cases} \dfrac{n_o(1-\rho)-\rho(1-\rho^{n_o})}{(1-\rho)^2} \leq \dfrac{\alpha\mu}{\beta} < \dfrac{(n_o+1)(1-\rho)-\rho(1-\rho^{n_o+1})}{(1-\rho)^2}, & \text{if } \rho \neq 1, \\ \dfrac{n_o(n_o+1)}{2} \leq \dfrac{\alpha\mu}{\beta} < \dfrac{(n_o+1)(n_o+2)}{2}, & \text{if } \rho = 1. \end{cases} \tag{6.1.22}$$

Under the socially optimal policy, one would choose to join at state $x \in \mathbb{N}_0$ if and only if $x < n_o$. Since n_o is the *smallest* integer for which the right-hand inequality in (6.1.22) holds, this implies that one would prefer joining to balking at state x if and only if:

$$\begin{cases} \dfrac{\alpha\mu}{\beta} \geq \dfrac{(x+1)(1-\rho)-\rho(1-\rho^{x+1})}{(1-\rho)^2}, & \text{if } \rho \neq 1, \\ \dfrac{\alpha\mu}{\beta} \geq \dfrac{(x+1)(x+2)}{2}, & \text{if } \rho = 1. \end{cases}$$

This is exactly the same condition as (6.1.18), showing that the Whittle indices for a single-server queue have an equivalence with Naor's classical results for social optimisation in an $M/M/1$ queue; specifically, the conditions $W(x) \geq 0$ and $x < n_o$ are equivalent. Of course, this is entirely expected, since the Whittle indices $W_i(x)$ are defined in such a way that $W_i(x) \geq 0$ if and only if joining is preferable to balking at state x in a single-server problem involving only facility i.

Finally, it is also interesting to observe how the Whittle indices $W(x)$ given in (6.1.17) are affected in light-traffic and heavy-traffic limits. Let $W(x,\lambda)$ denote the Whittle index for state $x \in \mathbb{N}_0$ given a demand rate $\lambda > 0$. Then, from (6.1.17), one immediately obtains:

$$\lim_{\lambda \to 0} W(x,\lambda) = \alpha - \frac{\beta(x+1)}{\mu}.$$

This states that, in a light-traffic limit (as $\lambda \to 0$), the Whittle index $W(x,\lambda)$ tends to the expected net reward for an individual customer for joining under state x. Of course, this is exactly the same index as that used by the *selfish* policy $\tilde{\theta}$, which shows that the Whittle policy $\theta^{[W]}$ 'tends

to the selfish policy' as $\lambda \to 0$. This is what one might expect, given that the selfish policy was shown by Theorem 4.4.4 to be asymptotically optimal in a light-traffic limit. Next, considering the *heavy-traffic* limit (as $\lambda \to \infty$), by taking limits in (6.1.17) again one obtains:

$$\lim_{\lambda\to\infty} W(x,\lambda) = \begin{cases} \alpha - \dfrac{\beta}{\mu}, & \text{if } x = 0, \\ -\infty, & \text{if } x \geq 1. \end{cases}$$

Recall that $\alpha - \beta/\mu$ is always strictly positive by assumption, and hence as $\lambda \to \infty$ the Whittle policy $\theta^{[W]}$ tends to a policy which chooses to join if and only if the system is empty. This limiting policy is an example of a *vacancy policy* (see Definition 4.4.6), and Theorem 4.4.7 showed that vacancy policies are optimal in a heavy-traffic limit, so again this is as expected.

This example has shown that if one restricts attention to *single-server* queues, then the Whittle policy $\theta^{[W]}$ is asymptotically optimal in a light-traffic limit and also optimal in a heavy-traffic limit. Theorem 6.1.8 later in this section will prove that these limiting properties of the Whittle policy also hold in greater generality, i.e. when multiple servers are allowed. ⊠

More generally, conditions implying the asymptotic optimality of Whittle's heuristic for restless bandit problems have been investigated in the literature; see, for example, [85, 193].

The next example is a numerical example which shows the ability of the Whittle policy $\theta^{[W]}$ to closely approximate an optimal policy in a system with two dual-server facilities.

Example 6.1.6. ***(Whittle indices in a system with two dual-server facilities)***

This example uses the same parameters as Example 4.1.1 from Chapter 4. To recap, there is a demand rate $\lambda = 10$ and the parameters for the two facilities are as follows:

$$c_1 = 2, \qquad \mu_1 = 8, \qquad \beta_1 = 10, \qquad \alpha_1 = 2,$$
$$c_2 = 2, \qquad \mu_2 = 2, \qquad \beta_2 = 10, \qquad \alpha_2 = 6.$$

The selfishly optimal policy $\tilde{\theta}$ and socially optimal policy θ^* (found by relative value iteration) for this system were shown in Table 4.1, page 127. As discussed in that example, an important difference between $\tilde{\theta}$ and θ^* is that θ^* chooses to balk at the state $(2, 2)$, which ensures that the

x	0	1	2	3
$W_1(x)$	0.750	0.750	-0.416	-1.770

Table 6.1: The Whittle indices $W_1(x)$ for $x \in \{0, 1, 2, 3\}$.

recurrent state space S_{θ^*} consists of only 9 states which form a proper subset of $\tilde{S}$. Table 6.1 shows the Whittle indices $W_1(x)$ for facility 1 (computed using (6.1.13)) for $x \in \{0, 1, 2, 3\}$.

Similarly, Table 6.2 shows the Whittle indices $W_2(x)$ for $x \in \{0, 1, 2\}$.

x	0	1	2
$W_2(x)$	1.000	1.000	-12.214

Table 6.2: The Whittle indices $W_2(x)$ for $x \in \{0, 1, 2\}$.

These results imply that the Whittle policy $\theta^{[W]}$ (for states $\mathbf{x} \in \tilde{S}$) is as shown in Table 6.3. By comparing $\theta^{[W]}$ with the socially optimal policy θ^* in Table 4.1, one finds that $\theta^{[W]}$ differs from θ^* at the states $(0, 1)$ and $(1, 1)$. Since the system is small, one may easily use dynamic programming to evaluate the average reward $g_{\theta^{[W]}}$ earned by $\theta^{[W]}$. The resulting value is $g_{\theta^{[W]}} \approx 7.256$, whereas the optimal value is $g^* \approx 7.282$, implying that $\theta^{[W]}$ is within about 0.35% of optimality. Similarly, it can be checked that the sub-optimality of the selfish policy $\tilde{\theta}$ is about 3%.

	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	2	2	1
$x_1 = 1$	2	2	1
$x_1 = 2$	2	2	0
$x_1 = 3$	2	2	0

Table 6.3: The Whittle policy $\theta^{[W]}$ for the system in Example 6.1.6.

In this particular example, the optimal policy θ^* has a certain subtlety in that it chooses different decisions at the states $(0, 0)$ and $(0, 1)$, despite the fact that both states offer an available server at both facilities (since the facilities are both dual-server). One may view this policy as somewhat counter-intuitive. In fact, the Whittle policy $\theta^{[W]}$ is unable to replicate such a decision-making property. This is due to the fact that at any state $\mathbf{x} \in \tilde{S}$ with $x_i < c_i$ for some $i \in \{1, 2, ..., N\}$, it

must be the case that $W_i(x_i) = \alpha_i - \beta_i/\mu_i$, since an admission charge W greater than $\alpha_i - \beta_i/\mu_i$ would be required in order for balking to be preferred to joining i at $\mathbf{x}$. Hence, it can never be the case that the Whittle policy prefers facility i over facility $j \neq i$ at some state $\mathbf{x}$ with $x_i < c_i$ and $x_j < c_j$, but prefers j over i at some other state $\mathbf{y} \neq \mathbf{x}$ with $y_i < c_i$ and $y_j < c_j$.

As a further note, the Whittle indices in Tables 6.1 and 6.2 can easily be verified by using relative value iteration to compute optimal policies in single-facility problems with the rewards α_i adjusted. For example, Table 6.1 states that $W_1(2) \approx -0.416$, which implies that the reward α_1 would need to be increased by at least 0.416 in order for the optimal threshold in the corresponding single-facility problem to exceed 2. Indeed, by computing the optimal single-facility policy for facility 1 with the reward value α_1 increased from 2 to 2.42, one obtains a threshold policy with threshold 3. On the other hand, using the slightly smaller value $\alpha_1 = 2.41$ yields a threshold of 2. ⊠

By their nature, index-based policies are appealing not only due to their ease of application but also due to their preservation of intuitive structural properties (which may not be exhibited by a socially optimal policy). The next result establishes that the Whittle policy $\theta^{[W]}$ possesses several of the structural properties which have been thematic throughout this thesis.

Theorem 6.1.7. *The Whittle policy $\theta^{[W]}$ possesses the following properties:*

1. *(Containment.) Let S_W be the set of positive recurrent states under the policy $\theta^{[W]}$. Then $S_W \subseteq \tilde{S}$, where $\tilde{S}$ is the corresponding set for the selfish policy $\tilde{\theta}$.*

2. *(First monotonicity property.) Suppose $\theta^{[W]}(\mathbf{x}) = 0$ for some state $\mathbf{x} \in S$. Then $\theta^{[W]}(\mathbf{x}^{i+}) = 0$ for all facilities $i \in \{1, 2, ..., N\}$.*

3. *(Second monotonicity property.) Suppose $\theta^{[W]}(\mathbf{x}) = i$ for some state $\mathbf{x} \in S$ and facility $i \in \{1, 2, ..., N\}$ with $x_i \geq 1$. Then $\theta^{[W]}(\mathbf{x}^{i-}) = i$.*

4. *(Third monotonicity property.) Suppose $\theta^{[W]}(\mathbf{x}) = i$ for some state $\mathbf{x} \in S$ and facility $i \in \{1, 2, ..., N\}$. Then $\theta^{[W]}(\mathbf{x}^{j+}) = i$ for all facilities $j \in \{1, 2, ..., N\} \setminus \{i\}$.*

5. *(Sink property.) The policy $\theta^{[W]}$ is a sink policy with a sink state $\mathbf{z} \in S$ satisfying $z_i = \min\{x \in \mathbb{N}_0 : W_i(x) < 0\}$ for all facilities $i \in \{1, 2, ..., N\}$.*

6. *(Conservativity with demand.) Let $S_W(\lambda)$ be the set of positive recurrent states under the Whittle policy $\theta^{[W]}_{\lambda}$ given a demand rate $\lambda > 0$. Then, for any pair of demand rates $\lambda_1, \lambda_2 > 0$ with $\lambda_1 > \lambda_2$, it is the case that $S_W(\lambda_1) \subseteq S_W(\lambda_2)$.*

7. *(Non-idling property.) The policy $\theta^{[W]}$ does not choose to balk at any state $\mathbf{x} \in S$ with $x_i < c_i$ for at least one facility $i \in \{1, 2, ..., N\}$.*

Proof. The most trivial properties to prove are the monotonicity properties, numbered (2)-(4) in the theorem, since these are direct consequences of the indexability property proved by Theorem 6.1.2. Indeed, due to Theorem 5.1.4, any admission charge W which causes joining to be preferred to balking at some state $x + 1$ (with $x \in \mathbb{N}_0$) in a single-facility problem also causes joining to be preferred at state x. By Definition 6.1.3, the largest such charge would be $W_i(x + 1)$ (considering an arbitrary facility $i \in \{1, 2, ..., N\}$). That is, given an admission charge $W_i(x + 1)$, one can say that joining is preferable to balking at state x, from which it follows that $W_i(x) \geq W_i(x + 1)$ by definition of $W_i(x)$. This establishes that the indices $W_i(x)$ are monotonically decreasing in $x \in \mathbb{N}_0$ for all $i \in \{1, 2, ..., N\}$, whereupon the properties (2)-(4) follow trivially from (6.1.14) (recall that one assumes that there is a fixed ordering of the N facilities used to settle ties, so that i is always preferred to j (or vice versa) at any state $\mathbf{x} \in S$ for which $W_i(x_i) = W_j(x_j)$).

For the containment property (1), it is sufficient to observe that due to Theorem 5.1.4, the socially optimal threshold T^*_i for any facility $i \in \{1, 2, ..., N\}$ satisfies $T^*_i \leq \tilde{B}_i$, where $\tilde{B}_i$ is the selfish threshold defined in (4.1.2). This implies that $W_i(\tilde{B}_i) < 0$ for any facility i, and hence the policy $\theta^{[W]}$ never chooses to join facility i at any state $\mathbf{x} \in S$ with $x_i \geq \tilde{B}_i$. In particular, balking is chosen at any state with $x_i \geq \tilde{B}_i$ for all $i \in \{1, 2, ..., N\}$. It follows that no state outside $\tilde{S}$ can be accessible from state $\mathbf{0}$ under $\theta^{[W]}$; equivalently, the set S_W must be contained in $\tilde{S}$.

Next, let $\mathbf{z}$ be the unique state in S with $z_i = \min\{x \in \mathbb{N}_0 : W_i(x) < 0\}$ for all $i \in \{1, 2, ..., N\}$, and let $S_{\mathbf{z}} = \big\{\mathbf{x} \in S : x_i \leq z_i \text{ for all } i \in \{1, 2, ..., N\}\big\}$. By definition of $\mathbf{z}$, any state $\mathbf{x} \in S_{\mathbf{z}} \setminus \{\mathbf{z}\}$ has $W_i(x_i) \geq 0$ for at least one $i \in \{1, 2, ..., N\}$. Hence, $\mathbf{z}$ is the only state in $S_{\mathbf{z}}$ at which the policy $\theta^{[W]}$ chooses to balk. The state $\mathbf{z}$ must therefore be accessible from $\mathbf{0}$ (via an unbroken sequence of arrivals, for example) and hence positive recurrent under $\theta^{[W]}$, from which it follows that *any* state $\mathbf{x} \in S_{\mathbf{z}} \setminus \{\mathbf{z}\}$ is positive recurrent under $\theta^{[W]}$ since it can be reached from $\mathbf{z}$ via an appropriate sequence of service completions. Moreover, no state outside $S_{\mathbf{z}}$ is accessible from $\mathbf{0}$ under $\theta^{[W]}$,

since this would require joining some facility i at a state $\mathbf{x} \in S_{\mathbf{z}}$ with $x_i = z_i$ and hence $W_i(x_i) < 0$. It follows that S_W, the set of positive recurrent states under $\theta^{[W]}$, is equal to $S_{\mathbf{z}}$. Since $\mathbf{z}$ is the only state in $S_{\mathbf{z}}$ at which $\theta^{[W]}$ chooses to balk, property (5) follows by Definition 5.2.14.

Next, property (6) may be established using Theorem 5.1.8 from Chapter 5, which states that the socially optimal threshold found by relative value iteration is monotonically decreasing with the demand rate λ. Let $\lambda_1, \lambda_2 > 0$ be demand rates with $\lambda_1 > \lambda_2$. By property (5), the Whittle policy $\theta^{[W]}_{\lambda_2}$ corresponding to demand rate λ_2 is a sink policy with a sink state $\mathbf{z} \in S$ satisfying $z_i = \min\{x \in \mathbb{N}_0 : W_i(x, \lambda_2) < 0\}$ for all $i \in \{1, 2, ..., N\}$, where $W_i(x, \lambda_2)$ is the obvious generalisation of $W_i(x)$. In order for the set of recurrent states $S_W(\lambda_1)$ *not* to be contained in $S_W(\lambda_2)$, the policy $\theta^{[W]}_{\lambda_1}$ would have to choose to join some facility $i \in \{1, 2, ..., N\}$ at a state $\mathbf{x}$ with $x_i = z_i$ for some $i \in \{1, 2, ..., N\}$. This would require $\theta^{[W]}_{\lambda_1}$ to choose facility i at a state with $W_i(x_i, \lambda_2) < 0$, implying the relationship $W_i(x_i, \lambda_1) \geq 0 > W_i(x_i, \lambda_2)$. It will therefore be sufficient to show that $W_i(x_i, \lambda_1) \leq W_i(x_i, \lambda_2)$ for all $\mathbf{x} \in S$ and $i \in \{1, 2, ..., N\}$. Indeed, due to Theorem 5.1.8, any admission charge W that causes joining to be preferred to balking at an arbitrary state $x \in \mathbb{N}_0$ in a single-facility problem under demand rate λ_1 also causes joining to be preferred at x under demand rate λ_2. The largest such charge would be $W_i(x, \lambda_1)$ (considering an arbitrary facility $i \in \{1, 2, ..., N\}$). That is, given an admission charge $W_i(x, \lambda_1)$, one can say that joining is preferable to balking at state x under demand rate λ_2, from which it follows that $W_i(x, \lambda_2) \geq W_i(x, \lambda_1)$ by definition of $W_i(x, \lambda_2)$. Property (6) then follows from the previous arguments.

In order to establish property (7), consider an arbitrary facility $i \in \{1, 2, ..., N\}$ and suppose relative value iteration is applied to the corresponding single-facility problem, with the state space restricted to the finite set $\{0, 1, ..., \tilde{B}_i\}$. Here, $\tilde{B}_i$ is the selfishly optimal threshold for facility i defined in (4.1.2), and $\tilde{B}_i \geq c_i$ due to the assumption that $\alpha_i - \beta_i/\mu_i > 0$. Lemma 4.3.2 (applied to the single-facility problem) implies that joining is preferred to balking by the optimal policy θ^*_i at all states $x < c_i$; hence, it must be the case that $W_i(x) \geq 0$ for all $x < c_i$. Since the facility i was arbitrary, it follows that for any state $\mathbf{x} \in S$ with $x_i < c_i$ for some $i \in \{1, 2, ..., N\}$ one has $W_i(x_i) \geq 0$ and hence balking is not chosen by the Whittle policy $\theta^{[W]}$. This completes the proof. □

Naturally, the *selfish* policy $\tilde{\theta}$ (being an index-based policy) also possesses all of the properties discussed in Theorem 6.1.7, although property (6) is somewhat meaningless in the selfishly optimal

case, since $\tilde{\theta}$ is actually independent of the demand rate λ. In a sense, one might say that $\theta^{[W]}$ is simply an alternative index policy which is less naive (in the context of social optimisation) than $\tilde{\theta}$, since it takes into account more than just the welfare of the immediate customer.

It was proved in Section 4.4 that the selfishly optimal policy $\tilde{\theta}$ is asymptotically socially optimal in a light-traffic limit. The next result shows that the Whittle policy $\theta^{[W]}$ not only emulates the light-traffic optimality of $\tilde{\theta}$, but also attains optimality in a *heavy*-traffic limit. The notation $\theta^{[W]}_{\lambda}$ will be used to denote the Whittle index policy given a fixed demand rate $\lambda > 0$.

Theorem 6.1.8. *Let $g_{\theta^{[W]}}(\lambda)$ be the long-run average reward attained by the Whittle policy $\theta^{[W]}_{\lambda}$ given a demand rate λ, and let $g^*(\lambda)$ be the corresponding optimal value. Then:*

1. *$\theta^{[W]}$ is asymptotically optimal in a light-traffic limit. That is:*

$$\lim_{\lambda \to 0} \frac{g^*(\lambda) - g_{\theta^{[W]}}(\lambda)}{g^*(\lambda)} = 0.$$

2. *$\theta^{[W]}$ is optimal in a heavy-traffic limit. That is:*

$$\lim_{\lambda \to \infty} \left(g^*(\lambda) - g_{\theta^{[W]}}(\lambda) \right) = 0.$$

Proof. As before, it will be convenient to denote the Whittle indices by $W_i(x, \lambda)$ (for $i \in \{1, 2, ..., N\}$, $x \in \mathbb{N}_0$ and $\lambda > 0$) as opposed to $W_i(x)$ in order to allow for a dependence on the demand rate λ. Consider the light-traffic limit first. Given any demand rate $\lambda > 0$, one obtains the following expression for $W_i(0, \lambda)$ (for $i \in \{1, 2, ..., N\}$) by setting $x = 0$ in (6.1.13):

$$W_i(0, \lambda) = \alpha_i - \frac{\beta_i}{\mu_i}.$$

Of course this is as expected, since a 0-threshold policy can only be preferable to a 1-threshold policy at facility i if the admission charge is greater than a customer's expected net reward for joining when a server is available. It then follows from (6.1.14) (and the assumption that $\alpha_i - \beta_i/\mu_i > 0$ for all $i \in \{1, 2, ..., N\}$) that, regardless of the demand rate λ, the Whittle policy $\theta^{[W]}_{\lambda}$ always chooses a facility i at state $\mathbf{0}$ which maximises $\alpha_i - \beta_i/\mu_i$. The result then follows by Corollary 4.4.5, which states that any such policy is asymptotically optimal in a light-traffic limit.

In the heavy-traffic case, one may invoke Theorem 5.1.8. This states that in any single-facility problem, the socially optimal threshold $T^*(\lambda)$ tends to the number of servers c as $\lambda \to \infty$. Thus,

for each $i \in \{1, 2, .., N\}$ there exists some $\delta_i > 0$ such that, for all $\lambda > \delta_i$:

$$W_i(c_i, \lambda) < 0.$$

On the other hand, regardless of the value of λ, one must have $W_i(x_i, \lambda) \geq 0$ for any state $\mathbf{x} \in S$ with $x_i < c_i$ due to the monotonic nature of the convergence proved by Theorem 5.1.8. These observations imply that for sufficiently large values of λ, the Whittle policy $\theta_\lambda^{[W]}$ chooses to balk at a state $\mathbf{x} \in S$ if and only if $x_i \geq c_i$ for all $i \in \{1, 2, ..., N\}$. That is, $\theta_\lambda^{[W]}$ becomes a *vacancy policy* (see Definition 4.4.6) for sufficiently large values of λ. The result then follows by Theorem 4.4.7, which states that any vacancy policy must be optimal in a heavy-traffic limit. $\square$

Before concluding this section, it will be useful to establish an insightful and interesting property of the Whittle policy $\theta^{[W]}$ which involves a comparison with the optimal policy θ^* found by relative value iteration. In previous chapters, the term 'conservative' has been used to describe a policy whose set of positive recurrent states is contained within that of another policy; for example, Theorem 4.2.4 showed that the socially optimal policy θ^* is more conservative than the selfish policy $\tilde{\theta}$. Of course, the Whittle policy $\theta^{[W]}$ is a heuristic policy which is intended to earn an expected long-run average reward as close as possible to that of the optimal policy θ^*. Thus, one might expect $\theta^{[W]}$ to be as similar as possible to θ^* in terms of the decisions chosen at the various states $\mathbf{x} \in S$, although Example 6.1.6 has shown that it need not always be identical.

As shown by the counter-examples in Appendix A.8, structural properties of the optimal policy θ^* are not easy to prove in complete generality, and hence one would think that results based on decision-making comparisons between θ^* and $\theta^{[W]}$ would not be easy to obtain. It therefore comes as something of a surprise that the Whittle policy can be shown to be more conservative than the optimal policy θ^*; that is, the set of positive recurrent states S_W associated with the Whittle policy is contained within the corresponding set S_{θ^*} under θ^*. This is an interesting result, since it is not immediately obvious that the Whittle policy $\theta^{[W]}$ should have such a predisposition towards conservative decision-making. In order to prove the result, a lemma will be needed.

Lemma 6.1.9. *Let θ^* be the optimal policy found by relative value iteration. Then, for $\mathbf{x} \in \tilde{S}$:*

$$\theta^*(\mathbf{x}) = 0 \;\; \Rightarrow \;\; \theta^{[W]}(\mathbf{x}) = 0.$$

That is, the Whittle policy $\theta^{[W]}$ chooses to balk at any state $\mathbf{x} \in \tilde{S}$ where θ^* chooses to balk.

Proof. The proof can be established using an argument based on dynamic programming. Essentially, one can show that if balking is chosen at some state $\mathbf{x} \in \tilde{S}$ by the optimal policy θ^*, then balking would also be chosen by an optimal threshold policy in a single-facility problem involving any of the facilities $i \in \{1, 2, ..., N\}$ at the state with x_i customers present (where x_i is the i^{th} component of the state $\mathbf{x}$ in the N-facility problem). Since the Whittle policy $\theta^{[W]}$ makes decisions by considering each of the N facilities operating in isolation, this is sufficient to establish the result. The full details are somewhat messy, but can be found in Appendix A.6, page 449. □

The result of Lemma 6.1.9 enables the aforementioned conservativity property of the Whittle policy $\theta^{[W]}$ to be proved for a general N-facility system. This is stated below as a theorem.

Theorem 6.1.10. ***(Conservativity of the Whittle policy)***

Let S_W denote the set of positive recurrent states under the Whittle policy $\theta^{[W]}$, and let S_{θ^*} denote the corresponding set under the policy θ^* found by relative value iteration on the finite set $\tilde{S}$. Then:

$$S_W \subseteq S_{\theta^*}.$$

Proof. By Theorem 4.2.4, S_{θ^*} is contained in the finite set $\tilde{S}$ associated with the selfishly optimal policy $\tilde{\theta}$. Hence, there must exist some state in S_{θ^*} at which the policy θ^* chooses to balk; otherwise, an unbroken sequence of customer arrivals would cause the process to pass outside $\tilde{S}$ under θ^*. Let $\mathbf{z} \in S_{\theta^*}$ be a state at which θ^* chooses to balk, and let $S_{\mathbf{z}}$ be defined as follows:

$$S_{\mathbf{z}} := \left\{ \mathbf{x} \in \tilde{S} : x_i \leq z_i \ \ \forall\, i \in \{1, 2, ..., N\} \right\}.$$

That is, $S_{\mathbf{z}}$ is the set of states $\mathbf{x} \in \tilde{S}$ which satisfy the componentwise inequality $\mathbf{x} \leq \mathbf{z}$. Since $\mathbf{z} \in S_{\theta^*}$, it follows that all states in $S_{\mathbf{z}}$ are also included in S_{θ^*}, since they are accessible from $\mathbf{z}$ via service completions. Hence, $S_{\mathbf{z}} \subseteq S_{\theta^*}$. On the other hand, since balking is chosen by θ^* at $\mathbf{z}$, it follows by Lemma 6.1.9 that balking is also chosen at $\mathbf{z}$ by the Whittle policy $\theta^{[W]}$. By definition of the Whittle policy, this implies that $W_i(z_i) < 0$ for all $i \in \{1, 2, ..., N\}$. Therefore it is impossible for any state $\mathbf{x} \notin S_{\mathbf{z}}$ to be accessible from state $\mathbf{0}$ under the Whittle policy, since this would require joining some facility $i \in \{1, 2, ..., N\}$ to be chosen at a state $\mathbf{y} \in S_{\mathbf{z}}$ with $y_i = z_i$ and hence $W_i(y_i) < 0$. It follows that $S_W \subseteq S_{\mathbf{z}} \subseteq S_{\theta^*}$, which completes the proof. □

As a further note, it was established in Section 4.3 that the recurrent state space S_{θ^*} under *any*

stationary socially optimal policy θ^* must satisfy the following relationship:

$$S^\circ \subseteq S_{\theta^*} \subseteq \tilde{S}, \tag{6.1.23}$$

where S° is the set of states in S with no customers waiting in queues, defined in (4.3.3). It has been established by Theorem 6.1.7 that the Whittle policy $\theta^{[W]}$ is always a non-idling policy, and hence $S^\circ \subseteq S_W$. Thus, in the case where θ^* is the optimal policy found by relative value iteration on $\tilde{S}$, the result of Theorem 6.1.10 enables (6.1.23) to be strengthened as follows:

$$S^\circ \subseteq S_W \subseteq S_{\theta^*} \subseteq \tilde{S}.$$

In Section 6.3 the performance of the Whittle policy will be tested over a wide range of system parameters. In the next section, an alternative heuristic approach will be discussed.

6.2 Static routing heuristic

The second of the two heuristic policies to be discussed in this chapter is based on an approach which has received considerable attention in the literature, involving the derivation of a stationary policy from a *state-independent* randomised policy. As a starting point, let σ denote a policy under which, regardless of the system state, any customer who arrives is sent to facility $i \in \{1, 2, ..., N\}$ with probability $\sigma_i \in [0, 1]$, where $\sum_{i=1}^{N} \sigma_i \leq 1$. The probability of a customer balking upon arrival is then given by $\sigma_0 := 1 - \sum_{i=1}^{N} \sigma_i$. This type of policy might be referred to as a *static routing* policy, a *Bernoulli splitting* policy or simply as a state-independent routing policy. Moreover, it may be observed that policies of this form represent a natural N-facility generalisation of the policies for *unobservable* $M/M/1$ queueing systems discussed in Chapter 2. Throughout this section, state-independent randomised policies such as σ will be referred to as *static* policies.

The heuristic policy to be developed in this section is based on applying a single step of *policy improvement* to an optimal static policy (that is, a policy which performs best among all static policies). "Policy improvement", in this context, refers to the general strategy of improving upon a certain 'reference' policy (in this case, an optimal static policy) by allowing the decision at any state $\mathbf{x} \in S$ to be chosen in such a way as to maximise the aggregate expected net reward over a long period of time, under the assumption that the reference policy will be followed thereafter.

Recall that the Policy Improvement Algorithm (PIA), introduced in Section 3.7, is a dynamic programming algorithm which is guaranteed to converge to an optimal stationary policy after a finite number of iterations, but in order to do so it must be allowed to make as many policy improvement steps as should be required. The heuristic policy of this section is based on applying a *single* step of policy improvement, and as such its optimality is not guaranteed.

Heuristic policies of this nature have been examined by various authors. Krishnan [109] considered the problem of assigning customers to parallel Markovian queues in such a way as to minimise the average time spent in the system, and found that a heuristic derived from a "Bernoulli splitting" rule was able to perform better than an individually optimal policy in many cases. Sassen et al. [152] also considered the minimisation of average sojourn times, but assumed a general distribution for service times at any server; their "decomposition approach" follows a similar derivation to that of Krishnan's heuristic. Ansell et al. [4] considered the minimisation of long-run average holding costs, and generalised the problem further by incorporating heterogeneous customer classes; however, their formulation also assumes single-server facilities with exponentially-distributed service times, so that individual facilities operate as multi-class $M/M/1$ queues under a static routing policy. The development in this section mainly emulates that of Argon et al. [6], who considered long-run average cost minimisation in a problem involving single-server queues arranged in parallel with no balking, but also incorporated a more general cost structure than that in [4].

In seeking an optimal static policy, one aims to find a vector of routing probabilities $(\sigma_1, \sigma_2, ..., \sigma_N)$ which maximises the expected long-run average reward g_σ, which is given by:

$$g_\sigma = \sum_{i=1}^{N} \big(\lambda\sigma_i\alpha_i - \beta_i L_i(\lambda\sigma_i)\big),$$

where $L_i(\lambda\sigma_i)$ is the expected number of customers present at facility i, given that customers arrive via a Poisson process with rate $\lambda\sigma_i$. It should be noted that $L_i(\lambda\sigma_i)$ is finite if and only if $\lambda\sigma_i < c_i\mu_i$, and so it will be appropriate to define $g_\sigma = -\infty$ for any policy σ with $\lambda\sigma_i \geq c_i\mu_i$ for at least one $i \in \{1, 2, ..., N\}$. Naturally, due to elementary properties of Poisson processes (see, for example, [146] p. 310) each facility i operates as an independent $M/M/c_i$ queue with demand rate $\lambda\sigma_i$ under the static policy σ. Thus, a static routing policy may be represented as a vector of demand rates $(\lambda_1, \lambda_2, ..., \lambda_N)$, where $\lambda_i = \lambda\sigma_i$ for $i \in \{1, 2, ..., N\}$; the rate at which customers balk is then given by $\lambda_0 := \lambda - \sum_{i=1}^{N} \lambda_i$. Throughout this section, static policies will be represented by

vectors $(\lambda_1, \lambda_2, ..., \lambda_N)$, with $0 \leq \lambda_i \leq \lambda$ for each $i \in \{1, 2, ..., N\}$ and $\sum_{i=1}^{N} \lambda_i \leq \lambda$.

A fundamental assumption throughout this section is the existence and uniqueness of an optimal static policy $(\lambda_1^*, \lambda_2^*, ...\lambda_N^*)$ which maximises the objective function:

$$g(\lambda_1, \lambda_2, ..., \lambda_N) := \sum_{i=1}^{N} \big(\lambda_i \alpha_i - \beta_i L_i(\lambda_i)\big). \tag{6.2.1}$$

The next result shows that this assumption is theoretically justified.

Theorem 6.2.1. *There exists a unique optimal static policy.*

Proof. For each facility $i \in \{1, 2, ..., N\}$, let $g_i(\lambda_i)$ be defined as follows:

$$g_i(\lambda_i) := \lambda_i \alpha_i - \beta_i L_i(\lambda_i). \tag{6.2.2}$$

It is known from the literature (see [65, 114]) that $L_i(\lambda_i)$, the expected number of customers present at facility i, is a strictly *convex* function of the demand rate λ_i. This implies that $g_i(\lambda_i)$ is a strictly *concave* function of λ_i, since by differentiating twice one obtains:

$$g_i{}''(\lambda_i) = -\beta_i L_i{}''(\lambda_i) < 0.$$

Figure 6.3 illustrates the general shape of the function $g_i(\lambda_i)$ as λ_i increases. Note that the assumption that $\alpha_i - \beta_i/\mu_i > 0$ for each $i \in \{1, 2, ..., N\}$ ensures that $g_i(\lambda_i)$ is strictly positive for sufficiently small values of λ_i. To see this, suppose facility i is a single-server queue ($c_i = 1$), in which case one has $L_i(\lambda_i) = \rho_i/(1 - \rho_i)$, where $\rho_i = \lambda_i/\mu_i$. As shown in Section 2.3, by solving the equation $g_i(\lambda_i) = 0$ with $c_i = 1$ and λ_i assumed strictly positive, one finds:

$$\lambda_i = \mu_i - \frac{\beta_i}{\alpha_i} > 0,$$

Noting also that $g_i(0) = 0$, the concavity of $g_i(\lambda_i)$ then implies that $g_i(\lambda_i) > 0$ for demand rates $\lambda_i \in (0,\ \mu_i - \beta_i/\alpha_i)$. It is easy to show that the mean number of customers present in an $M/M/c$ queue is monotonically decreasing with the number of servers c; indeed, this was shown using a sample path argument in Example 3.8.3. Hence, (6.2.2) implies that $g_i(\lambda_i)$ monotonically increases with the number of servers, and it follows that $g_i(\lambda_i) > 0$ for $\lambda_i \in (0,\ \mu_i - \beta_i/\alpha_i)$ irrespective of the number of servers $c_i \in \mathbb{N}$ (since this has been shown to be true when $c_i = 1$).

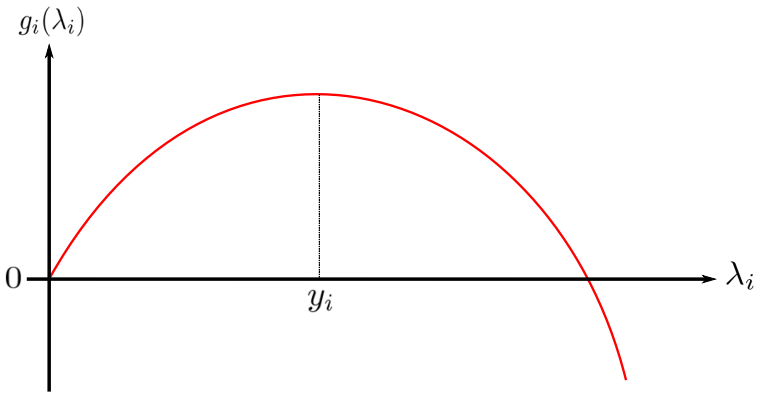


Figure 6.3: The general shape of the function $g_i(\lambda_i)$ as λ_i increases.

For each $i \in \{1, 2, ..., N\}$, there exists a unique maximiser $y_i > 0$ such that $g_i(y_i) > g_i(\lambda_i)$ for all $\lambda_i \in (0, \infty) \setminus \{y_i\}$ due to the strict concavity property. If $\sum_{i=1}^{N} y_i \leq \lambda$ then the result of the theorem is trivial, since the policy with $\lambda_i = y_i$ for each $i \in \{1, 2, ..., N\}$ is the unique optimal static policy. On the other hand, suppose $\sum_{i=1}^{N} y_i > \lambda$. Let the set B be defined as follows:

$$B := \left\{ (\lambda_1, \lambda_2, ..., \lambda_N) \in \mathbb{R}^N : \lambda_i \in [0, y_i] \; \forall \; i \in \{1, 2, ..., N\}, \; \sum_{i=1}^{N} \lambda_i \leq \lambda \right\}.$$

It is clear that any static policy with $\lambda_i > y_i$ for some $i \in \{1, 2, ..., N\}$ must be sub-optimal, since one would be able to improve the performance of such a policy without violating any constraints by setting $\lambda_i = y_i$. Hence, any static policy which is optimal among all policies in B must be an optimal static policy. The set B is a *compact* subset of $\mathbb{R}^N$; that is, it is closed and bounded. Moreover, since $g_i(\lambda_i)$ diverges to minus infinity as $\lambda_i \to c_i\mu_i$, one must have $y_i < c_i\mu_i$ for each $i \in \{1, 2, ..., N\}$, and hence the objective function (6.2.1) is well-defined and continuous on the set B. By the Weierstrass extreme value theorem (see [96], p. 689), it follows that $g(\lambda_1, \lambda_2, ..., \lambda_N)$ attains a maximum on the set B; that is, there exists an optimal static policy.

It remains to be shown that any optimal static policy must be unique. Consider the problem of maximising the objective function (6.2.1) subject to the requirements that $\lambda_i \in [0, y_i]$ for each $i \in \{1, 2, ..., N\}$ and $\sum_{i=1}^{N} \lambda_i \leq \lambda$. This problem fits the general framework of a *convex optimisation* problem, involving a strictly concave objective function and a set of linear constraints. It is known from the theory of such problems (see [23], p. 19) that a *unique* optimal solution exists under the stated conditions; that is, there must exist a unique static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ which maximises

the long-run average reward $g(\lambda_1, \lambda_2, ..., \lambda_N)$ defined in (6.2.1). Indeed, it is possible to construct a rigorous proof of the uniqueness of any optimal static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ which exploits concavity of the functions $g_i(\lambda_i)$; further details can be found in Appendix A.6, page 454. □

The concavity of the functions $g_i(\lambda_i) = \lambda_i \alpha_i - \beta_i L_i(\lambda_i)$ enables an interesting property of optimal static policies to be proved, which will prove to be useful later in this section.

Theorem 6.2.2. *Consider two different demand rates $\lambda, \bar{\lambda} > 0$, with $\lambda < \bar{\lambda}$. Let $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ and $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$ be the unique optimal static policies under the system demand rates λ and $\bar{\lambda}$ respectively. Then, for all facilities $i \in \{1, 2, ..., N\}$:*

$$\lambda_i^* \leq \bar{\lambda}_i^*.$$

Proof. As in the proof of Theorem 6.2.1, define $g_i(\lambda_i)$ for $i \in \{1, 2, ..., N\}$ as follows:

$$g_i(\lambda_i) = \lambda_i \alpha_i - \beta_i L_i(\lambda_i).$$

Given that the policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ is optimal under demand rate λ, it is possible to show that the optimal policy $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$ under demand rate $\bar{\lambda}$ satisfies:

$$\sum_{i=1}^{N} \bar{\lambda}_i^* \geq \sum_{i=1}^{N} \lambda_i^*. \tag{6.2.3}$$

Indeed, this may be argued via contradiction. Suppose the optimal policy $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$ under $\bar{\lambda}$ does *not* satisfy (6.2.3). In view of Theorem 6.2.1, this implies:

1. The policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$, which is optimal under λ, is sub-optimal under $\bar{\lambda}$.
2. There exists an optimal policy $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$ under $\bar{\lambda}$ such that $\sum_{i=1}^{N} \bar{\lambda}_i^* < \sum_{i=1}^{N} \lambda_i^*$.

However, given that $\sum_{i=1}^{N} \bar{\lambda}_i^* < \sum_{i=1}^{N} \lambda_i^* \leq \lambda$, one may construct a policy under demand rate λ with $\lambda_i = \bar{\lambda}_i^*$ for $i = 1, 2, ..., N$ (with $\lambda_0 = \lambda - \sum_{i=1}^{N} \lambda_i$ being the rate at which customers balk); then, by the first of the above statements (and the fact that the constructed policy induces the same long-run average reward under either of the demand rates λ and $\bar{\lambda}$) this policy yields an average reward strictly greater than that given by $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ under λ, which contradicts the optimality of $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ under λ. It follows that the relationship in (6.2.3) holds.

Next, arguing by contradiction, suppose the statement of the theorem is false. This implies that there exists a non-empty subset $I \subseteq \{1, 2, ..., N\}$ such that $\bar{\lambda}_i^* < \lambda_i^*$ for all facilities $i \in I$. Let the subset $M \in \{1, 2, ..., N\}$ and positive quantity ϵ be defined as follows:

$$M := \left\{ i \in \{1, 2, ..., N\} : \bar{\lambda}_i^* < \lambda_i^* \right\},$$
$$\epsilon := \sum_{i \in M} \left(\lambda_i^* - \bar{\lambda}_i^* \right).$$

Let $R := \{1, 2, ..., N\} \setminus M$. Due to (6.2.3), R must be non-empty and, in addition:

$$\bar{\lambda}_i^* \geq \lambda_i^* \qquad \forall\, i \in R, \tag{6.2.4}$$

$$\sum_{i \in R} \left(\bar{\lambda}_i^* - \lambda_i^* \right) \geq \epsilon. \tag{6.2.5}$$

The condition in (6.2.5) may be expressed as:

$$\sum_{i \in R} \left(\bar{\lambda}_i^* - \lambda_i^* \right) = \epsilon + \delta, \tag{6.2.6}$$

for some $\delta \geq 0$. Let $\{\epsilon_i\}_{i \in R}$ be any set of non-negative numbers for which $\sum_{i \in R} \epsilon_i = \epsilon$, and let $\{\delta_i\}_{i \in R}$ be any set of non-negative numbers with $\sum_{i \in R} \delta_i = \delta$ and, in addition:

$$\epsilon_i + \delta_i = \bar{\lambda}_i^* - \lambda_i^* \qquad \forall\, i \in R. \tag{6.2.7}$$

It is clearly always possible to find two sets $\{\epsilon_i\}_{i \in R}$ and $\{\delta_i\}_{i \in R}$ satisfying the aforementioned conditions, given that (6.2.4) and (6.2.6) hold. Indeed one way to do this is as follows:

1. Put the facilities $i \in R$ in some arbitrary order $i_1, i_2, ..., i_{|R|}$ and set $j = 1$.

2. Define ϵ_{i_j} as follows:
$$\epsilon_{i_j} := \min \left(\bar{\lambda}_{i_j}^* - \lambda_{i_j}^*,\ \epsilon - \sum_{k=1}^{j-1} \epsilon_{i_k} \right). \tag{6.2.8}$$

3. If $\sum_{k=1}^{j} \epsilon_{i_k} = \epsilon$ go to step 4; otherwise, increment j by 1 and return to step 2. Note that the condition $\sum_{k=1}^{j} \epsilon_{i_k} = \epsilon$ must hold after at most $|R|$ iterations due to (6.2.5).

4. Set $\epsilon_{i_k} = 0$ for all $k \in \{j+1, j+2, ..., |R|\}$ and $\delta_i = \bar{\lambda}_i^* - \lambda_i^* - \epsilon_i$ for all $i \in R$.

Under the supposition that the theorem is false, a policy $(\lambda_1, \lambda_2, ..., \lambda_N)$ with $\lambda_i = \lambda_i^*$ for all $i \in M$ and $\lambda_i = \lambda_i^* + \delta_i$ for all $i \in R$ must be sub-optimal under demand rate $\bar{\lambda}$. In particular, such a

policy is inferior to the optimal policy $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$ under demand rate $\bar{\lambda}$. Hence, recalling the fact that $\bar{\lambda}_i^* = \lambda_i^* + \delta_i + \epsilon_i$ for $i \in R$ due to (6.2.7), one has the following:

$$\sum_{i \in M} g_i(\lambda_i^*) + \sum_{i \in R} g_i(\lambda_i^* + \delta_i) < \sum_{i \in M} g_i(\bar{\lambda}_i^*) + \sum_{i \in R} g_i(\lambda_i^* + \delta_i + \epsilon_i). \tag{6.2.9}$$

Recall that $\sum_{i \in M} \bar{\lambda}_i^* = \sum_{i \in M} \lambda_i^* - \epsilon$, and hence (using the fact that $\sum_{i \in R} \epsilon_i = \epsilon$) one has $\sum_{i \in M} \bar{\lambda}_i^* + \sum_{i \in R} (\lambda_i^* + \epsilon_i) \leq \lambda$, so the policy with $\lambda_i = \bar{\lambda}_i^*$ for all $i \in M$ and $\lambda_i = (\lambda_i^* + \epsilon_i)$ for all $i \in R$ is a feasible static policy under demand rate λ in the sense that it satisfies the constraint $\sum_{i=1}^N \lambda_i \leq \lambda$. Hence, using the fact that the policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ is optimal under λ:

$$\sum_{i \in M} g_i(\lambda_i^*) + \sum_{i \in R} g_i(\lambda_i^*) \geq \sum_{i \in M} g_i(\bar{\lambda}_i^*) + \sum_{i \in R} g_i(\lambda_i^* + \epsilon_i). \tag{6.2.10}$$

By comparison between (6.2.9) and (6.2.10), one then obtains:

$$\sum_{i \in R} \Big(g_i(\lambda_i^* + \epsilon_i) - g_i(\lambda_i^*) \Big) < \sum_{i \in R} \Big(g_i(\lambda_i^* + \delta_i + \epsilon_i) - g_i(\lambda_i^* + \delta_i) \Big). \tag{6.2.11}$$

In order for (6.2.11) to hold, it must be the case that for at least one $j \in R$:

$$g_j(\lambda_j^* + \epsilon_j) - g_j(\lambda_j^*) < g_j(\lambda_j^* + \delta_j + \epsilon_j) - g_j(\lambda_j^* + \delta_j). \tag{6.2.12}$$

Given that $\delta_j \geq 0$ by definition, (6.2.12) can only be possible if $\delta_j > 0$. However, in that case one finds that the function $f_j(x) := g_j(x + \epsilon_j) - g_j(x)$ has increased over the interval $[\lambda_j^*, \lambda_j^* + \delta_j]$. Recalling that $\epsilon_j \geq 0$, this gives a contradiction with the fact that $g_j(\cdot)$ is a strictly *concave* function as shown in the proof of Theorem 6.2.1, and thereby completes the proof. □

The problem of finding an optimal static policy is a relatively straightforward non-linear optimisation problem which can be solved efficiently within mathematical computing packages. The next example illustrates the result of Theorem 6.2.2 by comparing optimal static policies under various different values of the overall demand rate λ in a system with 4 service facilities.

Example 6.2.3. ***(Optimal static policies)***

Consider a system with 4 facilities, with parameters given as follows:

$$\begin{aligned}
&c_1 = 4, \quad && \mu_1 = 1, \quad && \beta_1 = 4, \quad && \alpha_1 = 17, \\
&c_2 = 3, \quad && \mu_2 = 2, \quad && \beta_2 = 3, \quad && \alpha_2 = 10, \\
&c_3 = 2, \quad && \mu_3 = 4, \quad && \beta_3 = 4, \quad && \alpha_3 = 10, \\
&c_4 = 3, \quad && \mu_4 = 5, \quad && \beta_4 = 12, \quad && \alpha_4 = 5.
\end{aligned}$$

For this example, an experiment was conducted in which optimal static policies $(\lambda_1^*, \lambda_2^*, \lambda_3^*, \lambda_4^*)$ were found under 4000 different values of the system demand rate λ ranging from 0 to 40. Figure 6.4 shows the resulting values of λ_1^*, λ_2^*, λ_3^* and λ_4^* for the various values of λ.

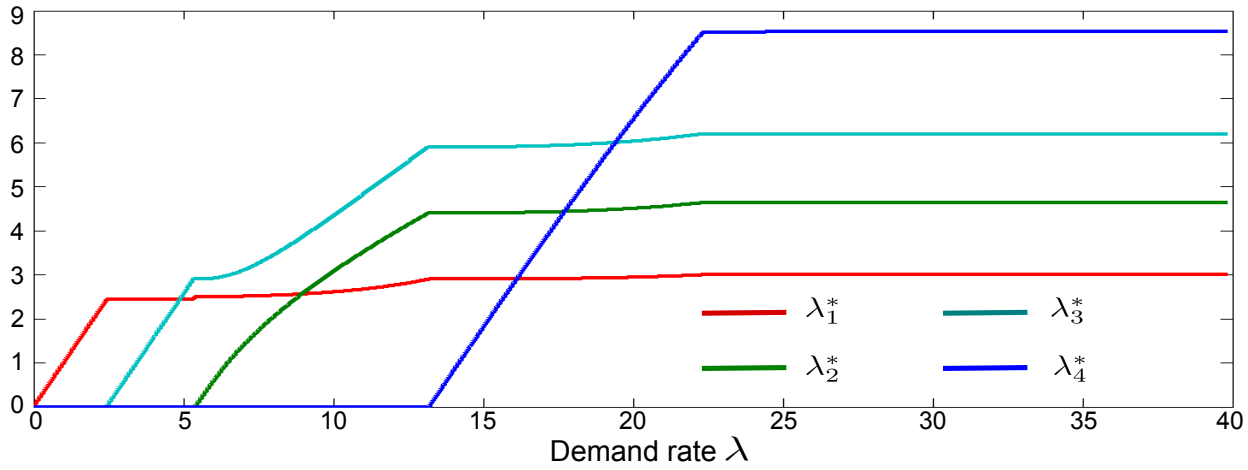


Figure 6.4: Values of λ_1^*, λ_2^*, λ_3^* and λ_4^* for various values of λ.

Figure 6.4 shows that, as expected, the values λ_1^*, λ_2^*, λ_3^* and λ_4^* increase monotonically with the overall demand rate λ. In addition, there are some interesting trends to note. For small values of λ, facility 1 (which has the largest value of service, but the slowest service rate) is preferred to all other facilities by the optimal static policy, but as λ becomes large, the joining rate at facility 1 eventually becomes the smallest of all the facilities. Conversely, facility 4 (which has the smallest value of service, but the fastest service rate) attains a joining rate larger than those of the other facilities for sufficiently large values of λ, but it is not used at all by the optimal static policy when λ is small. As λ increases, eventually all 4 joining rates reach a 'plateau'; naturally, this represents the point at which each 'marginal' long-run average reward $g_i(\lambda_i)$ has reached its unique maximum value (refer to Figure 6.3). Beyond this point, there is clearly no advantage to be gained by increasing the joining rate at any facility, and as such the optimal static policy $(\lambda_1^*, \lambda_2^*, \lambda_3^*, \lambda_4^*)$ remains unchanged (except for the rate at which customers balk) as $\lambda \to \infty$.

Incidentally, the sub-optimality of the optimal static policy (in comparison to the socially optimal policy found by dynamic programming) is negligible for very small values of λ, but appears to increase steadily with λ and reaches 20% when $\lambda \approx 20$, and 33% when $\lambda \approx 40$. Corollary 4.4.8 from Chapter 4 implies that the optimal long-run average reward g^* tends to $\sum_{i=1}^{N} c_i(\alpha_i\mu_i - \beta_i) = 214$

as $\lambda \to \infty$, whereas the average reward earned by the static policy remains at a stable value of approximately 129.8 as λ becomes large, causing sub-optimality of almost 40% . $\boxtimes$

Having shown that a unique optimal static policy exists, the next task is to derive a heuristic policy by applying a single step of policy improvement to the optimal static policy as described in the introduction to this section. Assume that the system has been *uniformised* as described in Section 3.3, so that it evolves in discrete time steps of size $\Delta \in \left(0,\ (\lambda + \sum_{i=1}^{N} c_i\mu_i)^{-1}\right]$. Let $\hat{r}_i(x_i, \lambda_i^*)$ denote the *expected single-step reward earned by facility* $i \in \{1, 2, ..., N\}$ when the number of customers present at i is $x_i \in \mathbb{N}_0$ and the Poisson queue-joining rate at i is $\lambda_i^* \in [0, \lambda]$. At any discrete time step, the probability of facility i being selected by the optimal static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ is λ_i^*/λ; of course, this probability is independent of the system state $\mathbf{x} \in S$. If facility i is selected at a particular time step when the system is in state $\mathbf{x}$, then the reward earned under the 'anticipatory' reward formulation (3.5.15) is $\hat{r}(\mathbf{x}, i)$, irrespective of whether or not a new customer arrival occurs. Hence, $\hat{r}_i(x_i, \lambda_i^*)$ is related to the reward $\hat{r}(\mathbf{x}, i)$ defined in (3.5.15) as follows:

$$\hat{r}_i(x_i, \lambda_i^*) = \frac{\lambda_i^*}{\lambda}\hat{r}(\mathbf{x}, i) = \begin{cases} \lambda_i^* \left(\alpha_i - \dfrac{\beta_i}{\mu_i}\right), & \text{if } x_i < c_i, \\ \lambda_i^* \left(\alpha_i - \dfrac{\beta_i(x_i + 1)}{c_i\mu_i}\right), & \text{if } x_i \geq c_i. \end{cases} \tag{6.2.13}$$

The average reward under the optimal static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ is then given by:

$$g(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*) = \sum_{i=1}^{N} \sum_{x=0}^{\infty} \pi_i(x, \lambda_i^*)\, \hat{r}_i(x, \lambda_i^*),$$

where $\pi_i(x, \lambda_i^*)$ is the steady-state probability of $x \in \mathbb{N}_0$ customers being present at facility i given that it operates with a Poisson arrival rate of λ_i^*; naturally, an expression for $\pi_i(x, \lambda_i^*)$ is given by standard results for $M/M/c$ queues (see, for example, [67]). The derivation of a heuristic routing policy from the optimal static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ to be given in this section begins by following Argon et al. [6]. Let $\delta(\mathbf{x}, i)$ denote the difference in the expected total net reward (over a long period of time) that would result from choosing facility $i \in \{1, 2, ..., N\}$ under state $\mathbf{x} \in S$ at an arbitrary discrete time step $n \in \mathbb{N}_0$ and then following the optimal static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ thereafter, as opposed to simply following the optimal static policy from time n onwards. Given that facilities operate independently under $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ and the probability of an arrival at any

time step is $\lambda\Delta$, an expression for $\delta(\mathbf{x}, i)$ analogous to that in [6] is as follows:

$$\begin{aligned}\delta(\mathbf{x}, i) &= \sum_{j=1}^{N} \frac{\lambda_j^*}{\lambda}\Big(\hat{r}(\mathbf{x}, i) - \hat{r}(\mathbf{x}, j) + \lambda\Delta\left(D_i(x_i, \lambda_i^*) - D_j(x_j, \lambda_j^*)\right)\Big) \\ &\quad + \left(1 - \sum_{j=1}^{N} \frac{\lambda_j^*}{\lambda}\right)\Big(\hat{r}(\mathbf{x}, i) + \lambda\Delta D_i(x_i, \lambda_i^*)\Big) \\ &= \hat{r}(\mathbf{x}, i) + \lambda\Delta D_i(x_i, \lambda_i^*) - \sum_{j=1}^{N} \frac{\lambda_j^*}{\lambda}\Big(\hat{r}(\mathbf{x}, j) + \lambda\Delta D_j(x_j, \lambda_j^*)\Big), \end{aligned} \tag{6.2.14}$$

where $D_i(x_i, \lambda_i^*)$ is the additional long-run aggregate reward gained from having an extra customer at facility i operating under demand rate λ_i^*, given that there are x_i already present. Of course, the option of balking is also available. Since balking does not earn any cost or reward, an expression for $\delta(\mathbf{x}, 0)$ (where 0 represents balking) analogous to (6.2.14) is simply given by:

$$\delta(\mathbf{x}, 0) = -\sum_{j=1}^{N} \frac{\lambda_j^*}{\lambda}\left(\hat{r}(\mathbf{x}, j) + \lambda\Delta D_j(x_j, \lambda_j^*)\right). \tag{6.2.15}$$

Let $\theta^{[B]}$ denote the heuristic policy obtained by applying a single step of policy improvement to the optimal static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$. As discussed previously, the general principle of policy improvement dictates that the actions $\theta^{[B]}(\mathbf{x})$ chosen at the various states $\mathbf{x} \in S$ should optimise the long-run aggregate reward under the assumption that the 'reference' policy to which the improvement step is applied (in this case, the optimal static policy) will be followed after the choice is made. As such, $\theta^{[B]}$ should simply choose an action $a \in \{0, 1, ..., N\}$ which maximises $\delta(\mathbf{x}, a)$ at any given state $\mathbf{x}$. Using (6.2.14) and (6.2.15), this implies that for any $\mathbf{x} \in S$:

$$\theta^{[B]}(\mathbf{x}) \in \begin{cases} \underset{i \in \{1,2,...,N\}}{\arg\max} \Big(\hat{r}(\mathbf{x}, i) + \lambda\Delta D_i(x_i, \lambda_i^*)\Big), & \text{if } \exists\ i \in \{1, 2, ..., N\} \text{ such that} \\ & \hat{r}(\mathbf{x}, i) + \lambda\Delta D_i(x_i, \lambda_i^*) \geq 0, \\ \{0\}, & \text{otherwise.} \end{cases} \tag{6.2.16}$$

The similarity with the criterion given in (6.1.14) for the Whittle index policy $\theta^{[W]}$ is clear; indeed, both heuristic policies are index-based policies, but an explicit expression for $D_i(x_i, \lambda_i^*)$ has yet to be determined. For this purpose, it will be helpful to recall some of the notation used in previous chapters. Recall from the discussion in Section 3.7 that the *relative value function* $h(\mathbf{x})$ represents the long-term advantage of initialising the system in state $\mathbf{x} \in S$, as opposed to beginning in an arbitrary reference state (e.g. state $\mathbf{0}$). Since each facility $i \in \{1, 2, ..., N\}$ operates independently

under the optimal static policy, it will be convenient to use $h_i(x, \lambda_i^*)$ to represent the expected long-term additional net reward *earned by facility i only* for starting with $x \in \mathbb{N}_0$ customers present as opposed to starting with none, given that arrivals occur at the rate $\lambda_i^* \in [0, \lambda]$ corresponding to the optimal static policy. That is, for each facility $i \in \{1, 2, ..., N\}$ and $x \in \mathbb{N}_0$:

$$h_i(x, \lambda_i^*) := \lim_{n \to \infty} \big(V_i^{(n)}(x, \lambda_i^*) - V_i^{(n)}(0, \lambda_i^*) \big), \tag{6.2.17}$$

where $V_i^{(n)}(x, \lambda_i^*)$ denotes the expected total reward at facility i over $n \in \mathbb{N}_0$ discrete time steps, given that the initial state is x and the arrival rate is λ_i^*. In fact, the relative value (also referred to as the *bias*) $h_i(x, \lambda_i^*)$ is analogous to the expression $K_i - gT_i$ used by the authors in [6], where (in their notation) K_i is the expected total reward earned until a particular facility is empty and T_i is the expected time until the facility is empty, in both cases starting with $i \geq 0$ customers present, and g is the long-run average reward earned by the facility. Then, by definition, the quantity $D_i(x_i, \lambda_i^*)$ in (6.2.15) and (6.2.16) is related to the values $h_i(x, \lambda_i^*)$ as follows:

$$D_i(x_i, \lambda_i^*) = h_i(x_i + 1, \lambda_i^*) - h_i(x_i, \lambda_i^*) \quad (i \in \{1, 2, ..., N\},\ x_i \in \mathbb{N}_0,\ \lambda_i^* \in [0, \lambda]).$$

Since $D_i(\cdot)$ and $h_i(\cdot)$ relate to a single facility i operating independently, it will be convenient to focus on an arbitrary facility $i \in \{1, 2, ..., N\}$ in the arguments that follow. The random state evolution at facility i under the optimal static policy is equivalent to that of an isolated $M/M/c_i$ queueing system with a Poisson arrival rate λ_i^*, which always chooses to accept any incoming customer. Hence, results from earlier in this thesis (in particular, those in Section 4.2) imply that the values $h_i(x, \lambda_i^*)$ relating to facility i satisfy the *policy evaluation equations*:

$$g_i(\lambda_i^*) + h_i(x, \lambda_i^*) = \hat{r}_i(x, \lambda_i^*) + \sum_{y \in \mathbb{N}_0} p_i(x, y, \lambda_i^*) h_i(y, \lambda_i^*) \qquad (x \in \mathbb{N}_0), \tag{6.2.18}$$

where $\hat{r}_i(x, \lambda_i^*)$ is the single-facility reward defined in (6.2.13), $g_i(\lambda_i^*)$ is as defined in (6.2.2), and the transition probabilities $p_i(x, y, \lambda_i^*)$ are given for $x, y \in \mathbb{N}_0$ as follows:

$$p_i(x, y, \lambda_i^*) = \begin{cases} \lambda_i^* \Delta, & \text{if } y = x + 1, \\ \min(x, c_i) \mu_i \Delta, & \text{if } y = x - 1, \\ 1 - \lambda_i^* \Delta - \min(x, c_i) \mu_i \Delta, & \text{if } y = x, \\ 0, & \text{otherwise.} \end{cases} \tag{6.2.19}$$

Note that the parameter Δ in (6.2.19) is the uniformisation parameter used for the N-facility system as a whole; that is, $\Delta \in \left(0, (\lambda + \sum_{j=1}^{N} c_j\mu_j)^{-1}\right]$. Since a single-facility scenario is being considered, it would be feasible to modify the value of Δ (e.g. by setting $\Delta = (\lambda_i^* + c_i\mu_i)^{-1}$) in such a way as to reduce the probability $p_i(x, x, \lambda_i^*)$ of a 'self-transition'; however, this approach is not necessary as the final expression obtained for the index $\delta(\mathbf{x}, i)$ will be independent of Δ. Note that one may define $h_i(0, \lambda_i^*) = 0$ in order to be consistent with (6.2.17) and to determine the values $h_i(x, \lambda_i^*)$ in (6.2.18) uniquely; effectively, this designates 0 as the 'reference state'.

One may proceed to use the equations (6.2.18) to derive expressions for the relative values $h_i(x, \lambda_i^*)$ and differences $D_i(x, \lambda_i^*)$ in terms of the system parameters and the average reward $g_i(\lambda_i^*)$. Firstly, however, it should be noted that although the system demand rate λ is always assumed positive, there is a possibility that the demand rate λ_i^* at facility i may be zero under the optimal static policy. In this case, one simply has $D_i(x_i, \lambda_i^*) = 0$ for all $x_i \in \mathbb{N}_0$, so that the expression $\hat{r}(\mathbf{x}, i) + \lambda\Delta D_i(x_i, \lambda_i^*)$ in (6.2.16) reduces to the single-step reward $\hat{r}(\mathbf{x}, i)$. The explanation for this is that $D_i(x_i, \lambda_i^*)$ essentially represents the cumulative increase in the expected waiting costs for customers following the optimal static policy as a result of an extra customer having been admitted to facility i (indeed, it will be shown later that $D_i(x_i, \lambda_i^*)$ is always non-positive). If customers never join i under the static policy, then there are no adverse effects caused to these customers as a result of an extra customer being present at i; thus, $D_i(x_i, \lambda_i^*) = 0$. The derivation of general expressions for $h_i(x, \lambda_i^*)$ and $D_i(x, \lambda_i^*)$ given here will proceed under the assumption that $\lambda_i^* > 0$.

By setting $x = 0$ in (6.2.18), one obtains the following expression for $h_i(1, \lambda_i^*) = 0$:

$$h_i(1, \lambda_i^*) = \frac{g_i(\lambda_i^*) - \hat{r}_i(0, \lambda_i^*)}{\lambda_i^*\Delta}.$$

Furthermore, by definition of $D_i(x, \lambda_i^*)$ and the fact that $h_i(0, \lambda_i^*) = 0$:

$$D_i(0, \lambda_i^*) = h_i(1, \lambda_i^*) - h_i(0, \lambda_i^*) = h_i(1, \lambda_i^*) = \frac{g_i(\lambda_i^*) - \hat{r}_i(0, \lambda_i^*)}{\lambda_i^*\Delta}. \tag{6.2.20}$$

In general, for integers $1 \leq x < c_i$, one obtains the following from (6.2.18):

$$\begin{aligned} g_i(\lambda_i^*) + h_i(x, \lambda_i^*) = \hat{r}_i(x, \lambda_i^*) + \lambda_i^*\Delta h_i(x+1, \lambda_i^*) + x\mu_i\Delta h_i(x-1, \lambda_i^*) \\ + (1 - \lambda_i^*\Delta - x\mu_i\Delta)h_i(x, \lambda_i^*). \end{aligned}$$

This implies, following some simple manipulations:

$$h_i(x+1,\lambda_i^*) - h_i(x,\lambda_i^*) = \frac{x\mu_i}{\lambda_i^*}\big(h_i(x,\lambda_i^*) - h_i(x-1,\lambda_i^*)\big) + \frac{g_i(\lambda_i^*) - \hat{r}_i(x,\lambda_i^*)}{\lambda_i^*\Delta}.$$

Thus, for $1 \leq x < c_i$, one has the following recurrence relationship for $D_i(x,\lambda_i^*)$:

$$D_i(x,\lambda_i^*) = \frac{x\mu_i}{\lambda_i^*}\, D_i(x-1,\lambda_i^*) + \frac{g_i(\lambda_i^*) - \hat{r}_i(x,\lambda_i^*)}{\lambda_i^*\Delta}. \tag{6.2.21}$$

Using recursive substitution with (6.2.21) one then obtains, for $0 \leq x < c_i$:

$$D_i(x,\lambda_i^*) = \sum_{k=0}^{x} \frac{x!}{k!} \left(\frac{\mu_i}{\lambda_i^*}\right)^{x-k} \left(\frac{g_i(\lambda_i^*) - \hat{r}_i(k,\lambda_i^*)}{\lambda_i^*\Delta}\right). \tag{6.2.22}$$

Equivalently, using the fact that $\hat{r}_i(x,\lambda_i^*) = \hat{r}_i(0,\lambda_i^*)$ for all non-negative integers $x < c_i$:

$$D_i(x,\lambda_i^*) = \left(\frac{g_i(\lambda_i^*) - \hat{r}_i(0,\lambda_i^*)}{\lambda_i^*\Delta}\right) \sum_{k=0}^{x} \frac{x!}{k!} \left(\frac{\mu_i}{\lambda_i^*}\right)^{x-k}. \tag{6.2.23}$$

On the other hand, for integers $x \geq c_i$, the recurrence relationship is as follows:

$$D_i(x,\lambda_i^*) = \frac{c_i\mu_i}{\lambda_i^*}\, D_i(x-1,\lambda_i^*) + \frac{g_i(\lambda_i^*) - \hat{r}_i(x,\lambda_i^*)}{\lambda_i^*\Delta}. \tag{6.2.24}$$

Using (6.2.23) and (6.2.24) and applying a simple inductive argument, one can then show that for integers $x \geq c_i$, the following expression is valid for $D_i(x,\lambda_i^*)$:

$$\begin{aligned} D_i(x,\lambda_i^*) = {} & \left(\frac{g_i(\lambda_i^*) - \hat{r}_i(0,\lambda_i^*)}{\lambda_i^*\Delta}\right) \sum_{k=0}^{c_i-1} \frac{c_i!\, c_i^{x-c_i}}{k!} \left(\frac{\mu_i}{\lambda_i^*}\right)^{x-k} \\ & + \sum_{k=c_i}^{x} \left(\frac{c_i\mu_i}{\lambda_i^*}\right)^{x-k} \left(\frac{g_i(\lambda_i^*) - \hat{r}_i(k,\lambda_i^*)}{\lambda_i^*\Delta}\right). \end{aligned} \tag{6.2.25}$$

Recall from (6.2.16) that the heuristic policy $\theta^{[B]}$ chooses a facility i at state $\mathbf{x} \in S$ which maximises $\hat{r}(\mathbf{x},i) + \lambda\Delta D_i(x_i,\lambda_i^*)$ (or chooses to balk, if all such quantities are negative). Using (6.2.23) and (6.2.25), one may write an expression for $\hat{r}(\mathbf{x},i) + \lambda\Delta D_i(x_i,\lambda_i^*)$ as follows:

$$\hat{r}(\mathbf{x},i) + \lambda\Delta D_i(x_i,\lambda_i^*) = \begin{cases} \hat{r}(\mathbf{x},i) + \dfrac{\lambda}{\lambda_i^*}\Big(g_i(\lambda_i^*) - \hat{r}_i(0,\lambda_i^*)\Big) \displaystyle\sum_{k=0}^{x_i} \frac{x_i!}{k!} \left(\frac{\mu_i}{\lambda_i^*}\right)^{x_i-k}, & \text{if } x_i < c_i, \\ \hat{r}(\mathbf{x},i) + \dfrac{\lambda}{\lambda_i^*}\Big(g_i(\lambda_i^*) - \hat{r}_i(0,\lambda_i^*)\Big) \displaystyle\sum_{k=0}^{c_i-1} \frac{c_i!\, c_i^{x_i-c_i}}{k!} \left(\frac{\mu_i}{\lambda_i^*}\right)^{x_i-k} \\ \qquad + \dfrac{\lambda}{\lambda_i^*} \displaystyle\sum_{k=c_i}^{x_i} \left(\frac{c_i\mu_i}{\lambda_i^*}\right)^{x_i-k} \Big(g_i(\lambda_i^*) - \hat{r}_i(k,\lambda_i^*)\Big), & \text{if } x_i \geq c_i. \end{cases} \tag{6.2.26}$$

As mentioned previously, it is assumed that $\lambda_i^* > 0$ in (6.2.26); if $\lambda_i^* = 0$, then one trivially has $D_i(x_i, \lambda_i^*) = 0$ for all $x_i \in \mathbb{N}_0$ and hence $\hat{r}(\mathbf{x}, i) + \lambda \Delta D_i(x_i, \lambda_i^*) = \hat{r}(\mathbf{x}, i)$.

The expressions for $\hat{r}(\mathbf{x}, i) + \lambda \Delta D_i(x_i, \lambda_i^*)$ given in (6.2.26) will serve as the indices used by the heuristic policy $\theta^{[B]}$ for choosing its preferred facility under any given state. It is important to note that these expressions depend only on the system parameters and the demand rates λ_i^* associated with the optimal static policy. The average rewards $g_i(\lambda_i^*)$ for the various facilities can be obtained easily using (6.2.2) after the optimal static policy has been found. Thus, although the relative values $h_i(x, \lambda_i^*)$ and differences $D_i(x, \lambda_i^*)$ were used in the derivation of the indices given in (6.2.26), the quantities $h_i(x, \lambda_i^*)$ and $D_i(x, \lambda_i^*)$ do not appear in the indices themselves. This enables the policy $\theta^{[B]}$ to be implemented without any requirement for time-consuming algorithmic methods to be used in order to find quantities such as $h_i(x, \lambda_i^*)$, $D_i(x, \lambda_i^*)$ or any other expressions of the type that might be involved in a dynamic programming algorithm; essentially, all that is required is the determination of the optimal static policy. As discussed previously, it is important that a heuristic policy should be easy to implement without being encumbered by the dimensionality of the state space, or any other obstacles which might be associated with a large-scale problem.

Some further manipulations of the indices in (6.2.26) are possible, although these are largely cosmetic in nature and serve only to suppress the appearance of the factor (λ/λ_i^*). Firstly, recall that $\hat{r}(\mathbf{x}, i) = \lambda w(\mathbf{x}, i)$, where $w(\mathbf{x}, i)$ is an individual customer's expected net reward for joining facility i under state $\mathbf{x}$, defined in (4.1.1). Since λ appears as a factor in all of the other terms in (6.2.26), it is reasonable to divide by λ throughout, so that effectively one considers the quantity $w(\mathbf{x}, i) + \Delta D_i(x_i, \lambda_i^*)$ as opposed to $\hat{r}(\mathbf{x}, i) + \lambda \Delta D_i(x_i, \lambda_i^*)$. Obviously, dividing by λ simply re-scales the indices and does not affect the actions chosen by the policy $\theta^{[B]}$. Note that whether one makes this adjustment or not, the indices remain independent of Δ. Also, consider the expression $\left(g_i(\lambda_i^*) - \hat{r}_i(x, \lambda_i^*)\right)/\lambda_i^*$. By introducing a quantity $w_i(x)$ to represent a customer's expected net reward for joining facility i when there are x customers present, one has the relationship $\hat{r}_i(x, \lambda_i^*) = \lambda_i^* w_i(x)$, and hence $\left(g_i(\lambda_i^*) - \hat{r}_i(x, \lambda_i^*)\right)/\lambda_i^* = g_i(\lambda_i^*)/\lambda_i^* - w_i(x)$. Then, by letting $M_i(\lambda_i^*)$ denote the average waiting time of customers who join facility i at a Poisson rate λ_i^*, one has $L_i(\lambda_i^*) = \lambda_i^* M_i(\lambda_i^*)$ due to Little's formula (see, [67], p. 10) and hence:

$$g_i(\lambda_i^*)/\lambda_i^* = \left(\lambda_i^* \alpha_i - \beta_i L_i(\lambda_i^*)\right)/\lambda_i^* = \alpha_i - \beta_i M_i(\lambda_i^*), \tag{6.2.27}$$

where the expression obtained in (6.2.27) is obviously equal to the steady-stage average expected net reward earned by individual customers arriving at a rate λ_i^*. Let this 'individual average reward' be denoted by $\overline{w}_i(\lambda_i^*)$. That is, for facilities $i \in \{1, 2, ..., N\}$ and $x \in \mathbb{N}_0$:

$$w_i(x) = \begin{cases} \alpha - \dfrac{\beta_i}{\mu_i}, & \text{if } x < c_i, \\ \alpha - \dfrac{\beta_i(x+1)}{c_i\mu_i}, & \text{if } x \geq c_i, \end{cases} \tag{6.2.28}$$

$$\overline{w}_i(\lambda_i^*) = \sum_{x=0}^{\infty} \pi_i(x, \lambda_i^*) w_i(x) = \alpha_i - \beta_i M_i(\lambda_i^*). \tag{6.2.29}$$

Of course, $M_i(\lambda_i^*)$ (like $L_i(\lambda_i^*)$) can be obtained using the standard $M/M/c$ formulae in texts such as [67] (p. 126). So, by the preceding arguments, $\left(g_i(\lambda_i^*) - \hat{r}_i(x, \lambda_i^*)\right)/\lambda_i^* = \overline{w}_i(\lambda_i^*) - w_i(x)$; that is, the difference between the expected long-run average reward for individual customers under demand rate λ_i^* and the expected reward for a customer who joins under state x. The next definition formalises these arguments by providing refined formulae for the indices used by $\theta^{[B]}$.

Definition 6.2.4. ***(Bernoulli index heuristic)***

Let the heuristic policy $\theta^{[B]}$ obtained by applying a single step of policy improvement to the optimal static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ be referred to as the Bernoulli index heuristic *policy. Then, for each state $\mathbf{x} \in S$, the action $\theta^{[B]}(\mathbf{x})$ chosen by the policy $\theta^{[B]}$ satisfies:*

$$\theta^{[B]}(\mathbf{x}) \in \begin{cases} \underset{i \in \{1,2,...,N\}}{\arg\max}\ P_i(x_i, \lambda_i^*), & \text{if } \exists\ i \in \{1, 2, ..., N\} \text{ such that } P_i(x_i, \lambda_i^*) \geq 0, \\ \{0\}, & \text{otherwise,} \end{cases} \tag{6.2.30}$$

where, for $i \in \{1, 2, ..., N\}$ and $x \in \mathbb{N}_0$, the index $P_i(x, \lambda_i^*)$ is given by:

$$P_i(x, \lambda_i^*) = \begin{cases} w_i(x) + \Big(\overline{w}_i(\lambda_i^*) - w_i(0)\Big) \displaystyle\sum_{k=0}^{x} \frac{x!}{k!} \left(\frac{\mu_i}{\lambda_i^*}\right)^{x-k}, & \text{if } \lambda_i^* > 0 \text{ and } x < c_i, \\ w_i(x) + \Big(\overline{w}_i(\lambda_i^*) - w_i(0)\Big) \displaystyle\sum_{k=0}^{c_i-1} \frac{c_i!\, c_i^{x-c_i}}{k!} \left(\frac{\mu_i}{\lambda_i^*}\right)^{x-k} & \\ \qquad + \displaystyle\sum_{k=c_i}^{x} \left(\frac{c_i\mu_i}{\lambda_i^*}\right)^{x-k} \Big(\overline{w}_i(\lambda_i^*) - w_i(k)\Big), & \text{if } \lambda_i^* > 0 \text{ and } x \geq c_i, \\ w_i(x), & \text{if } \lambda_i^* = 0. \end{cases} \tag{6.2.31}$$

In cases where two or more facilities attain the maximum in (6.2.30), it will be assumed that a decision is made according to some fixed ranking order of the N facilities.

The next example demonstrates the performance of the Bernoulli index heuristic policy in a small system consisting of two facilities with two servers each.

Example 6.2.5. ***(Bernoulli index heuristic with two dual-server facilities)***

This example revisits the same set of parameters used in Examples 4.1.1 and 6.1.6. To recap, there is a demand rate $\lambda = 10$ and the parameters for the two facilities are:

$$\begin{aligned} c_1 &= 2, \qquad \mu_1 = 8, \qquad \beta_1 = 10, \qquad \alpha_1 = 2, \\ c_2 &= 2, \qquad \mu_2 = 2, \qquad \beta_2 = 10, \qquad \alpha_2 = 6. \end{aligned}$$

In Example 4.1.1, the selfishly optimal policy $\tilde{\theta}$ and socially optimal policy θ^* (found by relative value iteration) for the system were derived; refer to Table 4.1 (page 127). As discussed in Example 6.1.6, the optimal policy θ^* has a slightly unusual characteristic, in that it does not always choose the same decision at states where both facilities have an idle server; specifically, the actions chosen by θ^* at the states $(0,0)$ and $(0,1)$ differ from each other, as do the actions chosen at $(1,0)$ and $(1,1)$. The Whittle index policy $\theta^{[W]}$ discussed in the previous section always prefers the same facility among those which have at least one idle server, regardless of how many idle servers are available. In this particular system, it chooses facility 2 at all four of the aforementioned states and as a result fails to attain optimality (its sub-optimality is about 0.35%).

Now consider the approach based on deriving a heuristic policy via an optimal static routing policy as discussed in this section thus far. Using Python's OpenOpt package [138], the optimal rates λ_1^* and λ_2^* (rounded to 3 d.p.) associated with the policy $(\lambda_1^*, \lambda_2^*)$ are obtained as:

$$\lambda_1^* = 6.230, \qquad \lambda_2^* = 0.981.$$

Thus, the rate at which customers balk under the static policy $(\lambda_1^*, \lambda_2^*)$ is $\lambda - \lambda_1^* - \lambda_2^* = 2.789$. The long-run average rewards $g_1(\lambda_1^*)$ and $g_2(\lambda_2^*)$ are then given by (6.2.2) as:

$$g_1(\lambda_1^*) = 3.281, \qquad g_2(\lambda_2^*) = 0.667,$$

from which it follows that the overall average reward under the optimal static policy is $g_1(\lambda_1^*) + g_2(\lambda_2^*) = 3.948$, which is approximately 45% short of the optimal value $g^* = 7.282$. The next stage is to derive the heuristic policy $\theta^{[B]}$ by applying a policy improvement step. Table 6.4 shows the Bernoulli indices $P_1(x, \lambda_1^*)$ for facility 1 (computed using 6.2.31) for $x \in \{0, 1, 2, 3\}$.

x	0	1	2	3
$P_1(x, \lambda_1^*)$	0.526	0.239	-0.786	-1.810

Table 6.4: The Bernoulli indices $P_1(x, \lambda_1^*)$ for $x \in \{0, 1, 2, 3\}$.

Similarly, Table 6.5 shows the Bernoulli indices $P_2(x, \lambda_2^*)$ for $x \in \{0, 1, 2\}$.

x	0	1	2
$P_2(x, \lambda_2^*)$	0.680	0.028	-3.285

Table 6.5: The Bernoulli indices $P_2(x, \lambda_2^*)$ for $x \in \{0, 1, 2\}$.

Note that the indices $P_1(0, \lambda_1^*)$ and $P_1(1, \lambda_1^*)$ differ from each other, as do $P_2(0, \lambda_2^*)$ and $P_2(1, \lambda_2^*)$. This shows that the Bernoulli index policy does not suffer from the same limitation as the Whittle policy, which (in general) always has $W_i(x_i) = W_i(0)$ for all $x_i < c_i$. The decisions chosen by $\theta^{[B]}$ at the various states $\mathbf{x} \in \tilde{S}$ are then specified by (6.2.30), and are shown in Table 6.6.

	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	2	1	1
$x_1 = 1$	2	1	1
$x_1 = 2$	2	2	0
$x_1 = 3$	2	2	0

Table 6.6: The Bernoulli index policy $\theta^{[B]}$ for the system in Example 6.2.5.

By comparison with Tables 4.1 and 6.3, it may be seen that the Bernoulli policy $\theta^{[B]}$ is equivalent to the optimal policy θ^* at all states $\mathbf{x} \in \tilde{S}$, and differs from the Whittle policy $\theta^{[W]}$ at states $(0, 1)$ and $(1, 1)$. Thus, for this system, $\theta^{[B]}$ succeeds in attaining optimality. Although this example has shown that $\theta^{[B]}$ has an advantage over $\theta^{[W]}$ in that it has the ability to distinguish between two different 'idle' states $x_i < c_i$ and $y_i < c_i$ at a particular facility i, this certainly does not imply

that $\theta^{[B]}$ always performs better than $\theta^{[W]}$; in fact, $\theta^{[W]}$ has certain appealing properties of its own (discussed in Section 6.1) which are not shared by $\theta^{[B]}$, as later results will show. ⊠

As one might expect, the Bernoulli policy $\theta^{[B]}$ determined by the indices in (6.2.31) possesses various structural properties which have been discussed in previous chapters and associated with the selfishly optimal policy $\tilde{\theta}$, the Whittle policy $\theta^{[W]}$ and (in some cases only) the optimal policy θ^* found by relative value iteration. The next result summarises these properties.

Theorem 6.2.6. *The Bernoulli index policy $\theta^{[B]}$ possesses the following properties:*

1. *(Containment.) Let S_B be the set of positive recurrent states under the policy $\theta^{[B]}$. Then $S_B \subseteq \tilde{S}$, where $\tilde{S}$ is the corresponding set for the selfish policy $\tilde{\theta}$.*

2. *(First monotonicity property.) Suppose $\theta^{[B]}(\mathbf{x}) = 0$ for some state $\mathbf{x} \in S$. Then $\theta^{[B]}(\mathbf{x}^{i+}) = 0$ for all facilities $i \in \{1, 2, ..., N\}$.*

3. *(Second monotonicity property.) Suppose $\theta^{[B]}(\mathbf{x}) = i$ for some state $\mathbf{x} \in S$ and facility $i \in \{1, 2, ..., N\}$ with $x_i \geq 1$. Then $\theta^{[B]}(\mathbf{x}^{i-}) = i$.*

4. *(Third monotonicity property.) Suppose $\theta^{[B]}(\mathbf{x}) = i$ for some state $\mathbf{x} \in S$ and facility $i \in \{1, 2, ..., N\}$. Then $\theta^{[B]}(\mathbf{x}^{j+}) = i$ for all facilities $j \in \{1, 2, ..., N\} \setminus \{i\}$.*

5. *(Sink property.) The policy $\theta^{[B]}$ is a sink policy with a sink state $\mathbf{z} \in S$ satisfying $z_i = \min\{x \in \mathbb{N}_0 : P_i(x, \lambda_i^*) < 0\}$ for all facilities $i \in \{1, 2, ..., N\}$.*

6. *(Conservativity with demand.) Let $S_B(\lambda)$ be the set of positive recurrent states under the Bernoulli index policy $\theta_\lambda^{[B]}$ given a demand rate $\lambda > 0$. Then, for any pair of demand rates $\lambda, \bar{\lambda} > 0$ with $\lambda < \bar{\lambda}$, it is the case that $S_B(\bar{\lambda}) \subseteq S_B(\lambda)$.*

Proof. Recall that the index $P_i(x, \lambda_i^*)$ defined in (6.2.31) is equivalent to the expression $w_i(x) + \Delta D_i(x, \lambda_i^*)$, where $w_i(x)$ is as defined in (6.2.28), $D_i(x, \lambda_i^*) = h_i(x+1, \lambda_i^*) - h_i(x, \lambda_i^*)$ and $h_i(x, \lambda_i^*) = \lim_{n\to\infty} \left(V_i^{(n)}(x, \lambda_i^*) - V_i^{(n)}(0, \lambda_i^*) \right)$ by definition. It follows that structural properties of the indices $P_i(x, \lambda_i^*)$ (for example, monotonicity) can be proved using the finite-stage values $V_i^{(n)}(x, \lambda_i^*)$, and this can be done using inductive arguments similar to those used in previous chapters. In fact, the arguments required in this proof are especially simple, for two main reasons:

1. It is sufficient to consider an arbitrary facility $i \in \{1, 2, ..., N\}$ operating in isolation, as was done in the derivation of the indices $P_i(x, \lambda_i^*)$ themselves;

2. Under the static policy, each facility $i \in \{1, 2, ..., N\}$ operates as an $M/M/c_i$ queue with a demand rate λ_i^* in which *every customer who arrives joins the queue*, so it is not necessary to consider different actions that may be chosen at the various states.

The proof of property (6) requires the result of Theorem 6.2.2, which states that each demand rate λ_i^* is monotonically increasing with λ. For details, see Appendix A.6, page 456. □

Theorem 6.2.6 makes no mention of a *seventh* structural property which was included in the analogous result (Theorem 6.1.7) for the Whittle policy $\theta^{[W]}$ in the previous section; specifically, the *non-idling* property. Somewhat surprisingly, this property fails to hold for the Bernoulli index policy. Indeed, the next example shows that $\theta^{[B]}$ may be an idling policy.

Example 6.2.7. ***(The Bernoulli index policy $\theta^{[B]}$ may be an idling policy)***

It is sufficient to consider a system consisting of only one facility ($N = 1$), with a demand rate $\lambda = 5$ and parameters for the single facility given as follows:

$$c = 5, \qquad \mu = 2, \qquad \beta = 15, \qquad \alpha = 8.$$

A notable characteristic of this system is that the reward for service α is only slightly greater than the expected cost incurred during a service time, β/μ. The optimal static policy λ_1^* (representing the optimal Poisson queue-joining rate) is, to 3 decimal places:

$$\lambda_1^* = 3.595.$$

Thus, customers balk at a rate $\lambda_0^* = \lambda - \lambda_1^* = 1.405$ under the static policy. Table 6.7 shows the indices $P_1(x, \lambda_1^*)$ for states $x \in \{0, 1, 2, 3, 4, 5\}$ obtained using (6.2.31), and the resulting decisions made by the Bernoulli policy $\theta^{[B]}$. Of course, $\theta^{[B]}(x) = 1$ if and only if $P_1(x, \lambda_1^*) \geq 0$.

Although there are 5 service channels available, $\theta^{[B]}(x)$ chooses to balk at states 3 and 4. Hence, $\theta^{[B]}$ fails to achieve the non-idling property. Theorem 4.3.3 has already established that, given any number of facilities N, the optimal policy θ^* found by relative value iteration is always a non-idling policy, which suggests that this characteristic of $\theta^{[B]}$ is somewhat undesirable.

x	0	1	2	3	4	5
$P_1(x, \lambda_1^*)$	0.406	0.353	0.242	-0.024	-0.761	-3.103
$\theta^{[B]}(x)$	1	1	1	0	0	0

Table 6.7: The Bernoulli indices $P_1(x, \lambda_1^*)$ and decisions $\theta^{[B]}(x)$ for $x \in \{0, 1, 2, 3, 4, 5\}$.

Some comments are appropriate in order to explain why $\theta^{[B]}$ fails to exhibit the non-idling property in this particular system. The selfish threshold $\tilde{B}_1$ in this system may be calculated using (4.1.2) as $\lfloor \alpha c\mu/\beta \rfloor = 5$, which is equal to the number of service channels c. Thus, even the selfish policy $\tilde{\theta}$ (which is the least conservative among all of the state-dependent policies considered in this thesis thus far) chooses to balk when all of the servers are occupied. An individual customer's expected net reward for joining under some state $x < 5$ is $\alpha - \beta/\mu = 0.5$, which is drastically smaller (in terms of absolute value) than the net reward $\alpha - 2\beta/\mu = -7$ that would be obtained by joining under the state $x = 5$. In general, it is clear that for each extra service time spent waiting in the system by a customer, the expected net reward decreases steeply, and therefore it is extremely important to prevent customers from joining at any state $x \geq 5$ if possible. The Bernoulli index policy $\theta^{[B]}$ chooses actions based on the assumption that the optimal static policy will be followed at all times thereafter. Since the static policy always applies the same randomised decision rule and allows customers to join with probability $\lambda_1^*/\lambda \approx 0.7$ regardless of the state, the Bernoulli policy $\theta^{[B]}$ is obliged to over-compensate by rejecting customers at states 3 and 4, thereby reducing the likelihood of any state $x \geq 5$ being reached after adoption of the static policy. ⊠

In Section 6.1, it was shown that the Whittle index policy $\theta^{[W]}$ is asymptotically optimal in a light-traffic limit and also optimal in a heavy-traffic limit. It is therefore of interest to investigate whether or not the Bernoulli index policy $\theta^{[B]}$ shares these properties. In doing so, it will also be possible to establish whether or not the optimal static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ itself (used to derive the Bernoulli policy) is optimal as $\lambda \to 0$ and/or as $\lambda \to \infty$. Before proceeding, it will be useful to make the observation that the optimal static policy can never be average reward optimal (over all admissible policies) for any fixed $\lambda > 0$. This will be stated here as a theorem.

Theorem 6.2.8. *Let the system demand rate $\lambda > 0$ be fixed. Then any static routing policy $(\lambda_1, \lambda_2, ..., \lambda_N)$ is sub-optimal, in the sense that:*

$$g(\lambda_1, \lambda_2, ..., \lambda_N) < g^*,$$

where $g(\lambda_1, \lambda_2, ..., \lambda_N)$ is the expected long-run average reward earned by the static policy and $g^* = \sup_\theta g_\theta$ is the optimal average reward over all admissible policies.

Proof. Under the assumption that $\alpha_i - \beta_i/\mu_i > 0$ for all $i \in \{1, 2, ..., N\}$, the optimal average reward g^* must be strictly positive, as discussed at the beginning of the proof of Theorem 4.4.4. Hence, the trivial static policy with $\lambda_i = 0$ for all i (and hence $g(\lambda_1, \lambda_2, ..., \lambda_N) = 0$) is sub-optimal. On the other hand, consider a static policy which has $\lambda_i > 0$ for some facility i. Then, assuming that $\lambda_i < c_i\mu_i$ (otherwise the average reward $g_i(\lambda_i)$ at facility i would be negatively infinite, again implying sub-optimality) all 'marginal' states $x_i \in \mathbb{N}_0$ at facility i are positive recurrent under the static policy; indeed, the stationary probability $\pi_i(x_i, \lambda_i)$ for any $x_i \in \mathbb{N}_0$ is strictly positive and given by standard $M/M/c_i$ queue formulae (e.g. [67], p. 69). In particular, states $x_i \geq \tilde{B}_i$ are positive recurrent, where $\tilde{B}_i$ is the selfishly optimal threshold defined in (4.1.2).

It follows that for any marginal state $x_i \geq \tilde{B}_i$, there will be infinitely many points in time at which the static policy allows a customer to join facility i when it is in state x_i, thereby earning a *negative* reward $\hat{r}_i(x_i, \lambda_i)$. It is clear, therefore, that the static policy is strictly inferior to a non-static, history-dependent policy ψ which chooses the same actions as the static policy $(\lambda_1, \lambda_2, ..., \lambda_N)$ at all times, *unless* the static policy chooses to admit a customer to facility i under some state $x_i \geq \tilde{B}_i$, in which case ψ chooses to balk. This can be established formally using a sample path argument completely analogous to the argument given in the proof of Theorem 4.2.5. □

The next result shows that, despite its sub-optimality for fixed values of λ, the optimal static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ *does* attain asymptotic optimality in a light-traffic limit. The fact that the Bernoulli policy $\theta^{[B]}$ shares the same property is an immediate consequence of this result. The notation $\theta_\lambda^{[B]}$ will be used, as opposed to $\theta^{[B]}$, in order to reflect the dependence of the Bernoulli policy on λ (similar to the notation $\theta_\lambda^{[W]}$ used for the Whittle policy in Theorem 6.1.8).

Theorem 6.2.9. *The optimal static policy $(\lambda_1^*, \lambda_2^*, ...\lambda_N^*)$ and the Bernoulli index policy $\theta_\lambda^{[B]}$ are both asymptotically optimal in a light-traffic limit, in the sense that:*

$$\lim_{\lambda \to 0} \left(\frac{g^*(\lambda) - g(\lambda_1^*, \lambda_2^*, ...\lambda_N^*)}{g^*(\lambda)} \right) = \lim_{\lambda \to 0} \left(\frac{g^*(\lambda) - g_{\theta^{[B]}}(\lambda)}{g^*(\lambda)} \right) = 0,$$

where $g^*(\lambda)$ and $g_{\theta^{[B]}}(\lambda)$ respectively denote the optimal expected long-run average reward over all admissible policies and the average reward earned by $\theta_\lambda^{[B]}$, given a demand rate $\lambda > 0$.

Proof. First, let $(\lambda_1, \lambda_2, ..., \lambda_N)$ be any static routing policy under which the system is stable; that is, $\lambda_i < c_i\mu_i$ for all $i \in \{1, 2, ..., N\}$. Then, by analogy with (6.2.27) and (6.2.28), the expected long-run average reward $g(\lambda_1, \lambda_2, ...\lambda_N)$ earned by this policy satisfies:

$$g(\lambda_1, \lambda_2, ...\lambda_N) = \sum_{i=1}^{N} \lambda_i \overline{w}_i(\lambda_i) = \sum_{i=1}^{N} \lambda_i \sum_{x=0}^{\infty} \pi_i(x, \lambda_i) w_i(x). \tag{6.2.32}$$

Recall that $\overline{w}_i(\lambda_i) = \alpha_i - \beta_i M_i(\lambda_i)$, where $M_i(\lambda_i)$ is the mean waiting time for customers at facility i under demand rate λ_i. It is a simple matter to show, using standard $M/M/c$ queue formulae ([67], p. 69) that $M_i(\lambda_i) \to 1/\mu_i$ as $\lambda_i \to 0$ (equivalently, a customer's expected time waiting in the queue at facility i tends to zero as $\lambda_i \to 0$). It is also trivial to show, for each facility i, that $\lambda_i \to 0$ as the overall demand rate λ tends to zero. Hence, for $i \in \{1, 2, ..., N\}$:

$$\lim_{\lambda \to 0} \overline{w}_i(\lambda_i) = w_i(0) = \alpha_i - \frac{\beta_i}{\mu_i}. \tag{6.2.33}$$

Let $J \subseteq \{1, 2, ..., N\}$ denote the set of facilities which maximise $\alpha_i - \beta_i/\mu_i$. That is:

$$J = \underset{i \in \{1,2,...,N\}}{\arg\max} \left\{ \alpha_i - \frac{\beta_i}{\mu_i} \right\}.$$

Let $(\bar{\lambda}_1, \bar{\lambda}_2, ..., \bar{\lambda}_N)$ be a static policy which, given any demand rate $\lambda > 0$, always chooses to join some facility $i \in J$ with probability one; that is, $\bar{\lambda}_i = \lambda$ for some $i \in J$ and $\bar{\lambda}_j = 0$ for all facilities $j \neq i$. Obviously the potential instability of the system under the policy $(\bar{\lambda}_1, \bar{\lambda}_2, ..., \bar{\lambda}_N)$ is not an issue, since the limiting behaviour as $\lambda \to 0$ is being considered, and as such one can restrict attention to demand rates $\lambda < c_i\mu_i$. Then, using (6.2.32) and (6.2.33):

$$\lim_{\lambda \to 0} \frac{g(\bar{\lambda}_1, \bar{\lambda}_2, ..., \bar{\lambda}_N)}{\lambda} = \lim_{\lambda \to 0} \sum_{i=1}^{N} \frac{\bar{\lambda}_i}{\lambda} \overline{w}_i(\bar{\lambda}_i) = \max_{i \in \{1,2,...,N\}} \left\{ \alpha_i - \frac{\beta_i}{\mu_i} \right\}. \tag{6.2.34}$$

Meanwhile, since $w_i(x) \leq w_i(0)$ for each $i \in \{1, 2, ..., N\}$ and $x \in \mathbb{N}_0$, it is clear from (6.2.32) that the average reward $g(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ for the optimal static policy must satisfy:

$$\frac{g(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)}{\lambda} \leq \max_{i \in \{1,2,...,N\}} w_i(0) = \max_{i \in \{1,2,...,N\}} \left\{ \alpha_i - \frac{\beta_i}{\mu_i} \right\}.$$

Hence, given any demand rate $\lambda > 0$, due to the (static) optimality of $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$:

$$\frac{g(\bar{\lambda}_1, \bar{\lambda}_2, ..., \bar{\lambda}_N)}{\lambda} \leq \frac{g(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)}{\lambda} \leq \max_{i \in \{1,2,...,N\}} \left\{ \alpha_i - \frac{\beta_i}{\mu_i} \right\}.$$

Then, applying the 'squeeze theorem' ([166], p. 909) and recalling (6.2.34):

$$\lim_{\lambda\to 0}\frac{g(\lambda_1^*,\lambda_2^*,...,\lambda_N^*)}{\lambda}=\max_{i\in\{1,2,...,N\}}\left\{\alpha_i-\frac{\beta_i}{\mu_i}\right\}. \tag{6.2.35}$$

In the proof of Theorem 4.4.4, it was shown (beginning on page 151) that the optimal expected long-run average reward (over all admissible policies) $g^*(\lambda)$ satisfies:

$$\lim_{\lambda\to 0}\frac{g^*(\lambda)}{\lambda}=\max_{i\in\{1,2,...,N\}}\left\{\alpha_i-\frac{\beta_i}{\mu_i}\right\}. \tag{6.2.36}$$

So, by combining (6.2.35) and (6.2.36), one can show:

$$\lim_{\lambda\to 0}\left(\frac{g^*(\lambda)-g(\lambda_1^*,\lambda_2^*,...\lambda_N^*)}{g^*(\lambda)}\right)=\frac{\max_i\{\alpha_i-\beta_i/\mu_i\}-\max_i\{\alpha_i-\beta_i/\mu_i\}}{\max_i\{\alpha_i-\beta_i/\mu_i\}}=0,$$

which establishes the asymptotic optimality of $(\lambda_1^*,\lambda_2^*,...,\lambda_N^*)$ as $\lambda\to 0$. Next, consider the Bernoulli index policy $\theta_\lambda^{[B]}$. By its construction, $\theta_\lambda^{[B]}$ should always earn an expected long-run average reward which improves upon the average reward earned by the optimal static policy $(\lambda_1^*,\lambda_2^*,...,\lambda_N^*)$, and so one would expect $\theta_\lambda^{[B]}$ to inherit the light-traffic optimality of $(\lambda_1^*,\lambda_2^*,...,\lambda_N^*)$. Indeed, $\theta_\lambda^{[B]}$ is a stationary policy which (by Theorem 6.2.6) induces a Markov chain on some finite set of states contained within the selfishly optimal state space $\tilde{S}$. Therefore, by Corollary 4.4.5, in order to establish light-traffic optimality of $\theta_\lambda^{[B]}$ it is sufficient to show that:

$$\theta_\lambda^{[B]}(\mathbf{0})\in J=\operatorname*{arg\,max}_{i\in\{1,2,...,N\}}\left\{\alpha_i-\frac{\beta_i}{\mu_i}\right\}. \tag{6.2.37}$$

It can be checked using (6.2.31) that the indices $P_i(0,\lambda_i^*)$ for $i\in\{1,2,...,N\}$ satisfy:

$$P_i(0,\lambda_i^*)=\begin{cases}\overline{w}_i(\lambda_i^*), & \text{if } \lambda_i^*>0,\\ w_i(0), & \text{if } \lambda_i^*=0.\end{cases} \tag{6.2.38}$$

Let the quantities $\phi^*>0$ and $\phi^\dagger\geq 0$ be defined as follows:

$$\begin{aligned}\phi^* &:= \max_{i\in\{1,2,...,N\}}\left\{\alpha_i-\frac{\beta_i}{\mu_i}\right\},\\ \phi^\dagger &:= \begin{cases}\max_{i\in\{0,1,...,N\}\setminus J}\left\{\alpha_i-\frac{\beta_i}{\mu_i}\right\}, & \text{if } \{1,2,...,N\}\setminus J \text{ is non-empty},\\ 0, & \text{otherwise.}\end{cases}\end{aligned} \tag{6.2.39}$$

Evidently, $\phi^*-\phi^\dagger$ yields the smallest possible deficit in a customer's expected net reward that would result from taking an action at state $\mathbf{0}$ *other* than joining a facility belonging to J. Let i,j

be two facilities with $i \in J$ and $j \in \{1, 2, ..., N\} \setminus J$, supposing that the latter set is non-empty. By (6.2.33), one has $\lim_{\lambda \to 0} \overline{w}_k(\lambda_k^*) = w_k(0)$ for all $k \in \{1, 2, ..., N\}$. Hence, (6.2.38) implies that regardless of whether or not the arrival rates λ_i^* and λ_j^* are non-zero, one has:

$$\lim_{\lambda \to 0} P_i(0, \lambda_i^*) = w_i(0) = \phi^*,$$
$$\lim_{\lambda \to 0} P_j(0, \lambda_j^*) = w_j(0) \leq \phi^\dagger.$$

It follows that there exists $\delta > 0$ such that for all demand rates $\lambda \in (0, \delta)$:

$$P_i(0, \lambda_i^*) > \phi^* - \frac{\phi^* - \phi^\dagger}{2} = \frac{\phi^* + \phi^\dagger}{2},$$
$$P_j(0, \lambda_j^*) < \phi^\dagger + \frac{\phi^* - \phi^\dagger}{2} = \frac{\phi^* + \phi^\dagger}{2}.$$

Noting that $(\phi^* + \phi^\dagger)/2$ is strictly positive, it follows that for any two facilities $i, j \in \{1, 2, ..., N\}$ with $i \in J$ and $j \notin J$, one has $P_i(0, \lambda_i^*) > \max(P_j(0, \lambda_j^*), 0)$ for sufficiently small values of λ. It then follows directly from the criterion in 6.2.30 that $\theta_\lambda^{[B]}(\mathbf{0}) \in J$ when $\lambda < \delta$; that is, the Bernoulli policy $\theta_\lambda^{[B]}$ chooses to join a facility belonging to the set J under state $\mathbf{0}$ when λ is sufficiently small. Applying the result of Corollary 4.4.5 completes the proof of the theorem. $\square$

In this section so far it has been shown that the heuristic index policies $\theta_\lambda^{[W]}$ and $\theta_\lambda^{[B]}$ and the optimal static policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ are all asymptotically optimal in a light-traffic limit. However, the next example shows that the heavy-traffic optimality of the Whittle policy $\theta_\lambda^{[W]}$ (proved by Theorem 6.1.8) is not a property which is shared by either $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ or $\theta_\lambda^{[B]}$.

Example 6.2.10. ***(Sub-optimality of*** $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ ***and*** $\theta_\lambda^{[B]}$ ***in a heavy-traffic limit)***

It will be sufficient to consider the same system as in Example 6.2.7. To recap, there is a demand rate $\lambda = 5$ and a single facility with parameters given as follows:

$$c = 5, \qquad \mu = 2, \qquad \beta = 15, \qquad \alpha = 8.$$

As discussed in the proof of Theorem 6.2.1, there must exist a unique value λ_1^* which maximises the strictly concave function $g(\lambda) = \lambda\alpha - \beta L(\lambda)$ (refer to Figure 6.3). In Example 6.2.7 it was found that the optimal queue-joining rate (among static policies) was $\lambda_1^* \approx 3.595$ under the demand rate $\lambda = 5$. Given that customers balk with a non-zero probability under this policy, it must be the

case (due to the shape of $g(\lambda)$) that λ_1^* remains constant at the same value 3.595 for all $\lambda > 5$. The long-run average reward $g(\lambda_1^*)$ under this policy is approximately 1.458.

On the other hand, applying the result of Corollary 4.4.8, the optimal long-run average reward $g^*(\lambda)$ over all admissible policies approaches the following limit as $\lambda \to \infty$:

$$\lim_{\lambda\to\infty} g^*(\lambda) = c(\alpha\mu - \beta) = 5. \tag{6.2.40}$$

Therefore the sub-optimality of the optimal static policy approaches approximately 70% as $\lambda \to \infty$; that is, the static policy λ_1^* is not optimal in a heavy-traffic limit. In Example 6.2.7, it was shown that the Bernoulli policy $\theta_\lambda^{[B]}$ chooses to balk at states 3 and 4 when the demand rate is $\lambda = 5$. Using arguments analogous to those in the proof of Theorem 4.4.7, one can show that the expected long-run average reward $g_\theta(\lambda)$ under any stationary policy θ satisfies:

$$\lim_{\lambda\to\infty} g_\theta(\lambda) = \min(c, T_\theta)\alpha\mu - \beta T_\theta, \tag{6.2.41}$$

where $T_\theta = \min\{x \in \mathbb{N}_0 : \theta(x) = 0\}$; that is, T_θ is the lowest-populated state at which θ chooses to balk (equivalently, the highest-populated state which is positive recurrent under θ). Since the Bernoulli policy $\theta_\lambda^{[B]}$ applies a threshold $T_{\theta_\lambda^{[B]}} = 3$ when $\lambda = 5$, it follows directly from property (6) of Theorem 6.2.6 (conservativity with demand) that states $x \geq 4$ must remain excluded from the positive recurrent set S_B when λ is increased; indeed, the proof of the theorem shows that the indices $P_1(x, \lambda_1^*)$ are monotonically decreasing with λ, so one can say that $\theta_\lambda^{[B]}(x) = 0$ for all $x \geq 3$ when $\lambda > 5$. Hence, due to (6.2.41), the average reward $g_{\theta^{[B]}}(\lambda)$ must satisfy:

$$g_{\theta^{[B]}}(\lambda) \leq 3(\alpha\mu - \beta) = 3 \qquad \forall\ \lambda \in (5, \infty), \tag{6.2.42}$$

which implies, due to (6.2.40), that the sub-optimality of $\theta_\lambda^{[B]}$ is at least 40% as $\lambda \to \infty$; that is, $\theta_\lambda^{[B]}$ also fails to attain heavy-traffic optimality. Alternatively, note that the indices $P_i(x, \lambda_i^*)$ in (6.2.31) are independent of the system demand rate λ, and depend only on the optimal static joining rates $\lambda_1^*, \lambda_2^*, ..., \lambda_N^*$ and the other system parameters. It has been shown that, in this particular example, λ_1^* remains at a constant value of approximately 3.595 for all $\lambda > 5$, which implies that the values $P_1(x, \lambda_1^*)$ remain unchanged for all $x \in \mathbb{N}_0$ as $\lambda \to \infty$; hence, the policy $\theta_\lambda^{[B]}$ also remains unchanged as $\lambda \to \infty$. This shows that the inequality in (6.2.42) becomes 'tight' as $\lambda \to \infty$, and $\lim_{\lambda\to\infty}(g^*(\lambda) - g_{\theta^{[B]}}(\lambda)) = 5 - 3 = 2$. In conclusion, $\theta_\lambda^{[B]}$ improves upon the performance of the static policy λ_1^* by more than 100% as $\lambda \to \infty$, but remains 40% short of optimality.

Intuitively, the fact that λ_1^* remains unchanged for all $\lambda > 5$ must imply that the same is true of the Bernoulli policy $\theta_\lambda^{[B]}$, since $\theta_\lambda^{[B]}$ decides whether or not to admit a customer at some state $x \in \mathbb{N}_0$ based on the assumption that the optimal static policy will be followed by all customers thereafter. If the static policy remains unchanged as $\lambda \to \infty$, then there is no effect on the decision-making criterion applied by $\theta_\lambda^{[B]}$ and therefore no reason for $\theta_\lambda^{[B]}$ to change its decisions. $\boxtimes$

Theorem 6.1.10 established that the Whittle policy $\theta^{[W]}$ is more conservative than the policy θ^* found by relative value iteration. Unfortunately, the analogous comparison between $\theta^{[B]}$ and θ^* does not yield any conclusive result, since $\theta^{[B]}$ may or may not be more conservative than θ^*, depending on the system parameters. Indeed, the previous example has shown that $\theta^{[B]}$ may be more conservative than θ^* (which is always a non-idling policy by Theorem 4.3.3), and in order to show that $\theta^{[B]}$ may *not* be more conservative than θ^*, it is sufficient to consider the 4-facility system in Example 6.2.3. For the parameters used in that example, Figure 6.4 indicates that the optimal (static) demand rate λ_4^* for facility 4 is equal to zero when the system demand rate λ is sufficiently small. For example, suppose $\lambda = 10$. Then, using relative value iteration to compute the optimal policy θ^*, one finds that θ^* is a sink policy (see Definition 5.2.14) with sink state $(8, 15, 11, 4)$. However, given that $\lambda_4^* = 0$, it follows from (6.2.31) that $P_4(x, \lambda_4^*) = w_4(x)$ for all $x \in \mathbb{N}_0$, and hence the Bernoulli policy $\theta^{[B]}$ is in favour of admitting a customer to facility 4 (as opposed to balking) at any state $\mathbf{x} \in S$ with $x_4 < \tilde{B}_4 = 6$. Indeed, it can be verified that the policy $\theta^{[B]}$ derived using the parameters in Example 6.2.3 is a sink policy with sink state $(7, 10, 9, 6)$. Therefore the set of recurrent states S_B under $\theta^{[B]}$ includes states $\mathbf{x} \in S$ with $x_4 > 4$, and hence S_B is not contained in the set S_{θ^*} associated with θ^*; that is, $\theta^{[B]}$ is *not* more conservative than θ^*.

Similarly, it is not possible to compare the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$ with respect to conservativity. Indeed, the Whittle policy $\theta^{[W]}$ is always a non-idling policy (by Theorem 6.1.7) but it is also more conservative than θ^* (by Theorem 6.1.10). Hence, $\theta^{[B]}$ is more conservative than $\theta^{[W]}$ in the system described in Example 6.2.10, but it is *not* more conservative than $\theta^{[W]}$ when one considers the parameters in Example 6.2.3 and sets $\lambda = 10$ (as described above).

In the next section, the performances of the Whittle policy $\theta^{[W]}$ and the Bernoulli policy $\theta^{[B]}$ will be compared using numerical tests with randomly-generated system parameters.

6.3 Numerical results

Sections 6.1 and 6.2 introduced the Whittle index heuristic and the static routing heuristic respectively. Both of these heuristic methods are designed for the purpose of obtaining near-optimal (and easily-implementable) policies in a reasonably short amount of time. Both methods also involve analysing the N service facilities *individually*, in order to derive a set of indices which characterise a stationary routing policy for the system as a whole. In order to assess whether or not the methods are consistently successful in finding near-optimal policies, it is desirable to carry out a large number of numerical experiments involving randomly-generated parameter sets.

The purpose of this section is to report the results of a series of experiments involving more than 60,000 randomly-generated sets of system parameters. In order to evaluate the *exact* sub-optimality of a heuristic policy, it is necessary to evaluate the expected long-run average reward earned by the relevant policy and compare this with the optimal value g^* associated with an *average reward optimal* policy. Usually, one would wish to carry out these tasks using dynamic programming algorithms, but this is only practical if the finite state space $\tilde{S}$ is of relatively modest size. Of course, the heuristic methods discussed in Sections 6.1 and 6.2 can easily be applied to systems in which $\tilde{S}$ is extremely large, but it is generally not feasible to evaluate the optimal value g^* in such systems, and therefore the only comparisons of interest that can be made in 'large' systems are comparisons between different heuristic policies (whose performances must be *approximated*, using simulation) and comparisons with simple decision rules such as the selfishly optimal decision rule, discussed in Section 4.1. As such, this section will be divided into two parts:

- In the first part, systems of relatively modest size will be considered. These are systems in which the size of $|\tilde{S}|$ facilitates the efficient computation of the optimal average reward g^* using DP algorithms, and also enables similar evaluations of the average rewards earned by the Whittle policy $\theta^{[W]}$, the Bernoulli policy $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$.

- In the second part, 'large' systems will be considered. These are systems in which the value of g^* is assumed to be unattainable, and the average rewards earned by the heuristic policies $\theta^{[W]}$, $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$ must be approximated using simulation.

For the sake of distinction, a 'modest-sized' system will be defined in this section as a system in

which the cardinality of $\tilde{S}$ is between 100 and 100,000. Although it is certainly possible to apply DP algorithms to systems of greater size than this without an extremely long running time being required, it is desirable to impose a relatively strict restriction on $|\tilde{S}|$ in order to allow a large number of experiments to be carried out in a reasonable amount of time. The remainder of this section will proceed to present the results obtained from numerical experiments.

Part 1: 36,136 systems of small-to-moderate size

A series of experiments were conducted, involving 36,136 randomly-generated sets of system parameters. For each system, the parameters were randomly generated as follows:

- The number of facilities, N, was sampled from the set $\{2, 3, 4\}$.
- Each service rate μ_i was sampled from a uniform distribution between 5 and 25.
- Each service capacity c_i was sampled unbiasedly from the set $\{2, 3, 4, 5\}$.
- Each holding cost β_i was sampled from a uniform distribution between 5 and 25.
- Each fixed reward α_i was sampled from a uniform distribution which was dependent upon the number of facilities N. This uniform distribution was between 2 and 18 in the cases $N = 2$ and $N = 3$, and between 2 and 12 in the case $N = 4$.
- The demand rate λ was sampled from a uniform distribution between 0 and $1.5 \times \sum_{i=1}^{N} c_i\mu_i$.

A constraint was imposed whereby parameter sets were accepted only if they caused the size of the selfish space $\tilde{S}$ to be between 100 and 100,000 states. In addition, all facilities i were required to satisfy the condition $\alpha_i > \beta_i/\mu_i$ in order to avoid degeneracy. Parameter sets which did not satisfy these criteria were rejected and, in these cases, all parameters were re-sampled.

Let $\rho := \lambda/\sum_{i=1}^{N} c_i\mu_i$ be a measure of the relative traffic intensity for a particular system. In this section, the three cases $\rho \in (0, 0.5)$, $\rho \in (0.5, 1)$ and $\rho > 1$ will be considered separately. Figure 6.5 illustrates the distributions of the percentage sub-optimality for the three policies $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$ in systems with $\rho \in (0, 0.5)$. As stated earlier, these results have been obtained by using DP algorithms to evaluate the average rewards earned by $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$, and comparing these with the optimal value g^* (also obtained using DP). Figures 6.6 and 6.7 show the corresponding findings

for the cases $\rho \in (0.5, 1)$ and $\rho > 1$ respectively. In each of the three cases, the number of systems tested was approximately 12,000. It should be noted that the categories on the horizontal axes in Figures 6.5-6.7 are not mutually exclusive; so, for example, any heuristic policy which is within 0.1% of optimality in a particular system is also counted as being within 0.5%, etc.

Figure 6.5 shows that the Whittle heuristic appears to out-perform the static routing heuristic in systems with $\rho < 0.5$. The selfish policy $\tilde{\theta}$ also performs more strongly in light-demand systems than in higher-demand systems, which is as expected. Figure 6.6 shows that in 'medium-demand' systems (with $0.5 < \rho < 1$), all three heuristics appear to fare worse than in the low-demand case. The Whittle heuristic again appears to out-perform the static routing heuristic in this category, although its superiority is less obvious. Figure 6.7 shows that the Whittle heuristic appears to have a clear superiority over the static routing heuristic in systems with $\rho > 1$. This may be related to the fact that (by Theorem 6.1.8) the Whittle heuristic policy is optimal in a heavy-traffic limit, whereas the Bernoulli policy $\theta^{[B]}$ is not. As expected, the selfish policy performs very poorly when the demand rate is high. Incidentally, the average reward earned by the Whittle heuristic policy $\theta^{[W]}$ was within 5% of optimality in all of the systems tested across all categories.

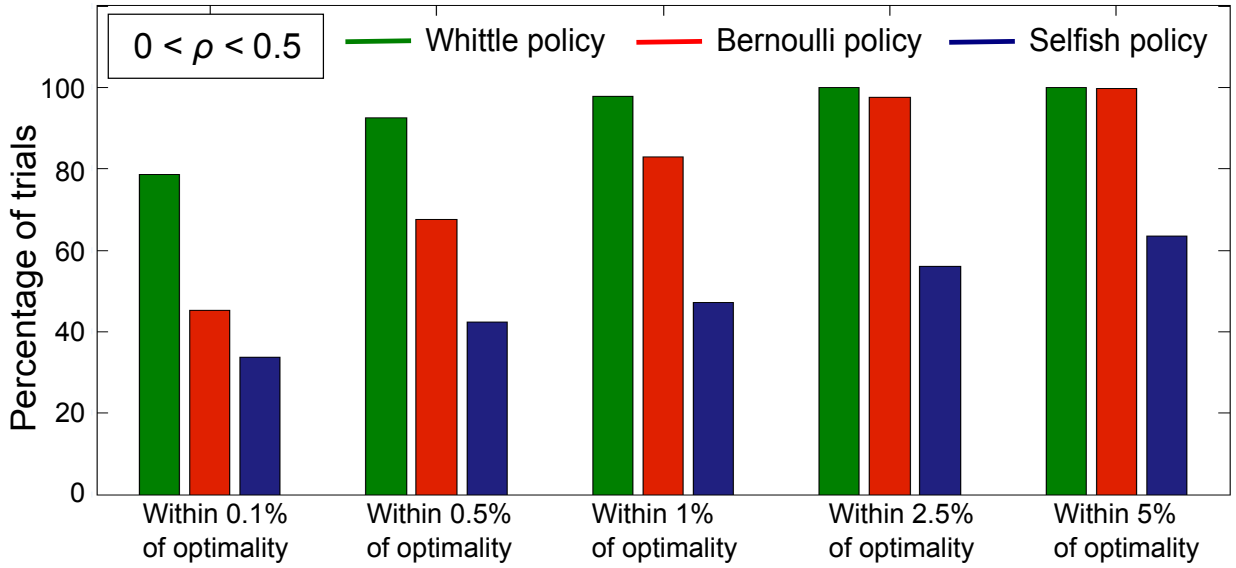


Figure 6.5: The distributions of the percentage sub-optimality for the Whittle policy $\theta^{[W]}$, the Bernoulli policy $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$ in 'light-demand' systems with $\rho \in (0, 0.5)$.

Table 6.8 shows, for each of the three policies $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$, further information about the

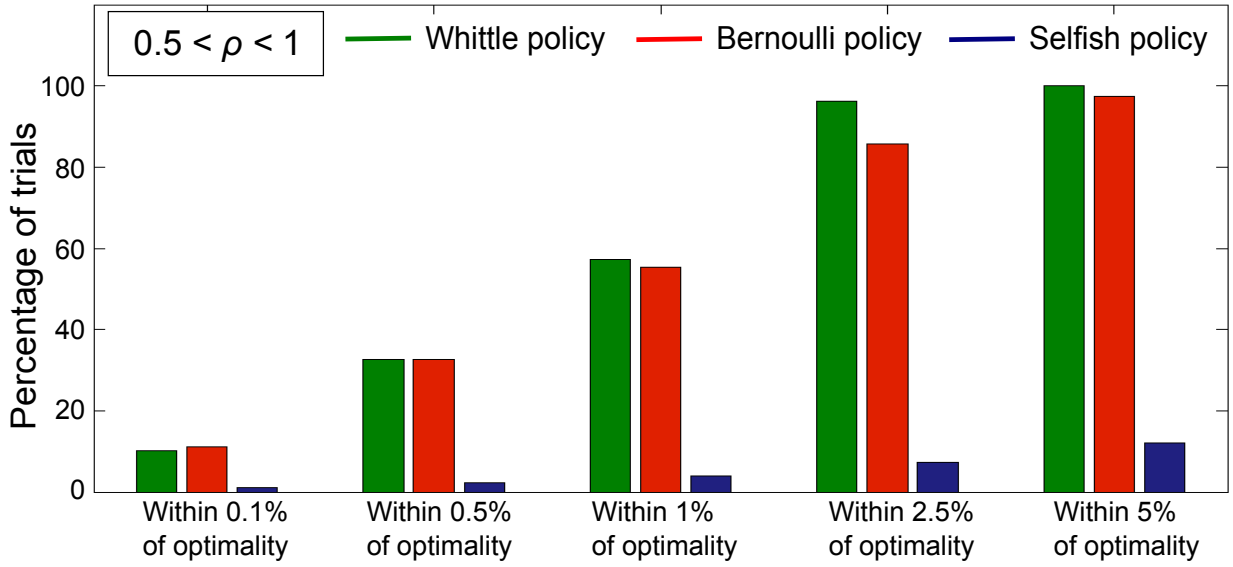


Figure 6.6: The distributions of the percentage sub-optimality for the Whittle policy $\theta^{[W]}$, the Bernoulli policy $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$ in 'medium-demand' systems with $\rho \in (0.5, 1)$.

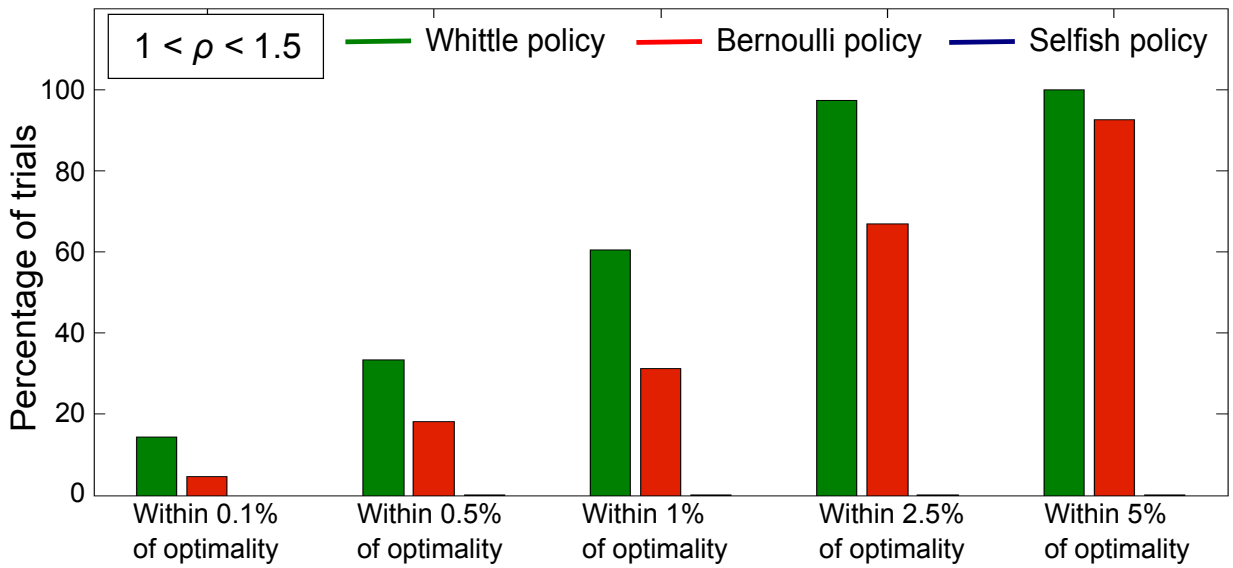


Figure 6.7: The distributions of the percentage sub-optimality for the Whittle policy $\theta^{[W]}$, the Bernoulli policy $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$ in 'high-demand' systems with $\rho \in (1, 1.5)$.

distribution of the percentage sub-optimality in each of the three categories $\rho \in (0, 0.5)$, $\rho \in (0.5, 1)$ and $\rho \in (1, 1.5)$. The combined results for all three categories are also given. For each policy (within each ρ category), the table shows the 25^{th}, 50^{th}, 75^{th} and 100^{th} percentiles of the percentage sub-optimality (over all trials), and a two-sided 99% confidence interval for the mean.

Table 6.9 shows pairwise comparisons between the three heuristic policies $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$, with systems again divided into categories according to the value of ρ and a fourth category included, showing the combined results for all categories. Each figure in the table shows the percentage of trials (within the relevant demand rate category) in which the policy referred to in the corresponding row of the table achieved a *strictly* greater average reward than the policy referred to in the corresponding column. For example, the Whittle policy $\theta^{[W]}$ out-performed the Bernoulli policy $\theta^{[B]}$ in 73.25% of the light-demand systems, 49.04% of the medium-demand systems and 69.03% of the high-demand systems. It is important to note that these figures indicate the percentage of systems in which there was a *strict* superiority of one policy over another. In some systems, two heuristic methods may yield identical policies, in which case neither policy has an advantage; indeed, this appears especially likely to happen when the demand rate λ is small.

Part 2: 24,074 systems of large size

A further series of experiments were performed, involving another 24,074 randomly-generated sets of system parameters. For each set of parameters, the number of facilities N was between 4 and 12, the number of channels c_i for each facility i was between 2 and 6, the fixed rewards α_i were sampled from a uniform distribution between 2 and 18 and the service rates μ_i, holding costs β_i and demand rate λ were generated in the same way as in Part 1. The intention was to investigate 'large' systems, in which the size of the selfish state space $\tilde{S}$ would preclude the use of dynamic programming algorithms. As such, parameter sets were accepted only if they gave a value of $|\tilde{S}|$ greater than 100,000 states. As in the previous part, the constraint $\alpha_i > \beta_i/\mu_i$ (for all facilities i) was also enforced. The median of $|\tilde{S}|$ over all of the 24,074 trials performed was approximately 7.3 million states, and the maximum was approximately 6.08×10^{22} (60.8 sextillion).

With the use of DP algorithms assumed to be infeasible, one cannot evaluate the optimal average reward g^* for a given system, nor is it possible to evaluate the performances of the heuristic

All demand rates					
Policy	Lower quartile	Median	Upper quartile	Maximum	99% C.I. for Mean
Whittle $\theta^{[W]}$	0.009	0.429	1.102	4.843	0.663 ± 0.010
Bernoulli $\theta^{[B]}$	0.180	0.788	1.864	20.382	1.313 ± 0.022
Selfish $\tilde{\theta}$	4.858	31.033	70.419	98.111	38.032 ± 0.447

Light demand ($\rho \in (0, 0.5)$)					
Policy	Lower quartile	Median	Upper quartile	Maximum	99% C.I. for Mean
Whittle $\theta^{[W]}$	0.000	0.001	0.062	3.144	0.111 ± 0.004
Bernoulli $\theta^{[B]}$	0.002	0.162	0.703	11.768	0.488 ± 0.010
Selfish $\tilde{\theta}$	0.005	1.349	11.541	76.983	8.363 ± 0.180

Medium demand ($\rho \in (0.5, 1)$)					
Policy	Lower quartile	Median	Upper quartile	Maximum	99% C.I. for Mean
Whittle $\theta^{[W]}$	0.355	0.849	1.455	4.607	0.977 ± 0.010
Bernoulli $\theta^{[B]}$	0.349	0.869	1.730	15.646	1.288 ± 0.019
Selfish $\tilde{\theta}$	12.043	26.668	42.196	86.909	28.320 ± 0.258

High demand ($\rho \in (1, 1.5)$)					
Policy	Lower quartile	Median	Upper quartile	Maximum	99% C.I. for Mean
Whittle $\theta^{[W]}$	0.335	0.795	1.335	4.843	0.902 ± 0.010
Bernoulli $\theta^{[B]}$	0.744	1.731	3.002	20.382	2.160 ± 0.026
Selfish $\tilde{\theta}$	69.265	80.490	88.407	98.111	77.233 ± 0.198

Table 6.8: The distribution of the percentage sub-optimality under the three policies $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$, with results categorised according to the value of ρ. All figures are rounded to 3 decimal places.

All systems			
	Whittle	Bernoulli	Selfish
Whittle	x	63.81%	93.77%
Bernoulli	31.61%	x	91.32%
Selfish	3.07%	6.07%	x

Light demand ($\rho \in (0, 0.5)$)			
	Whittle	Bernoulli	Selfish
Whittle	x	73.25%	84.19%
Bernoulli	14.85%	x	76.02%
Selfish	6.37%	16.16%	x

Medium demand ($\rho \in (0.5, 1)$)			
	Whittle	Bernoulli	Selfish
Whittle	x	49.04%	97.14%
Bernoulli	49.80%	x	97.97%
Selfish	2.86%	2.03%	x

High demand ($\rho \in (1, 1.5)$)			
	Whittle	Bernoulli	Selfish
Whittle	x	69.03%	99.99%
Bernoulli	30.34%	x	99.99%
Selfish	0.00%	0.00%	x

Table 6.9: Pairwise comparisons between $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$. Each figure shows the percentage of trials in which the policy referred to in the corresponding row out-performed the policy in the corresponding column.

policies $\theta^{[W]}$ and $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$ exactly. However, as discussed in previous sections, the indices used for decision-making by these three policies are simple to obtain (regardless of the size of $|\tilde{S}|$), since they can be determined by considering individual facilities one-by-one. One can then use simulation to *estimate* the average rewards earned by the three policies. Table 6.10 shows pairwise comparisons between the *estimated* performances of $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$ obtained from these trials. All of these estimates have been obtained using discrete-time simulation, with 10 million random transitions simulated for each of the three policies within each trial (including 'dummy' simulations caused by uniformisation) and the same random number seed used in all cases in order to ensure a fair comparison. All of the policies tested used the same discrete-time step size, given by $\Delta = (\lambda + \sum_{i=1}^{N} c_i\mu_i)^{-1}$. As in Table 6.9, the value of $\rho = \lambda / \sum_{i=1}^{N} c_i\mu_i$ has been used to divide the systems tested into three categories: $\rho \in (0, 0.5)$, $\rho \in (0.5, 1)$ and $\rho \in (1, 1.5)$.

By comparing Tables 6.9 and 6.10, one may observe the effect of increasing the size of the finite state space $\tilde{S}$ on the relative performances of the policies $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$. Of course, it is important to bear in mind that the results in Table 6.10 are subject to simulation error. In all of the four ρ categories, the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$ appear to perform slightly better in comparison to the selfish policy $\tilde{\theta}$ than they did in smaller-sized systems. However, the most interesting point to

All systems			
	Whittle	Bernoulli	Selfish
Whittle	x	58.76%	93.75%
Bernoulli	37.88%	x	92.53%
Selfish	6.18%	7.42%	x

Light demand ($\rho \in (0, 0.5)$)			
	Whittle	Bernoulli	Selfish
Whittle	x	57.85%	81.70%
Bernoulli	32.07%	x	77.87%
Selfish	18.10%	21.99%	x

Medium demand ($\rho \in (0.5, 1)$)			
	Whittle	Bernoulli	Selfish
Whittle	x	32.76%	99.64%
Bernoulli	67.24%	x	99.80%
Selfish	0.36%	0.20%	x

High demand ($\rho \in (1, 1.5)$)			
	Whittle	Bernoulli	Selfish
Whittle	x	85.99%	99.89%
Bernoulli	14.01%	x	99.89%
Selfish	0.11%	0.11%	x

Table 6.10: Further comparisons between $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$ in systems where $|\tilde{S}|$ is large.

note is that in systems with 'medium' demand rates ($\rho \in (0.5, 1)$), the Whittle policy $\theta^{[W]}$ appears to be clearly inferior to the Bernoulli policy $\theta^{[B]}$; this was not the case when smaller systems were considered earlier. On the other hand, in systems with 'high' demand rates ($\rho \in (1, 1.5)$), the Whittle policy $\theta^{[W]}$ appears to have a very strong advantage over the Bernoulli policy $\theta^{[B]}$. Thus, taking into account the results in Tables 6.9 *and* 6.10, it appears that there is a general tendency for $\theta^{[W]}$ to out-perform $\theta^{[B]}$ in systems where ρ is either 'small' or 'large', but for $\theta^{[B]}$ to gain an advantage over $\theta^{[W]}$ in systems where ρ lies within the 'intermediate' range $(0.5, 1)$.

6.4 Conclusions

The results in Chapter 4 (in particular, Theorem 4.2.4) have established that it is *theoretically* possible to find an average reward optimal policy for the MDP Υ formulated in Section 3.5 by truncating the state space S, and applying a dynamic programming algorithm to an MDP with the *finite* state space $\tilde{S}$ defined in Section 4.1. Unfortunately, the finite set $\tilde{S}$ might itself be very large in some problem instances, and for this reason it is necessary to look for heuristic approaches which can be relied upon to yield *near-optimal* policies in a short amount of time.

The *curse of dimensionality* has been widely discussed in the MDP literature (see [140, 141]).

Two *heuristic* methods have been introduced in this chapter, both of which are able to avoid the problems caused by a large number of service facilities (i.e. a high-dimensional state space) by analysing service facilities *individually*, and deriving a set of *indices* for each facility. In the case of the Whittle index policy $\theta^{[W]}$ discussed in Section 6.1, these indices are based on the properties of optimal routing policies in a *Lagrangian relaxation* of the original problem, while in the case of the static routing heuristic discussed in Section 6.2, the indices are based on the application of *policy improvement* to an optimal static routing policy. By relying on the simple principle of always choosing the facility with the largest index value at any given state, one can use the indices derived for the individual facilities to derive a *heuristic* stationary routing policy for the original N-facility problem. Furthermore, the empirical results in Section 6.3 have demonstrated that both of the policies $\theta^{[W]}$ and $\theta^{[B]}$ are frequently able to achieve a strong performance.

For any given set of system parameters, the indices which characterise the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$ can be calculated in a completely deterministic way. Thus, these heuristics do not rely on any iterative algorithm, nor do they involve any sort of randomisation. One might regard the deterministic nature of these heuristics as both a strength and a weakness. On one hand, their simplicity makes them extremely easy to implement; on the other hand, if the heuristics are found to perform *poorly* in a particular system, then it is not necessarily easy to see how the policies that they produce might be adjusted in order to achieve closer proximity to an optimal policy. An alternative approach to finding a near-optimal policy might involve repeatedly making updates to a 'test' policy according to a procedure which is somehow stochastic. The aim of this procedure would be to make gradual improvements to the test policy (somewhat in the style of a DP algorithm), with the ultimate goal of achieving convergence to an optimal policy. *Reinforcement learning*, which essentially utilises this type of approach, will be discussed extensively in Chapter 7.

7 Reinforcement learning

The challenge of finding a socially optimal policy for an MDP which models a large-scale real-world dynamic system is an extremely formidable one. As discussed in earlier chapters, dynamic programming algorithms might be able to accomplish the task in theory, but the time required to execute such procedures escalates quickly as the complexity of the problem increases, to the point where their effective use becomes unrealistic. The heuristic methods discussed in Chapter 6 have been shown to perform well in problems of reasonable size, and moreover these methods have strong theoretical backing. This chapter is devoted to an alternative approach, based on *simulation* of the real-world system under consideration. This approach is often referred to in the literature as *reinforcement learning* (a term which originated in the artificial intelligence community; see, for example, [53, 127, 188]), although various other names for it are commonly used, including simulation-based optimisation [62] and approximate dynamic programming [140].

Section 7.1 will present an overview of the general approach of reinforcement learning, and explain how it is applicable to the queueing system problems considered in this thesis. Section 7.2 will provide details of certain algorithms that can be found in the literature, and demonstrate their performances using some specific examples. Section 7.3 will present some 'specialised' algorithms which exploit the particular properties of the queueing systems under consideration. Section 7.4 will explore the method of *value function approximation*, which is essentially a technique for coping with problems where the state space is extremely vast. Section 7.5 will present numerical results demonstrating the performances of the algorithms discussed in the preceding sections. Finally, Sections 7.6 and 7.7 will consider certain generalisations of the queueing system described in Section 3.1, involving non-exponential distributions and heterogeneous customers respectively.

7.1 Introduction

Reinforcement learning differs from the optimisation methods considered in earlier chapters (such as dynamic programming) in that it relies upon *simulation* of the stochastic process to be optimised. Simulation itself is so well-established as a method for evaluating the performance of a queueing system that the idea of using it as a tool for optimising performance is a natural and appealing

one. While simulation, by its nature, cannot offer exact guarantees regarding the consequences of implementing a particular policy, the advances in computer processing power made in recent decades have enabled it to become an indispensable technique for mathematical modellers working in all areas of industry (see, for example, [113, 180]). An in-depth discussion of the advantages of simulation will not be given here, but it is appropriate to note some key points:

1. Firstly (and most obviously), creating a computer simulation of a queueing system saves time, effort and money in comparison to conducting experiments 'in real life'.

2. The level of detail in a simulation model can be extremely high. As such, it is often possible to simulate (with very little effort) a system whose random transitions follow conditional distributions which would be intractable using a more analytical approach.

3. Although the results of a simulation experiment are subject to random 'noise', this effect can be mitigated by conducting multiple trials and collecting summary statistics. If the model is designed well and a suitably large number of trials are performed, the resulting estimates of key performance measures for the system will generally be very reliable.

In the early part of this chapter, the system under consideration will once again be the 'basic' N-facility queueing system described in Section 3.1. Extensions to this model will be discussed in later sections. As discussed in Chapter 4, it is possible to find an *average reward optimal* policy θ^* for the system which operates in such a way that at any state $\mathbf{x} \in S$, the action $\theta^*(\mathbf{x})$ chosen by θ^* attains the maximum on the right-hand side of the optimality equations:

$$g^* + h(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}, a) + \sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y}) h(\mathbf{y}) \right\}, \tag{7.1.1}$$

where (as in previous chapters) g^* is the optimal expected long-run average reward over all admissible policies, and the function $h : S \to \mathbb{R}$ is referred to as the *relative value function*. Note that in (7.1.1), $r(\mathbf{x}, a)$ represents a generic reward function which is assumed to be suitable for the purpose of calculating the expected long-run average reward (results from previous chapters have shown that various different formulations for $r(\mathbf{x}, a)$ are possible). Let $Q(\mathbf{x}, a)$ be the 'Q-factor' associated with choosing action $a \in A_{\mathbf{x}}$ under state $\mathbf{x} \in S$, defined as follows:

$$Q(\mathbf{x}, a) := r(\mathbf{x}, a) - g^* + \sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y}) h(\mathbf{y}). \tag{7.1.2}$$

Accordingly, for all states $\mathbf{x} \in S$, one simply has:

$$h(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} Q(\mathbf{x}, a). \tag{7.1.3}$$

Hence, knowledge of the exact Q-factors for all state-action pairs is all that is required in order to characterise an average reward optimal policy. By substituting (7.1.3) into (7.1.1), one obtains an alternative set of optimality equations involving the Q-factors $Q(\mathbf{x}, a)$:

$$g^* + Q(\mathbf{x}, a) = r(\mathbf{x}, a) + \sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y}) \max_{a' \in A_{\mathbf{y}}} Q(\mathbf{y}, a'). \tag{7.1.4}$$

Essentially, the objective throughout this chapter will be to obtain reliable estimates for the Q-factors $Q(\mathbf{x}, a)$, under the premise that the dynamic programming algorithms relied upon in earlier chapters are rendered unusable due to the scale and/or complexity of the problem. This task will be accomplished using *reinforcement learning* (RL) algorithms. These RL algorithms will be very much based within the mathematical framework of a Markov Decision Process, and (at least in the first few sections) will rely upon state space, action space, transition probability and reward formulations similar to those introduced in Chapter 3. Indeed, these algorithms will have much in common with DP algorithms themselves; the essential difference will be that they avoid sweeping exhaustively through all states and actions on each iteration, as DP algorithms do.

Although RL algorithms have become popular tools for optimising Markov Decision Processes, the point should be made that reinforcement learning in general is a much broader field, with applications in statistics, economics, game theory and other disciplines (see, for example, [29, 51, 112]). Thus, this chapter concerns only one particular application area of RL, and certainly does not represent a comprehensive study of the field of reinforcement learning itself.

Before proceeding, some comments about the general philosophy of reinforcement learning are in order. RL may be summarised somewhat vaguely as the process of an intelligent being (often referred to as an *agent*) interacting with its environment, and learning from the consequences of its actions. At any point in time, the agent finds itself in a particular state, with various possible actions to choose from; it then selects an action and learns the consequences of the choice it has made. Depending on whether or not these consequences are favourable, it may become either more or less inclined to select the same action again in the future when it finds itself in the same state. As the process continues, the agent gradually acquires more experience and is able to improve

its own long-term welfare by almost always selecting actions which have resulted in favourable outcomes in the past; although one usually assumes that, regardless of how much experience has been gained, the agent will still make 'experimental' choices occasionally in order to continue the learning process, and avoid becoming completely settled on one particular policy.

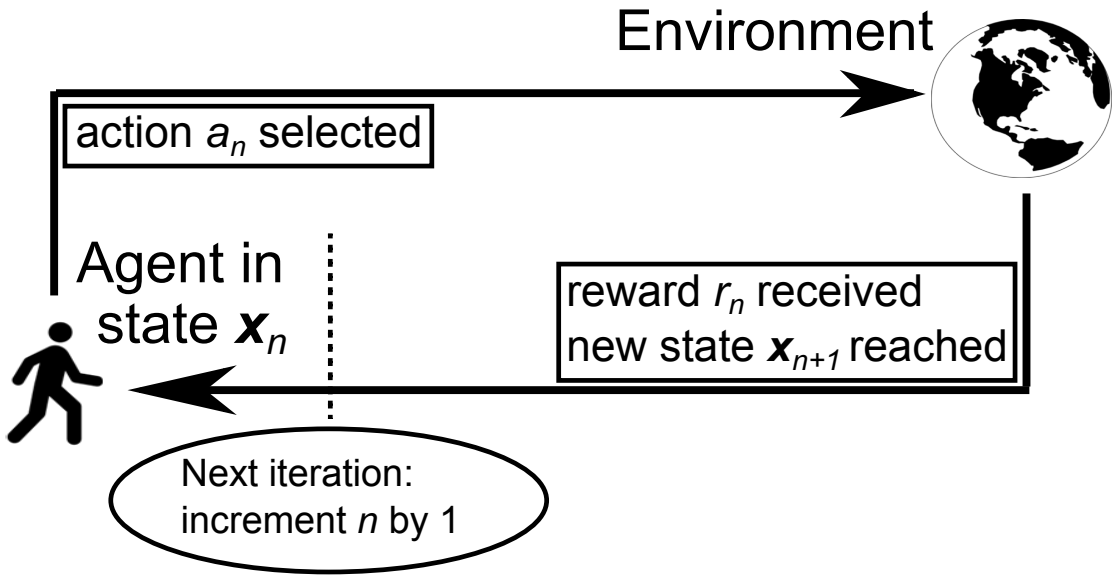


Figure 7.1: The 'agent-environment interface' in reinforcement learning.

Figure 7.1 is a diagram which bears similarity to illustrations found in several texts, including Sutton and Barto [175], in which it is referred to as the 'agent-environment interface'. For the sake of simplicity, let it be assumed for the time being that the process evolves in discrete time (this assumption will be relaxed in later sections). In keeping with the convention of this thesis by using vectors to represent states, let $\mathbf{x}_n$ denote the state of the system at some discrete time epoch $n \geq 0$. As depicted in Figure 7.1, an action a_n is chosen by the agent at time n, and two new pieces of information are acquired as a result: firstly, a reward r_n is received, and secondly the agent finds itself in a new state $\mathbf{x}_{n+1}$. The reward r_n may be either random or deterministic, but in either case one would assume that its value is influenced by both $\mathbf{x}_n$ and a_n. The new state $\mathbf{x}_{n+1}$ will generally be determined at random in a stochastic problem. The agent then uses both the reward r_n *and* the new state $\mathbf{x}_{n+1}$ to judge how 'favourable' the consequences of choosing action a_n in state $\mathbf{x}_n$ have turned out to be; for example, it might be the case that a particular outcome is deemed highly favourable by the agent even if the reward obtained is small, since it might entail a transition to

a state which is regarded as being extremely desirable. Similarly, if a particular action results in a transition to a state which has been associated only with poor outcomes in the past, then the agent may look upon this outcome unfavourably, regardless of the reward earned.

In general, RL algorithms assume that the 'new information' acquired by an agent after making a decision is accessible either by observation of an actual real-world process, or via a simulator; unsurprisingly, it is the latter assumption that is made in this thesis. In order to construct an adequate simulation model, it is necessary to have some knowledge (or inference) of the probability distributions which govern the rewards and transition probabilities for the system. However, it is usually *not* necessary to formulate exact expressions for the transition probabilities themselves. By relying upon simulation rather than exact computation, RL algorithms are said to avoid the "curse of modelling". Indeed, Gosavi [62] (p. 214) comments that "the avoidance of transition probabilities (by RL algorithms) is not a miracle, but a fact backed by mathematics".

To illustrate this principle, consider an $M/M/1$ queueing system with arrival rate $\lambda > 0$ and service rate $\mu > 0$. Assuming that the parameters λ and μ are known, the random evolution of this system can be completely determined by generating a sequence of exponentially-distributed inter-arrival times, and another sequence of exponentially-distributed service times; thus, it is not necessary to evaluate (for example) the probability of transferring from a particular state at time $t_0 \geq 0$ to another state at time $t_1 > t_0$ in order to simulate the progression of states, queue lengths, waiting times etc. Whilst the calculation of transition probabilities may not be problematic in the case of an $M/M/1$ queue, it may become an extremely difficult task when one considers more complicated queueing systems, especially in cases where it is not possible to apply uniformisation in order to discretise the system (cases such as these will be discussed further in later sections).

Even within the specific application area of Markov Decision Processes, RL algorithms in the literature are numerous and varied. However, as discussed previously, all of the algorithms considered in this chapter share a common purpose: to estimate the Q-factors $Q(\mathbf{x}, a)$ (which characterise an average reward optimal policy) as accurately as possible. From (7.1.2), it is clear that the value $Q(\mathbf{x}, a)$ for any state-action pair $(\mathbf{x}, a) \in S \times A$ may be regarded informally as a measure of the 'utility' of choosing action a under state $\mathbf{x}$. Recall (from Theorem 3.7.4) that the relative values $h(\mathbf{x})$ satisfying the average reward optimality equations (7.1.1) are unique, provided that one 'fixes'

one of their values (by, for example, setting $h(\mathbf{0}) = 0$). It then follows from (7.1.3) that the Q-factors $Q(\mathbf{x}, a)$ are also unique. Hence, the task of obtaining reliable estimates for these Q-factors is very much akin to the classical statistical problem of estimating the values of unknown population parameters by random sampling. Furthermore, the performance of an RL algorithm may be judged not only by whether or not it outputs a policy which is close to optimal, but also by whether or not the estimates obtained for the Q-factors are close to their theoretical values (assuming that one has the luxury of knowing these values exactly, which will generally only be the case in small-scale problems used for validation purposes). In the next section, two learning algorithms based on the general RL methodology discussed in this section will be introduced and compared.

7.2 TD learning and R-learning

The intention in this section is to explain in greater detail how RL algorithms may be applied to a Markov Decision Process, and to introduce some of the most well-known RL algorithms from the literature. In terms of their theoretical foundation, all of the RL algorithms considered in this chapter will be based (at least to some extent) on dynamic programming methods; however, these algorithms will differ significantly from DP algorithms in their mode of operation. Two important characteristics of the DP algorithms discussed in earlier chapters are as follows:

1. DP algorithms usually rely upon *synchronous updating*, which means that they update the value function on each iteration by looping over the entire state and action space.

2. DP algorithms rely upon knowledge of a deterministic reward $r(\mathbf{x}, a)$ and fully-specified probability distribution $\{p(\mathbf{x}, a, \mathbf{y})\}_{\mathbf{y} \in S}$ for each state-action pair $(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$.

In order to illustrate the way in which RL algorithms deviate from the characteristics of DP algorithms noted above, consider an N-facility queueing system which satisfies the assumptions of Section 3.1 and suppose the system is modelled by an MDP which has been uniformised as described in Section 3.3, so that it evolves in discrete time steps of size Δ, where $\Delta \in \left(0, (\lambda + \sum_{i=1}^{N} c_i \mu_i)^{-1}\right]$. RL algorithms operate by *simulating* random transitions. Suppose the process is in some state $\mathbf{x}_n \in S$ at time step $n \geq 0$, and an action $a_n \in \{0, 1, 2, ..., N\}$ is chosen. In this case, the task of simulating the next random transition is simply a matter of sampling a random number u from

a uniform distribution between 0 and 1, and then using the transition probabilities in (3.5.3) to determine the next state based on the particular range of values in which u is found. To be specific, let u be a random number between 0 and 1. Then, given that an action $a_n \in A_{\mathbf{x}_n}$ is chosen at state $\mathbf{x}_n \in S$ at time $n \geq 0$, the state $\mathbf{x}_{n+1}$ at time $n+1$ may be determined as follows:

$$\mathbf{x}_{n+1} = \begin{cases} \mathbf{x}_n^{i+}, & \text{if } u < \lambda\Delta \text{ and } a_n = i \text{ for some } i \in \{1, 2, ..., N\}, \\ \mathbf{x}_n^{i-}, & \text{if } \lambda\Delta + \sum_{j=1}^{i-1} \min((\mathbf{x}_n)_j, c_j)\mu_j\Delta \leq u < \lambda\Delta + \sum_{j=1}^{i} \min((\mathbf{x}_n)_j, c_j)\mu_j\Delta \\ & \text{for some } i \in \{1, 2, ..., N\}, \\ \mathbf{x}_n, & \text{otherwise,} \end{cases} \tag{7.2.1}$$

where, in keeping with the notation of previous chapters, $(\mathbf{x}_n)_j$ denotes the j^{th} component of the vector $\mathbf{x}_n$. Accordingly (making use of the RL terminology introduced in the previous section), an agent may be deployed to 'explore' the system by choosing actions and moving between states according to the simulation procedure described above. However, the question remains as to how actions should be selected. Since the ultimate objective is for the agent to 'learn' a policy which will optimise the performance of the system (i.e. a socially optimal control policy), it must obviously be allowed to experiment with many different policies. On the other hand, the simple strategy of selecting actions completely at random would not be the most efficient way for the agent to learn a socially optimal policy, since this would not make effective use of past experience.

What is really needed is some sort of compromise between *exploration* and *exploitation*; that is, the agent should sometimes choose actions randomly (in order to ensure that it explores a breadth of different options at each state), but should also sometimes act 'greedily' by focusing on the action(s) that have yielded the most favourable outcomes in the past. The importance of exploitation lies not only in the fact that it improves the agent's welfare in the short-term, but also in the fact that the action(s) which are perceived to be 'best' at any particular state are the most important ones to evaluate accurately, and therefore should be selected more often than actions for which there is strong evidence of sub-optimality. Of course, these ideas need to be formalised, and an RL algorithm must specify the exact means by which an agent selects actions. The theme of exploration vs. exploitation is important not only in reinforcement learning, but also in the design of meta-heuristics and the field of artificial intelligence in general (see [60, 139, 148]).

As the agent explores the consequences of choosing different actions at the various states it visits, it should gradually improve its estimates of the Q-factors $Q(\mathbf{x}, a)$ which characterise an average reward optimal policy for the system. In practice, this means that an array of values $\hat{Q}(\mathbf{x}, a)$ should be stored, where (for each state-action pair $(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$) $\hat{Q}(\mathbf{x}, a)$ is an estimate of $Q(\mathbf{x}, a)$. Each time the agent chooses some action $a \in A_{\mathbf{x}}$ at state $\mathbf{x} \in S$, it should then update its estimate $\hat{Q}(\mathbf{x}, a)$ based on the reward received and the new state reached after the resulting simulated random transition. The next question that must be addressed is exactly how the estimates $\hat{Q}(\mathbf{x}, a)$ should be calculated. Recalling (7.1.2) and (7.1.3), the *true* values $Q(\mathbf{x}, a)$ satisfy:

$$Q(\mathbf{x}, a) = r(\mathbf{x}, a) - g^* + \sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y}) \max_b Q(\mathbf{y}, b). \tag{7.2.2}$$

Of course, the Q-factors in (7.2.2) are those associated with an average reward optimal policy. Before introducing the RL algorithms of particular interest in this section (which aim to approximate an *optimal* policy), it will be useful to begin by addressing the problem of how to evaluate the expected long-run average reward under a *fixed* stationary policy using RL methods. Let θ be a stationary policy, and let $Q_\theta(\mathbf{x})$ denote the Q-factor for choosing the action $\theta(\mathbf{x}) \in A_{\mathbf{x}}$ prescribed by θ at state $\mathbf{x} \in S$, assuming that the policy θ is followed thereafter. By analogy to the policy evaluation equations in (3.7.5), the Q-factors $Q_\theta(\mathbf{x})$ must satisfy (for all $\mathbf{x} \in S$):

$$Q_\theta(\mathbf{x}) = r(\mathbf{x}, \theta(\mathbf{x})) - g_\theta + \sum_{\mathbf{y} \in S} p_\theta(\mathbf{x}, \mathbf{y}) Q_\theta(\mathbf{y}), \tag{7.2.3}$$

where g_θ and $p_\theta(\mathbf{x}, \mathbf{y})$ denote (respectively) the expected average reward under θ and the transition probabilities associated with the policy θ, and (as before) the reward function $r(\cdot)$ is not specified exactly at this stage, but is assumed to be fit for the purpose at hand. It is now appropriate to introduce the concept of *temporal-difference learning* (TD learning), which is discussed extensively in (for example) Sutton and Barto [175]. The TD learning procedure for evaluating a stationary policy θ may be described most easily by reference to the algorithm given below.

Temporal difference learning algorithm (TD learning)

1. Input θ, the stationary policy to be evaluated. Initialise values $n = 0$, $g_0 = 0$ and an array of values $\hat{Q}_\theta(\mathbf{x})$ with $\hat{Q}_\theta(\mathbf{x}) = 0$ for each $\mathbf{x} \in S$. Choose an initial state $\mathbf{x}_0 \in S$.

2. Use simulation to determine the reward r_n and the new state $\mathbf{x}_{n+1}$ reached at the next time

step. Then, update $\hat{Q}_\theta(\mathbf{x}_n)$ according to the following update rule:

$$\hat{Q}_\theta(\mathbf{x}_n) \leftarrow (1-\delta_n)\hat{Q}_\theta(\mathbf{x}_n) + \delta_n \left[r_n - g_n + \hat{Q}_\theta(\mathbf{x}_{n+1}) \right],$$

where $\delta_n \in [0,1]$ is a 'learning parameter' which either remains constant or tends to zero as $n \to \infty$. (Some possible definitions for δ_n are discussed later.)

3. Calculate g_{n+1}, the latest estimate of the average reward g_θ, as follows:

$$g_{n+1} := (1-\zeta_n)g_n + \zeta_n \left[r_n + \hat{Q}_\theta(\mathbf{x}_{n+1}) - \hat{Q}_\theta(\mathbf{x}_n) \right],$$

where $\zeta_n \in [0,1]$ is another learning parameter.

4. If $n = n_{\max}$, where $n_{\max}$ is a large integer, then output the array of values $\hat{Q}_\theta(\mathbf{x})$ as estimates for the values $Q_\theta(\mathbf{x})$ satisfying (7.2.3) and also output g_{n+1} as an estimate for the average reward g_θ, then stop. Otherwise, increment n by 1 and return to step 2.

Note that, as a consequence of the fact that a *fixed* policy θ is being evaluated, the Q-factors $Q_\theta(\mathbf{x})$ estimated by the TD learning algorithm are exactly equivalent to the relative values $h(\mathbf{x})$ in (3.7.5). An example involving an application to a queueing system will be given shortly. First, however, some further explanatory comments are in order. Essentially, in step 2 of the algorithm, the current estimate of the Q-factor at state $\mathbf{x}_n \in S$ (represented by $\hat{Q}_\theta(\mathbf{x}_n)$) is compared to the 'new' quantity $r_n - g_n + \hat{Q}_\theta(\mathbf{x}_{n+1})$, where the reward r_n and the new state $\mathbf{x}_{n+1}$ are both acquired via simulation. The link between the expression $r_n - g_n + \hat{Q}_\theta(\mathbf{x}_{n+1})$ and the expression on the right-hand side of (7.2.3) is obvious. As such, one can easily recognise that if the algorithm achieves its ultimate goal (rarely possible in practice!) of obtaining completely accurate estimates for the Q-factors $Q_\theta(\mathbf{x})$ and average reward g_θ, then the estimate $\hat{Q}_\theta(\mathbf{x}_n)$ will be an unbiased estimator of the random quantity $r_n - g_n + \hat{Q}_\theta(\mathbf{x}_{n+1})$ generated by the simulator. The difference $r_n - g_n + \hat{Q}_\theta(\mathbf{x}_{n+1}) - \hat{Q}_\theta(\mathbf{x}_n)$ is referred to as the *TD error* at stage n of the algorithm, and the estimate $\hat{Q}_\theta(\mathbf{x}_n)$ is updated (in step 2) by multiplying the TD error by the (typically small) quantity δ_n and 'adjusting' $\hat{Q}_\theta(\mathbf{x}_n)$ by the resulting amount. Similarly, in step 3, the estimate g_{n+1} for the expected average reward g_θ is calculated in a manner entirely consistent with the policy evaluation equations (7.2.3).

It should also be noted that, from a practical point of view, the initialisation of an array of values $\hat{Q}_\theta(\mathbf{x})$ for all $\mathbf{x} \in S$ (as described in step 1 of the algorithm) is obviously not possible if the state

space S is infinite. However, this does *not* mean to say that the TD learning algorithm cannot be applied to a problem with an infinite state space. Clearly, all that is required in the case of an infinite state space is for a computer program to be written in such a way that new values $\hat{Q}_\theta(\mathbf{x})$ are initialised for states $\mathbf{x} \in S$ as and when they are first visited during the simulated evolution of the process under θ. Assuming that the number of states visited during the running of the algorithm does not escalate in such a way that the system memory is unable to cope with the number of Q-factor estimates requiring storage, the program should be capable of at least obtaining a reliable estimate of the average reward g_θ, even if it is impossible to visit (and estimate the Q-factors for) *all* states in S during the finite running time of the algorithm. RL algorithms which do *not* need to store a unique value for each system state (and are thereby able to cope with extremely vast state spaces) will be discussed in Section 7.4. The fact that RL algorithms are theoretically able to cope with infinite state spaces gives them an important advantage over conventional DP algorithms, which are restricted by the need to sweep through *all* system states on each iteration.

In practice, the learning parameters δ_n and ζ_n are sometimes given small constant values; for example, the values $\delta_n = 0.1$ and $\zeta_n = 0.01$ (for all $n \in \mathbb{N}$) are used in the example given in Chapter 6.7 of Sutton and Barto [175]. Alternatively, δ_n and ζ_n may both tend to 0 as $n \to \infty$; however, in this case it is desirable to ensure that ζ_n decays to zero more rapidly than δ_n (see [18]). Generally speaking, it is not realistic to devise a rule for choosing δ_n and ζ_n which will suit any application; some 'tweaking' will usually be necessary in order to find the configuration which yields the best performance for the system under consideration. One possible approach is to allow the parameter δ_n to be state-dependent, and define it in such a way that it depends on the total number of visits made to state $\mathbf{x}_n$ up to and including time n. As explained by Gosavi [63], this strategy has its roots in the *Robbins-Monro* stochastic approximation scheme, originally introduced in [143]. Since the use of state-dependent learning parameters will be important in the development of this chapter, it will be useful to provide an outline of the Robbins-Monro procedure.

Robbins-Monro stochastic approximation scheme

Let X be a bounded random variable, and suppose that a sequence of samples $(X_1, X_2, X_3, ...)$ is available for estimating the expected value $E[X]$; that is, each sample X_i is a realisation of X. Let $(Y_1, Y_2, Y_3, ...)$ denote a sequence of estimates of $E[X]$, obtained as follows:

1. Initialise the values $n = 1$ and $Y_0 = 0$, and choose a small value $\epsilon > 0$.

2. Calculate Y_n, the n^{th} estimate of $E[X]$, as follows:

$$Y_n := (1 - d_n)Y_{n-1} + d_n X_n, \tag{7.2.4}$$

where it is assumed that d_n satisfies the following conditions:

$$\lim_{n\to\infty} \sum_{m=1}^{n} d_m = \infty, \tag{7.2.5}$$

$$\lim_{n\to\infty} \sum_{m=1}^{n} d_m^2 < \infty. \tag{7.2.6}$$

3. If $|Y_n - Y_{n-1}| < \epsilon$ then stop; otherwise, increment n by 1 and return to step 2.

If d_n satisfies the conditions (7.2.5)-(7.2.6), then Y_n is guaranteed to converge to $E[X]$ as $n \to \infty$ (see [63, 143]). If $d_n = 1/n$ for all $n \in \mathbb{N}$, then it is clear from (7.2.4) that the Robbins-Monro estimate Y_n is obtained by direct averaging of the samples $X_1, X_2, ..., X_n$; that is:

$$Y_n = \frac{X_1 + X_2 + ... + X_n}{n},$$

in which case the convergence of Y_n to $E[X]$ is implied directly by the strong law of large numbers (see, for example, [146], p. 78). However, the fact that alternative definitions for d_n are possible may be exploited in the choice of learning parameters for RL algorithms. Suppose that, in the simulation of an MDP operating under a stationary policy θ, $\nu_n(\mathbf{x})$ denotes the total number of visits made to state $\mathbf{x} \in S$ up to and including time $n \geq 0$. If the learning parameters δ_n in step 2 of the TD learning algorithm on page 292 are given by $1/\nu_n(\mathbf{x}_n)$ for all $n \in \mathbb{N}$, then each estimate $\hat{Q}_\theta(\mathbf{x}_n)$ is obtained via direct averaging of the 'data pieces' $r_m - g_m + \hat{Q}_\theta(\mathbf{x}_{m+1})$ acquired at all time steps $m \in [0, n]$ for which $\mathbf{x}_m = \mathbf{x}_n$. Therefore, in view of the fact that the $Q_\theta(\mathbf{x})$ values satisfy the evaluation equations (7.2.3), it is clear that setting $\delta_n = 1/\nu_n(\mathbf{x}_n)$ for $n \in \mathbb{N}$ is somewhat logical; however, there may exist other specifications for δ_n which enable a stronger performance (in terms of the rate of convergence of the estimates $\hat{Q}_\theta(\mathbf{x})$ to their true theoretical values).

It should also be noted that, unfortunately, the convergence properties of the Robbins-Monro approximation scheme (i.e. the fact that the estimates Y_n converge to $E[X]$) do not readily imply that the estimates $\hat{Q}_\theta(\mathbf{x})$ obtained by the TD learning algorithm are guaranteed to approach the

theoretical values $Q_\theta(\mathbf{x})$ as the running time tends to infinity. This is mainly due to the fact that each 'data piece' $r_n - g_n + \hat{Q}_\theta(\mathbf{x}_{n+1})$ acquired in step 2 of the algorithm depends on estimates of g_θ and $Q_\theta(\mathbf{x}_{n+1})$ (given by g_n and $\hat{Q}_\theta(\mathbf{x}_{n+1})$ respectively) which, in general, will not be exact. The task of establishing convergence properties for RL algorithms is far from trivial and presents an ongoing challenge for researchers; this topic is discussed further in Appendix A.9.

An important difference between the task of estimating the Q-factors $Q_\theta(\mathbf{x})$ using the 'data pieces' $r_n - g_n + \hat{Q}_\theta(\mathbf{x}_{n+1})$ acquired during the TD learning algorithm and the more simple problem of estimating an expected value $E[X]$ using sample realisations of the random variable X (addressed by the Robbins-Monro scheme) is that, assuming that the TD learning algorithm behaves as expected, each data piece $r_n - g_n + \hat{Q}_\theta(\mathbf{x}_{n+1})$ will become a more reliable estimator of $Q_\theta(\mathbf{x}_n)$ as the algorithm progresses; whereas, in the Robbins-Monro scenario, each sample realisation X_n is an equally good estimator of $E[X]$. This observation suggests that the learning parameters δ_n should be defined in such a way that the data pieces acquired at later stages of the algorithm are given more 'weight' than the data pieces acquired in the earlier stages, since the earlier-acquired data are likely to depend on less accurate estimates of g_θ and the Q-factors $Q_\theta(\mathbf{x})$. One way to achieve this (which again relies upon state-dependent learning parameters) is to define δ_n as follows:

$$\delta_n := \frac{T}{T + \nu_n(\mathbf{x}_n) - 1}, \tag{7.2.7}$$

where T is an integer greater than 1. Note that if $T = 1$ in (7.2.7), then $\delta_n = 1/\nu_n(\mathbf{x}_n)$ and this is again equivalent to direct averaging of the data pieces acquired at state $\mathbf{x}_n$ over the course of the process, with an equal 'weight' attached to each data piece. In experiments, values such as $T = 10$ have been found to yield satisfactory results; the examples given later in this chapter will demonstrate this. Note that, under the additional assumption that the stationary policy θ induces an ergodic Markov chain defined on a *finite* set of states S_θ (which has been shown, by the results in Section 4.2, to be true for any average reward optimal policy θ^* in the queueing system formulated in Section 3.1), it may be shown rigorously that the parameters δ_n defined in (7.2.7) satisfy the Robbins-Monro conditions $\sum_{n=1}^{\infty} \delta_n = \infty$ and $\sum_{n=1}^{\infty} \delta_n^2 < \infty$. Indeed, since $T > 1$ and $\nu_n(\mathbf{x}_n) \leq n$ for all $n \in \mathbb{N}$, the property $\sum_{n=1}^{\infty} \delta_n = \infty$ may be established by noting that:

$$\delta_n = \frac{T}{T + \nu_n(\mathbf{x}_n) - 1} > \frac{1}{n},$$

whereupon the property $\sum_{n=1}^{\infty} \delta_n = \infty$ follows using results for comparisons between infinite series (see [89], p. 50) and the fact that $\sum_{n=1}^{\infty} n^{-1} = \infty$. On the other hand, let $S_\theta \subset S$ denote the finite set of states which are positive recurrent under θ. Since any state $\mathbf{x} \notin S_\theta$ is visited only finitely many times under θ, it is sufficient (in order to show that $\sum_{n=1}^{\infty} \delta_n^2 < \infty$) to show:

$$\sum_{n=1}^{\infty} I(\mathbf{x}_n \in S_\theta)\delta_n^2 < \infty,$$

where $I(\cdot)$ is the indicator function. Then, again using standard results from analysis:

$$\begin{aligned}
\sum_{n=1}^{\infty} I(\mathbf{x}_n \in S_\theta)\delta_n^2 &= \sum_{n=1}^{\infty} I(\mathbf{x}_n \in S_\theta) \left(\frac{T}{T + \nu_n(\mathbf{x}_n) - 1} \right)^2 \\
&< \sum_{n=1}^{\infty} I(\mathbf{x}_n \in S_\theta) \left(\frac{T}{\nu_n(\mathbf{x}_n)} \right)^2 \\
&= T^2 \sum_{\mathbf{x} \in S_\theta} \sum_{n=1}^{\infty} \frac{1}{n^2} \\
&= T^2 |S_\theta| \pi^2 / 6, \qquad (7.2.8)
\end{aligned}$$

which implies $\sum_{n=1}^{\infty} \delta_n^2 < \infty$ as required. The next example revisits a small-scale problem considered earlier in this thesis, in order to graphically illustrate the speed at which the estimates $\hat{Q}_\theta(\mathbf{x})$ converge to the theoretical values $Q_\theta(\mathbf{x})$ during the TD learning algorithm.

Example 7.2.1. ***(Convergence of TD learning)***

This example uses the same parameters as Example 3.2.1. There is a demand rate $\lambda = 1$ and two facilities with one service channel each ($c_1 = c_2 = 1$). The other parameters are:

$$\begin{aligned}
\mu_1 &= 0.8, \qquad \beta_1 = 2, \qquad \alpha_1 = 6, \\
\mu_2 &= 0.4, \qquad \beta_2 = 1, \qquad \alpha_2 = 4.
\end{aligned}$$

Let θ be a stationary policy which chooses actions as shown in Table 7.1. It may be seen that θ induces an ergodic Markov chain defined on a set S_θ consisting of only 6 states.

The random transitions for the system operating under θ may be simulated by allowing each transition to depend on the value of a uniformly-distributed random number between 0 and 1, as described in (7.2.1). Let $r(\mathbf{x}, a, \mathbf{y})$ be defined for $(\mathbf{x}, a, \mathbf{y}) \in S \times A_{\mathbf{x}} \times S$ as follows:

$$r(\mathbf{x}, a, \mathbf{y}) := \sum_{i=1}^{N} \left(\alpha_i \min(y_i, c_i)\mu_i - \beta_i y_i \right).$$

	$x_2 = 0$	$x_2 = 1$
$x_1 = 0$	1	1
$x_1 = 1$	2	1
$x_1 = 2$	2	0

Table 7.1: The policy θ for the system in Example 7.2.1.

In this example, the reward r_n acquired after choosing action $a_n \in A_{\mathbf{x}_n}$ at the n^{th} iteration of TD learning will be defined in terms of the *next* state reached $\mathbf{x}_{n+1}$ as follows:

$$r_n := r(\mathbf{x}_n, a_n, \mathbf{x}_{n+1}). \tag{7.2.9}$$

Thus, the definition of r_n is similar to that of the 'real-time' reward $r(\mathbf{x})$ used in previous chapters except that it depends on the service completion rates and holding costs incurred at the *next* state $\mathbf{x}_{n+1}$, as opposed to the current state $\mathbf{x}_n$. Experiments have shown that defining the rewards r_n in this way leads to a slightly improved performance for TD learning; moreover, allowing r_n to depend on the randomly-generated state $\mathbf{x}_{n+1}$ (as opposed to being completely determined by the state $\mathbf{x}_n$ and action a_n) is somewhat more consistent with the general RL framework, which assumes that the reward r_n is not revealed to the agent until *after* the action a_n has been chosen.

The learning parameters δ_n will be given by (7.2.7) in this example, with $T = 10$. For the secondary learning parameters used in step 3 of the algorithm, it will be sufficient to use the state-independent rule $\zeta_n = 1/(n+1)$ for all $n \geq 0$ (note that adopting this definition ensures that ζ_n decays to zero faster than δ_n). Figure 7.2 shows, for each state $\mathbf{x} \in S_\theta$, the evolution of the estimate $\hat{Q}_\theta(\mathbf{x})$ as the number of iterations progresses during one trial of the TD learning algorithm. The 'true' values $Q_\theta(\mathbf{x})$ (calculated by solving the equations (7.2.3) exactly) are also shown as 'dashed' lines (- - -), in order to enable a comparison. It should be noted that the $Q_\theta(\mathbf{x})$ and $\hat{Q}_\theta(\mathbf{x})$ values shown in the figure are *relative* values, with $Q_\theta((0,0))$ set to a value of zero in order to obtain a unique solution to the equations (7.2.3). The estimated values $\hat{Q}_\theta(\mathbf{x})$ are scaled in the same way, so that $\hat{Q}_\theta((0,0)) = 0$; this is why the values $Q_\theta((0,0))$ and $\hat{Q}_\theta((0,0))$ are not shown in the figure (since they are both set equal to zero). The figure shows that each estimated value $\hat{Q}_\theta(\mathbf{x})$ appears to converge to its expected value $Q_\theta(\mathbf{x})$ as the number of learning iterations increases. ⊠

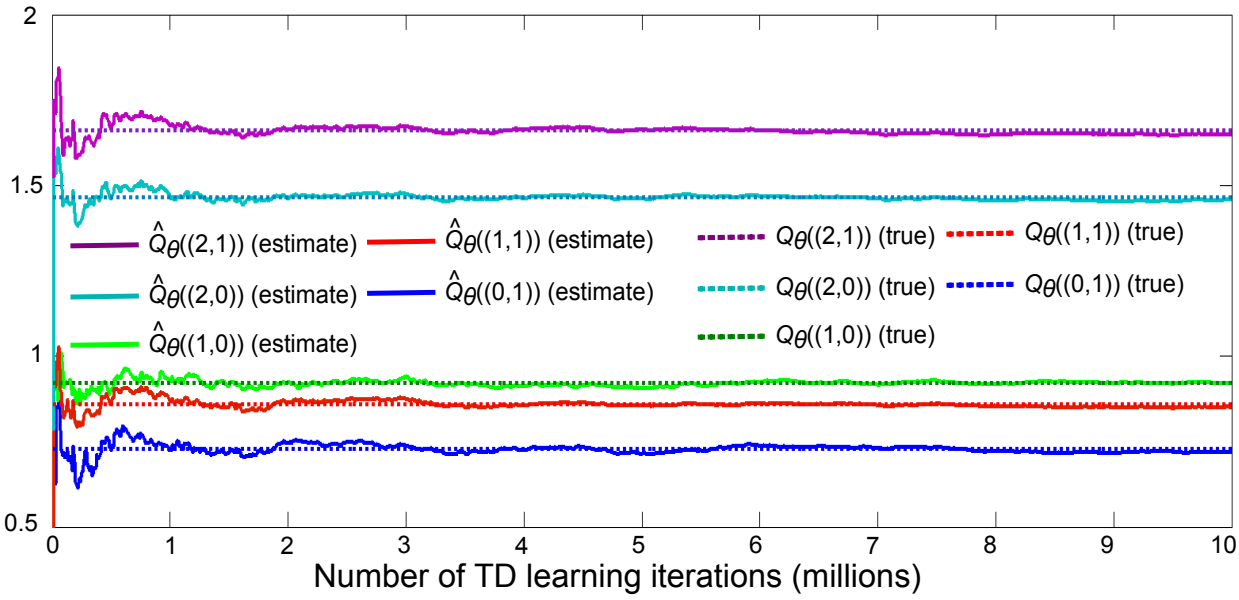


Figure 7.2: Convergence of the estimated values $\hat{Q}_\theta(\mathbf{x})$ to the true values $Q_\theta(\mathbf{x})$ in Example 7.2.1.

The TD learning algorithm on page 292 estimates the Q-factors and the expected long-run average reward under a *fixed* stationary policy θ. As one might expect, a more formidable challenge for an RL algorithm is to learn the Q-factors associated with an *average reward optimal* policy θ^* (without prior knowledge of this policy!) and the corresponding value of the average reward. As discussed earlier in this section, an RL algorithm will generally need to rely upon a mixture of *exploration* and *exploitation* in order to accomplish this task. In the present context, *exploration* refers to the act of choosing an action at random at a particular state, while *exploitation* refers to the choice of an action a which maximises $\hat{Q}(\mathbf{x}, a)$ (where $\mathbf{x} \in S$ is the current state, and $\hat{Q}(\mathbf{x}, a)$ is the latest available estimate of the value $Q(\mathbf{x}, a)$ associated with an *optimal* policy). One possible way for an RL algorithm to strike a balance between exploration and exploitation is to use a so-called *epsilon-greedy rule* for selecting actions. This mode of decision-making is defined below.

Definition 7.2.2. ***(Epsilon-greedy rule)***

An ϵ-greedy *rule is a rule for choosing actions which operates as follows:*

- *With probability* $\epsilon \in (0, 1)$*, a random choice is made; that is, each action* $a \in A_{\mathbf{x}}$ *is selected with probability* $1/|A_{\mathbf{x}}|$ *(where* $\mathbf{x} \in S$ *is the current state).*

- *With probability* $1 - \epsilon$*, an action* $a \in \arg\max_{a \in A_{\mathbf{x}}} \hat{Q}(\mathbf{x}, a)$ *is chosen. (In the case of ties, a*

random choice can be made among actions which attain the maximum.)

An RL algorithm may simply apply an ϵ-greedy rule, with some fixed value of $\epsilon \in (0, 1)$, at every decision epoch in order to ensure adequate coverage of the action sets at every state that it visits (or, at least, the states that it visits infinitely often). Typically, the value of ϵ will be small, since the priority of the algorithm (especially in its later stages) should be to ensure that the Q-factors associated with actions that are perceived as 'optimal' at the various states in S are estimated as accurately as possible, and therefore these actions should be 'sampled' more often than actions that are likely to be sub-optimal. Provided that exploratory actions are chosen at least *some* of the time, the RL algorithm should always have the opportunity to 'correct itself' by updating the set of actions that it believes to be optimal at any particular state; as such, it should be capable of eventually learning an optimal policy. An alternative to an ϵ-greedy rule is a rule for selecting actions by which exploratory choices are made almost exclusively in the early stages of the algorithm, but the probability of selecting exploratory actions decays towards zero (or a small positive number) as the number of iterations increases. For example, consider a rule which is similar to an ϵ-greedy rule, except that the probability of 'exploring' is given by:

$$P_n := \frac{K}{n + K}, \tag{7.2.10}$$

where n is the number of iterations completed, and K is a large integer (e.g. $K = 10^6$). The probability of choosing an 'exploitative' action is then given by $1 - P_n = n/(n+K)$, which increases with n. Applying a rule of this form yields the desirable effect of allowing a lot of exploration to be done in the early stages of the algorithm; however, the choice of the constant K can create a dilemma. If K is too small, then P_n will decay too quickly and the algorithm may therefore be unable to carry out enough exploration to find an optimal policy; instead, it may become 'stuck' evaluating a sub-optimal policy. On the other hand, if K is too large then the algorithm will simply waste too much time evaluating Q-factors associated with sub-optimal actions; although one might still expect it to find an optimal policy eventually, the convergence of the estimated values $\hat{Q}(\mathbf{x}, a)$ to their true values (if any such convergence occurs at all) will be undesirably slow.

Throughout this chapter, a number of different RL algorithms for learning average reward optimal policies will be considered. Experiments have shown that, although there usually does exist some value of K such that the rule given in (7.2.10) for selecting actions appears to out-perform any

ϵ-greedy rule with respect to speed of convergence to a near-optimal policy, the optimal choice of K tends to vary quite drastically according to the particular RL algorithm under consideration and the scale of the problem (in terms of dimensionality of the state space, etc.) to which the algorithm is applied. For the sake of simplicity and to ensure the fairest possible comparison between the various algorithms to be discussed in this chapter, the examples given later will assume that an ϵ-greedy rule is used for action selection, with the same value of ϵ used in each case.

It is possible to derive an RL algorithm for learning an optimal policy by making only a few minor adjustments to the TD learning algorithm given on page 292. The first adjustment required is that, instead of the actions selected by the agent being determined by a fixed stationary policy θ, actions should be selected using a rule which allows for both exploration and exploitation (such as an ϵ-greedy rule). The second modification involves a change to the update rule used for the Q-factor estimates, so that the quantities being estimated are the values $Q(\mathbf{x}, a)$ associated with an *optimal* policy, as opposed to the values $Q_\theta(\mathbf{x})$ associated with a fixed policy θ. The update rule used for the estimated long-run average reward g_n must also be modified, so that the optimal value g^* is approximated, as opposed to the policy-dependent value g_θ. The specific changes required to the update rules for $\hat{Q}(\mathbf{x}, a)$ and g_n can easily be inferred via a comparison between the optimality equations (7.2.2) and the policy evaluation equations (7.2.3), and details are provided in the algorithm given below, referred to in the literature as *R-learning* (see [62, 154]).

R-learning algorithm

1. Initialise the values $n = 0$, $g_0 = 0$ and an array of values $\hat{Q}(\mathbf{x}, a)$ with $\hat{Q}(\mathbf{x}, a) = 0$ for each state-action pair $(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$. Choose an initial state $\mathbf{x}_0 \in S$.

2. Select an action $a_n \in A_{\mathbf{x}_n}$ according to a rule which allows for both exploration and exploitation (e.g. an ϵ-greedy rule, or the rule described in (7.2.10)).

3. Use simulation to determine the reward r_n and the new state $\mathbf{x}_{n+1}$ reached at the next time step. Then, update $\hat{Q}(\mathbf{x}_n, a_n)$ according to the following update rule:

$$\hat{Q}(\mathbf{x}_n, a_n) \leftarrow (1 - \delta_n)\hat{Q}(\mathbf{x}_n, a_n) + \delta_n \left[r_n - g_n + \max_{a' \in A_{\mathbf{x}_{n+1}}} \hat{Q}(\mathbf{x}_{n+1}, a') \right], \tag{7.2.11}$$

where $\delta_n \in [0, 1]$ is a learning parameter which may be state-dependent.

4. If the action chosen in step 2 was an *exploitative* action, calculate g_{n+1} as follows:

$$g_{n+1} := (1-\zeta_n)g_n + \zeta_n \left[r_n + \max_{a' \in A_{\mathbf{x}_{n+1}}} \hat{Q}(\mathbf{x}_{n+1}, a') - \max_{a \in A_{\mathbf{x}_n}} \hat{Q}(\mathbf{x}_n, a) \right], \tag{7.2.12}$$

where $\zeta_n \in [0,1]$ is another learning parameter. Otherwise, set $g_{n+1} = g_n$.

5. If $n = n_{\max}$, where $n_{\max}$ is a large integer, then output a stationary policy $\theta^{[R]}$ which, at each state $\mathbf{x} \in S$, chooses an action $\theta^{[R]}(\mathbf{x})$ maximising the expression $\hat{Q}(\mathbf{x}, a)$ over all actions $a \in A_{\mathbf{x}}$. Also output g_{n+1} as an estimate for the optimal expected average reward g^*, then stop. Alternatively, if $n < n_{\max}$, increment n by 1 and return to step 2.

The R-learning algorithm was introduced by Schwartz [154] as a variation on a much more famous RL algorithm known as *Q-learning*, which is due to Watkins [191]. Q-learning, in its conventional form, is applicable to problems with *discounted rewards*, whereas R-learning applies to average reward problems. Unfortunately, while the convergence properties of Q-learning have been proved rigorously, there does not exist any formal proof that R-learning converges towards an average reward optimal policy. This topic is discussed in greater detail in Appendix A.9.

Of course, the main indicator of the performance of the R-learning algorithm is the expected long-run average reward attained under the policy $\theta^{[R]}$ found after $n_{\max}$ iterations, i.e. when the procedure is terminated. However, it is also possible to interrupt the R-learning procedure at certain points during its running time in order to evaluate its latest 'guess' for an average reward optimal policy; in other words, the procedure can be temporarily halted after n_j iterations (where $n_j < n_{\max}$) in order to examine the stationary policy obtained by choosing an action a at each state $\mathbf{x} \in S$ which maximises $\hat{Q}(\mathbf{x}, a)$. The evaluation of this policy can either be carried out using dynamic programming, or by using simulation; obviously, simulation will be less exact than dynamic programming, but also considerably faster. The use of simulation after a fixed number of iterations of R-learning to evaluate the latest guess for an optimal policy is referred to by Gosavi [64] as *Frozen Phase* (FP) updating, so-called because the latest available estimates for the Q-factors are 'frozen' during the simulation, as opposed to being updated after each random transition (as in the main phase of R-learning itself). The steps of an FP update are as follows:

Steps in 'Frozen Phase' (FP) updating

a) Initialise the values $k = 0$, $r_{tot} = 0$ and $t_{tot} = 0$. Choose an initial state $\mathbf{x}_0 \in S$.

b) Choose an action $a_k \in A_{\mathbf{x}_k}$ satisfying $a_k \in \arg\max_{a \in A_{\mathbf{x}_k}} \hat{Q}(\mathbf{x}_k, a)$, where $\hat{Q}(\mathbf{x}_k, a)$ is the latest available estimate of $Q(\mathbf{x}_k, a)$ obtained via R-learning.

c) Use simulation to determine the reward r_k and the new state $\mathbf{x}_{k+1}$ reached at the next time step. Then update the quantities r_{tot} and t_{tot} as follows:

$$r_{tot} \leftarrow r_{tot} + r_k,$$
$$t_{tot} \leftarrow t_{tot} + 1.$$

d) If $k = k_{\max}$, where $k_{\max}$ is a large integer, then output $\hat{g}_{n_j} := r_{tot}/t_{tot}$ (where n_j is the number of iterations of R-learning completed before interrupting the procedure), then resume the R-learning algorithm. Otherwise, increment k by 1 and return to step (b).

FP updates can be performed at regular stages of the R-learning algorithm in order to monitor its progress towards finding an average reward optimal policy. For example, suppose these updates are to be performed after every 1000 iterations of R-learning. Then, setting $n_j = 10^3 j$ for $j \in \mathbb{N}$, one obtains a sequence of values $(\hat{g}_{n_1}, \hat{g}_{n_2}, ...)$ which, ideally, should converge towards the optimal long-run average reward g^* (although it is unlikely that this convergence will be monotonic; nor should one expect it to follow any predictable pattern). It should also be pointed out that in [64], Gosavi relies upon FP updates as part of an algorithm which he refers to as R-SMART. In R-SMART, the estimates $\hat{g}_{n_j}$ obtained from FP updates are actually retained and used as estimates of g^* in the main part of the algorithm itself, effectively taking the place of the estimates g_n used by R-learning. It is therefore important to emphasise here that the R-learning algorithm given on page 301 has *no dependence* whatsoever on the values $\hat{g}_{n_j}$ obtained from FP updates; rather, FP updates (if desired) may be performed periodically in order to monitor the performance of the R-learning algorithm, but R-learning then resumes after each update as if no interruption had occurred.

The next example demonstrates the application of the R-learning algorithm to a problem which (unlike the problem considered in the previous example) is non-trivial in terms of the amount of computational effort required in order to find an average reward optimal policy via dynamic programming. At this stage, however, it is desirable to remain within the realms of problems that *can* be solved using DP in a reasonable amount of time, in order to enable a comparison between the policy found by R-learning and an optimal policy found using value iteration.

Example 7.2.3. ***(Convergence of R-learning)***

Consider a system with demand rate $\lambda = 25$ and 4 service facilities, each with multiple service channels available. The parameters for the 4 facilities are given below.

$$\begin{array}{llll} c_1 = 4, & \mu_1 = 1, & \beta_1 = 4, & \alpha_1 = 14, \\ c_2 = 3, & \mu_2 = 4, & \beta_2 = 5, & \alpha_2 = 7, \\ c_3 = 3, & \mu_3 = 8, & \beta_3 = 8, & \alpha_3 = 4, \\ c_4 = 2, & \mu_4 = 16, & \beta_4 = 5, & \alpha_4 = 2. \end{array} \tag{7.2.13}$$

This will be a running example considered throughout this chapter. The results from Section 4.2 imply that an average reward optimal policy exists which induces an ergodic Markov chain defined on a finite set of states contained in $\tilde{S}$, where $\tilde{S}$ is the selfishly optimal state space defined in (4.1.3). In order to improve the performance of the R-learning algorithm, it is therefore reasonable to restrict the action sets $A_{\mathbf{x}}$ at states $\mathbf{x}$ which are on the boundary of $\tilde{S}$ so that the process remains contained in the set $\tilde{S}$, although (as discussed earlier) RL algorithms are not necessarily restricted to a finite state space in general. In this example, the 'selfish thresholds' $\tilde{B}_1$, $\tilde{B}_2$, $\tilde{B}_3$ and $\tilde{B}_4$ are equal to 14, 16, 12 and 12 respectively. Hence, $\tilde{S}$ consists of a total of 43,095 states.

The purpose of this example is to show the results of running the R-learning algorithm with a value $\epsilon = 0.15$ used for ϵ-greedy action selection, rewards r_n defined as in (7.2.9) and a rule for the learning parameters δ_n which depends upon states and actions as follows:

$$\delta_n := \frac{T}{T + \nu_n(\mathbf{x}_n, a_n) - 1}, \tag{7.2.14}$$

where $\nu_n(\mathbf{x}_n, a_n)$ denotes the total number of times that action a_n has been chosen at state $\mathbf{x}_n$ up to and including time $n \geq 0$. Note that the rule given in (7.2.14) is the natural generalisation of the rule (7.2.7) to a problem in which various decision options are available at each state, as opposed to actions being prescribed by a fixed policy θ. As in Example 7.2.1, the value $T = 10$ will be used in (7.2.14), and the secondary learning parameters ζ_n will be given by $\zeta_n = 1/(n+1)$.

Figure 7.3 shows the evolution of the estimated value g_n over the course of 40 million iterations of R-learning, and also the progression of the values $\hat{g}_{n_j}$ obtained by performing Frozen Phase (FP) updates after every 10,000 iterations. It should be noted that the random transition occurring on a particular iteration may be a 'dummy' transition, since these transitions are simulated in discrete

time (using the technique of uniformisation). The optimal average reward may be calculated exactly using value iteration, and the resulting value $g^* \approx 126.611$ is indicated in the figure.

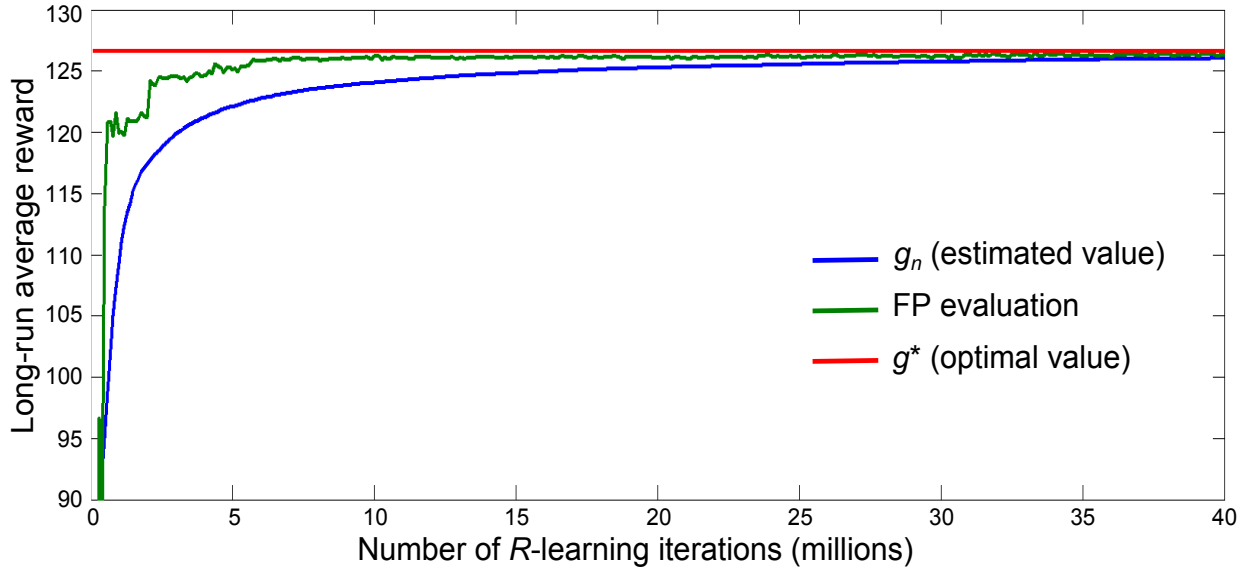


Figure 7.3: Convergence of g_n and $\hat{g}_{n_j}$ to the optimal value g^* in Example 7.2.3.

Figure 7.3 shows that, in this example, g_n appears to converge in a smooth fashion towards the optimal value g^* as the number of iterations n increases. On the other hand, the progression of the FP estimates is somewhat erratic in the early stages of the algorithm, but these values begin to converge towards g^* after approximately 100,000 iterations have been completed. The FP estimates are, in a sense, the most important indicator of the algorithm's performance, since each FP estimate shows (subject to a small margin of error caused by simulation) the performance of the policy that would be obtained by stopping the procedure at that particular point. The figure suggests that (except in the initial stages of the algorithm) the g_n values actually tend to underestimate the long-run average reward associated with the 'best' policy found by R-learning thus far. After 40 million iterations, the policy $\theta^{[R]}$ returned in step 5 of the R-learning algorithm attains an average reward of approximately 126.28, within 0.3% of the known optimal value $g^* \approx 126.61$.

Although the R-learning algorithm succeeds in finding a policy $\theta^{[R]}$ which almost attains average reward optimality in this particular example, it is interesting to note that this near-optimal policy is obtained *without* exploring a large proportion of the state-action pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$. This can be seen by examining the values $\nu_n(\mathbf{x}, a)$ for pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ after 40 million iterations. The

distribution of $\nu_n(\mathbf{x}, a)$ values (with $n = 40$ million) is illustrated in Table 7.2.

Range	No. of pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$	Pct. of pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$
$\nu_n(\mathbf{x}, a) = 0$	185,259	94.806
$1 \leq \nu_n(\mathbf{x}, a) \leq 9$	4,183	2.141
$10 \leq \nu_n(\mathbf{x}, a) \leq 99$	3,090	1.581
$100 \leq \nu_n(\mathbf{x}, a) \leq 999$	1,793	0.918
$1000 \leq \nu_n(\mathbf{x}, a) \leq 9999$	754	0.386
$10{,}000 \leq \nu_n(\mathbf{x}, a) \leq 99{,}999$	267	0.137
$100{,}000 \leq \nu_n(\mathbf{x}, a) \leq 999{,}999$	55	0.028
$\nu_n(\mathbf{x}, a) \geq 1{,}000{,}000$	7	0.004

Table 7.2: Distribution of values $\nu_n(\mathbf{x}, a)$ for $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ after 40 million iterations of R-learning.

Remarkably, almost 95% of state-action pairs in $\tilde{S} \times A_{\mathbf{x}}$ are not visited at all during these 40 million iterations! Of course, if a particular state-action pair $(\mathbf{x}, a)$ remains unvisited during the evolution of the algorithm, then the Q-factor estimate $\hat{Q}(\mathbf{x}, a)$ remains at its default value of zero. Thus, this example shows that R-learning is capable of finding a near-optimal policy by obtaining meaningful Q-factor estimates at only a small minority of the state-action pairs that it is allowed to explore. This also indicates the amount of redundancy involved in using a dynamic programming algorithm (which repeatedly sweeps through *all* state-action pairs) in problems consisting of tens of thousands of system states, in which it may not be necessary to find an *exact* optimal policy.

In this example, an ϵ-greedy rule was used for action selection, with $\epsilon = 0.15$. Figure 7.4 shows the results of repeating the experiment with various alternative choices of ϵ (recall that ϵ is the probability that an 'exploratory' action will be chosen on a particular iteration). The figure shows the progression of the estimates $\hat{g}_{n_j}$ obtained from FP evaluations, for various different values of ϵ. Since the FP estimates appear to converge quite quickly towards the optimal value g^* in most cases, only the behaviour observed during the first 5 million iterations of R-learning has been shown. The most obvious conclusion to draw from the figure is that $\epsilon = 0.7$ and $\epsilon = 0.9$ are not good choices in this particular example. In general, if the value of ϵ is set too high, the convergence of the FP estimates towards g^* will tend to be slow, since the algorithm will not have sufficient opportunity

to approximate the Q-factors accurately for state-action pairs which are likely to feature as part of an optimal policy. On the other hand, when the *smallest* value of ϵ ($\epsilon = 0.1$) is used, the FP estimates appear to behave quite erratically in the early stages of the algorithm.

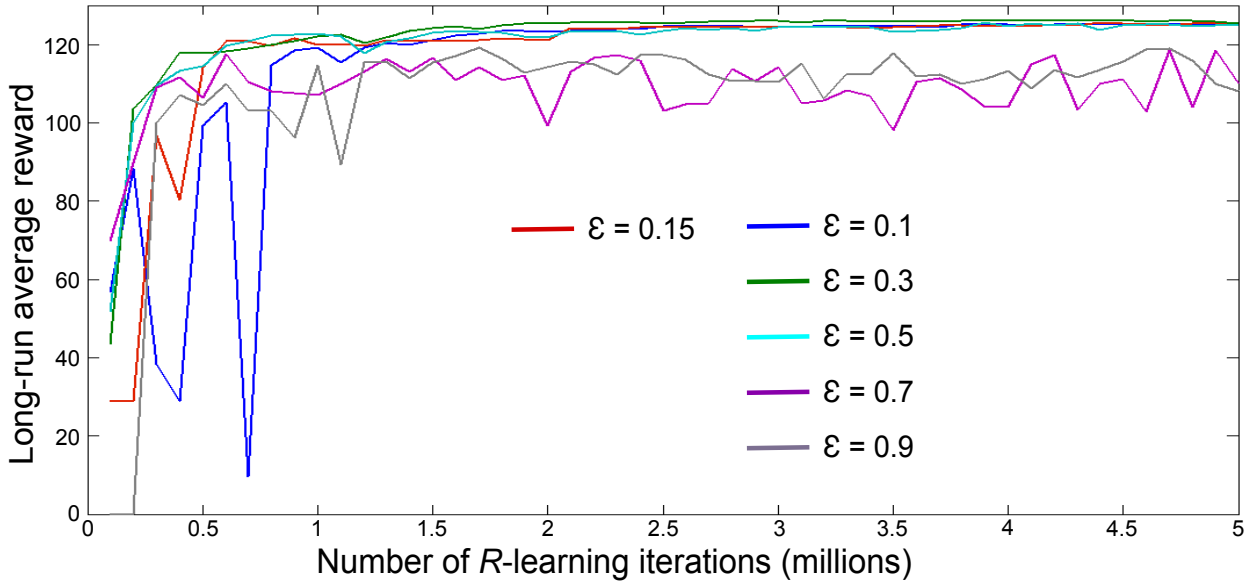


Figure 7.4: The effect of varying the exploration parameter ϵ in Example 7.2.3.

Further experiments (details of which are not provided here) appear to suggest that in general, the major pitfall to be avoided when using an ϵ-greedy rule is a value of ϵ which is too *high*. Small values of ϵ may result in quite slow convergence in the *initial* stages of R-learning, but this is arguably a price worth paying in order to ensure that the algorithm focuses almost exclusively on 'optimal' actions in the later stages of its finite running time. In fact, as noted earlier in this section, it may be preferable to avoid using an ϵ-greedy rule completely; one can instead use a rule such as (7.2.10), under which the probability of exploring decays with time. However, in this case one is faced with the dilemma of choosing a rate of decay which is suitable for the given problem.

It should also be noted that, although Figure 7.3 might appear to suggest that a very large number of iterations are required before R-learning begins to approach an optimal policy, an RL algorithm generally requires much less time than a DP algorithm to perform a fixed number of iterations. This is because each iteration of an RL algorithm involves evaluating (via simulation) only one state-action pair, whereas DP algorithms evaluate *all* such pairs on every iteration. Of course, the estimation of g^* is also required in R-learning, but this is insignificant in comparison to the

computational requirements of a DP-style update. Thus, the requirement for millions of iterations to be performed usually does not cause a significant problem for an RL algorithm. ⊠

In general, a fast computer should be capable of performing tens of millions of iterations of the R-learning algorithm in less than a minute. Table 7.3 shows the results of another set of experiments, in which the running times of the R-learning algorithm were compared with those of the RVIA (see page 101) over a number of different test systems. Ten tests were performed, and in each test, the RVIA and R-learning algorithms were applied to a system which featured the same set of parameters as in Example 7.2.3, but with the values of the rewards α_i scaled either upwards or downwards in order to increase or decrease the size of the finite state space $\tilde{S}$. In each test, the RVIA used a stopping parameter $\epsilon = 10^{-6}$, and the R-learning algorithm used the same parameter settings as specified in Example 7.2.3 (for ϵ, δ_n, etc.) and was allowed to run for 50 million iterations. The tests were performed on a computer with an Intel Core i7 processor ($\sim$ 2.8 Ghz) and 4GB of RAM. It is important to clarify that Frozen Phase (FP) evaluations were *not* performed in any of the tests, so the running times shown in the table for the R-learning algorithm are simply the times taken to perform 50 million iterations, *without* interrupting the procedure at any stage. After each test, the average reward earned by the R-learning policy $\theta^{[R]}$ was evaluated using dynamic programming and the resulting sub-optimality (relative to the known optimal value g^*) is shown.

No. of states in $\tilde{S}$	RVIA running time	R-learning running time	Sub-optimality of $\theta^{[R]}$
4032	6.79 secs	56.88 secs	0.04%
12,600	22.02 secs	56.20 secs	0.08%
24,624	44.65 secs	55.72 secs	0.10%
33,800	1 min, 7 secs	56.14 secs	0.08%
45,800	1 min, 31 secs	56.13 secs	1.16%
104,346	3 mins, 53 secs	56.79 secs	0.34%
257,439	11 mins, 7 secs	56.50 secs	3.84%
508,950	31 mins, 58 secs	56.66 secs	0.83%
750,288	48 mins, 35 secs	56.55 secs	0.50%
1,014,816	71 mins, 55 secs	56.71 secs	0.28%

Table 7.3: Comparisons between the running times of the RVIA and R-learning algorithms.

The results in Table 7.3 indicate that, as expected, the running time of the RVIA increases steeply as the number of states that it needs to 'loop through' on each iteration increases. However, the running time of the R-learning algorithm (with 50 million iterations) remains almost constant at roughly 56 seconds. It should be noted, however, that all of the 10 systems tested in this experiment featured the same number of facilities ($N = 4$). If the number of facilities N was increased, then the R-learning algorithm *would* require a longer running time, since it would need to inspect a greater number of actions at any given state in order to identify the action with the largest Q-factor. In 4-facility systems, it appears that the 'critical value' of $|\tilde{S}|$ which causes the RVIA to be slower than 50 million iterations of R-learning is roughly 30,000. The table also indicates that the R-learning algorithm is consistently successful in attaining policies which earn average rewards very close to the optimal value g^* (although there is one apparent outlier in the seventh row).

Ideally, one would like to have a theoretical guarantee that by running an RL algorithm such as R-learning for an indefinite amount of time (as opposed to stopping it after a finite number of iterations), an average reward optimal policy would eventually be obtained. Unfortunately, there does not exist any formal proof that the R-learning algorithm presented on page 301 converges to an average reward optimal policy, although its strong performance has been demonstrated in numerical experiments (see the discussions in [123, 154, 162]). The TD learning algorithm presented on page 292, which evaluates a fixed stationary policy θ, may be recognised as a special case of the R-learning algorithm with only one action allowed at each state (in the same way that the Policy Evaluation Algorithm given on page 98 is derivable from the Relative Value Iteration Algorithm on page 101), and as such a proof of its convergence to the theoretical value g_θ is not attainable either. However, there *does* exist an RL algorithm with proven convergence properties, known as *relative Q-learning*, which can be applied to average reward problems. For a brief literature survey regarding the convergence properties of RL algorithms, please refer to Appendix A.9.

7.3 Tailored RL algorithms

The RL algorithms discussed in this chapter so far have been *general-purpose* algorithms. Algorithms such as TD learning, R-learning etc. can be applied to virtually any real-world problem which can theoretically be cast as an MDP, with the intended result that a set of Q-factors are

obtained which closely approximate a set of values satisfying the Bellman optimality equations (or, in the case of TD learning, evaluation equations). The next objective is to explore ways of modifying these algorithms by incorporating knowledge of the characteristics of the queueing problems considered in this thesis, in order to improve overall efficiency. In other words, the aim is now to *tailor* the RL algorithms discussed previously to the specific problem(s) at hand.

According to the general paradigm of reinforcement learning (as illustrated by Figure 7.1), an action a_n is chosen by the agent at each discrete epoch of time $n \in \mathbb{N}_0$. A random transition then occurs, which results in a reward r_n being earned and a new state $\mathbf{x}_{n+1}$ being reached. In the queueing system problems considered in this thesis, the random transition is determined by a 'random event', such as a new arrival, a service completion or (possibly) neither of these. Let ω_n denote the random event that occurs at time step n (following the choice of action a_n). Using notation similar to that in previous chapters, the event ω_n is an element of the finite set W given by:

$$W := \{0, \Lambda, M_1, M_2, ..., M_N\},$$

where 0 denotes a 'non-event' (i.e. no arrivals or service completions), Λ denotes a customer arrival, and M_i (for $i \in \{1, 2, ..., N\}$) denotes a service completion at facility i. Suppose the simulated random transitions of the system take place according to the transition rule given in (7.2.1). It is clear that (7.2.1) may be expressed in terms of the random event ω_n as follows:

$$\mathbf{x}_{n+1} = \begin{cases} \mathbf{x}_n^{i+}, & \text{if } \omega_n = \Lambda \text{ and } a_n = i \text{ for some } i \in \{1, 2, ..., N\}, \\ \mathbf{x}_n^{i-}, & \text{if } \omega_n = M_i \text{ for some } i \in \{1, 2, ..., N\}, \\ \mathbf{x}_n, & \text{otherwise.} \end{cases} \tag{7.3.1}$$

In order for (7.3.1) to be consistent with (7.2.1), ω_n should be determined as follows:

$$\omega_n = \begin{cases} \Lambda, & \text{if } u < \lambda\Delta, \\ M_i, & \text{if } \lambda\Delta + \sum_{j=1}^{i-1} \min((\mathbf{x}_n)_j, c_j)\mu_j\Delta \leq u < \lambda\Delta + \sum_{j=1}^{i} \min((\mathbf{x}_n)_j, c_j)\mu_j\Delta \\ & \text{for some } i \in \{1, 2, ..., N\}, \\ 0, & \text{otherwise,} \end{cases} \tag{7.3.2}$$

where u is a uniformly-distributed random number between 0 and 1. From (7.3.2), it is clear that the event ω_n has *no dependence* on the action a_n chosen at time n. To make sense of this, recall

that in the discrete-time MDP Υ formulated in Section 3.5, the action a_n chosen at a particular time step n may be interpreted as a hypothetical action which determines the fate of any customer who arrives at time n; if no arrival occurs, then the action a_n has no effect on the system state. The probability of an arrival occurring at any time step is $\lambda\Delta$, regardless of the action chosen; similarly, the probability of a service completion at facility $i \in \{1, 2, ..., N\}$ at time n is $\min((\mathbf{x}_n)_i, c_i)\mu_i\Delta$, which again is independent of a_n. Assuming that the reward formulation (7.2.9) is used, the reward r_n is then completely determined by the current state $\mathbf{x}_n$, action a_n and random event ω_n, and the same is true of the destination state $\mathbf{x}_{n+1}$. The fact that ω_n is independent of a_n is crucial in the conception of the first of the 'tailored' RL algorithms to be discussed in this section.

In the general RL model described in Section 7.1, the agent discovers the 'new information' r_n and $\mathbf{x}_{n+1}$ after choosing an action a_n at time step $n \in \mathbb{N}_0$. Importantly, it is not assumed that the agent has any means of knowing the reward and the new state that *would* have resulted from choosing a different action at time n; that is, the agent learns only the consequences of choosing one particular action. As such, one might say that the agent does not have the power of *hindsight*.

However, when one considers an N-facility queueing system which evolves according to simulated random transitions as described by (7.3.1) and earns rewards given by (7.2.9), it becomes clear that, in fact, a more efficient RL algorithm can be designed which makes use of the fact that the random event ω_n occurring at any time step n completely determines (in conjunction with the state $\mathbf{x}_n$) both the reward r_n and the new state $\mathbf{x}_{n+1}$ that would result from choosing *any* action $a \in A_{\mathbf{x}_n}$. For example, suppose the system is in some state $\mathbf{x}_n \in S$ at time n, and the action chosen is $a_n = i$ for some $i \in \{1, 2, ..., N\}$. If the event $\omega_n = \Lambda$ occurs, then (with certainty) the next state is $\mathbf{x}_{n+1} = \mathbf{x}_n^{i+}$. However, one can also say that if the action $a_n = j$ had been chosen (for some $j \in \{1, 2, ..., N\}$ with $j \neq i$), then the same random event ($\omega_n = \Lambda$) *would* have caused the new state to be $\mathbf{x}_{n+1} = \mathbf{x}_n^{j+}$. In general, although the agent is only permitted to choose one action at any given state, one can always say (following observation of the random event ω_n) what the reward r_n and next state $\mathbf{x}_{n+1}$ *would* have been if any other action had been chosen.

By exploiting this principle, it is possible to design a new RL algorithm which is similar to R-learning except that, on any given iteration n, the estimated values $\hat{Q}(\mathbf{x}_n, a)$ for *all* actions $a \in A_{\mathbf{x}_n}$ are updated, as opposed to only the value $\hat{Q}(\mathbf{x}_n, a_n)$ associated with the action a_n actually chosen

by the agent. In a sense, the new algorithm 'learns from hypothetical actions', by observing the random event ω_n at each time step $n \in \mathbb{N}_0$ and then using 'hindsight' to update the values $\hat{Q}(\mathbf{x}_n, a)$ associated with actions a *not* chosen by the agent. This new RL algorithm will be referred to here as 'HYPOT R-learning'. A complete description of the algorithm is given below.

HYPOT R-learning algorithm

1. Initialise the values $n = 0$, $g_0 = 0$ and an array of values $\hat{Q}(\mathbf{x}, a)$ with $\hat{Q}(\mathbf{x}, a) = 0$ for each state-action pair $(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$. Choose an initial state $\mathbf{x}_0 \in S$.

2. Use simulation to determine the random event ω_n that occurs at time step n.

3. Let $\mathbf{y} : S \times A_{\mathbf{x}} \times W \to S$ be defined for $(\mathbf{x}, a, \omega) \in S \times A_{\mathbf{x}} \times W$ by:

$$\mathbf{y}(\mathbf{x}, a, \omega) = \begin{cases} \mathbf{x}^{i+}, & \text{if } \omega = \Lambda \text{ and } a = i \text{ for some } i \in \{1, 2, ..., N\}, \\ \mathbf{x}^{i-}, & \text{if } \omega = M_i \text{ for some } i \in \{1, 2, ..., N\}, \\ \mathbf{x}, & \text{otherwise.} \end{cases}$$

   Similarly, let $R : S \times A_{\mathbf{x}} \times W \to \mathbb{R}$ be defined for $(\mathbf{x}, a, \omega) \in S \times A_{\mathbf{x}} \times W$ by:

$$R(\mathbf{x}, a, \omega) := \sum_{i=1}^{N} \left(\alpha_i \min(y_i, c_i)\mu_i - \beta_i y_i\right),$$

   where y_i is the i^{th} component of $\mathbf{y}(\mathbf{x}, a, \omega)$. Then, for *each action* $a \in A_{\mathbf{x}_n}$ available at state $\mathbf{x}_n$, update $\hat{Q}(\mathbf{x}_n, a)$ according to the following update rule:

$$\hat{Q}(\mathbf{x}_n, a) \leftarrow (1 - \delta_n(a))\hat{Q}(\mathbf{x}_n, a) + \delta_n(a)\left[R(\mathbf{x}_n, a, \omega_n) - g_n + \max_{a' \in A_{\mathbf{y}}} \hat{Q}(\mathbf{y}(\mathbf{x}_n, a, \omega_n), a')\right], \tag{7.3.3}$$

   where $A_{\mathbf{y}}$ denotes the set of actions available at state $\mathbf{y}(\mathbf{x}_n, a, \omega_n)$, and $\delta_n(a) \in [0, 1]$ is a learning parameter which may be dependent on the state $\mathbf{x}_n$ and action a.

4. Select an action $a_n \in A_{\mathbf{x}_n}$ according to a rule which allows for both exploration and exploitation (e.g. an ϵ-greedy rule). Then, let $r_n = R(\mathbf{x}_n, a_n, \omega_n)$ be the 'actual' reward earned at time n, and similarly let $\mathbf{x}_{n+1} = \mathbf{y}(\mathbf{x}_n, a_n, \omega_n)$ be the 'actual' next state.

5. If the action a_n chosen in step 4 was an *exploitative* action, calculate g_{n+1} as follows:

$$g_{n+1} := (1 - \zeta_n)g_n + \zeta_n\left[r_n + \max_{a' \in A_{\mathbf{x}_{n+1}}} \hat{Q}(\mathbf{x}_{n+1}, a') - \max_{a \in A_{\mathbf{x}_n}} \hat{Q}(\mathbf{x}_n, a)\right],$$

where $\zeta_n \in [0,1]$ is another learning parameter. Otherwise, set $g_{n+1} = g_n$.

6. If $n = n_{\max}$, where $n_{\max}$ is a large integer, then output a stationary policy $\theta^{[R]}$ which, at each state $\mathbf{x} \in S$, chooses an action $\theta^{[R]}(\mathbf{x})$ maximising the expression $\hat{Q}(\mathbf{x}, a)$ over all actions $a \in A_{\mathbf{x}}$. Also output g_{n+1} as an estimate for the optimal long-run average reward g^*, then stop. Alternatively, if $n < n_{\max}$, increment n by 1 and return to step 2.

On the n^{th} iteration of HYPOT R-learning, the simulated random event ω_n is used to update the estimated Q-factors for $|A_{\mathbf{x}_n}|$ different state-action pairs, where $|A_{\mathbf{x}_n}|$ is the number of actions available at the current state $\mathbf{x}_n \in S$. Intuitively, it is clear this procedure enables the various decision options $a \in A_{\mathbf{x}_n}$ at state $\mathbf{x}_n$ to be compared in a more robust way than the conventional R-learning algorithm presented in Section 7.2, since the consequences of each action $a \in A_{\mathbf{x}_n}$ are evaluated according to exactly the same sequence of random events. In conventional R-learning, a particular action $a \in A_{\mathbf{x}_n}$ might erroneously be believed to be 'superior' to another action $a' \in A_{\mathbf{x}_n}$ at some stage of the procedure simply because the random events observed after choosing action a are in some way more favourable than those observed after choosing action a' (although this type of error should become increasingly unlikely as the number of iterations increases).

Due to the greater use made of the random information obtained on each iteration, one would expect the HYPOT R-learning algorithm to perform better than R-learning over a fixed number of iterations; however, from a computational point of view there is clearly a compromise involved, since HYPOT R-learning requires more work to be done on each iteration. A comparison between the two algorithms with respect to the system in Example 7.2.3 will be given later.

Clearly, HYPOT R-learning has more in common with a dynamic programming algorithm than R-learning does, since it loops over all actions $a \in A_{\mathbf{x}_n}$ on each iteration n, much in the same way that an algorithm such as the RVIA (page 101) considers all possible actions at a given state $\mathbf{x} \in S$ in order to determine the value of its latest iterate $h_n(\mathbf{x})$. However, HYPOT R-learning is still very far from being a DP-style algorithm, since it still follows the fundamental RL principle of updating the estimated Q-factors at states 'as and when they are visited' during the simulation of the process; as such, the manner in which it updates the Q-factor estimates is still completely asynchronous. As discussed previously, the fact that it is possible to use the random event ω_n to update the Q-factors associated with actions $a \in A_{\mathbf{x}_n}$ *not* chosen by the agent on iteration n is

due to the particular MDP formulation from which the RL algorithm is derived, and would not be possible in a more general RL scenario (e.g. a scenario unrelated to queues) in which it might be impossible to know the consequences of a particular action without choosing it.

At this point it should be noted that there do, in fact, exist DP algorithms which do not use a synchronous updating style; see, for example, the asynchronous DP algorithms discussed in [175]. These algorithms differ from more conventional DP algorithms by updating the value function at only a *selected* number of states (or state-action pairs) per iteration, rather than sweeping through the entire state-action space on every iteration; for example, they might update only one pair on each iteration, based on the simulated path of an agent. Nevertheless, as explained in Section 7.2, RL algorithms such as TD learning and R-learning rely upon an update rule for the Q-factor estimates which is derived from the theory of stochastic approximation schemes. In this respect, they differ fundamentally from asynchronous dynamic programming algorithms.

One might reasonably ask why the similarities between HYPOT R-learning and DP algorithms should not be extended even further. In the formulation used by HYPOT R-learning, the probability distribution of the event ω_n occurring on the n^{th} iteration depends only on the state $\mathbf{x}_n \in S$ and can easily be determined using (7.3.2). For each $a \in A_{\mathbf{x}_n}$, the reward $R(\mathbf{x}_n, a, \omega_n)$ and the new state $\mathbf{y}(\mathbf{x}_n, a, \omega_n)$ are then determined using $\mathbf{x}_n$ and ω_n, and this information is used to update $\hat{Q}(\mathbf{x}_n, a)$ in step 3 of the algorithm. It would therefore be entirely feasible to update $\hat{Q}(\mathbf{x}_n, a)$ for each $a \in A_{\mathbf{x}_n}$ based on the *expected value* of the new data piece $R(\mathbf{x}_n, a, \omega_n) - g_n + \max_{a' \in A_{\mathbf{y}}} \hat{Q}(\mathbf{y}(\mathbf{x}_n, a, \omega_n), a')$, rather than an actual observation of this quantity arising from a random realisation of ω_n. Using this approach, one would update $\hat{Q}(\mathbf{x}_n, a)$ for each $a \in A_{\mathbf{x}_n}$ in step 3 as follows:

$$\hat{Q}(\mathbf{x}_n, a) \leftarrow (1-\delta_n(a))\hat{Q}(\mathbf{x}_n, a) + \delta_n(a) \sum_{\omega \in W} P(\omega_n = \omega) \left[R(\mathbf{x}_n, a, \omega) - g_n + \max_{a' \in A_{\mathbf{y}}} \hat{Q}(\mathbf{y}(\mathbf{x}_n, a, \omega), a') \right]. \tag{7.3.4}$$

Step 2 of the algorithm (in which a realisation of ω_n is sampled using simulation) would still be required in order to determine the value of the 'actual' reward r_n and the next state $\mathbf{x}_{n+1}$. One would expect the update rule (7.3.4) to be somewhat more robust than the rule (7.3.3), in which the event ω_n is sampled at random. Indeed, numerical tests using the parameters in Example 7.2.3 have shown that the update rule (7.3.4) appears to yield a slightly faster rate of convergence than (7.3.3) with respect to the number of iterations performed (as opposed to the computation time

elapsed). However, despite their apparent advantages, 'expectation-style' update rules of the form (7.3.4) will not be considered any further in this chapter, for the following reasons:

1. The computational requirements associated with this type of update rule are significantly greater than for those which rely on a single realisation of ω_n on each iteration.

2. More importantly, the rule in (7.3.4) depends upon the probability distribution of ω_n being known. Although this is clearly not a problem in the MDP formulations considered in this chapter thus far, the emphasis will shift later in this chapter towards formulations in which the transition probabilities are difficult (or, from a practical point of view, impossible) to state exactly. Recall that one of the main advantages of an RL algorithm should be its application to MDPs in which exact expressions for transition probabilities are not available.

The second of the two points above portends the introduction of the next RL algorithm to be discussed in this section, which is based on a re-formulation of the MDP used to derive the R-learning and HYPOT R-learning algorithms in which the transition probabilities are much more difficult to write down exactly. Recall that in the continuous-time MDP Ψ introduced in Chapter 3 (which was used to obtain a discrete-time MDP via uniformisation), the transitions of the system were always associated with either a new customer's arrival (denoted by $\omega = \Lambda$) *or* a service completion occurring (denoted by $\omega = M_i$, for some $i \in \{1, 2, ..., N\}$). Accordingly, the set of actions available at any decision epoch would either be $\{0, 1, ..., N\}$ or the singleton $\{0\}$, depending on whether an arrival or a service completion was associated with the relevant epoch.

The very fact that the formulation of Ψ referred to above requires (inconsequential) decisions to be made whenever a service completion occurs would appear to suggest some kind of inherent redundancy, even though the formulation itself is entirely sound and is consistent with CTMDP formulations found in the literature (for example, Puterman's $M/M/1$ formulation in [141], p. 568). This raises the question of whether it is possible to amend the formulation in such a way that decision epochs are *always* associated with arrivals (and not service completions), so that control can be exercised at *every* decision epoch, as opposed to only some decision epochs.

Assuming that the type of re-formulation described above is possible, a DP algorithm would effectively be able to take 'larger steps' through the process, since the time in between any two decision

epochs would always be the time in between two consecutive customer arrivals, regardless of the number of service completions occurring in the meantime. However, in considering the number of possible states accessible from an arbitrary state $\mathbf{x} \in S$ via a single transition, one would need to take into account *all* possibilities for the numbers of service completions occurring at the various facilities prior to the next customer's arrival, which could be an overwhelming task.

The fact that it is possible to amend the MDP formulation in the manner described above without losing the ability to evaluate the expected long-run average reward under any given stationary policy can be shown using rigorous arguments. Consider the evolution of the continuous-time process Ψ operating under some arbitrary stationary policy θ. Figure 7.5 shows a typical sample path that might be followed by the process over a small interval of time. Decision epochs are represented by vertical lines, and the states shown underneath these lines (e.g. $(\mathbf{x}, \Lambda)$, $((\mathbf{x}^{i+})^{j-}, M_j)$ etc.) are the system states at these decision epochs; the actions chosen (under the policy θ) when new customers arrive are also shown where appropriate. The vectors shown inside rectangles ($\mathbf{x}^{i+}$, $(\mathbf{x}^{i+})^{j-}$, etc.) represent the head counts at the various facilities *in between* decision epochs.

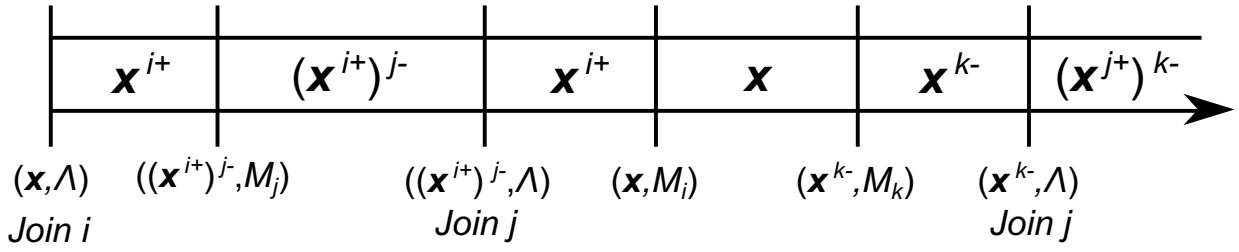


Figure 7.5: A typical state-time evolution of the process Ψ.

For example, the figure shows that the system is in state $(\mathbf{x}, \Lambda)$ at the beginning of the time period, at which point a new customer is sent to facility i. At the next decision epoch, the event M_j occurs (i.e. a service completion at facility j) and accordingly the next state is $((\mathbf{x}^{i+})^{j-}, M_j)$. However, in the physical process modelled by the MDP, the number of customers present at facility i in between these two decision epochs would obviously have been x_i^{i+} (i.e. $x_i + 1$, where x_i is the i^{th} component of the vector $\mathbf{x}$ associated with the initial state $(\mathbf{x}, \Lambda)$), since one imagines that the new customer who arrives at the first decision epoch joins facility i immediately, and the service completion at facility j does not occur until the second epoch is reached. Similarly, $(\mathbf{x}^{i+})^{j-}$ is a vector of head counts at the various facilities in between the second and third decision epochs (after

a service completion at j has occurred), $\mathbf{x}^{i+}$ represents the head counts in between the third and fourth epochs (after a new customer has joined facility j at the third epoch), etc.

In order to make the development in the next few pages as coherent as possible, it will be helpful to introduce some terminology. Let the sequence of states at the *decision epochs* of the process (in Figure 7.5, the sequence $((\mathbf{x}, \Lambda), ((\mathbf{x}^{i+})^{j-}, M_j), ...)$) be referred to as the sequence of *transitional states*, and let the sequence of vectors which represent the head counts at the various facilities *in between* the decision epochs (in Figure 7.5, the sequence $(\mathbf{x}^{i+}, (\mathbf{x}^{i+})^{j-}, ...)$) be referred to as the sequence of *physical states* of the process. The transitional states are elements of the 'augmented' state space $\mathcal{S}$ defined in (3.2.7), whereas the physical states are elements of the simplified state space S defined in (3.5.1). Note that, as in Chapter 3, the transitional state of the system exists continuously over time (as opposed to being defined only at decision epochs); so, for example, if the transitional state is $(\mathbf{x}, \Lambda) \in \mathcal{S}$ then this means that the event Λ occurred at the most recent decision epoch, at which point there were x_i customers at facility i ($i = 1, 2, ..., N$). Assuming that the stationary policy θ induces an ergodic, irreducible continuous-time Markov chain defined on some subset of $\mathcal{S}$, let $\{\pi_\theta^{(T)}((\mathbf{x}, \omega))\}_{(\mathbf{x},\omega)\in\mathcal{S}}$ denote the stationary distribution for the transitional states under θ, and let $\{\pi_\theta^{(P)}(\mathbf{x})\}_{\mathbf{x}\in S}$ denote the corresponding distribution for the physical states. By Lemma 3.5.1, $\{\pi_\theta^{(T)}((\mathbf{x}, \omega))\}_{(\mathbf{x},\omega)\in\mathcal{S}}$ and $\{\pi_\theta^{(P)}(\mathbf{x})\}_{\mathbf{x}\in S}$ are related as follows:

$$\pi_\theta^{(P)}(\mathbf{x}) = \sum_{(\mathbf{x},\omega)\in F_{\mathbf{x}}\cup G_{\mathbf{x}}} \pi_\theta^{(T)}((\mathbf{x}, \omega)) \qquad (\mathbf{x} \in S),$$

where the sets $F_{\mathbf{x}}$ and $G_{\mathbf{x}}$ (both subsets of $\mathcal{S}$) are as defined in (3.5.5); however, this relationship is somewhat incidental here. Next, let the *MRA state* of the system be defined as the physical state observed by the most recent customer to arrive in the system, *before* joining a facility (or balking). Here, MRA stands for 'Most Recent Arrival' and the MRA states are obviously elements of S. Figure 7.6 shows the same random sample path as Figure 7.5, with the MRA states (inside rectangles) taking the place of the physical states shown in the previous diagram.

Let $\{\pi_\theta^{(M)}(\mathbf{x})\}_{\mathbf{x}\in S}$ denote the stationary distribution for the MRA states under θ. Obviously, the MRA state of the system changes less frequently than the physical state. However, due to the well-known PASTA property (Poisson Arrivals See Time Averages) for queueing systems in which customers arrive via a Poisson process (see, for example, Wolff [202]), one can assert that the MRA states and the physical states share the same stationary distribution; that is, $\pi^{(M)}(\mathbf{x}) = \pi^{(P)}(\mathbf{x})$

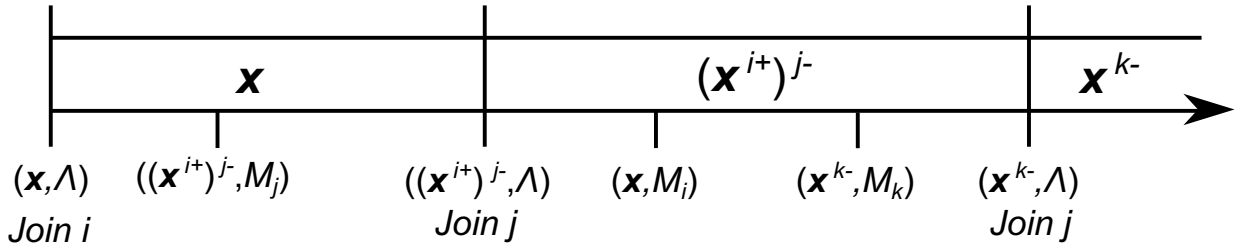


Figure 7.6: The progression of the 'MRA states' along a typical sample path of Ψ.

for all $\mathbf{x} \in S$. Hence, if one is able to formulate a CTMDP in which the state of the system is always the MRA state, then by finding the stationary distribution under the policy θ, one would also obtain the stationary distribution for the physical states under the same policy.

Consider the formulation of a CTMDP in which the system state at any given time is equal to the MRA state (and which otherwise is based on the assumptions listed in Section 3.1). In such a process, sojourn times will be uniformly distributed with mean $1/\lambda$. However, specifying the transition probabilities for such a process will not be an easy task. This is due to the fact that many service completions may take place in between two consecutive arrivals, and as such the number of states accessible from a given state $\mathbf{x} \in S$ via a single transition may be very large. Indeed, given any MRA state $\mathbf{x} \in S$ and any other state $\mathbf{y} \in S$ which satisfies the componentwise inequality $\mathbf{y} \leq \mathbf{x}$, there will be a non-zero probability of transferring from state $\mathbf{x}$ to $\mathbf{y}$ in a single transition, regardless of the policy being followed. It is easy to see that even if the transition probabilities for the process can be formulated exactly, the amount of time required to evaluate a given policy θ (or find an optimal policy) using a dynamic programming algorithm will be prohibitively large, due to the sheer number of stored values that must be 'looked up' on each iteration. On the other hand, an RL algorithm should not be encumbered by such problems, provided that it is designed in such a way that (in keeping with the general methodology of RL) it can simulate the progression of the MRA states without requiring knowledge of the underlying transition probabilities.

The next example demonstrates the validity of the preceding arguments in the case of an $M/M/1$ queue, by showing that the stationary distribution for the MRA states under a simple threshold policy does indeed match the corresponding distribution for the physical states.

Example 7.3.1. *($M/M/1$ balance equations for the MRA process)*

Let the demand rate $\lambda > 0$ be arbitrary, and suppose there is a single facility with a single service channel and service rate $\mu > 0$. The system is controlled by means of a threshold policy θ_T, whereby customers join the facility if and only if the number of customers x present when they arrive is smaller than the threshold $T \in \mathbb{N}$. In order to model the evolution of the 'physical state' of the system, one may construct a CTMC with infinitesimal generator matrix given by:

$$\mathcal{Q} := \begin{pmatrix} -\lambda & \lambda & 0 & 0 & \dots & 0 & 0 & 0 \\ \mu & -(\lambda+\mu) & \lambda & 0 & \dots & 0 & 0 & 0 \\ 0 & \mu & -(\lambda+\mu) & \lambda & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \mu & -(\lambda+\mu) & \lambda \\ 0 & 0 & 0 & 0 & \dots & 0 & \mu & -\mu \end{pmatrix} \tag{7.3.5}$$

From (7.3.5), it is immediate that the stationary probabilities $\pi^{(P)}_{\theta_T}(x)$ for states $x \leq T$ satisfy the familiar steady-state balance equations for a finite-capacity $M/M/1$ queue:

$$\begin{aligned} &\lambda\pi^{(P)}_{\theta_T}(0) = \mu\pi^{(P)}_{\theta_T}(1), \\ &(\lambda+\mu)\pi^{(P)}_{\theta_T}(x) = \lambda\pi^{(P)}_{\theta_T}(x-1) + \mu\pi^{(P)}_{\theta_T}(x+1) \qquad (1 \leq x \leq T-1), \\ &\lambda\pi^{(P)}_{\theta_T}(T-1) = \mu\pi^{(P)}_{\theta_T}(T). \end{aligned} \tag{7.3.6}$$

Equivalently, for $0 \leq x \leq T-1$:

$$\lambda\pi^{(P)}_{\theta_T}(x) = \mu\pi^{(P)}_{\theta_T}(x+1). \tag{7.3.7}$$

Now suppose that one wishes to model the evolution of the *MRA state* of the system under θ_T; that is, the physical state observed by the most recent customer to arrive. If the process is in some state $\mathbf{x}$ at an arbitrary point in time, where $1 \leq x \leq T$, then (by standard results) the probability that a service completion will occur before the next customer arrival is given by:

$$\int_0^\infty \lambda e^{-\lambda t} \int_0^t \mu e^{-\mu s} ds \; dt = \frac{\mu}{\lambda+\mu}.$$

For $k \leq x$, the probability that exactly k completions occur before the next arrival is:

$$\begin{cases} \dfrac{\lambda\mu^k}{(\lambda+\mu)^{k+1}}, & \text{if } k < x, \\ \dfrac{\mu^x}{(\lambda+\mu)^x}, & \text{if } k = x. \end{cases}$$

Hence, for any two states $x, y \in \{0, 1, ..., T\}$ (interpreted as MRA states, rather than physical states) the probability of transferring from state x to y in a single transition is:

$$\begin{cases} \dfrac{\lambda\mu^{z-y}}{(\lambda+\mu)^{z-y+1}}, & \text{if } y \geq 1, \\ \dfrac{\mu^z}{(\lambda+\mu)^z}, & \text{if } y = 0, \end{cases} \tag{7.3.8}$$

where $z = \min(x+1, T)$. Thus, noting that the infinitesimal transition rates for the MRA process are obtained by multiplying the probabilities in (7.3.8) by the uniform sojourn time parameter λ, one obtains the following generator matrix (which, like $\mathcal{Q}$, is of order $T+1$):

$$\hat{\mathcal{Q}} := \begin{pmatrix} \hat{q}_{00} & \dfrac{\lambda^2}{\lambda+\mu} & 0 & ... & 0 & 0 & 0 \\ \dfrac{\lambda\mu^2}{(\lambda+\mu)^2} & \hat{q}_{11} & \dfrac{\lambda^2}{\lambda+\mu} & ... & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \dfrac{\lambda\mu^{T-1}}{(\lambda+\mu)^{T-1}} & \dfrac{\lambda^2\mu^{T-2}}{(\lambda+\mu)^{T-1}} & \dfrac{\lambda^2\mu^{T-3}}{(\lambda+\mu)^{T-2}} & ... & \hat{q}_{T-2,T-2} & \dfrac{\lambda^2}{\lambda+\mu} & 0 \\ \dfrac{\lambda\mu^T}{(\lambda+\mu)^T} & \dfrac{\lambda^2\mu^{T-1}}{(\lambda+\mu)^T} & \dfrac{\lambda^2\mu^{T-2}}{(\lambda+\mu)^{T-1}} & ... & \dfrac{\lambda^2\mu^2}{(\lambda+\mu)^3} & \hat{q}_{T-1,T-1} & \dfrac{\lambda^2}{\lambda+\mu} \\ \dfrac{\lambda\mu^T}{(\lambda+\mu)^T} & \dfrac{\lambda^2\mu^{T-1}}{(\lambda+\mu)^T} & \dfrac{\lambda^2\mu^{T-2}}{(\lambda+\mu)^{T-1}} & ... & \dfrac{\lambda^2\mu^2}{(\lambda+\mu)^3} & \dfrac{\lambda^2\mu}{(\lambda+\mu)^2} & \hat{q}_{TT} \end{pmatrix}, \tag{7.3.9}$$

where, as usual, the diagonal elements $\hat{q}_{xx}$ (for $x = 0, 1, ..., T$) satisfy:

$$\hat{q}_{xx} = -\sum_{y \neq x} \hat{q}_{xy}.$$

In order to show that the stationary distribution $\{\pi^{(M)}_{\theta_T}(x)\}_{x\in\mathbb{N}_0}$ for the MRA process is identical to the corresponding distribution $\{\pi^{(P)}_{\theta_T}(x)\}_{x\in\mathbb{N}_0}$ for the physical process, it will be sufficient to show that the balance equations (7.3.7) hold with $\pi^{(P)}_{\theta_T}(x)$ replaced by $\pi^{(M)}_{\theta_T}(x)$ for each

$x \in \{0, 1, ..., T\}$. Recall that these balance equations are obtained by setting $\boldsymbol{\pi}_{\theta_T}^{(P)} \mathcal{Q} = \mathbf{0}$, where $\boldsymbol{\pi}_{\theta_T}^{(P)} = (\pi_{\theta_T}^{(P)}(0), \pi_{\theta_T}^{(P)}(1), ..., \pi_{\theta_T}^{(P)}(T))$. Consider the analogous set of balance equations given by $\boldsymbol{\pi}_{\theta_T}^{(M)} \hat{\mathcal{Q}} = \mathbf{0}$. By inspecting the rightmost column of $\hat{\mathcal{Q}}$ in (7.3.9), one finds:

$$\frac{\lambda^2}{\lambda+\mu} \pi_{\theta_T}^{(M)}(T-1) = \pi_{\theta_T}^{(M)}(T) \left[\frac{\lambda^2 \mu}{(\lambda+\mu)^2} + \frac{\lambda^2 \mu^2}{(\lambda+\mu)^3} + ... + \frac{\lambda^2 \mu^{T-1}}{(\lambda+\mu)^T} + \frac{\lambda \mu^T}{(\lambda+\mu)^T} \right]. \tag{7.3.10}$$

It can be shown using induction that for all integers $n \in \mathbb{N}$:

$$\frac{\lambda\mu}{\lambda+\mu} + \frac{\lambda\mu^2}{(\lambda+\mu)^2} ... + \frac{\lambda\mu^{n-1}}{(\lambda+\mu)^{n-1}} + \frac{\mu^n}{(\lambda+\mu)^{n-1}} = \mu. \tag{7.3.11}$$

Hence, after dividing by $\lambda/(\lambda+\mu)$ in equation (7.3.10) and using (7.3.11), one obtains:

$$\lambda \pi_{\theta_T}^{(M)}(T-1) = \mu \pi_{\theta_T}^{(M)}(T). \tag{7.3.12}$$

Proceeding in an inductive fashion, consider an arbitrary state $x \in \{1, 2, ..., T-1\}$ and assume that the following relationship has been verified for all $y \in \{x+1, x+2, ..., T\}$:

$$\lambda \pi_{\theta_T}^{(M)}(y-1) = \mu \pi_{\theta_T}^{(M)}(y). \tag{7.3.13}$$

Then, by inspecting the x^{th} column of $\hat{\mathcal{Q}}$ and applying (7.3.13) repeatedly, one can show:

$$\begin{aligned} &\frac{\lambda^2}{\lambda+\mu} \pi_{\theta_T}^{(M)}(x-1) + \pi_{\theta_T}^{(M)}(x) \left[\frac{\lambda^3 \mu}{(\lambda+\mu)^3} + \frac{\lambda^4 \mu}{(\lambda+\mu)^4} + ... + \frac{\lambda^{T-x+1} \mu}{(\lambda+\mu)^{T-x+1}} + \frac{\lambda^{T-x+2}}{(\lambda+\mu)^{T-x+1}} \right] \\ &= \pi_{\theta_T}^{(M)}(x) \left[\frac{\lambda^2}{\lambda+\mu} + \frac{\lambda^2 \mu^2}{(\lambda+\mu)^3} + ... + \frac{\lambda^2 \mu^x}{(\lambda+\mu)^{x+1}} + \frac{\lambda \mu^{x+1}}{(\lambda+\mu)^{x+1}} \right]. \end{aligned} \tag{7.3.14}$$

Upon dividing by $\lambda/(\lambda+\mu)$ and applying (7.3.11) to the right-hand side, (7.3.14) yields:

$$\begin{aligned} &\lambda \, \pi_{\theta_T}^{(M)}(x-1) + \pi_{\theta_T}^{(M)}(x) \left[\frac{\lambda^2 \mu}{(\lambda+\mu)^2} + \frac{\lambda^3 \mu}{(\lambda+\mu)^3} + ... + \frac{\lambda^{T-x} \mu}{(\lambda+\mu)^{T-x}} + \frac{\lambda^{T-x+1}}{(\lambda+\mu)^{T-x}} \right] \\ &= \pi_{\theta_T}^{(M)}(x) \left(\lambda + \mu - \frac{\lambda\mu}{\lambda+\mu} \right). \end{aligned} \tag{7.3.15}$$

Hence, in order to establish that $\lambda \pi_{\theta_T}^{(M)}(x-1) = \mu \pi_{\theta_T}^{(M)}(x)$ for all $x \in \{0, 1, ..., T-1\}$, it is sufficient to show that the following property holds for integers $n \geq 1$:

$$\frac{\lambda\mu}{\lambda+\mu} + \frac{\lambda^2 \mu}{(\lambda+\mu)^2} + ... + \frac{\lambda^{n-1} \mu}{(\lambda+\mu)^{n-1}} + \frac{\lambda^n}{(\lambda+\mu)^{n-1}} = \lambda. \tag{7.3.16}$$

In fact, (7.3.16) is the same identity as (7.3.11) (with λ and μ interchanged). This completes the proof that the following balance equations hold for $0 \leq x \leq T-1$:

$$\lambda \pi_{\theta_T}^{(M)}(x) = \mu \pi_{\theta_T}^{(M)}(x+1).$$

Therefore, as expected, $\pi_{\theta_T}^{(M)}(x) = \pi_{\theta_T}^{(P)}(x)$ for all $x \in \{0, 1, ..., T\}$. That is, the stationary distribution of the number of customers in the system seen by the most recent customer to arrive corresponds to the distribution of the physical state of the system itself. ⊠

Example 7.3.1 addresses $M/M/1$ queues, but the PASTA property holds in much greater generality (see [45, 126, 186, 202]). However, in a more general system with N facilities and multiple servers at each facility, the transition probabilities for the so-called 'MRA process' will obviously be somewhat more complicated than those in (7.3.8). Fortunately, as discussed earlier, an RL algorithm need not rely upon knowledge of these probabilities. Indeed, in order to simulate the evolution of the system under a particular decision-making scheme, all that is required is a sequence of inter-arrival times sampled from an exponential distribution with parameter λ, and another randomly-generated sequence which represents the service-time requirements of individual customers. Naturally, the latter sequence will also depend on the actions chosen when new customers arrive.

At this point, it will be useful to make some brief comments regarding the design of a simulation model. Suppose one wishes to simulate the evolution of a *single-facility* system (either an $M/M/1$ or an $M/M/c$ queueing system) with demand rate $\lambda > 0$ and service rate $\mu > 0$, under the assumption that every customer who arrives joins the queue (so that there is no admission control). Suppose the first customer arrives at time $t_1 > 0$ and let $(t_2, t_3, ...)$ denote the sequence of inter-arrival times, so that the n^{th} customer arrives at time $\sum_{j=1}^{n} t_j$. Also let $(s_1, s_2, ...)$ be the sequence of service-time requirements for the individual customers. For each $n \in \mathbb{N}$, t_n and s_n are exponentially distributed and can be sampled using the well-known method of *inverse transform sampling* (see, for example, [146], p. 668). Using this method, one would determine t_n and s_n as follows:

$$t_n = -\frac{1}{\lambda}\log(u_n), \qquad s_n = -\frac{1}{\mu}\log(v_n),$$

where u_n and v_n are both sampled from a uniform distribution on $[0, 1]$. The sequences $(t_1, t_2, ...)$ and $(s_1, s_2, ...)$ then completely determine the arrival time, the amount of time spent waiting in the queue, the time at which service begins and the service completion time for any individual customer who arrives in the system. Moreover, the elements of these sequences are mutually independent and, as such, there is no reason why they should not be sampled in the initial phase of a simulation algorithm, enabling the subsequent evaluation of performance measures (such as mean waiting times, etc.) based on observations of the resulting outcomes for individual customers.

In an RL algorithm applied to a general N-facility system, the situation is more complicated only in the sense that the service-time requirement of a particular customer will depend on which facility (if any) they have been sent to, which in turn will often depend on the consequences of actions chosen in the past; that is, there is a history-dependence for customers' service times. However, there is no reason why the service-time requirement for an individual customer should not be determined immediately (using inverse transform sampling) following selection of an action by the agent. Using this approach, one may determine the total amount of time spent in the system by a customer *immediately* after their destination is chosen, and then use this information as part of a Q-factor update. Essentially, by adopting this type of approach, one allows the agent to 'learn' from consequences which (in the physical process being modelled) would not actually occur until some time after the relevant action is chosen; the idea is to allow the agent to learn from a customer's overall time spent in the system following the choice of a particular action, as opposed to the immediate changes in head counts at the various facilities caused by that action.

The RL algorithm given below is called 'A-HYPOT R-learning', where the A stands for 'accelerated'. This algorithm differs from HYPOT R-learning (page 312) by associating decision epochs only with customer arrivals, and thereby taking longer 'steps' through the process. Q-factors are updated based on the total time spent in the system (and associated waiting costs incurred) by individual customers. As discussed above, this approach involves calculating the system departure times of customers *immediately* after they are sent to service facilities; however, it should be emphasised that this 'advance knowledge' of customers' exit times is *not* used as part of the decision-making process, since this would violate the underlying assumptions of the model. It is also worthwhile to note that, unlike the previous RL algorithms presented in this chapter, the A-HYPOT algorithm does *not* rely on a discretisation of the system, and instead simulates the system in continuous time. This will enable certain generalisations (of a somewhat heuristic nature) to be made easily when non-exponential distributions are considered later in this chapter.

A-HYPOT R-learning algorithm

1. Initialise the values $n = 1$, $g_0 = 0$, $t_0 = 0$ and an array of values $\hat{Q}(\mathbf{x}, a)$ with $\hat{Q}(\mathbf{x}, a) = 0$ for each state-action pair $(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$. Assume that the system is initialised in state $\mathbf{0}$, and accordingly set $\mathbf{x}_1 = \mathbf{0}$ (the state observed by the first customer to arrive). For each facility

$i \in \{1, 2, ..., N\}$ and service channel $j \in \{1, 2, ..., c_i\}$, initialise the value:

$$f(i, j) = 0.$$

During the simulation process, $f(i, j)$ will store the time at which the j^{th} channel at facility i becomes 'free' (which occurs when the channel is unoccupied and there are no customers waiting in the queue at facility i); initially, all channels are free. Also, use inverse transform sampling to determine $t_1 > 0$, the arrival time of the first customer.

2. Use inverse transform sampling to determine d_n, the time between the n^{th} and $(n + 1)^{th}$ customer arrivals, then let $t_{n+1} = t_n + d_n$. For each facility $i \in \{1, 2, ..., N\}$, determine the number of service completions that occur between times t_n and t_{n+1} using the known actions $a_1, a_2, ..., a_{n-1}$ and exit times $z_1, z_2, ..., z_{n-1}$ of previous customers. Denote these values by κ_i $(i = 1, 2, ..., N)$. Then determine the state $\mathbf{y}_n \in S$ using $\mathbf{x}_n$ as follows:

$$\mathbf{y}_n = \mathbf{x}_n - \sum_{i=1}^{N} \kappa_i \mathbf{e}_i,$$

where, as previously, $\mathbf{e}_i$ is an N-vector with i^{th} component equal to one and all other components equal to zero. One may interpret $\mathbf{y}_n$ as the state of the system at time t_{n+1} in the event that the n^{th} customer balks, as opposed to joining some facility.

3. For each action $a \in A_{\mathbf{x}_n}$, carry out the following sub-procedure:

   (a) If $a = i$ for some $i \in \{1, 2, ..., N\}$, then use inverse transform sampling to sample a random service-time requirement at facility i, and denote this by $s(i)$. Then calculate the (hypothetical) system departure time $Z(i)$ as follows:

   $$Z(i) = \min_{j \in \{1, 2, ..., c_i\}} f(i, j) + s(i).$$

   One may interpret $Z(i)$ as the departure time of the n^{th} customer in the event that they go to facility i. On the other hand, if $a = 0$ then set $Z(a) = t_n$.

   (b) Let the reward $R(a)$ be given by:

   $$R(a) = \begin{cases} \lambda\Big(\alpha_i - \beta_i\big(Z(i) - t_n\big)\Big), & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\}, \\ 0, & \text{if } a = 0. \end{cases}$$

(c) If $a = i$ for some $i \in \{1, 2, ..., N\}$ and $Z(i) \geq t_{n+1}$, update $\hat{Q}(\mathbf{x}_n, a)$ as follows:

$$\hat{Q}(\mathbf{x}_n, a) \leftarrow (1 - \delta_n(a))\hat{Q}(\mathbf{x}_n, a) + \delta_n(a)\left[R(a) - g_{n-1} + \max_{a' \in A_{\mathbf{y}_n^{i+}}} \hat{Q}(\mathbf{y}_n^{i+}, a')\right], \quad (7.3.17)$$

where $\delta_n(a) \in [0, 1]$ is a learning parameter. Note that $\mathbf{y}_n^{i+}$ is the state of the system at time t_{n+1} if facility i is chosen when the n^{th} customer arrives, and this customer remains in the system at time t_{n+1}. On the other hand, if $a = i$ for some $i \in \{1, 2, ..., N\}$ and $Z(i) < t_{n+1}$, *or* if $a = 0$, then apply (7.3.17) with $\mathbf{y}_n^{i+}$ replaced by $\mathbf{y}_n$.

4. Select an action $a_n \in A_{\mathbf{x}_n}$ according to a rule which allows for both exploration and exploitation (e.g. an ϵ-greedy rule). Then, let $r_n = R(a_n)$ and $z_n = Z(a_n)$, where $R(a_n)$ and $Z(a_n)$ are the values obtained for action $a_n \in A_{\mathbf{x}_n}$ in step 3. Also, define:

$$\mathbf{x}_{n+1} = \begin{cases} \mathbf{y}_n^{i+}, & \text{if } a_n = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } Z(a_n) \geq t_{n+1}, \\ \mathbf{y}_n, & \text{otherwise.} \end{cases}$$

5. If the action a_n chosen in step 4 was an *exploitative* action, calculate g_n as follows:

$$g_n := (1 - \zeta_{n-1})g_{n-1} + \zeta_{n-1}\left[r_n + \max_{a' \in A_{\mathbf{x}_{n+1}}} \hat{Q}(\mathbf{x}_{n+1}, a') - \max_{a \in A_{\mathbf{x}_n}} \hat{Q}(\mathbf{x}_n, a)\right],$$

where $\zeta_{n-1} \in [0, 1]$ is another learning parameter. Otherwise, set $g_n = g_{n-1}$.

6. If $a_n = i$ for some $i \in \{1, 2, ..., N\}$, then let k be a service channel satisfying:

$$k \in \underset{j \in \{1,2,...,c_i\}}{\arg\min} f(i, j).$$

Then update the 'channel free' time $f(i, k)$ as follows:

$$f(i, k) \leftarrow f(i, k) + s(i),$$

where $s(i)$ is the same service-time value obtained in step 3(a).

7. If $n = n_{\max}$, where $n_{\max}$ is a large integer, then output a stationary policy $\theta^{[R]}$ which, at each state $\mathbf{x} \in S$, chooses an action $\theta^{[R]}(\mathbf{x})$ maximising the expression $\hat{Q}(\mathbf{x}, a)$ over all actions $a \in A_{\mathbf{x}}$. Also output g_n as an estimate for the optimal long-run average reward g^*, then stop. Alternatively, if $n < n_{\max}$, increment n by 1 and return to step 2.

The A-HYPOT R-learning algorithm emulates the previous HYPOT algorithm by, on the n^{th} iteration, updating the Q-factor estimates $\hat{Q}_n(\mathbf{x}_n, a)$ for *all* actions $a \in A_{\mathbf{x}_n}$. However, its two main advantages over the HYPOT algorithm are its ability to learn from *future* service completion times and the fact that it 'steps through' an entire inter-arrival period on each iteration, which allows a greater amount of randomly-generated information to be used per iteration. The next example provides a comparison between the convergence rates (per iteration) of the three versions of R-learning considered in this chapter so far, in the case of a 4-facility system.

Example 7.3.2. ***(Convergence rates of HYPOT and A-HYPOT algorithms)***

This example reports the results of an experiment involving the same parameters as Example 7.2.3. Recall that the system under consideration consists of four service facilities, and the selfish state space $\tilde{S}$ comprises approximately 43,000 states. The purpose of the experiment was to compare the convergence rates (per iteration) of the R-learning, HYPOT R-learning and A-HYPOT R-learning algorithms. In the experiment, 25 million iterations were performed using each of the three algorithms in turn, with the same random number seed used for each algorithm. All three algorithms used an ϵ-greedy rule for selecting actions, with $\epsilon = 0.15$. The R-learning algorithm used learning parameters δ_n given by (7.2.14) for the $\hat{Q}$-factor updates (with $T = 10$), and the HYPOT and A-HYPOT algorithms both used an analogous definition for $\delta_n(a)$:

$$\delta_n(a) := \frac{T}{T + \nu_n(\mathbf{x}_n, a) - 1}, \tag{7.3.18}$$

with $T = 10$ used for each action $a \in A_{\mathbf{x}_n}$ on each iteration n. All three algorithms used secondary parameters given by $\zeta_n = 1/(n+1)$ for calculating the estimated average rewards g_n. Figure 7.7 shows, for each of the three RL algorithms under consideration, the progression of the estimated values g_n over 25 million iterations and also the values $\hat{g}_{n_j}$ obtained by performing Frozen Phase (FP) updates after every 10,000 iterations. As expected, the A-HYPOT algorithm appears to yield the fastest convergence rate per iteration. Not only do the values g_n converge quickly towards the optimal value g^* using A-HYPOT R-learning, but the values obtained from FP updates (which, as discussed previously, are a somewhat more relevant indicator of the algorithm's performance) also appear to be considerably more stable than the corresponding values obtained using the other two algorithms. The HYPOT algorithm, despite being somewhat inferior to A-HYPOT, nevertheless

exhibits a considerable improvement over the 'ordinary' R-learning algorithm.

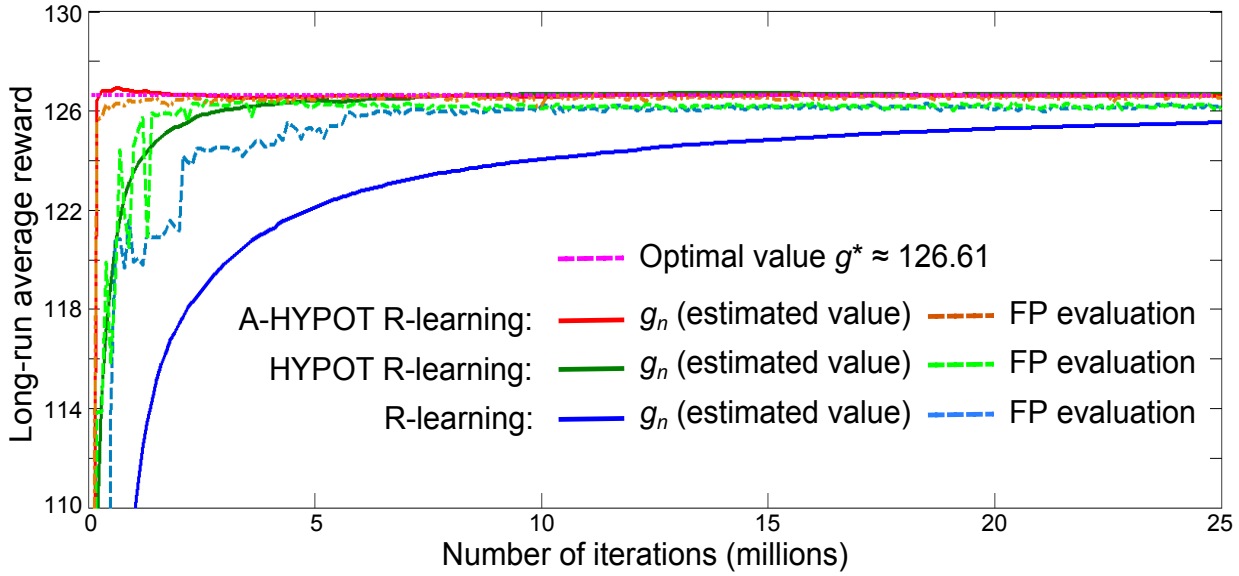


Figure 7.7: Comparison between the convergence rates (per iteration) for the R-learning, HYPOT R-learning and A-HYPOT R-learning algorithms, using the parameters from Example 7.2.3.

Of course, while comparisons with respect to convergence rates per iteration are interesting from a theoretical point of view, from a more practical perspective one would also be concerned about the running times required by these algorithms. For example, it is obvious that the HYPOT R-learning algorithm must perform more computations per iteration than 'regular' R-learning; moreover, this additional burden will increase with the dimensionality of the system, since it is caused by the HYPOT algorithm having to loop over all available actions a at whichever state the agent happens to be visiting. Figure 7.8 presents an additional comparison between the three algorithms with the time scale on the horizontal axis adjusted, so that it now measures (for each algorithm) the total number of Q-factor updates performed, rather than the number of iterations performed. Essentially, the only difference between Figures 7.7 and 7.8 is that the graphs for the HYPOT and A-HYPOT algorithms have been stretched horizontally by a factor of 5 (because they both have to update the Q-factors for 5 different actions on each iteration); thus, this presents a comparison which is much more favourable to the original version of R-learning introduced in Section 7.2.

The A-HYPOT R-learning algorithm, like the previous RL algorithms discussed in this chapter, relies upon the assumption that a unique value $\hat{Q}(\mathbf{x}, a)$ can be stored for each state-action pair

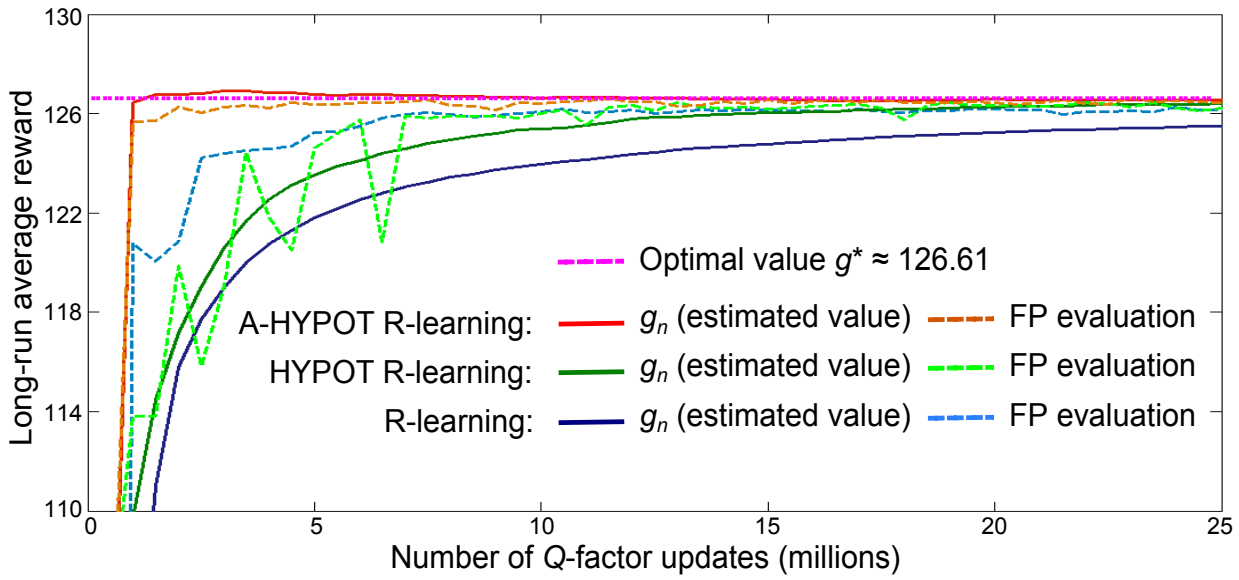


Figure 7.8: Comparison between the convergence rates (per Q-factor update) for the R-learning, HYPOT R-learning and A-HYPOT R-learning algorithms, using the parameters from Example 7.2.3.

$(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$ while the algorithm is running. This assumption may be somewhat unwieldy in systems where millions or billions of pairs are likely to be sampled. The next section will address the topic of *value function approximation*, which is useful in problems where the state space is simply too vast for the storage of unique values $\hat{Q}(\mathbf{x}, a)$ to be a viable approach.

7.4 Value function approximation

The examples given in the previous sections have shown that RL algorithms such as R-learning are capable of finding near-optimal policies for queueing systems with tens of thousands of possible states (after truncation of the infinite state space), without requiring more than a few minutes of running time on a fast computer. However, systems with truly vast state spaces have yet to be considered. A further aim in this chapter is to investigate the performance of RL algorithms in systems of higher dimensionality. As illustrated previously, RL algorithms are able to cope with vast state spaces to a certain extent by 'zoning in' on the most frequently-visited states under a near-optimal policy, but certain problems with their implementation still remain.

All of the RL algorithms considered in this chapter so far have relied upon the assumption that a

unique value $\hat{Q}(\mathbf{x}, a)$ can be stored (and updated when necessary) for each state-action pair $(\mathbf{x}, a)$ 'visited' by an agent. Furthermore, if a rule for the learning parameters δ_n such as (7.2.14) is used, then another value $\nu(\mathbf{x}, a)$ must be stored to track the number of times that the pair $(\mathbf{x}, a)$ has been updated since the beginning of the process. Obviously, a computer program can be written in such a way that the values $\hat{Q}(\mathbf{x}, a)$ and $\nu(\mathbf{x}, a)$ are not initialised until the first time the pair $(\mathbf{x}, a)$ is visited, so that only state-action pairs that are actually sampled during the finite running time of the algorithm impose upon the memory requirements of the system; however, the fact remains that in a system of high dimensionality, one must allow for the possibility of a very large number of $\hat{Q}(\mathbf{x}, a)$ and $\nu(\mathbf{x}, a)$ values requiring storage. Unfortunately, a computer's memory may simply be unable to cope with the size of the arrays that would be required for this purpose.

Even if one assumes that it is possible to store all of the $\hat{Q}(\mathbf{x}, a)$ and $\nu(\mathbf{x}, a)$ values that would be needed for an RL algorithm to obtain a reasonable approximation of an optimal policy after some finite amount of time, a further (arguably more superficial) problem is that the algorithms described so far have relied upon what is known as a *look-up table representation* for a stationary policy (see [15], p. 4). This means that when the algorithm reaches completion, an action $\theta^{[R]}(\mathbf{x}) \in \arg\max_{a \in A_{\mathbf{x}}} \hat{Q}(\mathbf{x}, a)$ is determined for each state $\mathbf{x} \in S$, where $\theta^{[R]}$ is the policy returned by the algorithm. Then, in order to implement the policy $\theta^{[R]}$ in a practical situation (or a simulation), one must continually 'look up' the decisions $\theta^{[R]}(\mathbf{x})$ associated with the various states $\mathbf{x} \in S$ during the evolution of the process. In other words, the policy $\theta^{[R]}$ lacks the ability of the index-based heuristic policies discussed in Chapter 6 to prescribe a decision to be chosen at a particular state $\mathbf{x} \in S$ according to a relatively simple function of the head counts at state $\mathbf{x}$; instead, like the policy given by a DP algorithm, it must refer to a vast array of decisions which may not follow any easily recognisable structure. In this section, it will be shown that an RL algorithm need not be encumbered by having to use a look-up table representation for its output policy.

Various strategies have been proposed in the literature for avoiding the need to store unique values for each state-action pair during the evolution of an RL algorithm; some of these will be discussed in this section. As one would expect, the storage of less information brings with it some disadvantages. To put things simply, one must be prepared to accept the likelihood of a reduction in the quality of solutions found when less information is stored; in other words, the final policy obtained by the algorithm may be slightly further from optimality when fewer $\hat{Q}(\mathbf{x}, a)$ values are stored explicitly.

However, this will be a necessary price to pay in systems where the state space is so vast that algorithms such as R-learning (in their conventional form) are no longer practical.

Essentially, the techniques to be discussed in this chapter are based on the premise that the Q-factors $Q(\mathbf{x}, a)$ which characterise an optimal policy follow *roughly* some kind of structure or pattern. The next example illustrates this idea, using the example of an $M/M/1$ queue.

Example 7.4.1. ***(Value function approximation in an*** $M/M/1$ ***queue)***

Consider an $M/M/1$ queue with system parameters given as follows:

$$\lambda = 2, \qquad \mu = 3, \qquad \beta = 5, \qquad \alpha = 35.$$

With these parameter values, a threshold policy θ_T with threshold $T = 8$ is average reward optimal. Let it be assumed that the system is formulated as an MDP, with a formulation given by (3.5.1)-(3.5.4). From a purely superficial point of view, an advantage of using the reward formulation in (3.5.3) is that the resulting optimal value function $h(x)$ has the property that $h(x+1) \geq h(x)$ if and only if joining is chosen at state $x \in \mathbb{N}_0$ under the optimal policy θ_T. Thus, after plotting the values $h(x)$ on a graph, one can easily identify the threshold T as the value of x that maximises $h(x)$ (or, in the event of a tie, the largest of these x values). Figure 7.9 shows the values $Q(x, 0)$ and $Q(x, 1)$ (computed using dynamic programming) plotted for $x \in \{0, 1, ..., 20\}$.

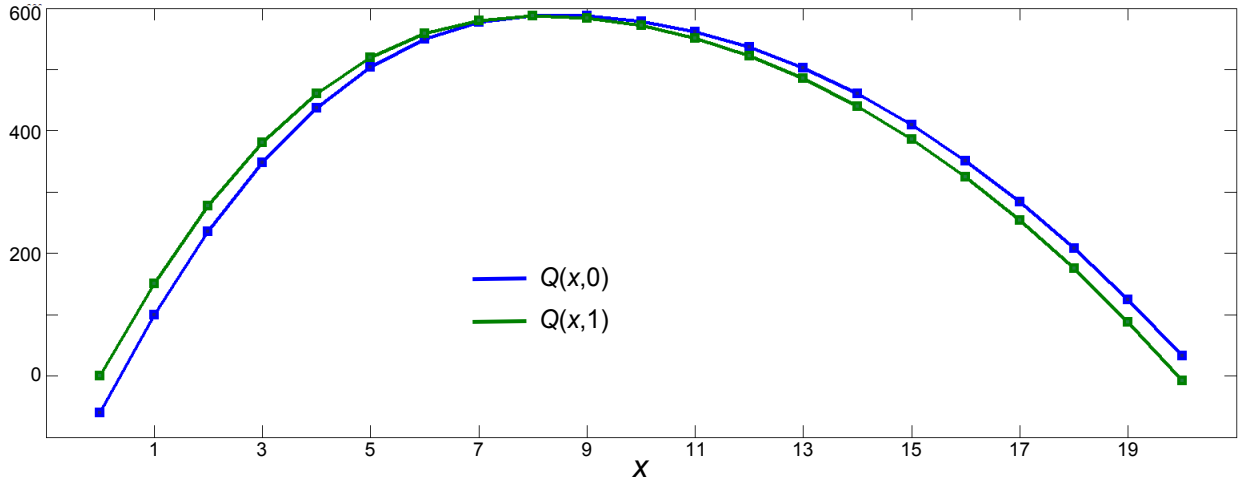


Figure 7.9: The values of $Q(x, 0)$ and $Q(x, 1)$ for $x \in \{0, 1, ..., 20\}$ in Example 7.4.1.

Note that, by definition, $h(x) = \max\big(Q(x, 0), Q(x, 1)\big)$ for each $x \in \mathbb{N}_0$, and hence it can be seen

from Figure 7.9 that the optimal value function $h(x)$ also follows a shape similar to those of $Q(x, 0)$ and $Q(x, 1)$. Examining the general trends depicted in the figure, it is reasonable to propose that a polynomial function of x should be suitable for approximating the $Q(x, 0)$ and $Q(x, 1)$ values; indeed, the curves in the figure look somewhat similar to quadratic curves. Figure 7.10, in which the curves in Figure 7.9 are plotted separately, shows the results of fitting polynomial models of orders 2, 3 and 4 to the $Q(x, 0)$ and $Q(x, 1)$ values using multivariate linear regression.

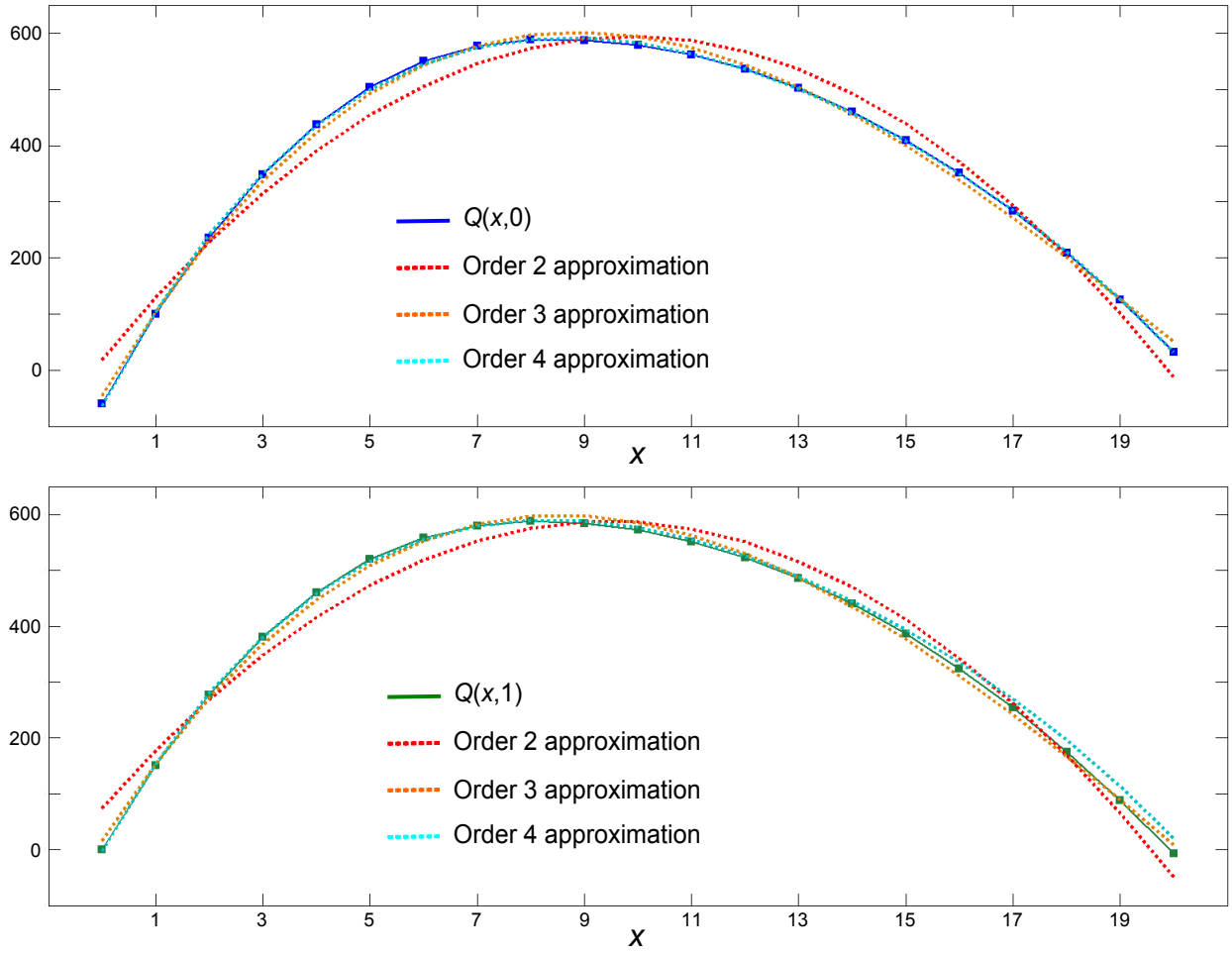


Figure 7.10: Polynomial approximations of orders 2, 3 and 4 for the functions $Q(x, 0)$ and $Q(x, 1)$.

The order of each polynomial approximating function in Figure 7.10 corresponds to the number of predictor variables used in the associated regression model. As one would expect, polynomials of higher degree tend to yield more accurate approximations. The figure shows that the quadratic (order 2) polynomials are not entirely satisfactory for representing the shape of the Q functions,

since they tend to misrepresent the position of the apex. However, when the order is increased to 4, the approximations are almost exact; indeed, the light blue dotted lines in the figure are almost indistinguishable from the $Q(x,0)$ and $Q(x,1)$ curves. The values of R^2 (the regression coefficient) associated with the polynomial models of order 2, 3 and 4 used for $Q(x,0)$ are 0.970, 0.997 and 0.999 respectively, and the corresponding R^2 values for $Q(x,1)$ are almost identical.

The implication of this example is that, in the case of a simple $M/M/1$ queue, the Q-factors which characterise an optimal policy follow a pattern which can be traced approximately by polynomial functions of the system state. Assuming that the same principle holds in higher dimensions, this suggests the possibility of employing well-established mathematical techniques such as interpolation, curve-fitting, regression and other related methods to estimate Q-factors even when the amount of information available is only sparse. However, given that an RL algorithm begins with *no* available information about the Q-factors which define an optimal policy, the exact procedure that one might use to acquire even a *small* amount of information (without resorting to explicit storage of a large number of Q-factor estimates in the system memory) remains unclear at this stage. ⊠

As discussed previously, the problem to be addressed in this section concerns the approximation of average reward optimal policies in systems with an extremely vast state space. However, it will always be assumed that a finite truncation of the state space can be applied, and this is justified by the results in Chapter 4. From this stage onwards, it will be assumed that the use of any algorithm which depends upon a unique value (of any kind) being associated with every state in the finite state space $\tilde{S}$ (or, similarly, every state-action pair in $\tilde{S} \times A_{\mathbf{x}}$) is prohibited. However, it is clearly reasonable to suppose that values can be stored for *some* states (or pairs), since a computer program will always afford the user *some* memory space in which to store data arrays.

Accordingly, let it be assumed that there exists a finite subset $R \subset \tilde{S}$ of states (or, in the case of state-action pairs, $R \subset \tilde{S} \times A_{\mathbf{x}}$) for which some quantity of interest can be stored (and updated) explicitly during the course of an RL algorithm. The set R may not necessarily be 'static'; indeed, the constituent elements of R may change over the course of the algorithm (so that different states or state-action pairs can be shuffled in and out of the set), but it should be assumed that there is some upper bound on the size of $|R|$ which, in general, will be much smaller than $|\tilde{S}|$.

Three broadly-defined methods for overcoming the difficulties posed by a vast state space will be

discussed here, although ultimately this section will focus on only one of these methods. The three techniques to be discussed are: state aggregation, interpolation, and function fitting.

State aggregation methods

The first method to be discussed here, known as *state aggregation*, is arguably also the crudest from a mathematical point of view. Assuming that it is not feasible to estimate Q-factors for all state-action pairs individually, one might instead allow each Q-factor to represent multiple pairs. To be more exact, suppose the finite state space $\tilde{S}$ is partitioned into a set $\{\tilde{S}_1, \tilde{S}_2, ..., \tilde{S}_K\}$ where K is reasonably small, so that it is possible to store a set of values in an array of size $K(N+1)$ (where $N+1$ is the maximum size of any action set). The set $\{\tilde{S}_1, \tilde{S}_2, ..., \tilde{S}_K\}$ is referred to as the 'aggregated' state space. On each iteration of an RL algorithm, the state-action pair visited is given (for aggregation purposes) by $(\tilde{S}_i, a)$ for some $i \in \{1, 2, ..., K\}$ and $a \in A_{\tilde{S}_i}$. After the simulation step of the iteration has been performed, a value $\hat{Q}(\tilde{S}_i, a)$ may be updated (using an update rule similar to those discussed in earlier sections) and stored in the system memory.

It should be emphasised that, when using aggregation methods, one still allows the RL algorithm to keep track of the *exact* state of the system (an element of $\tilde{S}$, rather than $\{\tilde{S}_1, \tilde{S}_2, ..., \tilde{S}_K\}$) at all times. The random transitions depend upon the *exact* states in the same way as normal, so that the simulation of the process is still entirely accurate; the aggregation of the state space is used purely as a means of reducing the number of Q-factors that require storage.

The drawbacks of this approach are obvious. In most applications, the state transitions will no longer possess the Markov property when the aggregate representation is used, since the transition probabilities from any aggregate state $\tilde{S}_i$ will depend upon the individual state $\mathbf{x} \in \tilde{S}_i$ that the process is in. As a result, RL algorithms lose much of their theoretical foundation when they rely upon state space aggregation. Indeed, it cannot be assumed that the values $Q(\tilde{S}_i, a)$ which the algorithm is trying to estimate will satisfy a set of equations of the form (7.1.4), which raises the question of whether the Q-factor update rules seen in earlier sections (which are directly based on these optimality equations) remain appropriate to use. Given the inherent ugliness of the method, it would appear that the use of state space aggregation can only be justified if one can show empirically that it produces good results. Gosavi [62] (p. 215) describes aggregation as a "robust" approach, but its effectiveness seems to depend upon the context in which it is used.

The question of exactly *how* states should be 'lumped together' may not be easy to resolve. Ideally, it is desirable for states with similar characteristics to be grouped together (see [62], p. 260). In the case of the queueing systems considered in this thesis, one might derive some measure of the 'overall congestion level' for each state $\mathbf{x} \in \tilde{S}$ and then construct various categories according to these congestion levels. Alternatively, taking into account the general principle that any policy which maximises the expected long-run average reward is likely to cause the process to spend a significant proportion of its time in states which earn 'high' rewards, it may be logical to categorise states according to the expected rewards earned; in this case, however, matters are slightly complicated by the fact that rewards are also likely to be action-dependent. The MDP formulation used in Example 7.4.1 avoids this difficulty, since the rewards $r(x)$ happen to depend only on the state, and (to make things even simpler) the state is one-dimensional. In the slightly more general case of an $M/M/c$ queue, a partition scheme based on the rewards $r(x)$ defined in (5.1.3) would be quite different from a scheme based simply on the congestion level x, since it would involve states 'close to c' being separated from states 'distant from c'. Moreover, using $\tilde{T}$ to denote the selfishly optimal threshold, one has $r(\tilde{T}) \approx r(0) = 0$ and therefore a partition scheme based on rewards would group together states at opposite extremities of the selfishly optimal state space $\{0, 1, ..., \tilde{T}\}$.

For further discussion of state aggregation methods, including some examples, refer to Gosavi [62] (p. 260). Aggregation methods are also discussed (albeit in the context of value iteration, rather than reinforcement learning) by Bertsekas and Tsitsiklis [15] (p. 215) (see also [185]).

Interpolation methods

Interpolation methods are based upon the premise that it is possible to store *some* Q-factors in the system memory but not *all*, as discussed earlier. Let $R \subset \tilde{S}$ be a set of *exemplar* states, at which the Q-factor estimates $\hat{Q}(\mathbf{x}, a)$ are to be stored and updated explicitly during the finite running time of the algorithm. The set R may be pre-determined and fixed, so that the same exemplar states are used throughout the process, or (in the case of a more sophisticated algorithm) it may be adaptable, so that system states are eligible to be 'dropped' from the set of exemplars and replaced by other states at various stages of the procedure. When one of the exemplar states $\mathbf{x} \in R$ is visited by the agent during the course of the RL algorithm, the value $\hat{Q}(\mathbf{x}, a)$ corresponding to the action chosen $a \in A_{\mathbf{x}}$ is updated using a rule similar to those given in earlier sections; however, a difficulty

arises in that update rules such as (7.2.11) depend on Q-factor estimates being available for states which may not be included in the set of exemplar states. As such, one must use *interpolation* to estimate Q-factors for non-exemplar states using the values stored for exemplar states.

Of course, the values $\hat{Q}(\mathbf{x}, a)$ stored for exemplar states $\mathbf{x} \in R$ are themselves only approximations of the *true* values $Q(\mathbf{x}, a)$, so interpolation effectively adds another layer of imprecision to proceedings; however, this is merely in keeping with the general theme of this section, which involves compromises being made between accuracy and practicability. In general, the exemplar states should be reasonably spread-apart in order to ensure sufficient coverage of the state space; on the other hand, there is little point in storing values explicitly for states located in regions of the state space which are unlikely ever to be visited under an optimal policy. Thus, there is a certain balancing act to be performed in deciding which regions of the state space should be 'covered' (in the sense of being 'near' an exemplar state) by the set of exemplars R. Tadepalli and Ok [177] discuss a method in which non-exemplar states $\mathbf{x} \in \tilde{S} \setminus R$ are added to the set R if it is found that their values $h(\mathbf{x})$ cannot be estimated within a given tolerance using the values stored for the other exemplar states. When a new state $\mathbf{x}$ is added to R, a check is performed to see whether any states which lie within a certain vicinity of $\mathbf{x}$ can safely be deleted from R due to their Q-factors being estimable within the specified tolerance following the addition of the new state to R.

It is important to emphasise the advantage of allowing the set of exemplars R to be modified during the running of the algorithm, since in general an RL algorithm will begin without any knowledge of the characteristics of an optimal policy. As such, exemplar states should be re-positioned gradually as the algorithm learns which regions of the state space are visited frequently under a strong-performing policy (and thereby require a relatively high concentration of exemplars).

The exact method of interpolation used to estimate Q-factors for non-exemplar states is likely to depend upon the complexity of the state space $\tilde{S}$. In the simple case where $\tilde{S}$ is one-dimensional (as in Example 7.4.1), one may simply use linear interpolation, so that the value $Q(z, 1)$ for any state z lying between two exemplar states x and y (with $x < y$) is approximated by:

$$\hat{Q}(z, 1) = \hat{Q}(x, 1) + \frac{z - x}{y - x} \left(\hat{Q}(y, 1) - \hat{Q}(x, 1) \right).$$

In the case of a multi-dimensional state space, two different approaches will be mentioned here, both of which are suggested by Gosavi [62] (p. 265) and are covered in considerable more detail

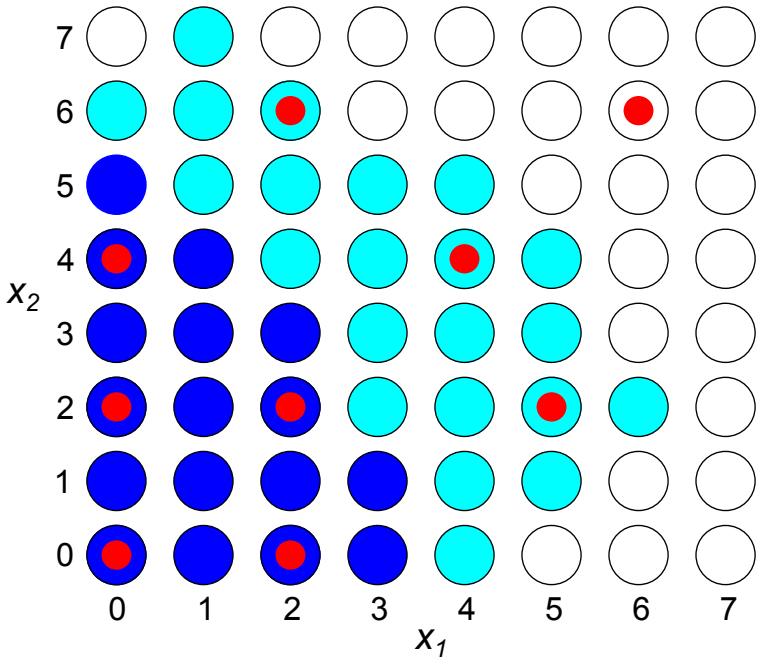


Figure 7.11: A conceptual example of how exemplar states might be distributed in a system with a two-dimensional state space according to the frequency with which individual states are visited under an optimal policy. States (x_1, x_2) are represented as circles and categorised as 'often visited, 'seldom visited' or 'very rarely visited' according to whether they are coloured dark blue, light blue or white respectively. States marked by red centres are 'exemplar states'. Frequently-visited states are more likely to be exemplars.

(in the broader context of statistical learning) by Hastie et. al. [77] (see also [39]). The first of these methods is known as the *k-nearest-neighbours* approach. Suppose it is necessary to obtain an estimate for some Q-factor $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$, where $\mathbf{x}$ is not an exemplar state. One may inspect the set of exemplars R in order to determine the k elements of R (where k is some pre-determined integer) which are closest to $\mathbf{x}$ with respect to some distance metric. An obvious choice for the distance metric is the Euclidean distance, given for two arbitrary states $\mathbf{x}, \mathbf{y} \in \tilde{S}$ by:

$$d(\mathbf{x}, \mathbf{y}) := \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}. \tag{7.4.1}$$

After identifying the k 'nearest neighbours' of $\mathbf{x}$ within R, the value $Q(\mathbf{x}, a)$ can be estimated by simply averaging the values $\hat{Q}(\mathbf{y}, a)$ for these states, or (at greater computational expense) using regression to predict $Q(\mathbf{x}, a)$ by fitting a model to the $\hat{Q}(\mathbf{y}, a)$ values for the k nearest neighbours. Unfortunately, even if the simpler averaging method is used, the process of *finding* the k nearest neighbours of $\mathbf{x}$ in the first place may require a significant amount of computation time. For this

reason, k-nearest-neighbours is not considered a feasible approach in this section.

An alternative to the k-nearest-neighbours approach is to use a *kernel-based method*. Essentially, this involves estimating a value $\hat{Q}(\mathbf{x}, a)$ for some non-exemplar state $\mathbf{x} \in \tilde{S} \setminus R$ and action $a \in A_{\mathbf{x}}$ by taking a weighted average of the values $\hat{Q}(\mathbf{y}, a)$ for the exemplar states $\mathbf{y} \in R$, where the weights are determined by the distances of the exemplar states from $\mathbf{x}$. Specifically, one would estimate the value $Q(\mathbf{x}, a)$ for some state-action pair $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ with $\mathbf{x} \notin R$ as follows:

$$\hat{Q}(\mathbf{x}, a) = \frac{\sum_{\mathbf{y} \in R} w(\mathbf{x}, \mathbf{y})\, \hat{Q}(\mathbf{y}, a)}{\sum_{\mathbf{y} \in R} w(\mathbf{x}, \mathbf{y})}, \tag{7.4.2}$$

where $w(\mathbf{x}, \mathbf{y})$ is a weight which depends in some way on the Euclidean distance $d(\mathbf{x}, \mathbf{y})$ defined in (7.4.1). The weight $w(\mathbf{x}, \mathbf{y})$ may be defined in such a way that it equals zero if $d(\mathbf{x}, \mathbf{y})$ exceeds a certain value, so that the weights 'die off' with increasing distance from $\mathbf{x}$. For more details on how the weights may be defined, see [62] (p. 267). Kernel-based methods avoid the need to locate the 'nearest neighbours' of a particular state in order to estimate a required Q-factor; obviously, however, computing a weighted average of estimated Q-factors for *all* states $\mathbf{y} \in R$ will itself be a time-consuming process if the set of exemplars R is of reasonable size. For this reason, kernel-based methods must also be considered impractical in systems of high dimensionality.

With a little thought, one begins to realise that interpolation methods suffer from a drawback which also applies to state aggregation methods, since they rely upon being able to store a limited number of Q-factors explicitly. In the case of aggregation, each Q-factor must be representative of a (possibly large) collection of states which, ideally, should be relatively homogeneous with respect to some particular characteristic. In the case of interpolation, the exemplar states should ideally be positioned in such a way that every state which might be visited by an optimal policy is within a certain proximity of an exemplar. In either case, the assumption that the desired conditions can be achieved becomes increasingly unrealistic as the complexity of the state space escalates. What is really needed is a radically different method which, instead of attempting to learn Q-factors individually, will instead attempt to learn some kind of formula or index for *approximating* the Q-factors. This is the idea behind the *function fitting* methods to be discussed next.

Function fitting methods

Function fitting methods are arguably the most flexible and aesthetically appealing of the approx-

imation methods discussed in this section. In Example 7.4.1, it was shown (in the case of an $M/M/1$ queue) that the values $Q(x,0)$ and $Q(x,1)$ could be approximated quite accurately using polynomial functions of the state x. Function fitting methods are based on the assumption that the same principle holds in much greater generality; that is, the Q-factors $Q(\mathbf{x},a)$ which characterise an optimal policy for a system with a multi-dimensional state space may be approximated by functions of the vector $\mathbf{x}$. A natural generalisation of the technique used in Example 7.4.1 would involve estimating $Q(\mathbf{x},a)$ for an arbitrary pair $(\mathbf{x},a) \in \tilde{S} \times A_{\mathbf{x}}$ as follows:

$$\hat{Q}(\mathbf{x},a) = B(a) + \sum_{k=1}^{K} \sum_{i=1}^{N} C_i^{(k)}(a)\, x_i^k, \tag{7.4.3}$$

where K is a pre-determined integer which represents the order of the approximation, N is (as usual) the dimensionality of the state space, the values $B(a)$ and $C_i^{(k)}(a)$ are weights to be determined by an RL algorithm, and x_i^k denotes the i^{th} component of $\mathbf{x}$ raised to the power k. For example, by setting $K = 2$ in (7.4.3) one obtains a quadratic function of N variables:

$$\hat{Q}(\mathbf{x},a) = B(a) + \sum_{i=1}^{N} C_i^{(1)}(a)\, x_i + \sum_{i=1}^{N} C_i^{(2)}(a)\, x_i^2. \tag{7.4.4}$$

It should be noted that the right-hand side in (7.4.4) is missing cross-product terms of the form $x_i x_j$ (where $i \neq j$); in general, cross-product terms should be included in a quadratic expression of N variables. However, since the number of cross-product terms increases steeply with N (and with K), it will be desirable throughout this section to simplify matters by restricting attention to approximating functions of the form (7.4.3). The total number of weights in (7.4.3) is $KN + 1$, which increases only linearly with N; this will be an important advantage when the RL algorithms to be discussed in this section are applied to systems with very large state spaces.

RL algorithms which use function fitting *do not need to store any Q-factors explicitly*; they simply aim to learn values of the weights in (7.4.3) which will accurately approximate the function $Q(\mathbf{x},a)$ for a given action a. Note that a unique set of weights is required for each action a that may be chosen during the course of the process; in general, this can cause a problem in MDPs where the action sets $A_{\mathbf{x}}$ vary considerably among the states $\mathbf{x} \in \tilde{S}$. Obviously, however, this does not cause any problems in the queueing systems considered in this thesis, since the action sets $A_{\mathbf{x}}$ depend on the states $\mathbf{x} \in \tilde{S}$ only to the extent that certain actions are prohibited at states which are on the boundary of the finite state space $\tilde{S}$. An RL algorithm will need to store and update values for all

of the weights associated with all possible actions; as such, the total number of weights requiring storage will be $(N+1)(KN+1)$, where (in keeping with the context of the problems considered in this thesis) $N+1$ is the cardinality of the set $\bigcup_{\mathbf{x}\in\tilde{S}} A_{\mathbf{x}}$, i.e. the total number of distinct actions available. It is intended that, provided the order of approximation K is reasonably small and the number of facilities N is not excessively large, a function-fitting RL algorithm should be applicable to a system with billions or trillions of states without any major practical difficulties.

The term *architecture* is used to describe the general parametric class of functions in which one attempts to find suitable approximators for the Q-factors $Q(\mathbf{x}, a)$ (see, for example, [15], p. 5). Ideally, an approximation architecture should be 'rich' enough to allow the $Q(\mathbf{x}, a)$ values to be approximated with an acceptable level of accuracy; this means that a sufficiently large number of free parameters (equivalently, weights) should be used. However, as discussed by Bertsekas and Tsitsiklis [15] (p. 60), the computational complexity of the algorithm increases with the number of parameters, so (as with most RL algorithms) there is a trade-off to be made between the ease with which the algorithm can be applied in practice and its fitness for purpose. In this section, the only architecture considered will be the class of polynomial functions described by (7.4.3).

It may be seen as a disadvantage of function fitting RL algorithms that a suitable approximation architecture must be determined *a priori*, without (in general) knowledge of the shape or characteristics of the function(s) to be approximated. If possible, it is appropriate to perform some kind of validation test to ensure that the architecture being used is suitable for the purpose at hand. In the case of the queueing systems under consideration, an obvious possibility for validating the architecture in (7.4.3) is to use dynamic programming to calculate the Q-factors exactly in a moderately-sized system, and then determine values of the weights $B(a)$ and $C_i^{(k)}(a)$ by applying multivariate linear regression to the resulting data. The quality of the approximation (for each action a) would then be measured by the regression coefficient R^2. Arguably, there are some limitations to this approach, including the fact that the model fitted by regression will implicitly assume that all state-action pairs are equally 'important'; this will be discussed later.

In this section, the particular design of the RL algorithm to be considered will make the validation method described above slightly more difficult to apply. This is because the algorithm will essentially be an extension of the A-HYPOT R-learning algorithm presented on page 323 which will use

function fitting methods as opposed to the storage of individual Q-factor estimates.

It was shown in Section 7.3 that A-HYPOT R-learning is capable of improving upon the performances of R-learning and HYPOT R-learning in terms of the number of iterations required to obtain a near-optimal policy; this is one reason for it being preferred to the other two algorithms as the basis for a function-fitting algorithm. However, as discussed in Section 7.3, A-HYPOT R-learning relies implicitly upon a re-formulated MDP in which the state of the system at any given time is the vector of head counts at the various facilities observed by the most recent customer to arrive. This re-formulated MDP has been referred to previously as the *MRA process*. The optimal Q-factors for the MRA process will not be the same as those which would be obtained using the MDP formulation given by (3.5.1)-(3.5.4) (which is based on the *physical* state of the system), despite the fact that both processes will have the same class of optimal stationary policies. Unfortunately, it is not practical to use dynamic programming to calculate the Q-factors associated with an optimal policy in the MRA process, due to the complexity of the associated transition probabilities (as illustrated, in the $M/M/1$ case, by Example 7.3.1); as such, there is no convenient means available of obtaining the exact Q-factors, even in a moderate-sized system.

One way around this difficulty is to *assume* (with the backing of the empirical evidence in Example 7.3.2) that the Q-factor estimates output by the A-HYPOT R-learning algorithm are reasonably close approximators of the true Q-factors, and apply multivariate regression to these estimated Q-factors. The next example shows the results obtained using this validation method.

Example 7.4.2. ***(Validation of the quadratic architecture for one system)***

This example uses the same parameters as Example 7.2.3. To summarise, there are 4 service facilities and the selfish state space $\tilde{S}$ consists of approximately 43,000 states. The optimal value of the expected long-run average reward is $g^* \approx 126.61$. As discussed previously, one may use multivariate linear regression to fit the quadratic model (7.4.4) to the estimated Q-factors found using the A-HYPOT algorithm as a means of validating the quadratic architecture.

However, there are some difficulties associated with this approach. The A-HYPOT algorithm, like the other RL algorithms considered in this chapter, operates in an *asynchronous* manner; as such, assuming that the run-time is finite, it will generally obtain much more accurate estimates of the

Q-factors for state-action pairs that are visited often than for pairs which are not. Realistically, this means that in a system with tens of thousands of states, only a very small proportion of the $Q(\mathbf{x}, a)$ values will be estimated with any reasonable accuracy. As a result, applying least-squares linear regression to a dataset consisting of estimated Q-factors for *all* state-action pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ will be meaningless, since most of these estimated values will be wildly inaccurate.

For the purposes of this validation test, a reasonable way of overcoming this problem is to apply the A-HYPOT algorithm *synchronously*; that is, one may update the values for *all* state-action pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ once each on every iteration, as opposed to following the simulated path of an agent. Of course, the reason for this approach being feasible is that the state space is only moderately-sized, which makes it possible to sweep through all state-action pairs repeatedly in the manner of a DP algorithm; in a larger system, this approach would not be possible.

In the case of A-HYPOT R-learning, the algorithm given on page 323 must be modified quite significantly in order to use a synchronous updating style; this is because the algorithm relies upon the storage of an array of values $z_1, z_2, ...$ which represent the service completion times of customers, and these values are used to determine future random transitions. This means that the order in which state-action pairs are updated *must* follow the simulated evolution of the system, otherwise the algorithm simply does not make sense. In the case of a synchronous algorithm, it must be possible to simulate a random transition from a particular state without any dependence on the transitions simulated previously. A full description of the changes required in order to use the A-HYPOT algorithm synchronously will not be given here, but essentially it is only a minor programming task to modify the algorithm so that transitions are simulated according to the principle that, given that the system is in some physical state $\mathbf{x} \in \tilde{S}$ at an arbitrary point in time, the probability of an arrival occurring before the next service completion is:

$$\frac{\lambda}{\lambda + \sum_{i=1}^{N} \min(x_i, c_i)\mu_i}.$$

This example presents the result of an experiment in which 10,000 iterations of the synchronous version of A-HYPOT R-learning were performed (in other words, the values $\hat{Q}(\mathbf{x}, a)$ for all pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ were updated 10,000 times each) in order to estimate the Q-factors with a reasonable level of accuracy. Following the termination of the algorithm, the quadratic model (7.4.4) was fitted to the final $\hat{Q}(\mathbf{x}, a)$ values using linear regression with $\{x_1, x_2, x_3, x_4, x_1^2, x_2^2, x_3^2, x_4^2\}$ as the

set of predictor variables, and a different set of weights obtained for each action $a \in \{0, 1, 2, 3, 4\}$. Table 7.4 shows the weights and the regression coefficient R^2 obtained for each a.

	$a = 0$	$a = 1$	$a = 2$	$a = 3$	$a = 4$
$B(a)$	372.492	487.581	485.875	458.737	421.840
$C_1^{(1)}(a)$	-188.248	-201.253	-191.044	-189.977	-188.866
$C_2^{(1)}(a)$	-73.915	-76.511	-86.368	-75.554	-74.423
$C_3^{(1)}(a)$	-15.638	-16.761	-16.667	-23.638	-15.998
$C_4^{(1)}(a)$	-3.821	-4.102	-4.061	-4.160	-8.072
$C_1^{(2)}(a)$	0.880	0.318	1.015	0.964	0.909
$C_2^{(2)}(a)$	-0.113	-0.015	-0.001	-0.053	-0.097
$C_3^{(2)}(a)$	-0.571	-0.552	-0.562	-0.657	-0.576
$C_4^{(2)}(a)$	-0.172	-0.176	-0.179	-0.178	-0.168
R^2	**0.995**	**0.996**	**0.996**	**0.996**	**0.995**

Table 7.4: Results of the multivariate linear regression experiment described in Example 7.4.2.

Table 7.4 shows that, for each action $a \in \{0, 1, 2, 3, 4\}$, the regression coefficient R^2 obtained by fitting the model (7.4.4) to the estimated values $\hat{Q}(\mathbf{x}, a)$ is greater than 0.995. Arguably, this can be regarded as a strong indication that, in the case of the particular 4-facility system under consideration, the true values $Q(\mathbf{x}, a)$ approximately fit the quadratic model. ⊠

Of course, as alluded to previously, the least-squares regression model considered in Example 7.4.2 assumes that all 'data pairs' $((\mathbf{x}, a), \hat{Q}(\mathbf{x}, a))$ are equally important; it has no means of ascertaining which Q-factors are the most important to approximate accurately. Clearly, this kind of impartiality is not really desirable. It would be preferable for the model-fitting to be done in a *biased* manner, in order to ensure a very close fit for the Q-factors at often-visited states, without great importance being attached to the quality of the fit at rarely-visited states. RL algorithms which use function fitting will naturally tend to prioritise 'influential' states in exactly this manner.

The next objective in this section will be to describe exactly how function fitting can be carried out as part of an RL algorithm. It is worth re-emphasising the fundamental assumption of function-fitting methods that *at no stage* of the algorithm can a Q-factor $\hat{Q}(\mathbf{x}, a)$ (nor any other type of

quantity, such as the number of times $(\mathbf{x}, a)$ has been visited) be stored for any individual state-action pair $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$. Hence, it is not possible in general to carry out the procedure described in Example 7.4.2, in which model-fitting is applied to a set of $\hat{Q}(\mathbf{x}, a)$ values which have already been acquired by an algorithm which depends upon storing these values explicitly.

Given that the value $Q(\mathbf{x}, a)$ for any individual pair $(\mathbf{x}, a)$ depends upon the Q-factors for *other* state-action pairs (as shown by the equations (7.2.2)), any update to the value $\hat{Q}(\mathbf{x}, a)$ made as part of an RL algorithm must be made according to estimates of the Q-factors for the *other* pairs; so, if these estimates are not stored in the memory, they must be acquired using the approximating function(s) that are constructed as part of the algorithm. However, these approximating functions themselves will need to be constructed by fitting a model to the latest available estimates for the Q-factors. In summary, fitting a model to approximate the Q-factors requires estimates for the Q-factors to be available, but calculating these estimates can only be done by using some kind of fitted model. This creates a 'chicken-and-egg' situation, and therefore both tasks must be carried out at the same time; that is, estimates for the Q-factors must be obtained using a model which is updated each time a new Q-factor estimate becomes available. Roughly speaking, an RL algorithm which uses function-fitting must perform the following steps on each iteration n:

- Simulate a random transition from the current state $\mathbf{x}_n$ following the choice of action a_n. Observe the random reward r_n and the new state $\mathbf{x}_{n+1}$.
- Use an approximating function to estimate the values $Q(\mathbf{x}_{n+1}, a)$ at the new state $\mathbf{x}_{n+1}$. Accordingly, calculate the new 'data piece', $y_n := r_n - g_n + \max_a \hat{Q}(\mathbf{x}_{n+1}, a)$.
- Update the relevant approximating function using the new data piece y_n in such a way that (ideally) $\hat{Q}(\mathbf{x}_n, a_n)$ becomes a more accurate approximation of $Q(\mathbf{x}_n, a_n)$.

Of course, the problem of constructing an approximating function $\hat{Q}(\mathbf{x}, a)$ for a given action a is somewhat similar to the classical regression problem of fitting a model to a given set of data, in the sense that one uses a set of 'data pairs' $((\mathbf{x}_n, a_n), y_n)$ to determine suitable values for the model parameters; however, classical regression methods are applicable in situations where all of the data values are *known* before the model is fitted. In the context of RL, data pairs are acquired piece-by-piece, and the values $y_n = r_n - g_n + \max_a \hat{Q}(\mathbf{x}_{n+1}, a)$ themselves depend upon the latest available

estimates for the model parameters. As such, one has no choice but to fit the model *incrementally* in the manner described above, which makes the procedure somewhat more difficult.

One possible method of constructing the approximating functions involves using *incremental linear regression*, which is similar to classical linear regression except that one must continually append new rows of data to the design matrix (consisting of values of the predictor variables) and the vector of observed values of the dependent variable. This is also referred to as *online linear regression*; see, for example, Strehl et. al. [173]. However, this approach tends to be very computationally-intensive, as one must store large arrays of data and perform time-consuming matrix computations on each iteration; as such, it is not feasible for the problems considered in this section.

The method of function-fitting used in this section will be based on *artificial neural networks* (ANNs). The term 'neural network' may be understood in a broad sense to refer to any procedure or algorithm by which one learns to associate a particular response with any given set of inputs or 'stimuli'. Applications of ANNs are common in various areas of computer science, statistics, and other fields; see, for example, [17, 72, 80, 133, 190]. Typical applications of ANNs might include the training of a machine to recognise an image or pattern according to a particular layout of pixels on a computer screen, or the medical diagnosis of a patient based on their observed symptoms. The origins of ANNs can be traced back at least as far as the 1960s and early 1970s (see [194, 200]); for a detailed review of ANNs which includes a comparison with classical statistical methodology, see [30]. In this section, the ANNs considered will be *linear neural networks*; these are considerably simpler in their formulation than non-linear networks, but are entirely suitable for the purpose of fitting a linear model such as (7.4.3) to a set of data which is acquired incrementally.

A linear neural network, also referred to as a *neuron*, applies in general to a situation where some output value $y \in \mathbb{R}$ is to be predicted using a set of input values $x_1, x_2, ..., x_N$ using the model $\hat{y} = w_0 + \sum_{i=1}^{N} w_i x_i$, where the w_i are weights to be determined. As depicted by Figure 7.12, the role of the neural network is simply to attach weights w_i to the inputs x_i in such a way that y is estimated with the greatest possible accuracy. In this sense, the linear neural network is used for the same purpose as classical linear regression; however, its essential feature is that it performs the model-fitting *incrementally*, using a procedure which is not at all computationally intensive. Note that in Figure 7.12, an extra input $x_0 = 1$ is included; this represents the *bias* term (or *intercept*)

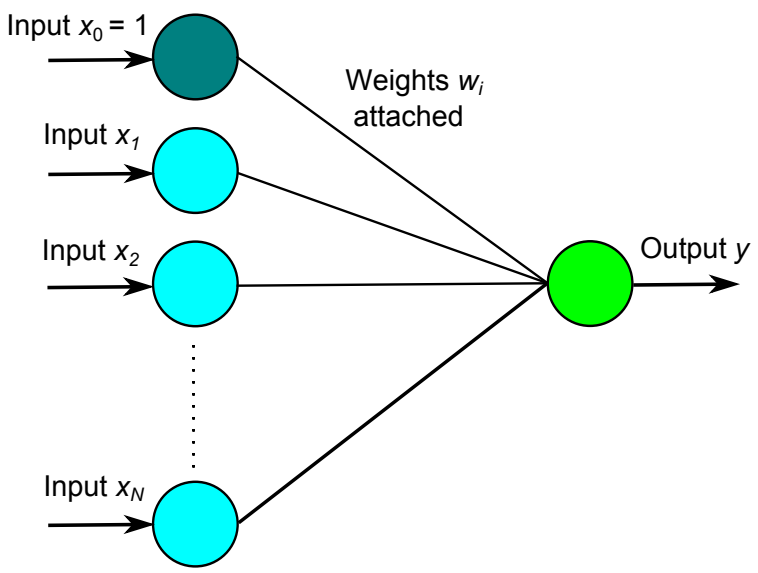


Figure 7.12: A simple linear neural network with N input nodes and only one output node.

used in the linear model, and the weight attached to this input will simply be w_0. As such, the neural network must attach $N + 1$ weights in order to determine the output value y. Clearly, the polynomial model (7.4.3) is of the required form to be fitted using a linear ANN.

The structure of a neural network becomes considerably more complicated when one attempts to model *non-linear* behaviour. In a non-linear ANN, the input nodes will typically be connected to an additional set of nodes referred to as *hidden nodes*, which comprise the *hidden layer* of the network. Broadly speaking, the purpose of the hidden layer is to apply a transformation to the input values in order to allow for a possible non-linear relationship between the input values and the output; for example, after computing a weighted sum of the input values, a hidden layer might then convert this sum into a value between 0 and 1 using a *sigmoidal function* (see [15], p. 62). If the relationship being modelled is thought to be highly non-linear, then several hidden layers may be used; the last hidden layer will then be connected to an output node (or, if several different outputs are required, an output layer). For further discussion of non-linear ANNs, see [62] (p. 75). As mentioned earlier, attention will be restricted to linear ANNs in this section.

The particular function-fitting RL algorithm considered in this section will be based on the *Widrow-Hoff* (WH) algorithm, which was first developed by Widrow and Hoff [200] and originally named

'Adaline', which is short for 'adaptive linear'. The algorithm is recommended by Gosavi [62] (p. 69) for calibrating the weights of a linear ANN. In order to illustrate the algorithm's underlying simplicity, it will be easiest to present it in the context of a general problem involving the fitting of an adaptive linear model to a set of data pairs which are acquired incrementally.

Widrow-Hoff (WH) algorithm

1. Assume that the model to be fitted is $\hat{y} = w_0 + \sum_{i=1}^{N} w_i x_i$, where the w_i are weights to be determined. Set $n = 0$ and initialise the values $w_i = 0$ for $i = 0, 1, ..., N$.

2. Acquire a new data piece $(\mathbf{x}_n, y_n)$, where $\mathbf{x}_n$ is a vector with N components and y_n is a scalar. Calculate q_n, the output of the neural network at the n^{th} stage, as follows:

$$q_n = w_0 + \sum_{i=1}^{N} w_i (\mathbf{x}_n)_i,$$

   where $(\mathbf{x}_n)_i$ denotes the i^{th} component of the vector $\mathbf{x}_n$.

3. For each $i \in \{0, 1, ..., N\}$, update the weight w_i as follows:

$$w_i \leftarrow w_i + \delta_n \left(y_n - q_n\right) (\mathbf{x}_n)_i, \tag{7.4.5}$$

   where δ_n is a learning parameter which may decay with n, and $(\mathbf{x}_n)_0 := 1$.

4. If $n > n_{\max}$, where $n_{\max}$ is a large integer, then return the set of weights w_i as the final calibration for the ANN and stop. Otherwise, increment n by 1 and return to step 2.

The update rule (7.4.5) is well-founded in theory and in fact is based on the *gradient descent* optimisation algorithm, which is commonly used in the field of non-linear optimisation (see, for example, [7]). Consider a problem in which one has some finite number M of data pairs $(\mathbf{x}_n, y_n)$, and the objective is to determine values of the weights $w_0, w_1, ..., w_N$ which will best approximate the values $y_1, y_2, ..., y_M$ via the model $\hat{y}_n = w_0 + \sum_{i=1}^{N} w_i (\mathbf{x}_n)_i$ (for $n = 1, 2, ..., M$). In the spirit of classical linear regression, let the sum-of-squares error $SSE(\mathbf{w})$, be defined by:

$$SSE(\mathbf{w}) := \frac{1}{2} \sum_{n=1}^{M} \left(y_n - w_0 - \sum_{i=1}^{N} w_i (\mathbf{x}_n)_i \right)^2, \tag{7.4.6}$$

where $\mathbf{w} = (w_0, w_1, ..., w_N)$ is the vector of weights. Naturally, the optimal set of weights should minimise $SSE(\mathbf{w})$. Using gradient descent, one would initialise a vector of weights $\mathbf{w}_0$ with components $(\mathbf{w}_0)_i = 0$ for $i = 0, 1, ..., N$ and then calculate $\mathbf{w}_t$ for $t \geq 1$ as follows:

$$\mathbf{w}_t := \mathbf{w}_{t-1} - \delta_t \nabla SSE(\mathbf{w}_{t-1}), \tag{7.4.7}$$

where ∇ denotes the gradient operator and, as usual, δ_t is a step-size (learning parameter) rule. Provided that δ_t satisfies the conditions (A.9.3), it can be shown that $\mathbf{w}_t$ converges to a minimiser of (7.4.6) as $t \to \infty$ (see [77], p. 395). Computing the gradient ∇SSE, one finds:

$$\nabla SSE(\mathbf{w}) = -\sum_{n=1}^{M} \left(y_n - w_0 - \sum_{i=1}^{N} w_i (\mathbf{x}_n)_i \right) \mathbf{x}_n.$$

Hence, (7.4.7) implies that one should derive $(\mathbf{w}_t)_i$ (for $i = 0, 1, ..., N$) on the t^{th} iteration of gradient descent using the data pairs $(\mathbf{x}_n, y_n)$ (with $(\mathbf{x}_n)_0 := 1$ for all n) as follows:

$$(\mathbf{w}_t)_i := (\mathbf{w}_{t-1})_i + \delta_t \sum_{n=1}^{M} \left(y_n - (\mathbf{w}_{t-1})_0 - \sum_{i=1}^{N} (\mathbf{w}_{t-1})_i (\mathbf{x}_n)_i \right) (\mathbf{x}_n)_i. \tag{7.4.8}$$

This type of procedure is referred to as *batch gradient descent*, since it involves using *all* of the data pairs $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_M, y_M)$ on each iteration t of the algorithm. Although the number of data pairs is assumed finite at this stage of the discussion, if M is large then it is clearly not ideal to have to inspect the entire set of data on each iteration. Fortunately an alternative approach is available, referred to as *stochastic* gradient descent (also *incremental* gradient descent), which in fact may be regarded as a simplification of the batch-updating procedure. Essentially, in stochastic gradient descent, one sweeps through the set of M data pairs repeatedly but updates the vector of weights using only one data pair at a time (see [134] for further explanation), which considerably speeds up the process in a real-time sense. Let $(\mathbf{x}_t, y_t)$ denote the single data pair used on the t^{th} iteration of stochastic gradient descent. Then the update rule for $(\mathbf{w}_t)_i$ is:

$$(\mathbf{w}_t)_i := (\mathbf{w}_{t-1})_i + \delta_t \left(y_t - (\mathbf{w}_{t-1})_0 - \sum_{i=1}^{N} (\mathbf{w}_{t-1})_i (\mathbf{x}_t)_i \right) (\mathbf{x}_t)_i. \tag{7.4.9}$$

which is equivalent to the update rule (7.4.5) used by the Widrow-Hoff algorithm. Thus, the Widrow-Hoff update rule is based on the application of stochastic gradient descent to a problem involving minimisation of the sum-of-squares error defined in (7.4.6). Moreover, the convergence properties of gradient descent imply that if the WH algorithm is applied to a finite set of data

pairs and allowed to sweep through the data pairs one-by-one an indefinite number of times, then (assuming the learning parameters δ_t satisfy the conditions (A.9.3)) it will eventually converge to the same set of weights that one would obtain using least-squares linear regression.

Of course, the context of the discussion given in the previous two pages is quite different from the context of attempting to find a socially optimal control policy for a queueing system using an RL algorithm. In an RL context, one considers an infinite sequence of data pairs $((\mathbf{x}_n, a_n), y_n)$ in which the values $y_n = r_n - g_n + \max_a \hat{Q}(\mathbf{x}_{n+1}, a)$ are intended to be estimates of the true values $Q(\mathbf{x}_n, a_n)$ that one wishes to approximate, but in fact are subject to both estimation errors and random noise. In such circumstances, it does not appear possible to devise an RL algorithm which will provably converge to a set of weights (and resulting estimates $\hat{Q}(\mathbf{x}, a)$) that will minimise the sum-of-squares error over all state-action pairs in $\tilde{S} \times A_{\mathbf{x}}$. Moreover, even if it were possible to achieve such convergence, the minimisation of a quantity such as $\sum_{(\mathbf{x},a)} \left(Q(\mathbf{x}, a) - \hat{Q}(\mathbf{x}, a)\right)^2$ is not necessarily what one wants to achieve in order to obtain a near-optimal policy, since (as discussed previously) it is actually more desirable to obtain closer approximations for some Q-factors than for others. For these reasons, as is usually the case with RL in general, it is appropriate to approach the problem of value function approximation in something of a heuristic spirit, and aim to justify any particular methodology by demonstrating its strong performance empirically.

Having discussed the relevant background material, it will now be possible to present the RL algorithm that will be used in this section to estimate the Q-factors for an optimal policy using value function approximation. The algorithm relies upon linear neural networks (one for each permissible action a) to fit the quadratic model (7.4.4), with the Widrow-Hoff updating rule used to update the weights of the networks. It is also based very closely on the A-HYPOT R-learning algorithm discussed in the previous section; in fact, essentially the only difference is that it relies on approximating functions to estimate Q-factors, rather than storing values explicitly. Although some of the steps are simply repetitions of those found in the A-HYPOT algorithm, for the purposes of clarity it will be appropriate to present the new algorithm in a complete form.

NEURO-HYPOT algorithm

1. Initialise the values $n = 1$ and $t_0 = 0$. For each action $a \in \bigcup_{\mathbf{x} \in \tilde{S}} A_{\mathbf{x}}$, initialise the weights $B(a) = 0$, $C_i^{(1)}(a) = 0$ and $C_i^{(2)}(a) = 0$ for each $i \in \{1, 2, ..., N\}$. Set the initial state $\mathbf{x}_1 = \mathbf{0}$.

For each $i \in \{1, 2, ..., N\}$ and $j \in \{1, 2, ..., c_i\}$, initialise the value:

$$f(i, j) = 0,$$

which represents the time at which the j^{th} channel at facility i becomes 'free'. Also, use inverse transform sampling to obtain t_1, the arrival time of the first customer.

2. Use inverse transform sampling to determine d_n, the time between the n^{th} and $(n+1)^{th}$ customer arrivals, then let $t_{n+1} = t_n + d_n$. For each facility $i \in \{1, 2, ..., N\}$, determine the number of service completions that occur between times t_n and t_{n+1} using the known actions $a_1, a_2, ..., a_{n-1}$ and exit times $z_1, z_2, ..., z_{n-1}$ of previous customers. Denote these values by κ_i $(i = 1, 2, ..., N)$. Then determine the state $\mathbf{y}_n \in \tilde{S}$ using $\mathbf{x}_n$ as follows:

$$\mathbf{y}_n = \mathbf{x}_n - \sum_{i=1}^{N} \kappa_i \mathbf{e}_i,$$

where, as previously, $\mathbf{e}_i$ is an N-vector with i^{th} component equal to one and all other components equal to zero. One may interpret $\mathbf{y}_n$ as the state of the system at time t_{n+1} in the event that the n^{th} customer balks, as opposed to joining some facility.

3. For each action $a \in A_{\mathbf{x}_n}$, carry out the following sub-procedure:

   (a) If $a = i$ for some $i \in \{1, 2, ..., N\}$, then use inverse transform sampling to sample a random service-time requirement at facility i, and denote this by $s(i)$. Then calculate the (hypothetical) system departure time $Z(i)$ as follows:

   $$Z(i) = \min_{j \in \{1,2,...,c_i\}} f(i, j) + s(i).$$

   One may interpret $Z(i)$ as the departure time of the n^{th} customer in the event that they go to facility i. On the other hand, if $a = 0$ then set $Z(a) = t_n$.

   (b) Let the reward $R(a)$ be given by:

   $$R(a) = \begin{cases} \lambda\Big(\alpha_i - \beta_i\big(Z(i) - t_n\big)\Big), & \text{if } a = i \text{ for some } a \in \{1, 2, ..., N\}, \\ 0, & \text{if } a = 0. \end{cases}$$

   (c) Define the state $\mathbf{y}$ as follows:

   $$\mathbf{y} := \begin{cases} \mathbf{y}_n^{i+}, & \text{if } a = i \text{ for some } i \in \{1, 2, ..., N\} \text{ and } Z(i) \geq t_{n+1}, \\ \mathbf{y}_n, & \text{otherwise.} \end{cases}$$

Thus, $\mathbf{y}$ is the state of the system at time t_{n+1} given that action a is chosen. For each action $a' \in A_{\mathbf{y}}$, obtain an estimate for $Q(\mathbf{y}, a')$ as follows:

$$\hat{Q}(\mathbf{y}, a') := B(a') + \sum_{i=1}^{N} C_i^{(1)}(a') \frac{y_i}{b_i} + \sum_{i=1}^{N} C_i^{(2)}(a') \left(\frac{y_i}{b_i}\right)^2, \tag{7.4.10}$$

where $b_i = \max_{\mathbf{x} \in \tilde{S}} x_i$ for each $i \in \{1, 2, ..., N\}$, i.e. b_i is dependent on the boundaries of the finite state space. Let the 'new data piece', $L(a)$, be defined as follows:

$$L(a) := R(a) - B_{\max} + \max_{a' \in A_{\mathbf{y}}} \hat{Q}(\mathbf{y}, a'), \tag{7.4.11}$$

where $B_{\max} := \max_a B(a)$, i.e. the maximum of the bias weights.

(d) For notational convenience, let $\mathbf{x} = \mathbf{x}_n$. Then, let $\hat{Q}(\mathbf{x}, a)$ be given by:

$$\hat{Q}(\mathbf{x}, a) := B(a) + \sum_{i=1}^{N} C_i^{(1)}(a) \frac{x_i}{b_i} + \sum_{i=1}^{N} C_i^{(2)}(a) \left(\frac{x_i}{b_i}\right)^2. \tag{7.4.12}$$

Then, update the weights of the neural network as follows:

$$\begin{aligned}
&B(a) \leftarrow B(a) + \delta_n \left(L(a) - \hat{Q}(\mathbf{x}, a)\right), \\
&C_i^{(1)}(a) \leftarrow C_i^{(1)}(a) + \delta_n \left(L(a) - \hat{Q}(\mathbf{x}, a)\right) \frac{x_i}{b_i} \qquad (i = 1, 2, ..., N), \\
&C_i^{(2)}(a) \leftarrow C_i^{(2)}(a) + \delta_n \left(L(a) - \hat{Q}(\mathbf{x}, a)\right) \left(\frac{x_i}{b_i}\right)^2 \qquad (i = 1, 2, ..., N),
\end{aligned} \tag{7.4.13}$$

where $\delta_n \in [0, 1]$ is a learning parameter which does *not* depend on $(\mathbf{x}_n, a)$.

4. Select an action $a_n \in A_{\mathbf{x}_n}$ according to a rule which allows for both exploration and exploitation (e.g. an ϵ-greedy rule). Then, let $r_n = R(a_n)$ and $z_n = Z(a_n)$, where $R(a_n)$ and $Z(a_n)$ are the values obtained for action $a_n \in A_{\mathbf{x}_n}$ in step 3. Also, let $\mathbf{x}_{n+1} = \mathbf{y}$.

5. If $a_n = i$ for some $i \in \{1, 2, ..., N\}$, then let k be a service channel satisfying:

$$k \in \underset{j \in \{1, 2, ..., c_i\}}{\arg\min} f(i, j),$$

Then update the 'channel free' time $f(i, k)$ as follows:

$$f(i, k) \leftarrow f(i, k) + s(i),$$

where $s(i)$ is the same service-time value obtained in step 3(a).

6. If $n = n_{\max}$, where $n_{\max}$ is a large integer, then output a stationary policy $\theta^{[R]}$ which, at each state $\mathbf{x} \in \tilde{S}$, chooses an action $\theta^{[R]}(\mathbf{x})$ maximising the expression $\hat{Q}(\mathbf{x}, a)$ over all actions $a \in A_{\mathbf{x}}$. Also output $B_{\max} = \max_a B(a)$ as an estimate for the optimal average reward g^*, then stop. Alternatively, if $n < n_{\max}$, increment n by 1 and return to step 2.

Steps 3(c) and 3(d) represent the function-fitting steps of the NEURO-HYPOT algorithm, in which Q-factors are estimated using the latest available weights for the neural network and then the weights themselves are updated according to the resulting Q-factor estimates. The update rules for the weights in (7.4.13) are based on the Widrow-Hoff algorithm, with $L(a) - \hat{Q}(\mathbf{x}, a)$ representing the difference between the new data value acquired and the output of the neural network (essentially, the value predicted by the quadratic model). Although the algorithm is similar to A-HYPOT R-learning in many respects, there are some extra nuances which should be discussed.

Firstly, the algorithm differs from the various R-learning algorithms presented in previous sections by *not* using a sequence of values g_n to estimate the optimal average reward g^*. Instead, it uses the value $B_{\max} = \max_a B(a)$. Note that, due to (7.4.4), $B_{\max} = \max_{a \in A_{\mathbf{0}}} \hat{Q}(\mathbf{0}, a)$, and therefore this update rule is consistent with the relative Q-learning algorithm discussed in Appendix A.9 (see Example A.9.3). By Lemma A.9.2, the optimal average reward g^* is equal to $\max_{a \in A_{\mathbf{0}}} Q(\mathbf{0}, a)$, which is why $B_{\max}$ is output as an estimate for g^* in the final step of the NEURO-HYPOT algorithm. Obviously, there is no guarantee that the final value of $B_{\max}$ will be an accurate estimate for the average reward that one would actually obtain under the stationary policy $\theta^{[R]}$; indeed, $B_{\max}$ might even be greater than g^*. In order to evaluate the performance of the policy $\theta^{[R]}$ accurately, one may perform regular 'Frozen Phase' (FP) evaluations as discussed in Section 7.2.

The reason for the 'relative Q-learning'-style rule being preferred to the 'R-learning'-style rule for defining the data pieces in (7.4.11) (in other words, g_n being omitted in favour of $B_{\max}$) is that, based on experimental results, it appears that the stability of the algorithm is improved by using this method. When function-fitting is used, the weights of the neural network may be extremely volatile in the early stages of the algorithm, in the sense that they may fluctuate wildly and take undesirably large or small values. If one attempts to use the R-learning approach and rely upon a sequence of values g_n which are updated using a rule similar to (7.2.12), then the calculation of each value g_n must be made according to estimates for two Q-factors, which in turn are dependent

upon the weights of the neural network. When this approach is used, it appears that the behaviour of the sequence $(g_n)_{n\in\mathbb{N}}$ is simply too unpredictable in the initial phase of the algorithm; it may even diverge (in either a positive or negative direction) and cause the weights of the network to do the same. When $B_{\max}$ is used instead of g_n, the estimates for the average reward are made according to a single bias term, and thus are less susceptible to divergent behaviour.

Secondly, the neural networks used by the algorithm are calibrated in such a way that they do *not* simply use the components x_i of the state vector $\mathbf{x}$ (and the squared components x_i^2) as inputs, but instead use x_i/b_i and $(x_i/b_i)^2$, where $b_i = \max_{\mathbf{x}\in\tilde{S}} x_i$ for each $i \in \{1, 2, ..., N\}$ (see steps 3(c) and 3(d)). This is a way of *normalising* the inputs, so that they always take values between 0 and 1. Again, this adjustment is made in order to improve the algorithm's performance; it appears to behave in a more stable manner when the inputs are restricted to the range $[0, 1]$ (see [62], p. 77). Of course, even when the inputs x_i/b_i and $(x_i/b_i)^2$ are used, the theoretical model being fitted by the algorithm still has the form (7.4.4); effectively, one simply re-scales the values of the weights w_i. The boundary values b_i are dependent on the bounds of the finite state space $\tilde{S}$, which can be chosen in the initial configuration of the algorithm. An obvious possibility is to let b_i correspond to the selfish threshold $\tilde{B}_i$ defined in (4.1.2), but in fact this may yield poor results in systems where average reward optimality is achieved by policies which cause the process to remain 'distant' from the boundaries of $\tilde{S}$ at all times. This will be discussed further in the next example.

A further important point to make is that the learning parameters δ_n must *not* be dependent upon the number of visits made to any particular state or state-action pair, nor must they be determined in any other way which would require a unique value to be stored for every individual state or pair. Naturally, this is in keeping with the constraints assumed throughout this section. The examples given earlier in this chapter relied upon rules such as (7.2.14) for the learning parameters, so this means that a new rule for δ_n must be found for the NEURO-HYPOT algorithm. There are many ways of defining δ_n so that it decays with n whilst also preserving the properties (A.9.3); however, in practice, setting δ_n to a small constant value (i.e. removing the dependence on n) can also work well in some scenarios. Again, this will be discussed in the example which follows.

Example 7.4.3. ***(Structural properties of the NEURO-HYPOT policy)***

The purpose of this example is not only to test the performance of the NEURO-HYPOT algorithm, but also to examine the structural properties of the policy that it obtains. The latter will be done by examining the final values of the weights of the neural network in order to determine whether or not various structural properties discussed in earlier chapters (e.g. containment, monotonicity) are preserved by the 'neural' policy $\theta^{[R]}$. Once again, the parameters from Example 7.2.3 will be considered. The system under consideration has a demand rate $\lambda = 25$, 4 service facilities with parameters given in (7.2.13) and $|\tilde{S}| \approx 43{,}000$, where $\tilde{S}$ is the selfish state space.

An experiment was performed in which the NEURO-HYPOT algorithm was allowed to run for 40 million iterations. As in previous examples, an ϵ-greedy rule was used for selecting actions, with $\epsilon = 0.15$. Frozen Phase (FP) updates (i.e. simulation runs using the algorithm's latest 'guess' for an optimal policy) were performed after every 10,000 iterations in order to gauge its progress. Some initial tests showed that when the 'caps' b_i used in steps 3(c) and 3(d) of the algorithm were set equal to the selfish thresholds $\tilde{B}_i$ defined in (4.1.2), a relatively large number of iterations were required before the algorithm began to approach an optimal policy. It appears that in general, the progress of the algorithm may be slow in systems for which there exists an average reward optimal policy θ^* with an associated set of positive recurrent states S_{θ^*} much smaller than $\tilde{S}$.

As discussed previously, evidence from the literature (see [62], p. 77) suggests that it is desirable to restrict the neural network (ANN) inputs to the range $[0, 1]$ if it is feasible to do so. Based on this principle, the A-HYPOT algorithm has been designed in such a way that it uses the bounds of the finite state space $\tilde{S}$ to scale the ANN inputs so that they are guaranteed to lie in the range $[0, 1]$. While this approach may not be strictly necessary for ensuring stability, it appears to be sufficient for the purpose at hand. However, the problem remains of exactly how the bounds of the finite set $\tilde{S}$ should be determined. If $\tilde{S}$ is too large, then the inputs received by the ANN on most of its iterations will tend to be much smaller than 1, and this will cause the algorithm to perform poorly (in terms of convergence speed). This is likely to be the reason for the selfish state space being unsuitable for employment as the finite state space $\tilde{S}$ in this particular example.

In order to improve the performance of the algorithm, it was decided that the values b_i should be set approximately equal to the boundary values associated with an average reward optimal policy.

Of course, one begins the procedure without any knowledge of the structure of an optimal policy, so this poses a problem. The solution adopted in this particular experiment was to draw upon results from Section 6.1, and determine the values b_i according to the *Whittle indices* $W_i(x)$ defined in (6.1.13). The Whittle index policy is a heuristic policy which, based on the results in Chapter 6, appears to closely approximate an optimal policy in many given problem instances. Thus, it is reasonable to suppose that the positive recurrent state space S_W associated with the Whittle policy should have similar dimensions to the corresponding region S_{θ^*} associated with an optimal policy. However, it is also known (by Theorem 6.1.10) that the Whittle policy is actually *more conservative* than the optimal policy that one would find using relative value iteration. Taking this into account, a reasonable approach is to define the thresholds b_i as follows:

$$b_i := \min\{x \in \mathbb{N}_0 : W_i(x) < 0\} + K \qquad (i = 1, 2, ..., N), \tag{7.4.14}$$

where K is a non-negative integer. By using this approach, one defines the finite state space $\tilde{S}$ so that it includes all states $\mathbf{x} \in S_W$ which would be positive recurrent under the Whittle policy, *and* (by adding the extra term K in (7.4.14)) some extra states which would *not* be recurrent under the Whittle policy. As explained above, the idea of including these 'extra states' is to allow the policy $\theta^{[R]}$ found by the NEURO-HYPOT algorithm the freedom to be slightly less conservative than the Whittle policy; there is no guarantee that the algorithm will yield a policy which actually accesses these 'extra states', so the intention is merely to allow some extra flexibility.

Ideally, the value of K should be small, in order to allow the algorithm to converge quickly towards a strong policy. However, small values of K also restrict the algorithm's freedom to explore beyond the realms of the Whittle state space S_W, so (once again) there is a compromise to be made between the amount of running time required and the likelihood of obtaining a near-optimal policy. In this particular example, it was decided after some experimentation that $K = 2$ would be a suitable value. Using (7.4.14) to compute the resulting b_i values, one obtains 6, 6, 7 and 7 for b_1, b_2, b_3 and b_4 respectively. The corresponding selfish thresholds are 14, 16, 12 and 12, so in this example the rule (7.4.14) yields a very significant reduction in the size of the finite state space.

Experimental results have also shown that, perhaps surprisingly, choosing a small constant value for the learning parameters δ_n can yield satisfactory results in many problem instances. Perhaps the simplest explanation for this is that if a rule is chosen which causes δ_n to decay with n, then

the performance of the NEURO-HYPOT algorithm can be quite sensitive to the *rate* of decay. When the δ_n values become very small, essentially the changes made to the weights in step 3(d) become negligible, so that no further improvements to the policy are possible. In general, the more complicated the system (in terms of the number of facilities, size of the state space, etc.), the slower the rate of decay should be. Thus, to be on the safe side, it appears that choosing a very slow rate of decay should always be a good option, but if the running time of the algorithm is only finite then this is almost equivalent to using a constant value. It is worth noting that the RL algorithms considered in earlier sections were largely able to avoid the problem of making the rate of decay appropriate for the size of the system by using rules such as (7.2.14), under which the learning parameters depend on the numbers of visits made to individual state-action pairs, which naturally causes the rate of decay to be slower in larger systems. When the parameters δ_n are allowed to depend only on the iteration count n, it is more difficult to find an appropriate rule.

When a constant value for the learning parameters is used, one must abandon any hopes of 'convergence' being achieved in any rigorous mathematical sense (obviously, the second condition in (A.9.3) is no longer satisfied); however, as discussed at various stages earlier in this chapter, convergence of the Q-factor estimates is not necessarily either realistic or desirable where RL algorithms are concerned. The advantage of using a constant value for the learning parameters is that it ensures that the algorithm persists in 'exploring' throughout its finite run-time; at no stage does it simply 'stall out'. Returning to the present example, it was decided after some experimentation that the value $\delta_n = 0.001$ (for all $n \in \mathbb{N}$) would be suitable. Table 7.5 shows, for each action $a \in \bigcup_{\mathbf{x}} A_{\mathbf{x}}$, the values of the weights $B(a)$, $C_i^{(1)}(a)$ and $C_i^{(2)}(a)$ obtained after 40 million iterations.

It should be made clear that the weights shown in Table 7.5 are intended for use in the original form of the quadratic model (7.4.4), and *not* the 'adjusted' version of the model (7.4.12) in which the inputs x_i and x_i^2 are divided by b_i and b_i^2 respectively. In other words, the final values of the weights output by the NEURO-HYPOT algorithm have been re-scaled so that they fit the original model (7.4.4). This enables an easy comparison with the results from the multivariate regression experiment in Example 7.4.2 (shown in Table 7.4) to be made. Without conducting a detailed analysis, it is obvious that there are considerable differences between the two sets of results, with the NEURO-HYPOT algorithm appearing to produce larger values for the weights $C_i^{(2)}(a)$

	$a=0$	$a=1$	$a=2$	$a=3$	$a=4$
$B(a)$	-18.729	127.089	99.872	54.188	22.068
$C_1^{(1)}(a)$	-81.672	-86.183	-86.137	-81.222	-81.097
$C_2^{(1)}(a)$	-33.499	-35.760	-39.738	-32.437	-33.497
$C_3^{(1)}(a)$	0.971	0.498	0.335	0.889	1.020
$C_4^{(1)}(a)$	-0.756	-0.591	-0.487	-0.559	-1.271
$C_1^{(2)}(a)$	-11.037	-15.725	-10.688	-11.133	-11.112
$C_2^{(2)}(a)$	-4.141	-3.938	-5.699	-4.445	-4.143
$C_3^{(2)}(a)$	-1.748	-1.692	-1.731	-3.361	-1.762
$C_4^{(2)}(a)$	-0.068	-0.061	-0.044	-0.057	-0.246

Table 7.5: Weights obtained using the NEURO-HYPOT algorithm for the system in Example 7.4.3.

associated with the quadratic terms (except in the case $i = 4$), so that the approximating functions appear to have a more 'quadratic shape' than those obtained using regression. Incidentally, a quick simulation experiment shows that the stationary policy that would be obtained using the regression weights in Table 7.4 yields an average reward of approximately 123.6, whereas the NEURO-HYPOT policy $\theta^{[R]}$ attains an average reward within 0.1% of the optimal value $g^* \approx 126.6$.

Figure 7.13 provides a diagrammatic comparison between the performances of the A-HYPOT R-learning algorithm (see page 323) and the NEURO-HYPOT algorithm over the entire 40 million iterations. For convenience, let A-HYPOT and NEURO-HYPOT be abbreviated to A-H and N-H respectively. Since the N-H algorithm does not use a sequence of values g_n to track the estimated average reward, the evolution of the bias term $B_{\max}$ (which, as discussed previously, also provides an approximation for the average reward) is shown in the figure instead. Unsurprisingly, the behaviour of $B_{\max}$ is considerably more erratic than that of the values g_n used by the A-H algorithm, and the FP estimates are also somewhat slower to converge to the optimal value g^* in the N-H case than in the A-H case. It is to be expected that N-H will compare unfavourably with A-H in systems of relatively modest size, since the A-H algorithm enjoys the luxury of being able to store estimates for Q-factors individually, whereas the N-H algorithm must rely on approximating functions. The major advantage of the N-H algorithm, of course, is that it can be used in systems with state spaces which are much larger than those which A-H would be able to cope with. Also, it should be pointed

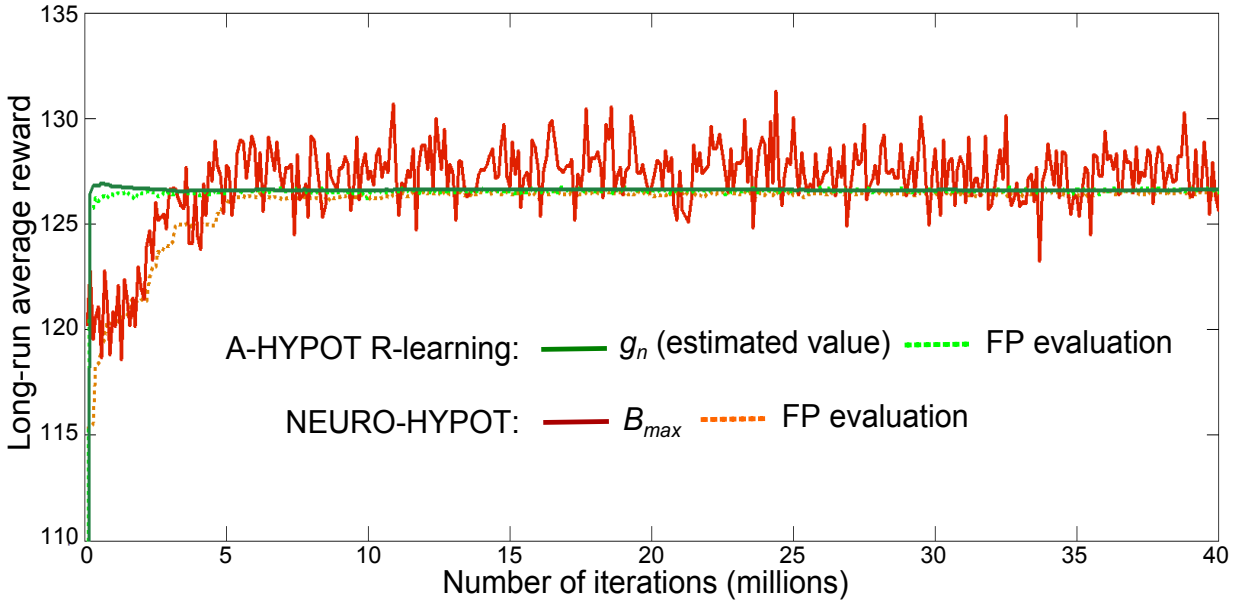


Figure 7.13: Performances of the A-HYPOT and NEURO-HYPOT algorithms in Example 7.4.3.

out that in this particular example, the N-H algorithm appears to perform very well in comparison to the other R-learning algorithms considered earlier in this chapter (see Figure 7.7).

As mentioned earlier, a further objective in this example is to analyse the structural properties of the NEURO-HYPOT policy $\theta^{[R]}$ using the weights in Table 7.5. First, consider the 'containment' property discussed in Section 4.2. It is known that the recurrent state space S_{θ^*} for any average reward optimal policy θ^* has the property $S_{\theta^*} \subseteq \tilde{S}$, where $\tilde{S}$ is the selfish state space. During the evolution of the N-H algorithm, the process was constrained to a finite subset of $\tilde{S}$ defined by thresholds $b_1 = 6$, $b_2 = 6$, $b_3 = 7$ and $b_4 = 7$. As such, the algorithm was not allowed to explore any states outside $\tilde{S}$. Nevertheless, one may still use the weights in Table 7.5 and the quadratic model (7.4.4) to determine decisions for each state in the *infinite* state space $\{(x_1, x_2, x_3, x_4) : x_1, x_2, x_3, x_4 \in \mathbb{N}_0\}$, although there is no guarantee that these decisions will be the same as those which would be obtained by allowing the algorithm to explore a larger finite state space. In the remainder of this example, S will denote the *infinite* state space defined in (3.5.1).

Let $\theta^{[R]}$ now refer to the NEURO-HYPOT policy which chooses an action at any state $\mathbf{x} \in S$ which maximises the estimate $\hat{Q}(\mathbf{x}, a)$ given by (7.4.4), assuming that *all* actions are permitted at every state; that is, $A_{\mathbf{x}} = A = \{0, 1, 2, 3, 4\}$ for every state $\mathbf{x} \in S$. In general, in order to show that the

recurrent region $S_{\theta^{[R]}}$ is contained within some finite set $S^\dagger \subset S$ with thresholds $b_i = \max_{\mathbf{x}\in S^\dagger} x_i$, it is sufficient to show that the estimates $\hat{Q}(\mathbf{x}, a)$ satisfy the following for each i:

$$\max_{\substack{\mathbf{x}\in S^\dagger \\ x_i = b_i}} \Big(\hat{Q}(\mathbf{x}, i) - \hat{Q}(\mathbf{x}, 0) \Big) < 0. \tag{7.4.15}$$

This states that $\theta^{[R]}$ will not prefer joining facility i to balking at any state $\mathbf{x} \in S^\dagger$ with $x_i = b_i$. Somewhat surprisingly, in this particular example the property (7.4.15) fails to hold when the thresholds b_i are set to the selfish thresholds $\tilde{B}_i$. Indeed, for $i = 4$ one finds:

$$\begin{aligned} \hat{Q}(\mathbf{x}, 4) - \hat{Q}(\mathbf{x}, 0) = {} & 40.797 + 0.575x_1 + 0.002x_2 + 0.049x_3 - 0.515x_4 \\ & - 0.075x_1^2 - 0.002x_2^2 - 0.014x_3^2 - 0.178x_4^2. \end{aligned} \tag{7.4.16}$$

The roots of the quadratic equation $40.797 - 0.515x_4 - 0.178x_4^2 = 0$ are (approximately) -16.655 and 13.762. This implies that one can set $x_1 = x_2 = x_3 = 0$ in (7.4.16) and choose any non-negative integer value smaller than 14 for x_4, with the result that $\hat{Q}(\mathbf{x}, 4) - \hat{Q}(\mathbf{x}, 0)$ is positive. In particular, $\hat{Q}((0, 0, 0, 12), 4) - \hat{Q}((0, 0, 0, 12), 0) > 0$, which implies that the policy $\theta^{[R]}$ favours joining facility 4 over balking at the state $(0, 0, 0, 12)$. However, the selfish threshold for facility 4 is $\tilde{B}_4 = 12$, so this provides an indication that $\theta^{[R]}$ may fail to conform to the containment principle for socially optimal policies discussed in earlier chapters. Indeed, further analysis shows that it is possible to identify certain states $\mathbf{x} \in S$ which possess all three of the following properties:

1. $\mathbf{x}$ is positive recurrent under $\theta^{[R]}$;
2. $\hat{Q}(\mathbf{x}, 4) > \hat{Q}(\mathbf{x}, a)$ for all $a \in \{0, 1, 2, 3\}$;
3. $x_4 \geq \tilde{B}_4$.

One such example is the state $\mathbf{x} = (6, 7, 7, 12)$. This shows that, in this particular example, $\theta^{[R]}$ fails to possess the containment property; that is, $S_{\theta^{[R]}}$ is *not* contained in $\tilde{S}$.

The fact that $\theta^{[R]}$ chooses to join facility 4 at certain states $\mathbf{x} \in S$ with $x_4 \geq \tilde{B}_4$ may be regarded as something of a peculiarity in this example, which appears to be a consequence of the parameter values chosen for that particular facility. Indeed, it transpires that the property (7.4.15) holds with threshold values $b_1 = 6$, $b_2 = 8$ and $b_3 = 7$ and $b_4 = 14$. Since these values of b_1, b_2 and b_3 are smaller than $\tilde{B}_1$, $\tilde{B}_2$ and $\tilde{B}_3$ respectively, it follows that under the policy $\theta^{[R]}$, the numbers of

customers present at facilities 1, 2 and 3 remain within the selfish thresholds at all times. Facility 4 has a much faster service rate than the other facilities (see (7.2.13)) and therefore one would think that it would tend to be the most sparsely-populated of the four facilities during the evolution of the NEURO-HYPOT algorithm (although obviously this depends on the decisions chosen at the various states). Hence, it might be the case that the weights $C_4^{(1)}(a)$ and $C_4^{(2)}(a)$ given by the algorithm (for the various actions $a \in A$) are distorted as a result of insufficient exploration of the state space; in other words, the algorithm is unable to 'test' enough different values of the component x_4, and fails to obtain appropriate values for $C_4^{(1)}(a)$ and $C_4^{(2)}(a)$.

On the other hand, in this particular example, the NEURO-HYPOT policy $\theta^{[R]}$ appears to be quite successful in preserving certain monotonicity properties which have been associated with optimal or near-optimal policies in earlier chapters. For example, it can be shown that if $\theta^{[R]}$ chooses to join some facility $i \in \{1,2,3,4\}$ at some state $\mathbf{x} \in S$ with $x_i \geq 1$, then the same action ($a = i$) is also chosen by $\theta^{[R]}$ at the state $\mathbf{x}^{i-}$. The easiest way to show this is to let i be fixed and examine the partial derivatives of $\hat{Q}(\mathbf{x}, i) - \hat{Q}(\mathbf{x}, a)$ (for all actions $a \neq i$) with respect to x_i, implicitly treating the component x_i as a continuous variable. Indeed, in the case $i = 1$, one finds:

$$\begin{aligned}
\frac{\partial}{\partial x_1}\Big(\hat{Q}(\mathbf{x},1) - \hat{Q}(\mathbf{x},0)\Big) &= -4.511 - 9.376x_1, \\
\frac{\partial}{\partial x_1}\Big(\hat{Q}(\mathbf{x},1) - \hat{Q}(\mathbf{x},2)\Big) &= -0.046 - 10.074x_1, \\
\frac{\partial}{\partial x_1}\Big(\hat{Q}(\mathbf{x},1) - \hat{Q}(\mathbf{x},3)\Big) &= -4.961 - 9.184x_1, \\
\frac{\partial}{\partial x_1}\Big(\hat{Q}(\mathbf{x},1) - \hat{Q}(\mathbf{x},4)\Big) &= -5.086 - 9.226x_1. \qquad (7.4.17)
\end{aligned}$$

Since all of these partial derivatives are negative, it follows that if $\hat{Q}(\mathbf{x}, 1) \geq \hat{Q}(\mathbf{x}, a)$ for all actions $a \neq 1$ at some state $\mathbf{x} \in S$ with $x_1 \geq 1$, then $\hat{Q}(\mathbf{x}^{1-}, 1) \geq \hat{Q}(\mathbf{x}^{1-}, a)$ for all $a \neq 1$ also, which establishes the required monotonicity property. One can show that the analogous property also holds for the other facilities $i \in \{2,3,4\}$. The conclusion of this example is that, in a moderate-sized system, the NEURO-HYPOT algorithm is capable of obtaining a policy $\theta^{[R]}$ which comes extremely close to attaining average reward optimality. Furthermore, structural properties such as monotonicity may be preserved by $\theta^{[R]}$, but this is not theoretically guaranteed. ⊠

The next section will present empirical evidence to show that the A-HYPOT and NEURO-HYPOT algorithms are both able to perform consistently well in tests involving randomly-generated param-

eter sets, without requiring an excessively large number of iterations to be performed.

7.5 Numerical results

The A-HYPOT and NEURO-HYPOT algorithms introduced in Sections 7.3 and 7.4 respectively both aim to find near-optimal policies by simulating the continuous-time evolution of the *MRA process*, which is basically a re-formulated MDP in which the state of the system at any given time is the 'physical' state observed by the most recent customer to arrive. The A-HYPOT (henceforth, A-H) algorithm relies on the assumption that it is computationally feasible to store unique values $\hat{Q}(\mathbf{x}, a)$ and $\nu(\mathbf{x}, a)$ for all state-action pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ during the running of the algorithm, whereas the NEURO-HYPOT (henceforth, N-H) algorithm instead relies upon value function approximation. Examples 7.3.2 and 7.4.3 have shown that both algorithms are capable of achieving a strong performance. However, in order to assess their capabilities more thoroughly, it is desirable to carry out an extensive series of tests involving randomly-generated parameters.

This section will report the results of numerical experiments, similar to those which were used in Section 6.3 to evaluate the performances of the Whittle index heuristic and the static routing heuristic. Three different system 'size categories' will be considered, as follows:

- Part 1 of this section will consider systems in which the size of the selfish space $\tilde{S}$ is between 100 and 100,000 states. In these systems, it will be possible to use a DP algorithm to evaluate the optimal average reward g^*, and also compute the average rewards earned by the policies given by the A-H and N-H algorithms. It will also be possible to compare the performances of these policies with those of the *heuristic* policies considered in Chapter 6.

- Part 2 of this section will consider systems in which $|\tilde{S}|$ is between 100,000 and 1 million states. These are intended to represent 'moderate-sized' systems, in which DP algorithms are unacceptably slow (relative to the time required for an RL algorithm to obtain a near-optimal policy), but nevertheless $|\tilde{S}|$ is *not* too large to allow the A-H algorithm (with its reliance upon storing unique values for individual state-action pairs) to be employed.

- Finally, Part 3 of this section will consider systems in which $|\tilde{S}|$ is greater (indeed, possibly much greater) than 1 million. These are intended to represent 'large' systems, in which the

use of *any* algorithm which relies upon using arrays of size $|\tilde{S} \times A_{\mathbf{x}}|$ to store unique values for individual state-action pairs is simply too computationally expensive to be feasible. These systems will therefore permit the use of the N-H algorithm, but not the A-H algorithm. As in the other parts, it will be possible to compare the performance of the policy given by the N-H algorithm with those of the *heuristic* policies discussed in Chapter 6.

The remainder of this section will proceed to present the results obtained from numerical experiments involving more than 80,000 sets of randomly-generated system parameters.

Part 1: 36,136 systems of small-to-moderate size

An experiment was performed in which the A-H and N-H algorithms were applied to 36,136 systems with randomly-generated parameters. In fact, these parameter sets were the same ones that were used in the first part of Section 6.3. This made it possible to use the data collected from earlier experiments to compare the performances of the A-H and N-H algorithms with those of the heuristic policies tested previously. The method for generating the random parameters was already described in Section 6.3. For each individual system, 10 million iterations of the A-H and N-H algorithms were performed; of course, one would expect the performances of these algorithms to improve if they were allowed to run for a larger number of iterations, but for the purposes of this experiment it was necessary to impose a strict limit. As in previous examples, an ϵ-greedy rule was used for action selection, with $\epsilon = 0.15$. The learning parameters were given by $\delta_n(a) = 10/(10 + \nu_n(\mathbf{x}_n, a) - 1)$ and $\zeta_n = 1/(n+1)$ in the A-H case, and $\delta_n(a) = 0.001$ in the N-H case. The 'caps' b_i used by the N-H algorithm were given by the rule (7.4.14) used in Example 7.4.3, with $K = 2$.

In this section, $\theta^{[AH]}$ and $\theta^{[NH]}$ will denote the policies given by the A-H and N-H algorithms respectively after 10 million iterations. Obviously, $\theta^{[AH]}$ and $\theta^{[NH]}$ cannot be derived in a deterministic fashion using the system parameters; they depend on the random simulation of the system. In this respect, they differ from the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$ considered in Chapter 6. As in Section 6.3, let $\rho := \lambda / \sum_{i=1}^{N} c_i \mu_i$ be a measure of the relative traffic intensity for a particular system. Figure 7.14 illustrates the distributions of the percentage sub-optimality for the policies $\theta^{[AH]}$ and $\theta^{[NH]}$ in 'light-demand' systems with $\rho \in (0, 0.5)$. For comparison purposes, the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$ are also represented in the figure. Figures 7.15 and 7.16 show the corresponding findings for the respective cases $\rho \in (0.5, 1)$ and $\rho > 1$.

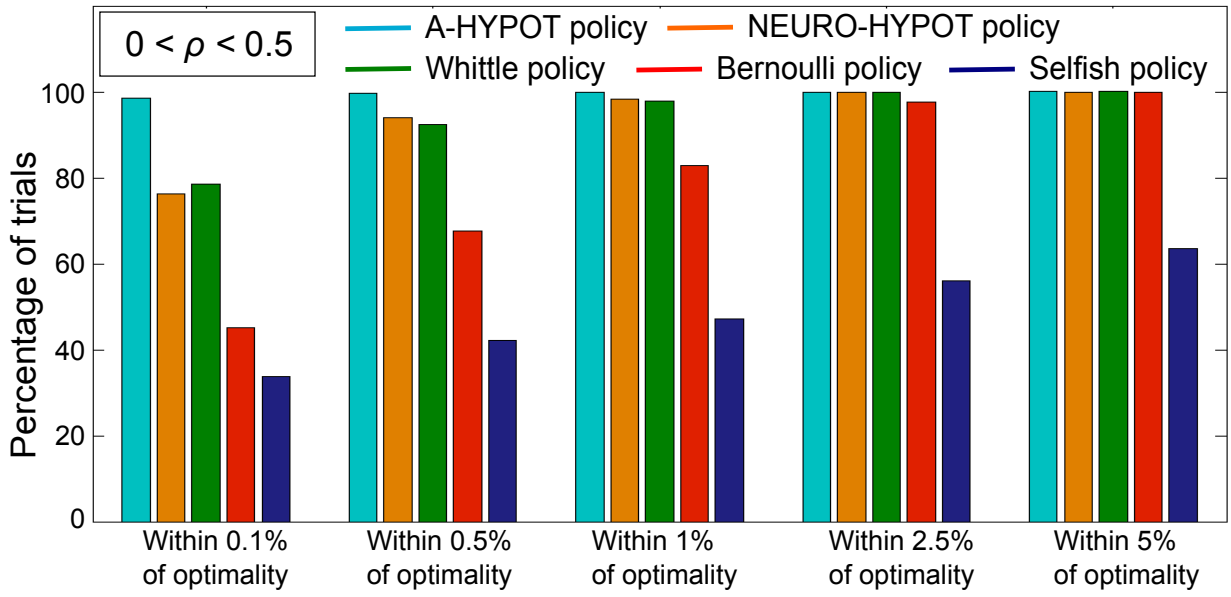


Figure 7.14: The distributions of the percentage sub-optimality for the A-H policy $\theta^{[AH]}$, the N-H policy $\theta^{[NH]}$, the Whittle policy $\theta^{[W]}$, the Bernoulli policy $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$ in systems with $\rho \in (0, 0.5)$.

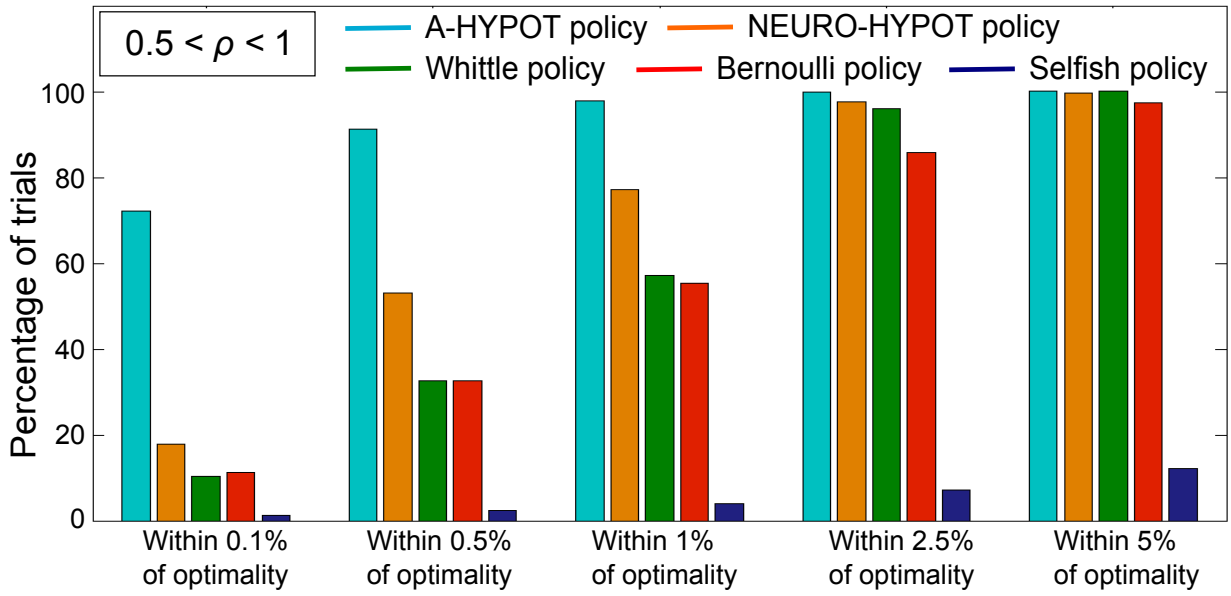


Figure 7.15: The distributions of the percentage sub-optimality for the A-H policy $\theta^{[AH]}$, the N-H policy $\theta^{[NH]}$, the Whittle policy $\theta^{[W]}$, the Bernoulli policy $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$ in systems with $\rho \in (0.5, 1)$.

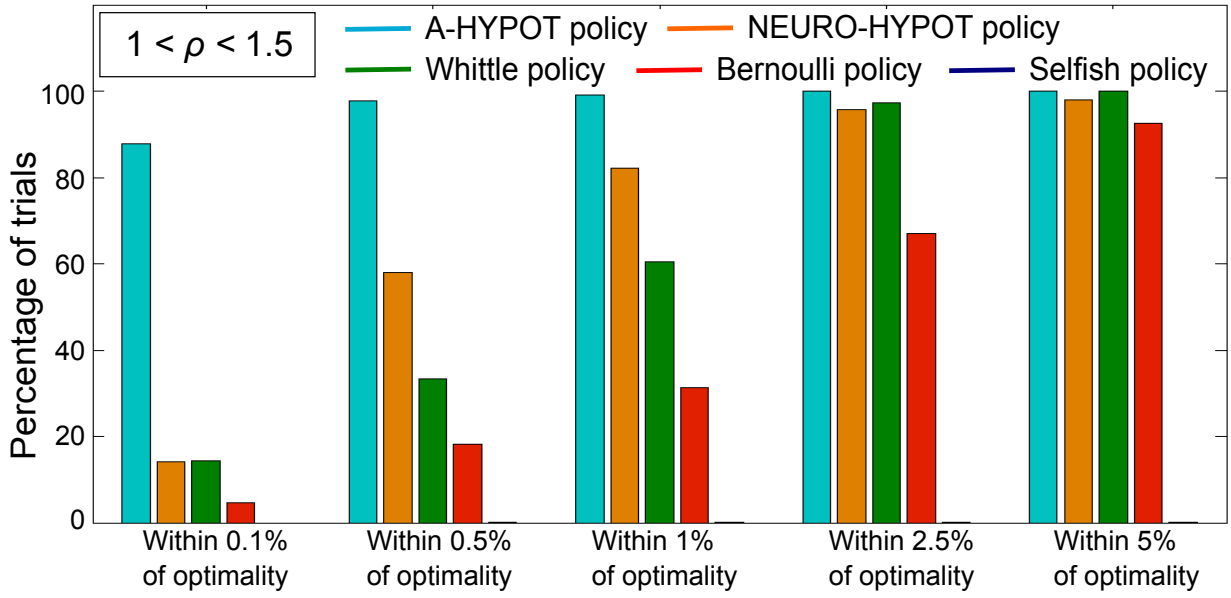


Figure 7.16: The distributions of the percentage sub-optimality for the A-H policy $\theta^{[AH]}$, the N-H policy $\theta^{[NH]}$, the Whittle policy $\theta^{[W]}$, the Bernoulli policy $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$ in systems with $\rho \in (1, 1.5)$.

Figures 7.14-7.16 show that the A-H policy $\theta^{[AH]}$ tends to achieve the strongest performance overall. It appears to be very consistent in earning average rewards within 1% of the optimal value (despite the relatively small number of iterations performed), whereas the N-H policy $\theta^{[NH]}$ and the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$ are somewhat less consistent, especially when ρ is large.

Table 7.6 shows, for both of the RL policies $\theta^{[AH]}$ and $\theta^{[NH]}$, further details of the distribution of the percentage sub-optimality in each of the three ρ categories. The combined results for all three categories are also shown. The corresponding results for the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$ were already given in Section 6.3 (see Table 6.8). For each policy (within each category), Table 7.6 shows the 25^{th}, 50^{th}, 75^{th} and 100^{th} percentiles of the percentage sub-optimality (over all trials performed), and a two-sided 99% confidence interval for the mean.

Although the A-H algorithm appears to perform consistently well in all categories, it is worth noting that in a very small number of cases, the N-H policy $\theta^{[NH]}$ appears to perform extremely poorly; indeed, its maximum sub-optimality over all of the 36,136 trials is approximately 75%. It is difficult to explain why the N-H algorithm is prone to catastrophic failures in a (very small) number of cases, but one possible theory is that the algorithm is prone to volatility due to the fact that the ANN weights which are updated on each iteration affect the Q-factor estimates at *all*

All demand rates					
Policy	Lower quartile	Median	Upper quartile	Maximum	99% C.I. for Mean
$\theta^{[AH]}$	0.000	0.007	0.043	3.509	0.069 ± 0.003
$\theta^{[NH]}$	0.029	0.243	0.637	74.838	0.567 ± 0.024

Light demand ($\rho \in (0, 0.5)$)					
Policy	Lower quartile	Median	Upper quartile	Maximum	99% C.I. for Mean
$\theta^{[AH]}$	0.000	0.001	0.004	3.422	0.012 ± 0.001
$\theta^{[NH]}$	0.001	0.011	0.091	5.516	0.111 ± 0.004

Medium demand ($\rho \in (0.5, 1)$)					
Policy	Lower quartile	Median	Upper quartile	Maximum	99% C.I. for Mean
$\theta^{[AH]}$	0.009	0.035	0.120	3.509	0.140 ± 0.004
$\theta^{[NH]}$	0.174	0.455	0.935	36.207	0.697 ± 0.016

High demand ($\rho \in (1, 1.5)$)					
Policy	Lower quartile	Median	Upper quartile	Maximum	99% C.I. for Mean
$\theta^{[AH]}$	0.000	0.011	0.048	2.268	0.055 ± 0.002
$\theta^{[NH]}$	0.186	0.409	0.792	74.838	0.893 ± 0.037

Table 7.6: The distribution of the percentage sub-optimality under the two RL policies $\theta^{[AH]}$ and $\theta^{[NH]}$, with results categorised according to the value of ρ. All figures are rounded to 3 decimal places.

system states. Occasionally, the weights may be updated in such a way that the algorithm's latest guess for an optimal policy changes dramatically and also disastrously. If the algorithm happens to be terminated at the precise moment that one of these 'dips' occurs, then the resulting policy will be poor. However, these 'dips' (if they occur) are likely to be corrected by the algorithm almost immediately. Therefore, in practice, if one is monitoring the performance of the algorithm (by performing regular FP evaluations, for example) then one should be able to ensure that the final policy obtained achieves an acceptable performance; in other words, it should be possible to ignore any weak policies which are caused by sudden dips in the algorithm's performance.

Table 7.7 shows pairwise comparisons between the RL policies $\theta^{[AH]}$ and $\theta^{[NH]}$, the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$ and the selfish policy $\tilde{\theta}$, similar to the comparisons presented in Section 6.3. Results are shown for each of the three categories $\rho \in (0, 0.5)$, $\rho \in (0.5, 1)$ and $\rho \in (1, 1.5)$, with an 'all demand rates' category also included, and each figure in the table shows the percentage of trials (within the relevant ρ category) in which the policy in the corresponding row achieved a strictly greater average reward than the policy in the corresponding column. The results show that, as expected, the A-H policy (which enjoys the advantage of being able to store Q-factor estimates explicitly) consistently out-performs the N-H policy over all categories, and it also achieves strong performances against both of the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$. There are some minor variations to note between the various categories; for example, the Whittle policy achieves quite a strong performance against both of the RL policies in the 'light demand' category (and even appears to out-perform the N-H policy), but fares substantially worse in the other categories.

Part 2: 26,277 systems of moderate size

A further experiment was performed, involving a further 26,277 sets of randomly-generated parameters. These parameters were generated in a similar way to those in Part 1, except that in this case, parameter sets were accepted only if they yielded a value of $|\tilde{S}|$ between 100,000 and 1 million states. As explained earlier, these systems were intended to represent 'moderate-sized' systems, in which one would not be able to compute an optimal policy efficiently using a DP algorithm, but the use of an RL algorithm involving the storage of Q-factor estimates for individual state-action pairs *would* be considered feasible. For each of the 26,277 systems tested, estimates of the average rewards earned by the RL policies $\theta^{[AH]}$ and $\theta^{[NH]}$, the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$ and the

All demand rates					
	$\theta^{[AH]}$	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[AH]}$	x	88.32	79.47	90.32	93.94
$\theta^{[NH]}$	8.58	x	55.80	72.37	91.21
$\theta^{[W]}$	19.55	42.19	x	63.82	93.77
$\theta^{[B]}$	8.58	25.30	31.61	x	91.32
$\tilde{\theta}$	4.98	6.57	3.07	6.07	x

Light demand ($\rho \in (0, 0.5)$)					
	$\theta^{[AH]}$	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[AH]}$	x	75.31	52.66	79.43	82.74
$\theta^{[NH]}$	20.41	x	29.23	63.29	75.48
$\theta^{[W]}$	44.42	64.76	x	73.25	84.19
$\theta^{[B]}$	17.36	29.91	14.85	x	76.02
$\tilde{\theta}$	14.02	17.87	6.37	16.16	x

Medium demand ($\rho \in (0.5, 1)$)					
	$\theta^{[AH]}$	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[AH]}$	x	94.28	94.46	93.18	99.11
$\theta^{[NH]}$	3.47	x	71.66	71.90	98.19
$\theta^{[W]}$	5.54	28.34	x	49.04	97.14
$\theta^{[B]}$	6.77	27.91	49.80	x	97.97
$\tilde{\theta}$	0.89	1.81	2.86	2.03	x

High demand ($\rho \in (1, 1.5)$)					
	$\theta^{[AH]}$	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[AH]}$	x	95.39	91.38	98.35	99.99
$\theta^{[NH]}$	1.84	x	66.60	81.90	99.98
$\theta^{[W]}$	8.62	33.40	x	69.03	99.99
$\theta^{[B]}$	1.63	18.10	30.34	x	99.99
$\tilde{\theta}$	0.01	0.02	0.00	0.01	x

Table 7.7: Pairwise comparisons between $\theta^{[AH]}$, $\theta^{[NH]}$, $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$. Each figure shows the percentage of trials in which the policy in the corresponding row out-performed the policy in the corresponding column.

selfish policy $\tilde{\theta}$ were obtained using simulation. The A-H and N-H algorithms were allowed to run for 10 million iterations each (as in Part 1), and the other parameters used by these algorithms were defined in the same way as in Part 1. All policies were evaluated by simulating 10 million random transitions (including 'dummy' transitions caused by uniformisation), with the same random number seed used in all cases and the same discrete-time step size used for each policy.

Pairwise comparisons between the five policies $\theta^{[AH]}$, $\theta^{[NH]}$, $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$ with respect to the performances achieved in these additional 26,277 tests are shown in Table 7.8. As in Part 1, the A-H policy $\theta^{[AH]}$ appears to achieve the strongest performance overall. In fact, its relative performance in comparison to the other four policies actually appears to be stronger than in Part 1 (the 'small systems' case). The N-H policy $\theta^{[NH]}$ appears to be slightly less consistent than the A-H policy, but nevertheless compares favourably with the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$.

All demand rates					
	$\theta^{[AH]}$	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[AH]}$	x	96.39	92.42	97.47	97.11
$\theta^{[NH]}$	3.61	x	65.70	72.92	92.78
$\theta^{[W]}$	7.22	34.30	x	57.04	93.50
$\theta^{[B]}$	2.17	27.08	41.88	x	90.61
$\tilde{\theta}$	2.17	7.22	3.97	8.30	x

Light demand ($\rho \in (0, 0.5)$)					
	$\theta^{[AH]}$	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[AH]}$	x	84.10	74.88	88.28	86.98
$\theta^{[NH]}$	12.72	x	43.56	74.57	77.02
$\theta^{[W]}$	19.07	53.34	x	71.40	82.38
$\theta^{[B]}$	6.75	22.49	19.79	x	72.57
$\tilde{\theta}$	7.31	19.36	8.92	21.58	x

Medium demand ($\rho \in (0.5, 1)$)					
	$\theta^{[AH]}$	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[AH]}$	x	99.58	99.87	96.89	99.95
$\theta^{[NH]}$	0.42	x	86.32	45.17	99.75
$\theta^{[W]}$	0.12	13.68	x	20.53	99.20
$\theta^{[B]}$	3.11	54.83	79.47	x	99.56
$\tilde{\theta}$	0.00	0.25	0.80	0.44	x

High demand ($\rho \in (1, 1.5)$)					
	$\theta^{[AH]}$	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[AH]}$	x	96.40	95.18	99.54	100.00
$\theta^{[NH]}$	3.60	x	78.69	89.31	100.00
$\theta^{[W]}$	4.82	21.31	x	72.81	100.00
$\theta^{[B]}$	0.46	10.69	27.19	x	100.00
$\tilde{\theta}$	0.00	0.00	0.00	0.00	x

Table 7.8: Pairwise comparisons between $\theta^{[AH]}$, $\theta^{[NH]}$, $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$ in 'moderate-sized' systems.

Part 3: 24,074 systems of large size

Finally, a further experiment was performed involving the application of the N-H algorithm to a further 24,074 systems with randomly-generated parameters. These parameter sets were, in fact, identical to the ones used in Part 2 of Section 6.3. In each of the systems tested, the number of facilities N was between 8 and 12, and the size of the selfish state space $\tilde{S}$ was at least 1 million states. Indeed, the median of $|\tilde{S}|$ over all trials performed was approximately 7.3×10^6, and the maximum (which occurred in a 12-facility system) was 6.08×10^{22}. The A-H algorithm was not considered in these trials, due to its requirement for explicit values to be stored for individual state-action pairs. Thus, the purpose of the experiment was to compare the results achieved by the N-H algorithm with those of the policies $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$ considered in Section 6.3. As in Part 2, the average rewards of these policies were estimated using discrete-time simulation.

Table 7.9 shows that, once again, the N-H policy is able to demonstrate a strong performance in

comparison to the other policies, especially in the 'medium demand' and 'high demand' categories. In fact, its relative performance against $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$ actually appears to be stronger than in Part 1 (the 'small systems' case) and Part 2 (the 'moderate-sized systems' case).

All demand rates				
	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[NH]}$	x	72.72	77.25	93.43
$\theta^{[W]}$	25.18	x	58.76	93.75
$\theta^{[B]}$	20.71	37.88	x	92.53
$\tilde{\theta}$	6.51	6.18	7.42	x

Light demand ($\rho \in (0, 0.5)$)				
	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[NH]}$	x	52.04	68.82	80.59
$\theta^{[W]}$	41.63	x	57.85	81.70
$\theta^{[B]}$	25.04	32.07	x	77.87
$\tilde{\theta}$	19.24	18.10	21.99	x

Medium demand ($\rho \in (0.5, 1)$)				
	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[NH]}$	x	91.54	65.11	99.79
$\theta^{[W]}$	8.46	x	32.76	99.64
$\theta^{[B]}$	34.89	67.24	x	99.80
$\tilde{\theta}$	0.21	0.36	0.20	x

High demand ($\rho \in (1, 1.5)$)				
	$\theta^{[NH]}$	$\theta^{[W]}$	$\theta^{[B]}$	$\tilde{\theta}$
$\theta^{[NH]}$	x	74.41	98.01	99.89
$\theta^{[W]}$	25.59	x	85.99	99.89
$\theta^{[B]}$	1.99	14.01	x	99.89
$\tilde{\theta}$	0.11	0.11	0.11	x

Table 7.9: Pairwise comparisons between $\theta^{[NH]}$, $\theta^{[W]}$, $\theta^{[B]}$ and $\tilde{\theta}$ in 'large-sized' systems.

In summary, this section has shown that both of the RL policies $\theta^{[AH]}$ and $\theta^{[NH]}$ are able to perform consistently well in comparison to the heuristic policies considered in Chapter 6, even when the number of RL iterations is restricted to a relatively modest number. As expected, the A-H algorithm demonstrates greater consistency than the N-H algorithm in systems where it can feasibly be applied. The N-H algorithm is prone to unpredictable behaviour in some cases, but the evidence of this section suggests that it can usually be relied upon to surpass the performances of the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$, provided that it is monitored in a sensible way.

7.6 Extension: Non-exponential distributions

The A-HYPOT and NEURO-HYPOT algorithms discussed in the previous sections both depend upon a re-formulated MDP, referred to in Section 7.3 as the *MRA process*, in which the state of

the system at any given time is the 'physical' state $\mathbf{x} \in S$ observed by the latest customer to arrive. Essentially, transitions of the MRA process occur only at customers' arrival times, and the single-step transition probabilities (which are not explicitly needed by RL algorithms, due to their reliance on *simulation* of the system's random transitions) must take into account all possibilities for the number of service completions occurring in between two customer arrivals.

In the MRA process, rewards are given by customers' *expected* net rewards when they enter the system, and therefore the exact points in time at which service completions occur are inconsequential; it is only important to determine (for each facility) the number of service completions that occur before the next arrival. As such, from a mathematical point of view, there is no loss of accuracy involved in assuming that service completions always occur at decision epochs (and not in between decision epochs), provided that the transition probabilities remain unaffected. One might imagine, for instance, that customers who have completed service are then held in another (costless) waiting area and are not physically ejected from the system until the next customer arrives. By adopting this perspective, it is easier to understand how the MRA process evolves as an MDP.

Throughout this thesis it has been assumed that customers arrive via a Poisson process, which ensures that the MRA process can be formulated as either a CTMDP or an MDP. However, since the transitions of the MRA process are associated only with arrivals (and *not* service completions), it is possible to obtain a *Semi-Markov Decision Process* (SMDP) by simply altering the problem formulation so that inter-arrival times follow an *arbitrary* probability distribution, as opposed to an exponential distribution. SMDPs were briefly discussed in Section 3.2; however, it will be appropriate to provide a proper definition before proceeding further with this section.

Definition 7.6.1. ***(Semi-Markov Decision Process)***

A Semi-Markov Decision Process (SMDP) is a collection of objects:

$$\Psi = \Big\{\mathcal{S}, \mathcal{A}_{\mathbf{x}}, F(\mathbf{x}, a), \sigma(\mathbf{x}, a, \mathbf{y}), \xi(\mathbf{x}, a), \xi_{\gamma}(\mathbf{x}, a), \gamma\Big\},$$

where $\mathcal{S}$ is the state space of the process (assumed either finite or countably infinite), $\mathcal{A}_{\mathbf{x}}$ is the set of actions available at state $\mathbf{x} \in \mathcal{S}$ (also assumed either finite or countably finite), $F_{\mathbf{x}}(t, a)$ is the cumulative distribution function for the amount of time t spent in state $\mathbf{x} \in \mathcal{S}$ given that action $a \in \mathcal{A}_{\mathbf{x}}$ is chosen, $\sigma(\mathbf{x}, a, \mathbf{y})$ is the probability that the process transfers from state $\mathbf{x} \in \mathcal{S}$ to $\mathbf{y} \in \mathcal{S}$

at its next transition given that action $a \in \mathcal{A}_{\mathbf{x}}$ is chosen, $\xi(\mathbf{x}, a)$ and $\xi_\gamma(\mathbf{x}, a)$ are (respectively) the undiscounted and discounted net rewards earned for choosing action $a \in \mathcal{A}_{\mathbf{x}}$ at state $\mathbf{x} \in \mathcal{S}$, and $\gamma \in (0, \infty)$ is a continuous discount rate applied to rewards earned in the future.

As discussed previously, a CTMDP is merely a special case of an SMDP. If the sojourn times for the process are all exponentially distributed, then the SMDP is a CTMDP.

When the queueing system formulation of Section 3.1 is modified so that inter-arrival times follow an arbitrary distribution, it remains the case that the arrival times of customers form a set of *regeneration points* for the process; in other words, the state of the system when a customer arrives provides, in conjunction with the action chosen, all information which is needed for calculating the probability distribution of the system state observed by the *next* customer. Results from the MDP literature (see [48, 157]) imply the existence of the following set of optimality equations for the SMDP obtained by allowing inter-arrival times to follow a general distribution:

$$h(\mathbf{x}) = \max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}, a) - g^* E[t(\mathbf{x}, a)] + \sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y}) h(\mathbf{y}) \right\} \quad \forall\, \mathbf{x} \in S, \tag{7.6.1}$$

where $t(\mathbf{x}, a)$ denotes the length of the sojourn in state $\mathbf{x}$ given that action $a \in A_{\mathbf{x}}$ is chosen, and the transition probabilities $p(\mathbf{x}, a, \mathbf{y})$ are associated with the MRA process. Note that the CTMDP formulation Ψ given in Section 3.2 would *not* remain valid under the assumption that inter-arrival times followed an arbitrary distribution, since in that case the decision epochs associated with service completions would not necessarily be regenerative; indeed, the sojourn times and transition probability distributions would depend upon the amount of time elapsed since the last customer arrival. This illustrates an important advantage of adopting the MRA formulation.

Throughout this section, attention will be restricted to the SMDP associated with the optimality equations (7.6.1), under the assumption that inter-arrival times follow an arbitrary distribution. The case of non-exponential *service-time* distributions will be discussed later. It is very easy to modify the A-HYPOT and NEURO-HYPOT algorithms presented on pages 323 and 348 respectively so that they can be used in the more general SMDP case. Indeed, both of these algorithms simply use inverse transform sampling to simulate inter-arrival times, and this technique can easily be applied to non-exponential distributions. As such, one can use these algorithms in exactly the

same way as in the case of Poisson arrivals; all that is required is for the pseudo-random numbers which determine inter-arrival times to be sampled from an alternative distribution.

Unfortunately, it is not easy to assess whether or not the A-HYPOT and NEURO-HYPOT algorithms succeed in finding near-optimal policies in the SMDP case, for the simple reason that it is too difficult to compute the optimal average reward g^* in most problem instances. Although it is theoretically possible to use dynamic programming algorithms such as value iteration to find optimal policies for SMDPs (after making a truncation of the infinite state space; see, for example, [48]), the SMDP being considered in this section is used to model the MRA process, in which the number of possible transitions from a state $\mathbf{x} \in \tilde{S}$ may be comparable in magnitude to the size of the finite state space $\tilde{S}$ itself. From a practical perspective, this rules out the use of dynamic programming algorithms. Another (related) problem is that, without the use of DP algorithms, it is not feasible to evaluate the performance of any given stationary policy exactly.

The approach in this section will be to compare the policies found by the A-HYPOT and NEURO-HYPOT algorithms with the policy found by the *Whittle index heuristic* discussed in Section 6.1. The Whittle heuristic can be adapted to the SMDP problem, although the computations required in order to compute steady-state probabilities and the Whittle indices themselves become very complicated when multiple service channels are allowed at the various facilities (this will be explained further below); for this reason, most of the examples given in this section will assume that each facility has only a single server available. The average rewards earned by the Whittle, A-HYPOT and NEURO-HYPOT policies will then be estimated via simulation, with multiple replicate trials performed in order to ensure that the results obtained are reliable.

Before proceeding with an example, it will be appropriate to discuss how the Whittle indices are obtained in the SMDP case. As in Section 6.1, one considers a *Lagrangian relaxation* of the original N-facility routing problem, in which the objective is to maximise the function:

$$g_\theta(W) := \sum_{i=1}^{N} \left(\alpha_i \eta_i(\theta) - \beta_i L_i(\theta) \right) + W \left(\lambda - \sum_{i=1}^{N} \eta_i(\theta) \right), \tag{7.6.2}$$

where $\eta_i(\theta)$ and $L_i(\theta)$ are (respectively) the effective queue-joining rate and mean number of customers present at facility $i \in \{1, 2, ..., N\}$ under policy θ, and W is a Lagrange multiplier which has an economic interpretation as a charge for admitting a customer. The maximisation is carried out

over all policies θ which belong to an expanded class of admissible policies Θ' with the ability to admit a newly-arrived customer to any *subset* of the set of facilities $\{1, 2, ..., N\}$. As in Section 6.1, one then obtains a *facility-wise decomposition* of the Lagrangian problem, in which the objective is to determine (for each facility i) the policy θ_i which will maximise the function:

$$g_{\theta_i}(W) = (\alpha_i - W)\,\eta_i(\theta_i) - \beta_i L_i(\theta_i), \tag{7.6.3}$$

and θ_i simply chooses between two permissible actions ('join' and 'balk') each time a new customer arrives. This single-facility problem is essentially the same as the analogous problem considered in Section 6.1, except that customers no longer arrive via a Poisson process. In this case, one considers a $GI/M/c$ queue (where GI stands for 'General Independent') as opposed to an $M/M/c$ queue. It will be assumed that there exists an optimal *threshold* policy (see Definition 5.1.3) for this $GI/M/c$ problem; indeed, this has been proved by Yechiali [204]. Furthermore, the fact that the facilities $i \in \{1, 2, ..., N\}$ are *indexable* (see Definition 6.1.1) may be established using an argument similar to that given in the proof of Theorem 6.1.2. Indeed, let $g_i(T, W)$ denote the value of $g_{\theta_i}(W)$ (as defined in (7.6.3)) given that θ_i is a threshold policy with threshold $T \in \mathbb{N}$. Then:

$$g_i(T, W) = \lambda(1 - \hat{\pi}_i(T, T))(\alpha_i - W) - \beta_i L_i(\theta_i), \tag{7.6.4}$$

where $\hat{\pi}_i(y, T)$ denotes the probability that a customer arrives under state $y \in \mathbb{N}_0$, given that a threshold T is applied. At this stage, an important point must be made. When systems with Poisson arrival rates were considered in earlier sections, it was always the case that the probability distribution for the system state observed by a newly-arrived customer was equivalent to the probability distribution for the system state at an arbitrary point in time; this was due to the PASTA (Poisson Arrivals See Time Averages) property discussed in Section 7.3. Unfortunately, however, the PASTA property does *not* hold when inter-arrival times follow an arbitrary distribution. Indeed, let $\{\hat{\pi}_i(x, T)\}_{x \in \mathbb{N}_0}$ denote the probability distribution for the single-facility system state observed at an arrival instant (this will be referred to as the *arrival distribution*), and let $\{\pi_i(x, T)\}_{x \in \mathbb{N}_0}$ denote the probability distribution for the system state at an arbitrary point in time (referred to as the *time-average distribution*), where in both cases a threshold T is applied. Then it is generally *not* the case that $\hat{\pi}_i(x, T) = \pi_i(x, T)$ for recurrent states $x \leq T$.

Nevertheless, the right-hand side of (7.6.4) is indeed a valid expression for the average reward $g_i(T, W)$ under threshold T, since $(1 - \hat{\pi}_i(T, T))$ is the probability that an arriving customer will

observe the system in some state $x < T$ (and hence join the queue). It follows from (7.6.4) that $g_i(T, W)$ is linearly decreasing with W, with gradient $-\lambda(1 - \hat{\pi}_i(T, T))$. Let $g_i^*(W)$ denote the optimal average reward for facility i given an admission charge W, and let $T_i^*(W)$ denote the corresponding optimal threshold. As observed by Glazebrook et. al. [59], $g_i^*(W)$ is the upper envelope of a collection of functions that are linear in W, and therefore $g_i^*(W)$ must be *convex* in W, which in turn implies that the gradient $-\lambda(1 - \hat{\pi}_i(T_i^*(W), T_i^*(W)))$ must be increasing with W. However, it can easily be shown that the probability $\hat{\pi}_i(T, T)$ is decreasing with T (this can be established using a continuous-time sample path argument; full details will not be given here, but it is sufficient to consider two finite-capacity $GI/M/c$ queues with thresholds T and $T + 1$ respectively and show, under the assumption that both queues follow the same sample path, that a customer arriving at the second queue cannot observe the state $T + 1$ unless a customer arriving at the first queue simultaneously observes the state T). It then follows that the threshold $T_i^*(W)$ must be decreasing with W, which establishes the required indexability property.

However, the formula (6.1.13) given in Section 6.1 for the Whittle indices $W_i(x)$ is not applicable in the SMDP case, since it was derived under the implicit assumption of an equivalence between the arrival distribution and the time-average distribution. Although the equation (7.6.4) is valid, it would be incorrect to express the expected number of customers present $L_i(\theta_i)$ in the form $\sum_{x=0}^{T} x\hat{\pi}_i(x, T)$, since the time-average distribution (rather than the arrival distribution) would be needed in order to compute this expected value. Therefore it is apparent that the right-hand side of (7.6.4) depends on both the arrival and time-average distributions. It will be useful to find an alternative expression for $g_i(T, W)$ which depends on only one of these distributions.

It transpires that the arrival distribution $\{\hat{\pi}_i(x, T)\}_{x \in \mathbb{N}_0}$ is somewhat easier to obtain than the time-average distribution, and so it will be convenient to consider only the arrival distribution. By again relying upon the strategy of calculating customers' *expected* net rewards when they enter the system, one can write $g_i(T, W)$ in a form which does not depend upon $L_i(\theta_i)$:

$$g_i(T, W) = \begin{cases} (1 - \hat{\pi}_i(T, T)) \left(\alpha_i - \dfrac{\beta_i}{\mu_i} \right), & \text{if } T \leq c_i, \\ \displaystyle\sum_{x=0}^{c_i - 1} \hat{\pi}_i(x, T) \left(\alpha_i - \frac{\beta_i}{\mu_i} \right) + \sum_{x=c_i}^{T-1} \hat{\pi}_i(x, T) \left(\alpha_i - \frac{\beta_i(x+1)}{c_i \mu_i} \right), & \text{if } T > c_i. \end{cases}$$

As mentioned earlier, the calculation of the arrival probabilities $\hat{\pi}_i(x, T)$ for a finite-capacity

$GI/M/c$ queue is an extremely laborious task in general (see [86]), but it is not especially difficult in the special case $c = 1$. Therefore the derivation of the Whittle indices will proceed under the assumption of a single-server queue, i.e. a $GI/M/1/T$ queue. In this case:

$$g_i(T, W) = \sum_{x=0}^{T-1} \hat{\pi}_i(x, T) \left(\alpha_i - \frac{\beta_i(x+1)}{\mu_i} \right). \tag{7.6.5}$$

As in Section 6.1, one may then obtain an expression for the Whittle index $W_i(x)$ by equating the expected average rewards under the thresholds x and $x+1$. The resulting formula, valid for states $x \in \mathbb{N}_0$ and given in terms of the arrival probabilities $\hat{\pi}_i(x, T)$, is as follows:

$$W_i(x) = \frac{\sum_{y=0}^{x} \hat{\pi}_i(y, x+1) \left(\alpha_i - \beta_i(y+1)/\mu_i\right) - \sum_{y=0}^{x-1} \hat{\pi}_i(y, x) \left(\alpha_i - \beta_i(y+1)/\mu_i\right)}{\sum_{y=0}^{x} \hat{\pi}_i(y, x+1) - \sum_{y=0}^{x-1} \hat{\pi}_i(y, x)}. \tag{7.6.6}$$

In the case of Poisson arrivals, the fact that (7.6.4) and (7.6.5) are both valid expressions for the average reward $g_i(T, W)$ ensures that the formula for $W_i(x)$ given in (7.6.6) is exactly equivalent to the formula (6.1.13) given in Section 6.1. Thus, in order to compute the Whittle indices $W_i(x)$, all that remains is to compute the arrival probabilities $\hat{\pi}_i(x, T)$ for thresholds $T \in \mathbb{N}$. In a $GI/M/1/T$ queue, this can be done by considering the *embedded Markov chain* formed by the points in time at which customers arrive (see, for example, [67], p. 248). This is defined below.

Definition 7.6.2. ***(Embedded Markov chain)***

Let Ψ be an SMDP operating under an arbitrary policy θ, and let $(\mathbf{x}_0, \mathbf{x}_1, ...)$ be the sequence of states visited under θ. Then $(\mathbf{x}_0, \mathbf{x}_1, ...)$ is referred to as an embedded Markov chain.

Let $f(t)$ denote the probability density function of the inter-arrival time distribution (assumed arbitrary at this point) and let $b_i : \mathbb{N}_0 \to \mathbb{R}$ be defined for integers $n \geq 0$ as follows:

$$b_i(n) := \int_0^{\infty} \frac{e^{-\mu_i t}(\mu_i t)^n}{n!} f(t) dt. \tag{7.6.7}$$

Thus, $b_i(n)$ may be interpreted as the probability that n service completions take place in between two consecutive customer arrivals in the single-facility problem under consideration (with exponential service rate μ_i and threshold T), assuming that the facility serves a continuous stream of customers and is never idle. It is also possible to interpet (7.6.7) as a mixture of Poisson probabilities; see, for example, [52]. It follows that if a customer arrives under some state $x \in \mathbb{N}_0$, then the

probability that the *next* customer to arrive observes state $y \in \mathbb{N}_0$ is given by:

$$\begin{cases} b_i(z-y), & \text{if } 1 \leq y \leq z, \\ 1 - \sum_{n=0}^{z-1} b_i(n), & \text{if } y = 0, \\ 0, & \text{otherwise,} \end{cases}$$

where $z = \min(x+1, T)$ is the state immediately after the first customer's arrival. Hence, the embedded Markov chain has a single-step transition probability matrix of the form:

$$P_i = \begin{pmatrix} 1-b_i(0) & b_i(0) & 0 & 0 & ... & 0 \\ 1-\sum_{n=0}^{1} b_i(n) & b_i(1) & b_i(0) & 0 & ... & 0 \\ 1-\sum_{n=0}^{2} b_i(n) & b_i(2) & b_i(1) & b_i(0) & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1-\sum_{n=0}^{T-1} b_i(n) & b_i(T-1) & b_i(T-2) & b_i(T-3) & ... & b_i(0) \\ 1-\sum_{n=0}^{T-1} b_i(n) & b_i(T-1) & b_i(T-2) & b_i(T-3) & ... & b_i(0) \end{pmatrix}$$

where (for clarity) the element in row x, column y of P_i is the probability that two consecutively-arriving customers observe states $x-1$ and $y-1$ respectively, and the matrix P_i is of order $T+1$. The arrival probabilities $\hat{\pi}_i(x,T)$ for $x \in \{0,1,...,T\}$ can then be found by solving the system of linear equations $\hat{\boldsymbol{\pi}}_i P_i = \hat{\boldsymbol{\pi}}_i$, where $\hat{\boldsymbol{\pi}}_i$ is the row vector of probabilities $(\hat{\pi}_i(0,T), \hat{\pi}_i(1,T), ..., \hat{\pi}_i(T,T))$, subject to the usual normalisation condition $\sum_{x=0}^{T} \hat{\pi}_i(x,T) = 1$. The Whittle indices are then obtained using (7.6.6), and (as in Section 6.1) the Whittle heuristic policy is defined as the policy which sends a customer arriving under state $\mathbf{x} \in S$ to a facility $i \in \{1,2,...,N\}$ which maximises $W_i(x_i)$, unless all of these indices are negative at $\mathbf{x}$, in which case balking is chosen.

Of course, the transition probabilities $b_i(n)$ are dependent upon $f(t)$, the density function for the arrival-time distribution. A computer program which calculates the Whittle indices will therefore need to perform integration in order to find these probabilities. The next example compares the

performances of the policies found using the A-HYPOT and NEURO-HYPOT algorithms with that of the Whittle heuristic policy under various different arrival-time distributions.

Example 7.6.3. ***(Alternative distributions for inter-arrival times)***

Consider a system with 5 single-server facilities, with the following parameters:

$$\begin{array}{llll} c_1 = 1, & \mu_1 = 1, & \beta_1 = 4, & \alpha_1 = 60, \\ c_2 = 1, & \mu_2 = 4, & \beta_2 = 5, & \alpha_2 = 20, \\ c_3 = 1, & \mu_3 = 8, & \beta_3 = 8, & \alpha_3 = 12, \\ c_4 = 1, & \mu_4 = 10, & \beta_4 = 7, & \alpha_4 = 9, \\ c_5 = 1, & \mu_5 = 16, & \beta_5 = 5, & \alpha_5 = 4. \end{array}$$

With these parameters, the selfish state space $\tilde{S}$ (which can be found independently of the inter-arrival distribution) consists of approximately 560,000 states. In this example, the A-HYPOT, NEURO-HYPOT and Whittle heuristic policies will be compared under various different inter-arrival distributions. Erlang distributions will be considered first, followed by constant and lognormal distributions. In each case, the two RL algorithms will be calibrated as in previous examples. Specifically, an ϵ-greedy rule with $\epsilon = 0.15$ will be used for action selection, the A-HYPOT algorithm will use learning parameters $\delta_n(a)$ given by the rule (7.3.18) (with $T = 10$) and secondary parameters $\zeta_n = 1/(n+1)$, and the NEURO-HYPOT algorithm will use a constant rule $\delta_n = 0.001$. Also, as in Example 7.4.3, the NEURO-HYPOT algorithm will use thresholds b_i given by the rule (7.4.14), with $K = 2$ and the Whittle indices $W_i(x)$ given by the new formula (7.6.6).

Case 1: Erlang distributions

Suppose inter-arrival times follow an Erlang distribution with mean $1/\lambda$ and shape parameter $k \in \mathbb{N}$ (also referred to as the number of *phases*; see [67], p. 127). The density function is:

$$f(t) = \frac{(k\lambda)(k\lambda t)^{k-1} e^{-k\lambda t}}{(k-1)!}, \qquad t \geq 0.$$

The case $k = 1$ represents an exponential distribution. An arrival rate $\lambda = 15$ will be used in this example, and values of k ranging from 2 to 5 will be considered. Figures 7.17-7.18 show the results of four experiments in which 50 million iterations of the A-HYPOT (A-H) and NEURO-HYPOT (N-H) algorithms were performed for each of the values $k \in \{2, 3, 4, 5\}$. For each value of k, the

(estimated) average reward attained by the Whittle heuristic policy is also shown. These average rewards were estimated by simulation, with 10 replicate trials performed in each case.

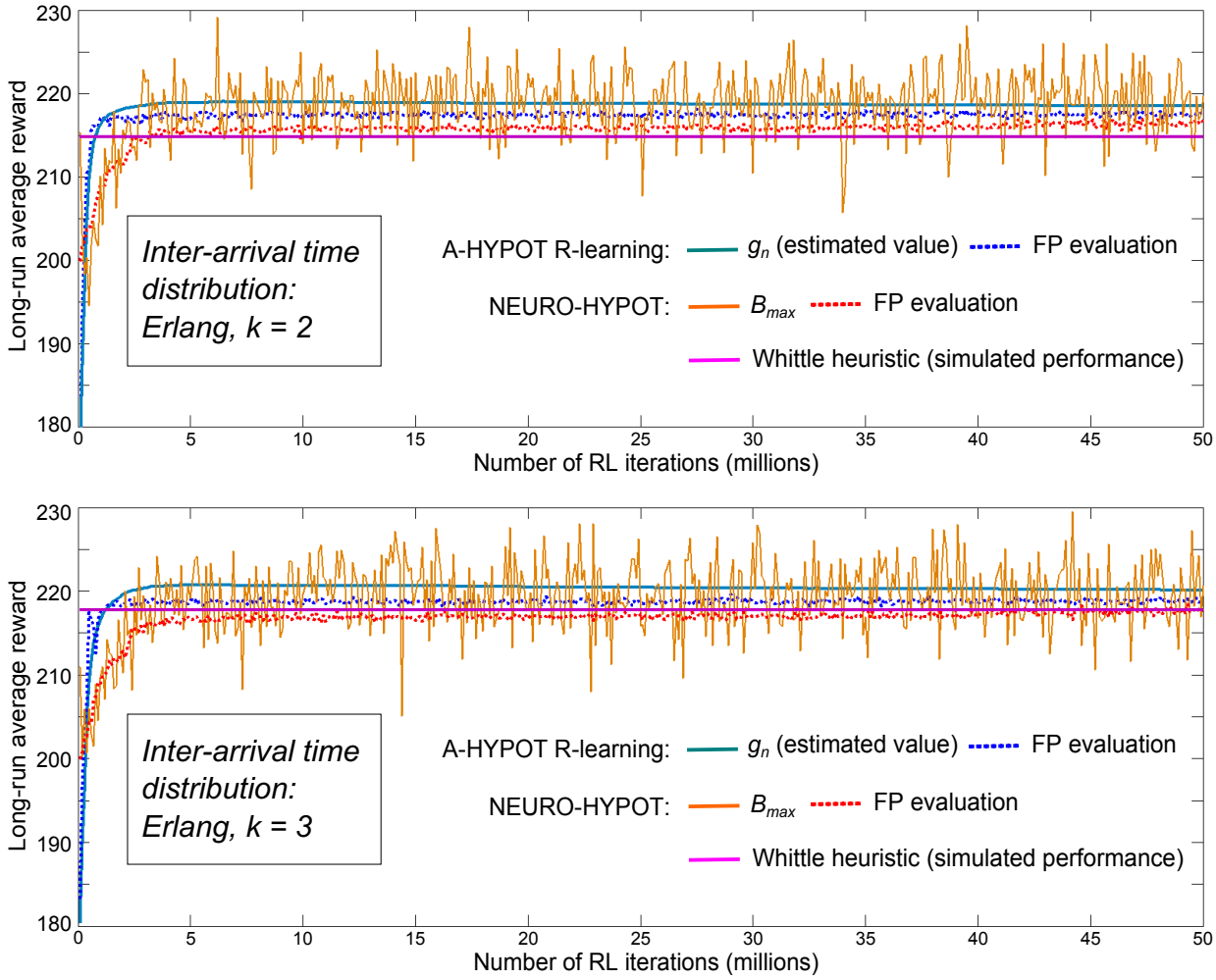


Figure 7.17: Comparisons between the A-HYPOT, NEURO-HYPOT and Whittle heuristic policies with inter-arrival times following an Erlang distribution with $k = 2$ (upper figure) and $k = 3$ (lower figure).

To some extent, the results shown in Figures 7.17-7.18 are comparable to the results of the experiment in Example 7.4.3, in which a Poisson arrival rate was assumed. In each of the four cases, the A-H algorithm manages to find a policy which out-performs the Whittle heuristic policy within about 1.3 million iterations (recall that the FP estimates, shown by dotted lines, indicate the performance of the latest policy found by the algorithm). As expected, the N-H algorithm exhibits a slightly worse performance than the A-H algorithm, and (except in the case $k = 2$) it also struggles to perform as well as the Whittle heuristic. One possible reason for this might be that the quadratic

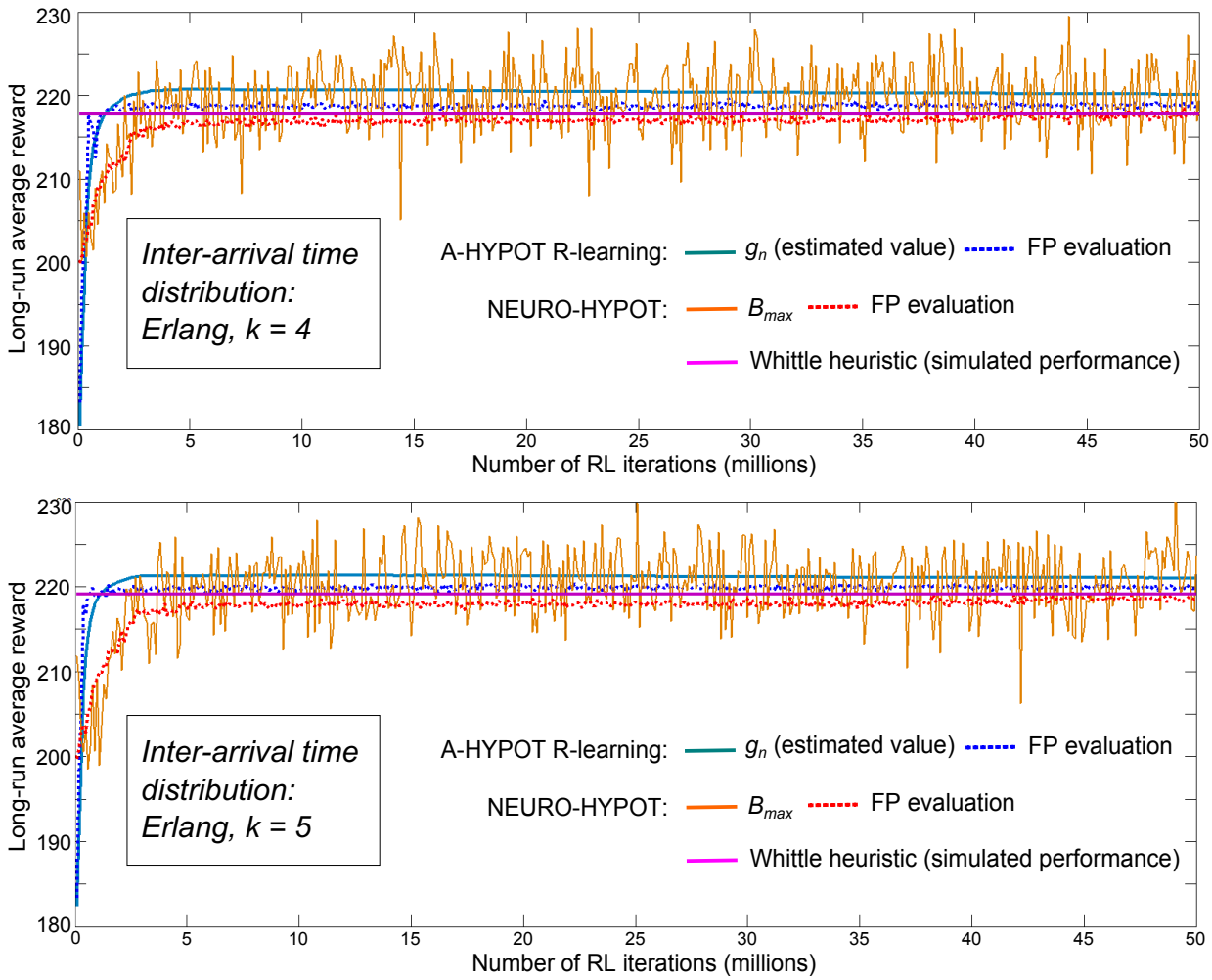


Figure 7.18: Comparisons between the A-HYPOT, NEURO-HYPOT and Whittle heuristic policies with inter-arrival times following an Erlang distribution with $k = 4$ (upper figure) and $k = 5$ (lower figure).

architecture (7.4.4) relied upon by the N-H algorithm does not fit the Q-factors so well when the arrival rate is non-Poisson; further analysis would be required in order to confirm this.

It is also interesting to note that the values g_n obtained by the A-H algorithm tend to over-estimate the average reward earned by the latest policy found by the algorithm; this did not occur when Poisson arrivals were considered previously. The reason for this is not known.

Another interesting point is that the Whittle heuristic policies appear to become more conservative as the shape parameter k increases. The indices $W_i(0)$ (for facilities $i \in \{1, 2, ..., N\}$) are actually independent of k, and are simply given by $\alpha_i - \beta_i/\mu_i$ (as in Section 6.1). However, for all integers

$x \geq 1$, the indices $W_i(x)$ appear to be monotonically decreasing with k. Table 7.10 illustrates this by showing the values $W_i(1)$ for $i \in \{1, 2, ..., 5\}$ and various different choices of k.

	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 10$
$i = 1$	-27.12	-36.68	-42.41	-46.24	-54.92
$i = 2$	11.52	10.87	10.48	10.23	9.64
$i = 3$	7.70	7.50	7.37	7.29	7.11
$i = 4$	3.04	3.03	3.02	3.01	2.99
$i = 5$	6.34	6.24	6.18	6.14	6.05

Table 7.10: Values of $W_i(1)$ for $i \in \{1, 2, ..., 5\}$ and various different values of k.

Of course, increasing the value of k reduces the amount of variation in the inter-arrival times (indeed, the variance of the Erlang distribution with parameters λ and k is $1/(k\lambda^2)$). It appears that the effect of increasing the value of k is similar to the effect of increasing the demand rate λ itself. Results in earlier chapters (specifically, Theorems 4.4.1 and 6.1.7) have established that, in the case of Poisson arrivals, the optimal average reward g^* increases with λ and the Whittle heuristic policy becomes increasingly conservative with λ. Figures 7.17-7.18 appear to show that, in the case of an Erlang arrival distribution, the Whittle and RL policies attain higher average rewards when k is increased, while the Whittle policy also appears to become more conservative. Thus, there is some evidence that reducing the amount of variation in the inter-arrival times works to the advantage of the system, just as increasing the arrival rate λ is also advantageous.

Case 2: Constant inter-arrival times

Suppose customers arrive at fixed intervals of length $1/\lambda$. In this case, the probabilities $b_i(n)$ defined in (7.6.7) are simply given by the Poisson probability mass function:

$$b_i(n) = \frac{e^{-\mu_i/\lambda}(\mu_i/\lambda)^n}{n!} \qquad \forall\, n \in \mathbb{N}_0.$$

Constant inter-arrival times arise in certain applications of queueing theory; for example, systems in which customers arrive at fixed appointment times, or production line systems in which items arrive via conveyor belts. In fact, a constant distribution is the limiting case of an Erlang distribution as the shape parameter k tends to infinity (see [67], p. 128), so this case is actually related to the previous one. Figure 7.19 shows the results of another experiment, identical to the 4 experiments

conducted in Case 1 except with inter-arrival times following a constant distribution.

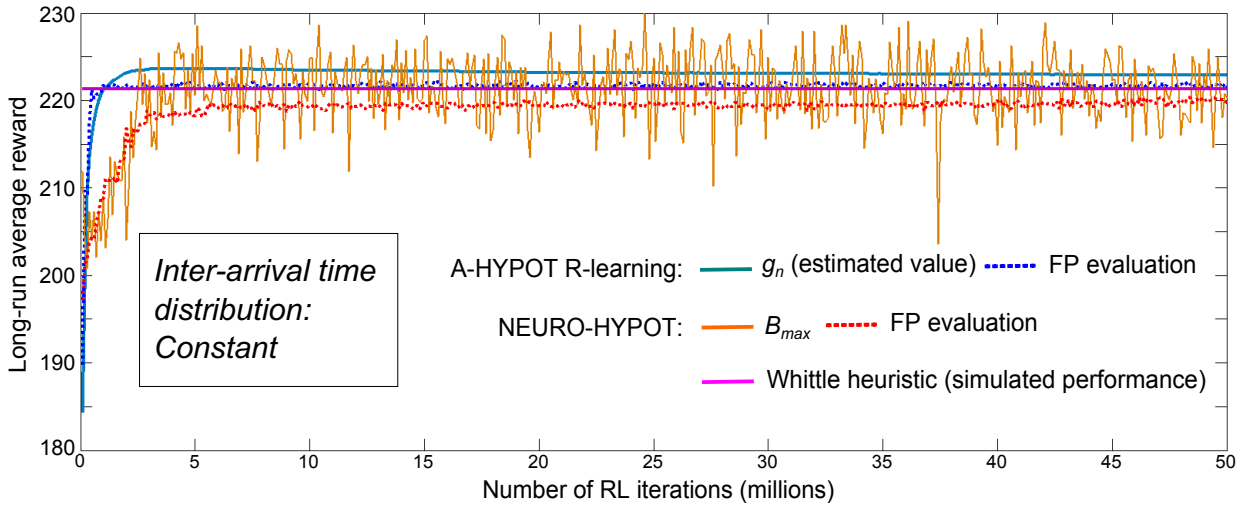


Figure 7.19: Comparison between the A-HYPOT, NEURO-HYPOT and Whittle heuristic policies with inter-arrival times following a constant distribution. All inter-arrival times are equal to $1/\lambda$.

In Case 1, it was observed that the average reward attained by the Whittle heuristic policy appears to increase with the Erlang parameter k. Given that constant inter-arrival times represent the case $k = \infty$, it therefore comes as no surprise that the Whittle policy is able to earn a greater average reward in this case than in any of the Erlang cases considered. Also, as in the previous cases, the Whittle heuristic policy appears to out-perform the NEURO-HYPOT policy (over 50 million iterations), but is not quite able to match the performance of the A-HYPOT policy.

Case 3: Lognormal distribution

Suppose inter-arrival times follow a lognormal distribution. The density function is:

$$f(t) = \frac{1}{t\sqrt{2\pi}\sigma} \exp\left(-\frac{(\log t - \nu)^2}{2\sigma^2}\right), \quad t \geq 0,$$

where $\nu \in \mathbb{R}$ and $\sigma > 0$ are the scale and shape parameters respectively. In this example, the scale parameter $\nu = \log(1/\lambda)$ will be used, where (as in the previous cases) $\lambda = 15$; this ensures that the median of the distribution is $1/\lambda$. Also, σ will be given by $|\nu|/5$. With these parameters, the lognormal distribution has a shape quite similar to that of the Erlang distribution with $k = 3$ or $k = 4$; see Figure 7.20. Figure 7.21 shows the results of another experiment, again similar to the previous experiments but with a lognormal distribution used for inter-arrival times.

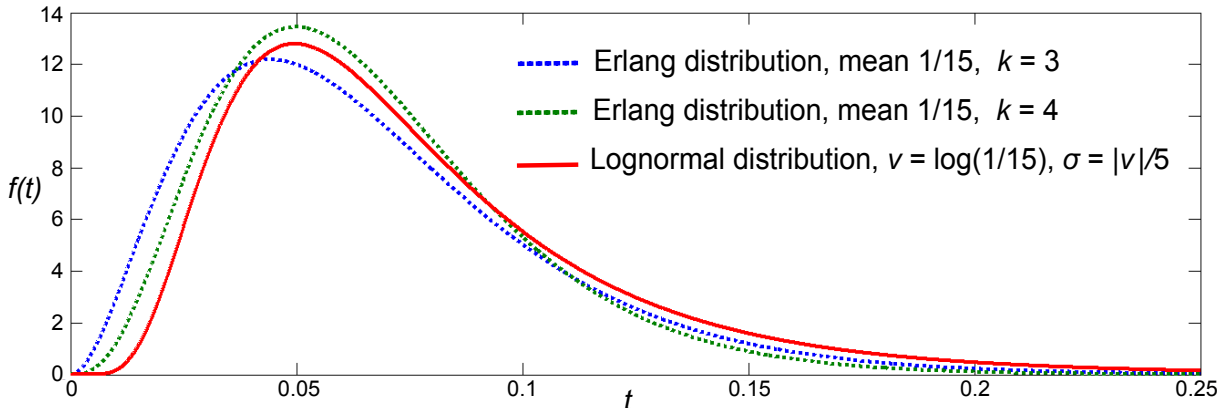


Figure 7.20: Comparison between the lognormal distribution considered in Case 3 of Example 7.6.3 and two of the Erlang distributions considered in Case 1 of the same example.

As in the previous cases, the values g_n and $B_{\max}$ used by the A-HYPOT and NEURO-HYPOT algorithms respectively tend to over-estimate the average rewards actually attained by these algorithms. Once again, the Whittle heuristic policy attains an average reward greater than that of the N-H policy (after 50 million iterations), but smaller than that of the A-H policy. ⊠

Before concluding this section, it will be appropriate to briefly discuss the case of non-exponential service times. Suppose (as in previous chapters) that customers arrive in the system via a Poisson process with parameter $\lambda > 0$, but the service times at the various facilities follow arbitrary distributions. For simplicity, this scenario will be referred to as the M/G case (Markovian arrivals and general service time distributions), whereas the scenario considered earlier in this section was the GI/M case (general arrival-time distribution and Markovian service times).

Unfortunately, the M/G scenario is much more difficult to model using an SMDP. In the GI/M case, it was possible to exploit the fact that customers' arrival times formed *regeneration points*, i.e. points at which past events had no predictive value. However, when service times follow arbitrary distributions, it is clearly not possible to use service completion times as regeneration points, since a service completion at one facility will generally take place while services are still in progress at other facilities. Even if the service completion times *did* have the regeneration property (for example, this would be the case in a single-facility system), it is generally necessary for actions to be chosen at regeneration points in an SMDP; but if the regeneration points in question correspond to service completions as opposed to customer arrivals then this poses a problem.

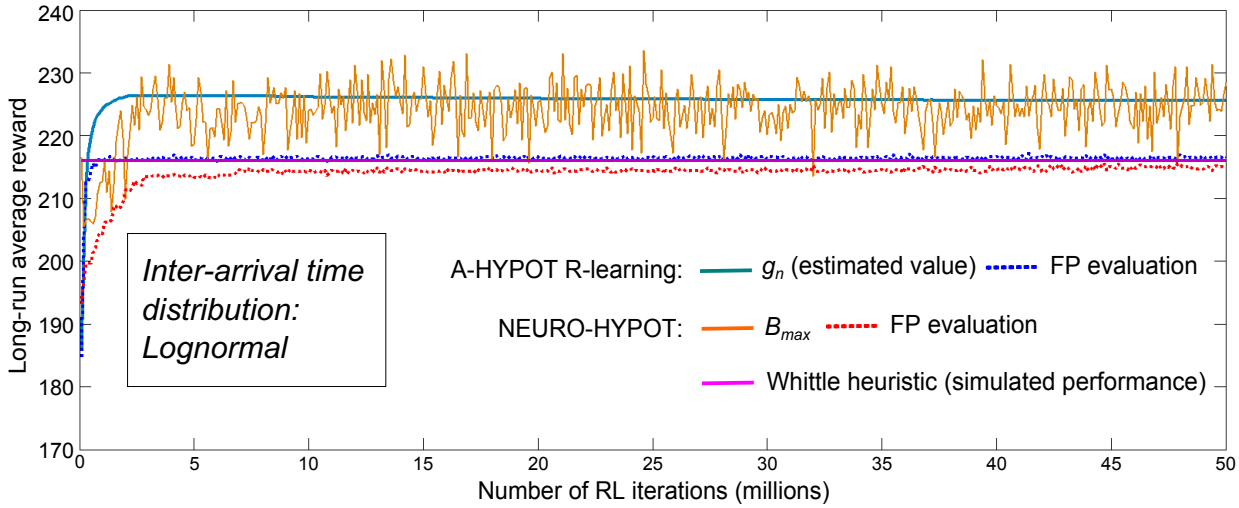


Figure 7.21: Comparison between the A-HYPOT, NEURO-HYPOT and Whittle heuristic policies with inter-arrival times following a lognormal distribution with median $1/\lambda$.

Nevertheless, the A-HYPOT and NEURO-HYPOT algorithms both use inverse transform sampling to simulate the service time requirements of individual customers, and it is therefore very simple to modify the steps of these algorithms so that service times at the various facilities follow arbitrary distributions. The M/G system can then be simulated accurately by both algorithms. Therefore the problem with applying these algorithms in the M/G case is *not* related to the simulation of random transitions, but is instead related to the estimation of Q-factors and the resulting characterisation of an optimal policy. The A-H and N-H algorithms both rely on the fundamental principle that the Q-factors that are being estimated satisfy a set of SMDP optimality equations of the form (7.6.1). In the case of the M/G system, these equations lose their basis in theory, since the transition probabilities $p(\mathbf{x}, a, \mathbf{y})$ no longer depend only on the state $\mathbf{x} \in S$ and action $a \in A_{\mathbf{x}}$ associated with the most recent customer to arrive; they also depend on the progress of services at the various facilities. Thus, although it is easy to modify the A-H and N-H algorithms so that they simulate the evolution of the M/G system, it is difficult to conceive what the results will be.

Figure 7.22 shows the results of applying the A-HYPOT and NEURO-HYPOT algorithms to a 4-facility system with the same parameters as in Example 7.2.3, but with the service time distributions modified so that each facility serves customers according to an Erlang distribution with mean μ_i^{-1} (where the values μ_i are given in (7.2.13)) and $k = 5$ as the Erlang shape parameter (i.e. the number of phases) at each facility. As discussed above, one would not necessarily expect these

algorithms to perform well in the case of non-exponential service times, since the SMDP optimality equations (7.6.1) are no longer valid. Indeed, the FP updates generated by the A-H algorithm exhibit quite strange behaviour; it seems that a peak value of approximately 125.5 is attained after a relatively small number of iterations, but subsequent estimates are markedly lower and are also quite inconsistent. Meanwhile, the N-H algorithm fails to obtain a policy which comes within close proximity (in terms of average reward) to the final policy given by the A-H algorithm.

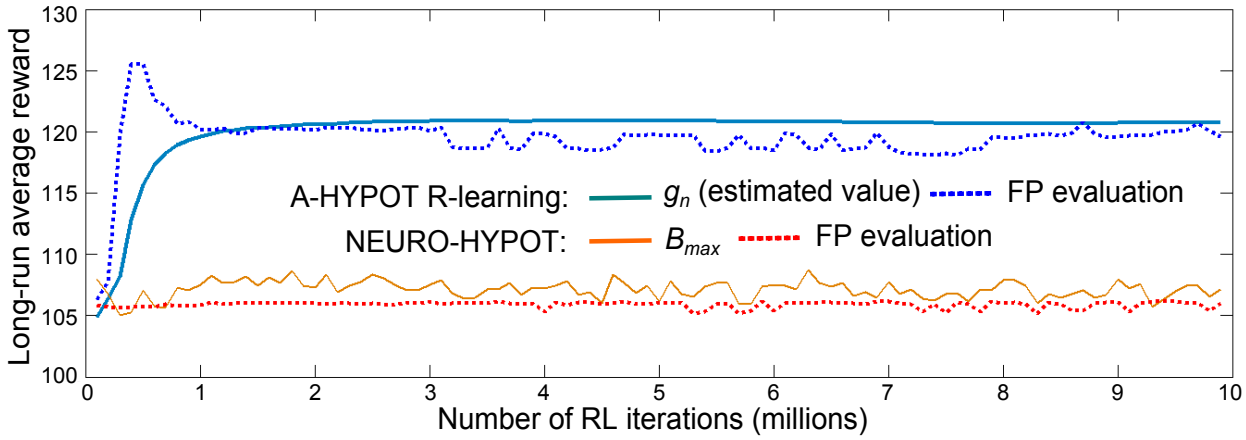


Figure 7.22: Performances of the A-HYPOT and NEURO-HYPOT algorithms in a system where customers arrive via a Poisson process, but the service-time distributions are all Erlang distributions with $k = 5$.

The conclusion of this section is that, although the A-HYPOT and NEURO-HYPOT algorithms both appear to perform strongly in the case of a GI/M system, this is not the case when an M/G system is considered. The GI/M system can be modelled as an SMDP by relying on the approach described in Section 7.3, whereas the M/G cannot be modelled as an SMDP in any obvious way. In general, it seems fair to surmise that the RL methods discussed throughout this chapter thus far are reliant on the principle that the system under consideration permits an MDP or an SMDP formulation; if no such formulation is possible, then these methods may be unsuitable.

It should be pointed out that the field of reinforcement learning is broad, and it is entirely plausible that a problem such as the M/G problem could be tackled effectively using RL techniques which are less closely linked with the theory of MDPs and SMDPs; this is a potential avenue for further work. Before concluding this chapter, one further type of problem will be considered, involving the application of RL algorithms to a system with heterogeneous customer classes.

7.7 Extension: Heterogeneous customers

Throughout this chapter, it has been assumed that all customers arriving in the system are subject to the same holding costs, fixed rewards and service time distributions. This section will reconsider the *heterogeneous customers* scenario discussed in Section 4.5, in which customers belong to heterogeneous groups or *classes*. Let it be assumed, as in Section 4.5, that each customer belongs to some class $i \in \{1, 2, ..., M\}$, where $M \geq 2$ is the number of classes. Customers of class i arrive via their own independent, time-homogeneous Poisson process with demand rate $\lambda_i > 0$. Any customer of class i who receives service at facility $j \in \{1, 2, ..., N\}$ incurs a holding cost $\beta_{ij} > 0$ per unit time while waiting in the system, and earns a reward $\alpha_{ij} > 0$ after completing service.

In Section 4.5, the service rates μ_j at the various facilities were assumed to be independent of customers' classes. This assumption, whilst being somewhat restrictive, was deemed necessary in order to enable a reasonably simple representation for the state space S of the associated MDP. However, in many types of applications one might wish to model a situation in which certain classes of customer require longer service times (on average) than others. Let it now be assumed, therefore, that the service time of a class i customer at facility j is exponentially distributed with parameter $\mu_{ij} > 0$. It will be shown in this section that the resulting MDP formulation, despite being significantly more complicated, may be tackled effectively using the RL methodology.

As discussed in Section 4.5, the state space representation $S = \{\mathbf{x} = (x_1, x_2, ..., x_N) : x_1, x_2, ..., x_N \in \mathbb{N}_0\}$ is satisfactory for the MDP which models the heterogeneous customers scenario if one assumes that single-step rewards are given by the 'anticipatory' reward formulation $\hat{r}(\mathbf{x}, \mathbf{a})$ defined in (4.5.2) and service rates are *not* dependent upon customer class. Indeed, if the μ_{ij} values have no dependence on i then one can easily calculate the single-step transition probabilities $p(\mathbf{x}, \mathbf{a}, \mathbf{y})$ for $\mathbf{x}, \mathbf{y} \in S$ by allowing $\mathbf{a} = (a_1, a_2, ..., a_M)$ to be a vector which prescribes the destinations for all customer classes individually, as in (4.5.1). However, in the case of class-dependent service rates, the state space representation S is no longer sufficient. Furthermore, one cannot simply use a matrix of values $X = (x_{ij})$ (where x_{ij} is the number of customers of class i at facility j) to represent the system state, since these x_{ij} values may not provide sufficient information to deduce the probability of a service completion occurring at some facility $j \in \{1, 2, ..., N\}$ before the next decision epoch if the facility in question is occupied by customers from more than one different class.

When service rates are class-dependent, it appears that one has no choice but to use a state space representation which includes not only the number of customers of class i (for each $i \in \{1, 2, ..., M\}$) waiting at each individual facility $j \in \{1, 2, .., N\}$, but also the positions of these customers in the queues. As previously, let x_j denote the number of customers waiting at facility j (regardless of class). Then the state of the system may be described by a vector $\mathbf{X}$ of the form:

$$\begin{aligned}\mathbf{X} = \Big(& (\mathcal{X}_{11}, \mathcal{X}_{12}, ..., \mathcal{X}_{1,x_1}), \\ & (\mathcal{X}_{21}, \mathcal{X}_{22}, ..., \mathcal{X}_{2,x_2}), \\ & \vdots \\ & (\mathcal{X}_{N1}, \mathcal{X}_{N2}, ..., \mathcal{X}_{N,x_N}) \Big), \end{aligned} \tag{7.7.1}$$

where $\mathcal{X}_{ik} \in \{1, 2, ..., M\}$ denotes the class of the customer at facility i, position k. At any facility i, the first c_i 'positions' will be taken by customers who have already begun service, so the customer at position c_i+1 will be the first customer waiting in the queue. Let $\mathbb{X}$ denote the set of vectors $\mathbf{X}$ of the form (7.7.1). After a suitable discretisation of the system, the transition probability $p(\mathbf{X}, \mathbf{a}, \mathbf{Y})$ for any two states $\mathbf{X}, \mathbf{Y} \in \mathbb{X}$ can be obtained exactly and depends primarily on the classes of the customers in service under state $\mathbf{X}$ and the action vector $\mathbf{a}$. However, it is clear that even if each head count x_j is constrained to a relatively small, finite range of values, the number of possible system states will generally be enormous. Indeed, suppose there are 4 facilities, 4 customer classes and the queue lengths are restricted so that no facility can have more than 4 customers present at any given time. A quick calculation shows that even with such stringent restrictions placed on the queue lengths, the number of possible system states is in excess of 10 billion.

For practical purposes, the complexity of the state space effectively rules out the use of dynamic programming algorithms. In addition, it also rules out the use of RL algorithms which depend upon storing unique values for individual state-action pairs (such as the A-HYPOT algorithm given on page 323). On the other hand, RL algorithms which use value function approximation methods (such as the NEURO-HYPOT algorithm presented on page 348) are not particularly encumbered by extremely vast state spaces; indeed, they are designed for use in such circumstances. As such, the discussion in this section will focus on the generalisation of the NEURO-HYPOT algorithm to the scenario with heterogeneous customer classes and class-dependent service rates.

As discussed in Section 7.6, RL algorithms such as NEURO-HYPOT (abbreviated from this point on to N-H) tend to rely on the premise that the system to which they are applied can theoretically be formulated as an MDP or an SMDP. Therefore, before discussing how the N-H algorithm can be applied to the heterogeneous customers problem, it is important to clarify the underlying MDP formulation. Recall (from Section 7.4) that the N-H algorithm models the so-called *MRA process*, which essentially means that the state of the system at any given time is defined as the 'physical' state observed by the most recent customer to arrive, regardless of how many service completions have occurred since then. Given that decision epochs for the MRA process are always associated with customer arrivals, it will be natural (and advantageous) to make a small adjustment to the state space formulation in (7.7.1), so that system states are denoted by *pairs* $(\mathbf{X}, i) \in \mathbb{X} \times \{1, 2, ..., M\}$, where $\mathbf{X}$ is a vector of the form (7.7.1) and i is the class of the latest customer to arrive. This ensures that the action chosen at any decision epoch can simply be a value $a \in \{0, 1, ..., N\}$, as opposed to a vector $\mathbf{a} = (a_1, a_2, ..., a_M)$ which prescribes decisions for all M classes.

The N-H algorithm operates by using the system state at any given decision epoch to derive a set of *inputs* to an Artificial Neural Network (ANN). The weights attached by the ANN depend upon the action chosen, and the ANN then produces an estimate of the Q-factor for the relevant state-action pair. This estimated value is compared with the latest randomly-sampled value obtained via simulation, and the weights of the ANN are then updated as appropriate. In the version of the algorithm given on page 348, the system state is simply an N-vector $(x_1, x_2, ..., x_N)$ and the components x_j are used directly as the inputs for the ANN (see Figure 7.12). Thus, it might be said that *all* of the information contained in the description of the system state is used by the ANN in order to produce a Q-factor estimate. Indeed, the quadratic approximation architecture given by (7.4.4) depends explicitly upon all of the components $x_1, x_2, ..., x_N$. An alternative approach would involve feeding only *some* of the state information into the ANN, in order to obtain a (probably less accurate) Q-factor estimate at a significantly lower computational expense. This alludes to the technique of *feature extraction*, described by Bertsekas and Tsitsiklis in [15] (p. 66).

Feature extraction, as described in [15], involves "selecting a set of *features* of the state and feeding them into an approximation architecture". These features should "capture the most important aspects of the current state", without necessarily including *all* such aspects. This approach makes sense in the present context, in which the state $(\mathbf{X}, i)$ is an element of the set $\mathbb{X} \times \{1, 2, ..., M\}$

and therefore is not compatible with a linear approximation architecture in any obvious way (note that the components $\mathcal{X}_{ik}$ of the vector $\mathbf{X}$ represent customer classes, which are categorical rather than ordinal; as such, it does not make sense to attach 'weights' to these components). Formally, feature extraction involves associating a vector of features $\mathbf{Z}$ with any given state $(\mathbf{X}, i)$, under the assumption that the following relationship holds for all actions $a \in \{0, 1, ..., N\}$:

$$Q((\mathbf{X}, i), a) \approx \hat{Q}(\mathbf{Z}, a). \tag{7.7.2}$$

In (7.7.2), $\hat{Q}$ should be interpreted as a relatively simple function of the vector of features $\mathbf{Z}$ and the action a; for example, it may be a linear function of the components of $\mathbf{Z}$. The system states $(\mathbf{X}, i)$ will generally be associated with feature vectors $\mathbf{Z}$ via a non-injective map, so that many different states can share the same feature vector. In this respect, feature extraction is implicitly related to the technique of *state aggregation* described in Section 7.4, since the same Q-factor estimate $\hat{Q}(\mathbf{Z}, a)$ will generally be associated with several different states $(\mathbf{X}, i)$. However, the approach is slightly different, since it does not involve explicitly storing values $\hat{Q}(\mathbf{Z}, a)$ for all possible pairs $(\mathbf{Z}, a)$ in the system memory; on the contrary, the estimates $\hat{Q}(\mathbf{Z}, a)$ are obtained via an approximation architecture, so that effectively one relies upon both function fitting *and* state aggregation. In summary, (7.7.2) postulates that the (theoretical) Q-factor for a particular state-action pair $((\mathbf{X}, i), a)$ can be approximated reliably using an easily-described function of $\mathbf{Z}$ and a, where $\mathbf{Z}$ is a *selected* set of features of the system state $(\mathbf{X}, i) \in \mathbb{X} \times \{1, 2, ..., M\}$.

In this section, the term 'extraction level' will be used, in an informal sense, to refer to the number of features (equivalently, the amount of information) extracted from a given system state in order to produce a Q-factor estimate. Three different levels of feature extraction will be considered; the first of these levels will be the lowest, and the third will be the highest (involving the richest approximation architecture). It is natural to suppose that, in order to give an RL algorithm the best possible chance of attaining a near-optimal policy, the level of feature extraction should always be as high as one can conceivably allow without over-complicating the underlying approximation architecture. However, as the next example will show, a high level of feature extraction may not necessarily yield better results than a low level of extraction over a fixed number of RL iterations. The three levels of extraction to be considered in this section are described below.

1. The first (lowest) level of feature extraction will involve defining the vector of features $\mathbf{Z}$

simply as an N-vector $\mathbf{x} = (x_1, x_2, ..., x_N)$, where x_j is the number of customers present at facility j (regardless of class). Essentially, by using this level of extraction, one completely ignores the fact that customers belong to heterogeneous classes and aims to approximate the Q-factors $Q((\mathbf{X}, i), a)$ using exactly the same approach that was relied upon in Section 7.4 (in which customers were assumed to be homogeneous). The approximation architecture will be the quadratic architecture used in Section 7.4. That is, for each action a:

$$\hat{Q}(\mathbf{x}, a) = B(a) + \sum_{j=1}^{N} C_j^{(1)}(a)\, x_j + \sum_{j=1}^{N} C_j^{(2)}(a)\, x_j^2. \tag{7.7.3}$$

For each action $a \in \{0, 1, ..., N\}$, the number of weights that must be fitted is $2N + 1$. Hence, using this extraction level requires $(N + 1)(2N + 1)$ weights to be fitted.

2. The second level of feature extraction will involve defining the vector of features $\mathbf{Z}$ as a *pair* $(\mathbf{x}, i)$, where $\mathbf{x} = (x_1, x_2, ..., x_N)$ is the same vector of head counts used in the previous case, and $i \in \{1, 2, ..., M\}$ is the class of the most recent customer to arrive. By using this level of extraction, one allows the weights of the ANN to depend upon the class of the customer whose destination is about to be determined, but *not* the classes of the customers who are already present at the various facilities. The approximation architecture will include weights which are dependent upon i (the new customer's class). That is, for each a:

$$\hat{Q}((\mathbf{x}, i), a) = B_i(a) + \sum_{j=1}^{N} C_{ij}^{(1)}(a)\, x_j + \sum_{j=1}^{N} C_{ij}^{(2)}(a)\, x_j^2. \tag{7.7.4}$$

In this case, the number of weights that must be fitted is $M(N + 1)(2N + 1)$.

3. The third (highest) level of feature extraction will involve defining the vector of features $\mathbf{Z}$ as a pair (X, i), where X is a matrix of order $M \times N$ and (as in the previous case) $i \in \{1, 2, ..., M\}$ is the class of the most recent customer to arrive. The component x_{jk} in row j, column k of the matrix X will be the number of customers of class $j \in \{1, 2, ..., M\}$ at facility $k \in \{1, 2, ..., N\}$ (regardless of their positions in the queue). By using this level of extraction, one allows the weights attached by the ANN to depend upon the class of the latest customer to arrive *and* some additional information regarding the classes of customers present at the individual facilities. The approximation architecture will incorporate two unique weights $C_{ijk}^{(1)}(a)$ and

$C^{(2)}_{ijk}(a)$ for each arrival class i, component x_{jk} of X and action a. That is:

$$\hat{Q}((X,i),a) = B_i(a) + \sum_{j=1}^{M}\sum_{k=1}^{N} C^{(1)}_{ijk}(a)\,x_{jk} + \sum_{j=1}^{M}\sum_{k=1}^{N} C^{(2)}_{ijk}(a)\,x^2_{jk}. \tag{7.7.5}$$

In this case, the number of weights that must be fitted is $M(N+1)(2MN+1)$.

It is worthwhile to emphasise that *none* of the extraction levels described above make *complete* use of the information $(\mathbf{X}, i)$ contained in the description of the system state; indeed, the third (highest) extraction level takes no account of the ordering of the customers waiting at any individual facility, whereas the system state $(\mathbf{X}, i) \in \mathbb{X}\times\{1,2,...,M\}$ *does* include this information. The N-H algorithm given on page 348 can easily be modified so that Q-factors are estimated using any of the three extraction levels described above. In the case of the first extraction level, there are no modifications required at all, since customers' classes are effectively ignored. In the cases of the second and third levels, the estimates $\hat{Q}(\mathbf{y}, a')$ and $\hat{Q}(\mathbf{x}, a)$ in equations (7.4.10) and (7.4.12) respectively should be calculated according to the relevant extraction level formula (either (7.7.4) or (7.7.5)), and there are some extra weights to update in step 3(d) which are updated in an analogous way to those in (7.4.13). Furthermore, the value $B_{\max}$ (used to provide an estimate of the optimal average reward) can be given by $\max_a B_i(a)$, where the choice of class $i \in \{1,2,...,M\}$ is arbitrary.

Another important point to be made is that, regardless of the level of feature extraction being used, the *simulation* of the system by the RL algorithm itself should always be done in a completely accurate way. The N-H algorithm on page 348 can easily be modified so that it faithfully reproduces the dynamics of a system with heterogeneous customers. Indeed, all that is required is for the inter-arrival times and service times (generated using inverse transform sampling in steps 2 and 3(a) respectively) to be sampled according to the class-dependent parameters λ_i and μ_{ij}. In general, the simulation of a stochastic process which evolves according to known parameters and probability distributions should always be completely accurate when an RL algorithm is used, regardless of whether or not techniques such as feature extraction (and other methods discussed in earlier sections, such as state aggregation) are employed. Feature extraction is used purely as a means of simplifying the process by which Q-factors are estimated. In summary, the estimation of Q-factors may be regarded as a somewhat separate process from the simulation of the system itself, even though both tasks must be performed at the same time when using RL.

The next example tests the performances of the N-H algorithm under the three different levels of feature extraction described above, in a 4-facility system with 4 customer classes.

Example 7.7.1. ***(Feature extraction in a system with heterogeneous customers)***

Consider a system with 4 heterogeneous customer classes, with demand rates given by:

$$\lambda_1 = 4, \qquad \lambda_2 = 8, \qquad \lambda_3 = 7, \qquad \lambda_4 = 6.$$

Suppose there are also 4 service facilities, with the following server capacities:

$$c_1 = 4, \qquad c_2 = 3, \qquad c_3 = 3, \qquad c_4 = 2.$$

The service rates, holding costs and fixed rewards for the 4 facilities are class-dependent, and are shown in Table 7.11. The parameters for Class 1 correspond to those in Example 7.2.3.

	Facility 1	Facility 2	Facility 3	Facility 4
Class 1	$\mu_{11} = 1$ $\beta_{11} = 4$ $\alpha_{11} = 14$	$\mu_{12} = 4$ $\beta_{12} = 5$ $\alpha_{12} = 7$	$\mu_{13} = 8$ $\beta_{13} = 8$ $\alpha_{13} = 4$	$\mu_{14} = 16$ $\beta_{14} = 5$ $\alpha_{14} = 2$
Class 2	$\mu_{21} = 2$ $\beta_{21} = 5$ $\alpha_{21} = 12$	$\mu_{22} = 5$ $\beta_{22} = 4$ $\alpha_{22} = 6$	$\mu_{23} = 6$ $\beta_{23} = 6$ $\alpha_{23} = 3$	$\mu_{24} = 15$ $\beta_{24} = 3$ $\alpha_{24} = 2$
Class 3	$\mu_{31} = 3$ $\beta_{31} = 6$ $\alpha_{31} = 17$	$\mu_{32} = 7$ $\beta_{32} = 4$ $\alpha_{32} = 9$	$\mu_{33} = 9$ $\beta_{33} = 7$ $\alpha_{33} = 5$	$\mu_{34} = 18$ $\beta_{34} = 4$ $\alpha_{34} = 3$
Class 4	$\mu_{41} = 2$ $\beta_{41} = 5$ $\alpha_{41} = 13$	$\mu_{42} = 3$ $\beta_{42} = 5$ $\alpha_{42} = 8$	$\mu_{43} = 7$ $\beta_{43} = 4$ $\alpha_{43} = 4$	$\mu_{44} = 13$ $\beta_{44} = 7$ $\alpha_{44} = 4$

Table 7.11: Parameters for the 4 service facilities in Example 7.7.1.

An experiment was performed in which the N-H algorithm was run using each of the three different levels of feature extraction described on page 387. In each case, 150 million RL iterations were performed, with an ϵ-greedy method used for action selection (with $\epsilon = 0.15$) and the constant

rule $\delta_n = 0.001$ used for the learning parameters. The Whittle index heuristic cannot easily be generalised to a problem involving heterogeneous customer classes, and therefore it was deemed necessary to find an alternative to the method for defining the facility 'thresholds' b_j that was used in Examples 7.4.3 and 7.6.3. The method chosen in this experiment was as follows:

- Under the first and second extraction levels, the thresholds b_j were given by:

$$b_j = \max_{i \in \{1,2,...,M\}} \left\lfloor \frac{\alpha_{ij} c_j \mu_j^*}{\beta_{ij}} \right\rfloor, \qquad j \in \{1, 2, ..., N\},$$

  with $\mu_j^* = \max(\mu_{1j}, \mu_{2j}, ..., \mu_{Mj})$ defined as the maximum service rate at facility j over all customer classes. This definition for b_j ensures that no customers of any class $i \in \{1, 2, ..., M\}$ have a 'selfish' incentive to join facility $j \in \{1, 2, ..., N\}$ when there are b_j customers present, since they would earn a negative expected net reward by doing so. Thus, this alludes to the principle of remaining 'contained' within the selfishly optimal state space.

- In the case of the third extraction level, the form of the underlying approximation architecture (see (7.7.5)) enables the facility thresholds b_{ij} to be class-dependent as well as facility-dependent. Essentially, b_{ij} represents the maximum number of customers of class i that one should expect to find at facility j. The following definition was used for b_{ij}:

$$b_{ij} = \left\lfloor \frac{\alpha_{ij} c_j \mu_{ij}}{\beta_{ij}} \right\rfloor, \qquad i \in \{1, 2, ..., M\},\ j \in \{1, 2, ..., N\}. \tag{7.7.6}$$

  This alludes, once again, to the principle of forcing the process to remain 'contained' within the selfishly optimal state space. A customer of class i who joins facility j when there are already b_{ij} (or more) customers *of their own class* present will earn a negative individual expected net reward, regardless of whether or not there are any customers of *other* classes also present. Therefore b_{ij} is an upper bound for x_{ij} (the number of customers of class i present at facility j) when the process operates under a selfishly optimal policy.

Under each extraction level, the N-H algorithm was allowed to run for 150 million iterations and Frozen Phase (FP) estimates were performed after every 10,000 iterations in order to simulate the performance of the latest 'guess' for an optimal policy. Figure 7.23 shows the results obtained. As in the N-H examples from previous sections, the $B_{\max}$ values provide a rough indication of the algorithm's progress towards finding a near-optimal policy, but the FP estimates provide the most

reliable measure of its performance, since these are essentially fixed-policy evaluations performed by simulation. Let $\theta_1^{(n)}$, $\theta_2^{(n)}$ and $\theta_3^{(n)}$ denote the policies obtained after n RL iterations under the first, second and third extraction levels respectively. The figure indicates that $\theta_3^{(n)}$ out-performs $\theta_2^{(n)}$ and $\theta_1^{(n)}$ when n is sufficiently large. After 150 million iterations, $\theta_2^{(n)}$ improves upon $\theta_1^{(n)}$ by about 11.5%, and $\theta_3^{(n)}$ achieves a further improvement of about 3% upon $\theta_2^{(n)}$.

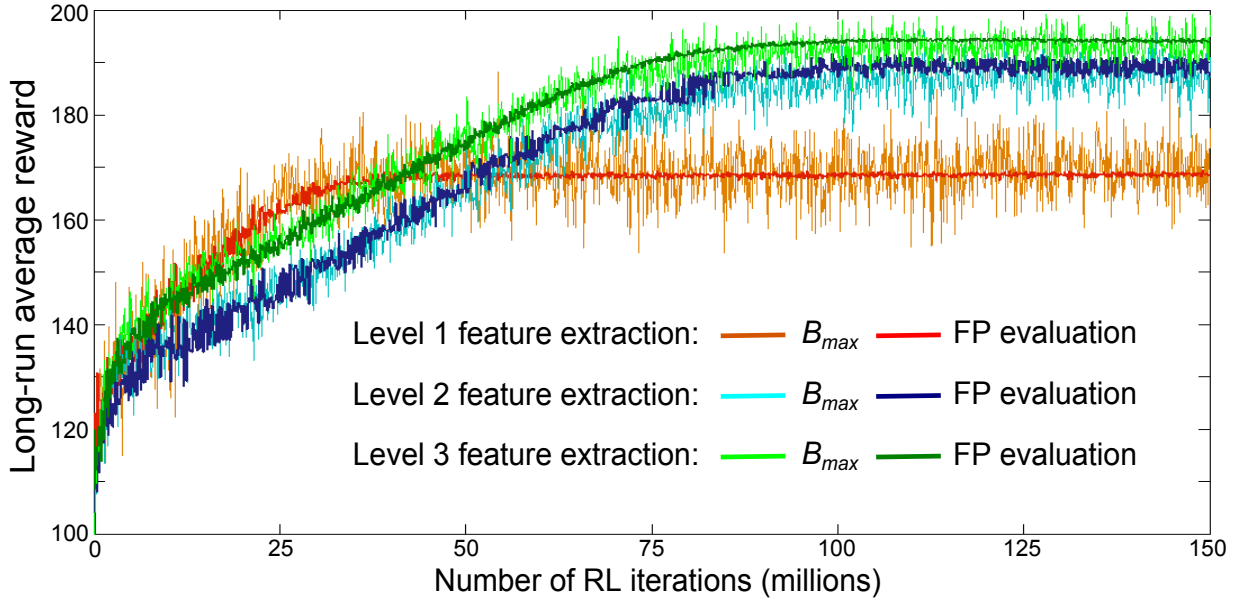


Figure 7.23: Performance of the NEURO-HYPOT algorithm with three different levels of feature extraction. The $B_{\max}$ values are given by $\max_a B(a)$ (for level 1) and $\max_a B_1(a)$ (for levels 2 and 3).

Essentially, by implementing the policy $\theta_3^{(n)}$ one allows routing decisions to be made according to the class of the newly-arrived customer *and* the classes of the customers already present at the various facilities. On the other hand, the routing decisions made by $\theta_2^{(n)}$ take no account of the classes of customers already present in the system, and the decisions made by $\theta_1^{(n)}$ take no account of customer classes at all. It is therefore unsurprising that $\theta_3^{(n)}$ performs better than either of the policies $\theta_2^{(n)}$ and $\theta_1^{(n)}$ when n is large. However, it is interesting to note that approximately 40 million RL iterations are needed before $\theta_3^{(n)}$ surpasses the performance of $\theta_1^{(n)}$, and more than 50 million iterations are required before $\theta_2^{(n)}$ is able to demonstrate its superiority to $\theta_1^{(n)}$.

The fact that $\theta_3^{(n)}$ and $\theta_2^{(n)}$ require so many iterations to be performed before they are able to out-perform $\theta_1^{(n)}$ may be explained by the fact that they rely upon more complicated approximation architectures, and therefore the ANN weights that they use require a greater amount of 'tuning'.

Indeed, given $M = 4$ and $N = 4$, the architecture (7.7.5) used by $\theta_3^{(n)}$ requires $M(N+1)(2MN+1) = 660$ weights to be fitted. The architecture (7.7.4) used by $\theta_2^{(n)}$ requires only $M(N+1)(2N+1) = 220$ weights to be fitted, whereas (7.7.3) (the simplest of the three architectures) requires only $(N+1)(2N+1) = 45$ weights to be fitted. Under any of the three extraction levels, the values of the ANN weights *relative to each other* are of paramount importance, since these values determine the ranking order which in turn determines the action chosen under a particular state when an exploitative decision is made. Thus, the higher-level extraction policies $\theta_2^{(n)}$ and $\theta_3^{(n)}$ are encumbered by the fact that they must determine suitable values for a large number of weights; on the other hand, the low-level extraction policy $\theta_1^{(n)}$, with only 45 weights, requires only a relatively small number of iterations in order to achieve the best performance that it is capable of.

It is also likely that the slow rate of convergence under the third extraction level is caused by the definition of the thresholds b_{ij} in (7.7.6). As discussed in Section 7.4, the N-H algorithm relies upon the technique of 'normalising' the inputs to the ANN, so that all inputs take values between 0 and 1. In the case of the third extraction level, this is done by dividing each component x_{ij} of the matrix X by the corresponding threshold b_{ij}. However, the threshold b_{ij} defined in (7.7.6) is a rather crude upper bound for the number of customers of class i that one would expect to find at facility j under an optimal policy, since it is essentially the minimum number of class i customers at facility j which would cause an additional class i customer's individual expected net reward to be negative if there were no customers of other classes present. In reality, it is likely that the presence of customers from *other* classes at facility j would deter a class i customer (acting under an optimal policy) from joining facility j even if x_{ij} was much smaller than b_{ij}; in other words, the threshold b_{ij} is probably not tight enough. If the ANN inputs are too small (i.e. too close to 0), then the convergence of the weights will tend to be slow. Ideally, one would wish to perform some kind of *heuristic* calculation in order to gain a rough estimate of the maximum number of class i customers that one would expect to observe at facility j under an optimal policy, in the same way that the Whittle index heuristic was used to define the thresholds b_j in Section 7.4; however, it is difficult to conceive how this could be done in the case of heterogeneous customers.

This example has shown that 'rich' approximation architectures such as (7.7.5) can achieve better results than simpler alternatives such as (7.7.3) and (7.7.4), but the advantage of a rich architecture

may not be seen until a large number of RL iterations have been performed. ⊠

In Example 7.7.1, the policy $\theta_3^{(n)}$ obtained after 150 million RL iterations attained an average reward of approximately 194.3 (subject to simulation error). One might suppose that this value should be close to the optimal value g^* which would be found using a dynamic programming algorithm; unfortunately, however this cannot be investigated further due to the practical difficulties discussed at the beginning of this section. Ideally, one would wish to compare the performance of the N-H algorithm (with feature extraction) with those of *heuristic* policies such as those discussed in Chapter 6, but neither the Whittle index heuristic (Section 6.1) nor the static routing heuristic (Section 6.2) can be generalised to the heterogeneous customers problem in any obvious way. In general, the challenge of devising and validating effective heuristics and RL algorithms for queueing systems which evolve according to complicated dynamics remains a formidable one.

7.8 Conclusions

Reinforcement learning is a term applied to a broad family of algorithms which aim to 'learn' an optimal pattern of behaviour through exploration and experience. In the context of the queueing system optimisation problems considered in this thesis, 'exploration' is carried out by simulating the consequences of choosing different actions at the various system states. By observing the random consequences of choosing particular actions, an RL algorithm is able to obtain empirical estimates of the Q-factors $Q(\mathbf{x}, a)$ which satisfy the average reward optimality equations (7.2.2) for the underlying MDP, and thereby characterise an average reward optimal policy.

Section 7.2 introduced some *general-purpose* RL algorithms, including R-learning and relative Q-learning. These algorithms can be applied to any average reward optimisation problem which permits a theoretical representation as an MDP. Section 7.3 discussed the use of *tailored* RL algorithms, which aim to exploit particular properties of the queueing systems considered in this thesis in order to achieve faster convergence to a near-optimal policy. The HYPOT R-learning algorithm presented on page 312 exploits an independence between the action chosen at a particular decision epoch and the random event that occurs after the action is chosen. Essentially, this independence implies that, at any particular state $\mathbf{x} \in S$, the Q-factor estimates $\hat{Q}(\mathbf{x}, a)$ for *all* actions $a \in A_{\mathbf{x}}$ can be updated in an even-handed manner according to the same sequence of random events. The

A-HYPOT algorithm presented on page 323 introduces a new MDP formulation in which the system state at any given time is given by the vector of head counts at the N facilities observed by the most recent customer to arrive. This MDP, referred to as the *MRA process*, has a set of transition probabilities which would realistically be too complicated to incorporate within a DP algorithm, and therefore it is ideally suited to the general approach of RL. The A-HYPOT algorithm is able to achieve a strong performance by simulating the MRA process in continuous time.

Section 7.4 discussed the practical problems caused by MDPs with extremely vast state spaces. The NEURO-HYPOT algorithm presented on page 348 uses the technique of *function-fitting* as a means of estimating the Q-factors $Q(\mathbf{x}, a)$ *without* the requirement for values to be stored for individual state-action pairs $(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$. The function-fitting is performed by an *Artificial Neural Network* (ANN), which operates in conjunction with an underlying *approximation architecture*. The ANN attaches action-dependent weights to a set of 'inputs' (given by the system state) in order to produce an estimate of the value $Q(\mathbf{x}, a)$ for any given state-action pair $(\mathbf{x}, a)$. The numerical results given in Section 7.5 have shown that the A-HYPOT and NEURO-HYPOT algorithms both perform consistently well in tests involving tens of thousands of randomly-generated parameter sets. The A-HYPOT algorithm is particularly strong, although its computational requirements imply that it may not be applicable in systems where the selfish state space $|\tilde{S}|$ is extremely large.

Section 7.6 discussed the application of the A-HYPOT and NEURO-HYPOT algorithms in systems which evolve according to non-exponential probability distributions. These methods were based on the theoretical formulation of the system as an SMDP. In the case of a non-exponential inter-arrival time distribution (and exponential service-time distributions), the system can easily be formulated as an SMDP by again relying upon the technique of associating decision epochs only with customers' arrival times (and not service completion times). The A-HYPOT and NEURO-HYPOT algorithms were both found to perform satisfactorily in comparison to an adapted version of the Whittle index heuristic from Section 6.1. Section 7.7 discussed the technique of *feature extraction*, which can essentially be used as a means of simplifying the approximation architecture by which Q-factors are estimated as part of a function-fitting RL algorithm. This is an appropriate technique to use in systems involving heterogeneous customer classes in which service rates are class-dependent, since the state space representation becomes extremely complicated in such circumstances.

8 Conclusions and further work

The conclusions given at the end of individual chapters have summarised the main findings of this thesis, and therefore the purpose of this final chapter will be to broadly assess the contributions made by this work and suggest some possible directions for further research.

As stated in the introductory chapter, an original motivation for this work was to investigate the contrasts between selfishly optimal and socially optimal policies in multiple-facility queueing systems, with patient choice in the NHS as a possible real-life application. However, the results of this thesis have not been restricted to any specific application, and indeed they might be applied to any scenario in which the problem of interest involves routing incoming traffic in such a way as to optimise the long-run economic performance, provided that the modelling assumptions made in Section 3.1 (e.g. linear holding costs, etc.) are justifiable in the relevant context.

All of the queueing systems considered in this thesis (with the possible exception of the system considered in Section 7.7, in which the combination of class-dependent service rates and multiple service channels creates some extra complexity) have been systems in which a customer's expected waiting time at a particular service facility can be calculated easily as a function of the number of customers already present. For this reason, it has generally been the case that selfishly optimal policies are simple *index-based* policies (see Section 6.1 for further elaboration). On the other hand, the characterisation of *socially* optimal policies has proved to be somewhat more difficult. Even in the case of an observable $M/M/1$ queue, the socially optimal threshold n_o must be obtained via a one-dimensional search for a value v_o satisfying the equation (2.3.3), whereas the selfishly optimal threshold is given immediately by (2.3.1). This theme persists in the more general case of a multiple-facility, multiple-server system, in which a socially optimal policy must be found by formulating the system as an MDP and applying dynamic programming methods.

The results in Chapter 3 have shown that there is room for some flexibility in formulating the discrete-time MDP used to find a socially optimal policy for the N-facility queueing system described in Section 3.1. Arguably, Theorem 3.5.3 is the most useful result in this chapter from a practical perspective, since it enables the 'real-time' and 'anticipatory' reward formulations ($r(\mathbf{x})$ and $\hat{r}(\mathbf{x}, a)$ respectively) to be interchanged without any loss of accuracy in evaluating the expected

long-run average reward under a fixed stationary policy. This equivalence between the two reward formulations has been put to advantageous use at various stages of this thesis; for example, in Section 4.4 the proof of the light-traffic (social) optimality of the selfish policy $\tilde{\theta}$ (Theorem 4.4.4) relies upon the anticipatory reward formulation, whereas the proof of the heavy-traffic optimality of vacancy policies (Theorem 4.4.7) instead utilises the real-time formulation.

Ideally, one would wish to make logical deductions about the structure of a socially optimal policy from the structure of the corresponding *selfishly* optimal policy $\tilde{\theta}$ (noting that the latter has a particularly simple formulation). Of course, the main result in this thesis which involves a comparison along these lines is Theorem 4.2.4, which establishes that a socially optimal policy θ^* exists which causes the process to remain 'contained' within the finite, selfish state space $\tilde{S}$. This result is interesting in the sense that it may be regarded as an N-fold generalisation of Naor's classical result for an $M/M/1$ queue, but it is also a result of great practical importance, since without it one would not have a theoretical justification for the strategy of searching within the finite set $\tilde{S}$ when employing dynamic programming algorithms. In addition, due to Theorem 4.4.4, it can be said that the bound $S_{\theta^*} \subseteq \tilde{S}$ becomes 'tight' as the demand rate λ tends to zero.

Chapters 6 and 7 herald a change in emphasis from earlier chapters of this thesis, since they consider the problem of finding *near-optimal* policies for systems with vast state spaces. The two heuristic approaches discussed in Chapter 6 are based on methods which have already received considerable attention in the literature. Both of these heuristic approaches involve analysing service facilities *individually* in order to derive index-based policies, and this makes it possible to prove various structural properties of the heuristic policies $\theta^{[W]}$ and $\theta^{[B]}$ by relying upon the properties of optimal *single-facility* policies proved in Section 5.1. For example, the Whittle policy $\theta^{[W]}$ is asymptotically optimal in light-traffic and heavy-traffic limits (by Theorem 6.1.8), and it can be ranked above the optimal policy θ^* in terms of conservativity (by Theorem 6.1.10).

While it is certainly useful to know that a particular heuristic method is likely to perform well in systems with very small or very large demand rates, it does not appear realistic to obtain general performance guarantees for the policies $\theta^{[W]}$ and $\theta^{[B]}$ (i.e. bounds on their sub-optimality). The numerical results presented in Section 6.3 have demonstrated that both heuristics are consistently able to perform well in randomly-generated problem instances, but no guarantees are available. The

various *reinforcement learning* algorithms discussed in Chapter 7 do not possess proven convergence properties either, but they do at least offer the possibility of continuous improvement; that is, one would hope that in most cases they would 'approach' an optimal policy over the course of their running time. The use of the term 'approximate dynamic programming' to describe the application of reinforcement learning algorithms is clearly appropriate, since these algorithms involve using simulation to obtain empirical estimates of the Q-factors which are computed exactly by a dynamic programming algorithm. Two extremely important advantages of RL algorithms are their ability to operate *without* exact knowledge of transition probabilities and rewards (exploited by the A-HYPOT and NEURO-HYPOT algorithms), and their natural tendency to focus attention on states which are visited *often* under an optimal policy. These properties enable RL algorithms to avoid, almost entirely, the turgid computations associated with dynamic programming.

As discussed in Section 2.1, virtually any queueing system formulation that one might conceive has the potential to be modified or generalised in some respect. It is therefore quite easy to propose generalisations of the various models considered in this thesis, and these may be recognised as possible directions for future research. Some possibilities are suggested below.

- All of the queueing systems considered in this thesis have assumed a cost-and-reward structure whereby an individual customer's waiting cost for spending a time $t > 0$ in the system, given that they join some facility i, is given by the *linear* function $\beta_i t$. This assumption may be seen as somewhat restrictive, and one might relax this condition by requiring only that holding costs are non-decreasing (one might also impose conditions such as convexity). Indeed, several of the classical papers from the 1970s on the optimal control of *single-facility* systems impose fairly general conditions on waiting costs; see, for example, [103, 120, 168].

- The assumption that customers never change their minds (by, for example, leaving a queue that they have already joined) clearly makes more sense in some applications than in others. In order to model the behaviour of *impatient customers*, one might allow *reneging from a queue* and *jockeying between queues* as permissible types of customer behaviour. A typical method of modelling impatient behaviour would involve assuming that if there are x_i customers waiting at a particular service facility i, then customers leave the facility at an exponential rate $\gamma_i x_i$, where $\gamma_i > 0$ is an 'impatience parameter' (see, for example, [59]). Customers who leave a

queue might be allowed to choose a new facility, or they might simply exit from the system immediately. There are some challenges involved in modelling this type of process as an MDP. For example, *uniformisation* of the system (as described in Section 3.3) requires a uniform upper bound to be obtained for the sum of the transition rates out of any state, but this type of upper bound is unattainable if one allows queue lengths to be arbitrarily large, since the rate at which customers 'quit' increases linearly with the queue length.

- The formulation of an N-facility system as an MDP or CTMDP makes sense in the case where the system is *fully observable*, so that the system state is always known exactly. A fundamentally different type of problem would involve a system that was only *partially observable*. In this case, one would model the system using a Partially Observable MDP (POMDP), rather than an MDP. The decision-maker would not know the system state exactly, and would make decisions according to their 'belief' of the system state, which would depend upon incomplete information. POMDPs are described in greater detail in [197] (p. 130).

Arguably, the most prominent theme throughout this thesis has been that of socially optimal policies; that is, policies which maximise expected long-run average reward. Some chapters have focused on structural properties of these policies, while others have explored methods for approximating these policies in systems with vast state spaces (or other sources of complexity). It is difficult to foresee the extent to which the results of this thesis would apply to the more general types of problems described above. The main findings of this thesis suggest the following:

- Average reward optimisation in systems with heterogeneous facilities is a difficult problem. One should not always expect socially optimal policies to have a logical structure.
- Depending on the cost-and-reward structure of the system, it might be possible to establish bounds for the attainable state space associated with a socially optimal policy.
- Methods based on heuristics and reinforcement learning (i.e. simulation-based optimisation) are capable of yielding very close approximations to socially optimal policies.

It is left as an objective of future research to investigate whether or not the general principles stated above are broadly applicable to a wide range of stochastic optimisation problems.

A Appendices

A.1 Discrete-time Markov chains

A Markov chain is a stochastic process which evolves over time. The theory of Markov chains is richly developed and this appendix will summarise basic results which can be found in many texts. A suitable reference is Chung [31] (see also [32, 66, 67, 95, 125, 144, 167, 171]).

Let (X_n) $(n = 0, 1, 2, ...)$ be a sequence of random variables defined on a state space S which is assumed to be either finite or denumerably infinite. The sequence (X_n) is said to be a *discrete-time Markov chain* if it satisfies the *Markov property* (or *memoryless property*); that is, its future evolution depends only on its present state, and not on the history of past states.

More formally, the sequence $(X_n)_{n\in\mathbb{N}_0}$ is a *discrete-time Markov chain* (MC) if:

$$P(X_n = j_n|X_{n-1} = j_{n-1}, X_{n-2} = j_{n-2}, ..., X_0 = j_0) = P(X_n = j_n|X_{n-1} = j_{n-1}), \tag{A.1.1}$$

for $n \geq 1$ and $j_k \in S,\ 0 \leq k \leq n$. If, in addition, $P(X_{n+1} = j|X_n = i)$ does not depend on n, then the Markov chain is called *stationary* or *time homogeneous*. In this case the notation $p(i, j)$ may be used to denote the conditional probability $P(X_{n+1} = j|X_n = i)$, and this is called the *transition probability* associated with transferring from state i to state j in a single time step. It is assumed that $\sum_{j\in S} p(i, j) = 1$ for all states $i \in S$, and hence the MC cannot leave S.

Let $p^{(n)}(i, j)$ denote an *n-step transition probability*. For $n = 0$ one may define:

$$p^{(0)}(i, j) = \delta_{ij} = \begin{cases} 0, & \text{if } i \neq j, \\ 1, & \text{if } i = j. \end{cases} \tag{A.1.2}$$

The n-step probabilities $p^{(n)}(i, j)$ can then be obtained recursively as follows:

$$p^{(n+1)}(i, j) = \sum_{k\in S} p^{(n)}(i, k)\ p^{(1)}(k, j). \tag{A.1.3}$$

It can be shown using the Markov property (A.1.1) that, for every $m \geq 0$, $p^{(n)}(i, j) = P(X_{m+n} = j|X_m = i)$; see Chung [31] (p. 8) for further details. It is from this property that one derives the term 'n-step transition probability'. The next theorem is proved in [31] (p. 9).

*** Theorem A.1.1.** ***(Chapman-Kolmogorov equations)***

For $m \geq 0$, $n \geq 0$ and $i, j \in S$, the $(m+n)$-step transition probabilities satisfy:

$$p^{(m+n)}(i,j) = \sum_{k \in S} p^{(m)}(i,k)\, p^{(n)}(k,j). \tag{A.1.4}$$

Let $P = (p(i,j))$ be the matrix whose $(i,j)^{th}$ entry is the transition probability $p(i,j)$. Thus P is a *stochastic matrix*, which means that it is a matrix of finite or (denumerably) infinite order with non-negative elements such that the sum of elements in each row is equal to one.

In addition, let $P^{(n)}$ be defined as the matrix whose $(i,j)^{th}$ entry is the n-step transition probability $p^{(n)}(i,j)$. It follows that $P^{(n)}$ is also a stochastic matrix (for all $n \geq 1$) and it may be referred to as the *n-step transition matrix*. From (A.1.3) it follows that, for $n \geq 1$:

$$P^{(n)} = P^n.$$

That is, the n-step transition probability $p^{(n)}(i,j)$ is given by the $(i,j)^{th}$ element of the transition matrix P^n. This is a simple yet powerful result which enables the n-step transition probabilities to be obtained by simply raising the one-step transition matrix P to the power n.

Definitions will now be provided for some of the terminology used in this thesis.

Definition A.1.2. ***(Accessibility)***

The state $j \in S$ is said to be accessible *from state $i \in S$ if $p^{(m)}(i,j) > 0$ for some $m \geq 0$.*

Definition A.1.3. ***(Communicating states)***

The states $i, j \in S$ are said to communicate *if i is accessible from j and j is accessible from i.*

Definition A.1.4. ***(Communicating classes)***

A communicating class *of states is a subset of S in which all states mutually communicate. By convention, a set containing a single state is also referred to by this name.*

Definition A.1.5. ***(Closed sets)***

A subset C of S is said to be closed *if no state outside of C is accessible from any state within C. If no proper subset of C is closed, then C is a communicating class.*

Definition A.1.6. ***(Absorbing states)***

A state which forms a closed set by itself is called an absorbing *state.*

Definition A.1.7. ***(Irreducibility)***

If S consists of a single communicating class, then S is said to be irreducible*.*

Some further notation must be introduced in order to define the important properties of recurrence and transience. First, let $f^{(n)}(i,j)$ denote the probability that a 'first visit' of the MC to state j starting from state i takes place in exactly n time steps. That is, for $n \geq 1$:

$$f^{(n)}(i,j) := P(X_u \neq j, 0 < u < n; X_n = j | X_0 = i). \tag{A.1.5}$$

In addition, let $f^*(i,j)$ be defined as follows:

$$f^*(i,j) = \sum_{n=1}^{\infty} f^{(n)}(i,j). \tag{A.1.6}$$

Thus, $f^*(i,j)$ is the probability that the MC will visit state j at least once, given that it starts in state i. Definitions of *recurrence* and *transience* are given below.

Definition A.1.8. ***(Recurrence and transience)***

Let $f^{(n)}(i,j)$ (for $n \geq 1$) and $f^*(i,j)$ be as defined in (A.1.5) and (A.1.6). Then:

- *If $f^*(i,i) = 1$, the state $i \in S$ is* recurrent*.*
- *If $f^*(i,i) < 1$, the state $i \in S$ is* transient *(or* nonrecurrent*).*

Definition A.1.9. ***(Unichains)***

If S consists of a single recurrent class of states and, in addition, a (possibly empty) class of transient states, then S is said to be unichain*.*

Further properties related to recurrence and transience may be found in classic texts such as [49, 104]. Some alternative definitions are possible. For example, given an initial state $X_0 = i \in S$, let $T_i > 0$ be a random variable denoting the time taken (i.e. number of time steps) to return to state i. Then i is recurrent if $P(T_i < \infty) = 1$, and transient if $P(T_i < \infty) < 1$. The important

principle is that if i is *transient*, there is a positive probability that the MC will leave state i and never return, whereas the corresponding probability for a *recurrent* state is always zero.

Let N_i denote the total number of visits to state $i \in S$ during an evolution of the MC (over an infinite amount of time), assuming that i is also the initial state. It can be shown using routine arguments (see [32], p. 123) that N_i has a *geometric distribution* given by:

$$P(N_i = m) = (f^*(j,j))^{m-1}\,(1 - f^*(j,j)) \qquad (m = 1, 2, ...). \tag{A.1.7}$$

As an immediate consequence of (A.1.7), one may write:

$$P(N_i < \infty) = \begin{cases} 1, & \text{if } f^*(j,j) < 1, \\ 0, & \text{if } f^*(j,j) = 1. \end{cases} \tag{A.1.8}$$

Therefore a transient state will be visited only finitely many times during the evolution of an MC, whereas a recurrent state will be visited infinitely often. Another property to note is that if i is a recurrent state, then all states accessible from i are also recurrent (see [32], p. 130).

The next definition provides a further classification scheme for recurrent states.

Definition A.1.10. ***(Positive recurrence and null recurrence)***

Let $i \in S$ be a recurrent state, and let $m(i,i)$ denote the expected time taken (i.e. the number of time steps) for a first return to state $i \in S$. Then:

- *If $m(i,i) = \infty$, the state i is said to be* null recurrent*.*
- *If $m(i,i) < \infty$, the state i is said to be* positive recurrent*.*

It is proved in [32] (p. 132) that if C is a closed set of finitely many states, then no state in C is null recurrent. Hence, a Markov chain whose state space S is *finite* cannot include null recurrent states. The next result is stated as a theorem; a proof may be found in [31] (p. 31).

*** Theorem A.1.11.** *Let R be an irreducible class. Then either all states in R are transient, or all states in R are null recurrent, or all states in R are positive recurrent. It is said that positive recurrence, null recurrence and transience are all* class properties*.*

The next property to be discussed here is periodicity.

Definition A.1.12. ***(Periodicity)***

Let $R \subseteq S$ be a closed irreducible class containing state $i \in S$, and let d be the greatest common divisor of all $m \in \mathbb{N}$ for which $p^{(m)}(i,i) > 0$. The integer d is referred to as the period *of state i. If $d \geq 2$ then R is said to be* periodic. *If $d = 1$ then R is said to be* aperiodic.

Periodicity is also a class property, and therefore all states in R must have the same period (see [31], p. 13). Note that if $p(i,i) > 0$, this immediately implies that R is aperiodic.

Definition A.1.13. ***(Ergodicity)***

An aperiodic, positive recurrent class of states is called ergodic.

For two states $i, j \in S$, let $v^{(n)}(i,j)$ be defined as follows:

$$v^{(n)}(i,j) =: \frac{1}{n} \sum_{m=0}^{n-1} p^{(m)}(i,j). \tag{A.1.9}$$

Thus, $v^{(n)}(i,j)$ denotes the expected number of visits to state j per unit time in the time interval $[0, n-1]$, when starting in state i. It is proved in [31] (p. 33) that $\pi(j) := \lim_{n\to\infty} v^{(n)}(j,j)$ exists. The limiting quantity $\pi(j)$ will be referred to as the *steady-state probability* of being in state j, and this may be regarded as the limiting average number of visits to j per unit time.

This appendix will conclude with two further results from the literature which have important practical implications. Concise proofs of both results may be found in [32] (p. 152).

*** Theorem A.1.14.** *Let R be an ergodic (i.e. irreducible and aperiodic) class of states. Then all states are positive recurrent if and only if the system of linear equations:*

$$\pi(j) = \sum_{i\in R} p(i,j)\pi(i), \qquad \sum_{j\in R} \pi(j) = 1, \tag{A.1.10}$$

has a solution. If there exists a solution to (A.1.10), then it is strictly positive, there are no other solutions and it is the case that $\pi(j) = \lim_{n\to\infty} p^{(n)}(i,j)$ for all states $i, j \in R$.

*** Corollary A.1.15.** *Let P be the transition probability matrix for an aperiodic, positive recurrent class of states R. Then $\lim_{n\to\infty} P^n$ exists and:*

$$\lim_{n\to\infty} P^n = \mathbf{e}\boldsymbol{\pi},$$

where $\mathbf{e}$ *is a column vector of ones, and* $\boldsymbol{\pi}$ *is the row vector with components* $\pi(j)$*.*

Theorem A.1.14 implies that $\{\pi(j)\}_{j\in S}$, which is referred to as the *stationary distribution* of the MC, can be found by solving a system of linear equations. Note that the steady-state probability $\pi(j)$ of the MC being in state j is independent of the initial state X_0. A probability distribution $\boldsymbol{\pi}$ which satisfies the vector equation $\boldsymbol{\pi} = \boldsymbol{\pi}P$ is also called an *invariant distribution*.

The matrix $P^* := \lim_{n\to\infty} P^n$ is known as the *limiting matrix*. By Corollary A.1.15, the stationary distribution may be approximated by inspecting the matrix P^n when n is large.

A.2 Continuous-time Markov chains

This appendix mainly follows the development in Puterman [141] and Ross [146]. Let S denote a countable set of states, and let $X = \{X(t) : t \geq 0\}$ be a stochastic process with state space S. The process $X = \{X(t) : t \geq 0\}$ is said to be a *continuous-time Markov chain* (CTMC) if:

$$P(X(t+u) = j | X(u) = i, X(v) = x(v), 0 \leq v < u) = P(X(t+u) = j | X(u) = i), \tag{A.2.1}$$

for any $u \geq 0$ and $i, j, x(v) \in S$. In addition, the CTMC is *time homogeneous* if:

$$P(X(t+u) = j | X(u) = i) = P(X(t) = j | X(0) = i), \tag{A.2.2}$$

for all $u \geq 0$. Suppose the process begins in state $i \in S$ at time 0 and remains there until time t, for some $t > 0$. The probability that the process then remains in state i until time $t + u$, for some $u > 0$, must necessarily (by the Markovian property) be the same as the unconditional probability that it remains in state i for a period of at least u. That is, if T_i is used to denote the amount of time that the process spends in state i before transferring to a different state, then:

$$P(T_i > t + u | T_i > t) = P(T_i > u). \tag{A.2.3}$$

It follows that the random variable T_i is *memoryless* and must therefore be *exponentially distributed* (see [146], p. 284 for further explanation). A CTMC may be regarded as a stochastic process that moves from state to state in a similar manner to that of a discrete-time Markov chain (see Appendix A.1), except that the amount of time that it spends in each state before proceeding to the next state is exponentially distributed with a parameter that may be state-dependent.

Suppose one has $X(t) = i \in S$ at some time t. In this case, the process remains in state i for a period of time determined by an exponential distribution with parameter $\tau(i)$, where $0 < \tau(i) < \infty$, and then jumps to state $j \in S$ with probability $\sigma(i,j)$. The parameter $\tau(i)$ is referred to as the *sojourn time parameter*. Let the *infinitesimal generator* $q(i,j)$ be given by:

$$q(i,j) = \begin{cases} -(1-\sigma(i,i))\tau(i), & \text{if } j = i, \\ \sigma(i,j)\tau(i), & \text{if } j \neq i. \end{cases} \tag{A.2.4}$$

The $q(i,j)$ values are sometimes referred to as *infinitesimal transition rates*, and the matrix Q of size $|S| \times |S|$ with components $q(i,j)$ is called the *generator matrix* of the CTMC. The generator matrix Q determines the probability distribution of the system state by means of the following differential equations, known as the *Kolmogorov equations* (see [171], p. 180):

$$\frac{d}{dt}\, p^{(t)}(i,j) = \sum_{k \in S} q(i,k)\, p^{(t)}(k,j), \tag{A.2.5}$$

where the notation $p^{(t)}(i,j)$ has been used to denote the conditional probability $P(X(t) = j | X(0) = i)$. It can be proved that $p^{(t)}(i,j)$ has a finite derivative for all pairs (i,j) (see [16], p. 266). Moreover, if δ_{ij} denotes the Kronecker delta function (defined in (A.1.2)), then:

$$q(i,j) = \lim_{t \to 0} \frac{p^{(t)}(i,j) - p^{(0)}(i,j)}{t} = \lim_{t \to 0} \frac{p^{(t)}(i,j) - \delta_{ij}}{t}. \tag{A.2.6}$$

The properties of transience and recurrence, discussed in Appendix A.1, can also be defined in the continuous-time case. Let $f(i,j)$ denote the 'first passage time' from state i to state j without visiting j in the meantime, and let F denote its distribution function. That is:

$$f(i,j) := \min_{t \geq 0} \left\{ P(X(t) = j | X(0) = i) \right\}, \tag{A.2.7}$$

$$F^{(t)}(i,j) := P(f(i,j) < t) \qquad (t > 0). \tag{A.2.8}$$

The state $i \in S$ is said to be *recurrent* if:

$$\lim_{t \to \infty} F^{(t)}(i,i) = 1. \tag{A.2.9}$$

A recurrent state i is said to be *positive recurrent* if $E\left[f(i,i)\right] < \infty$, and *null recurrent* if $E\left[f(i,i)\right] = \infty$. If i is not recurrent, then it is *transient*. Equivalently, i is transient if and only if:

$$\int_0^\infty p^{(t)}(i,i)\; dt < \infty. \tag{A.2.10}$$

One may also adapt the terminology used for discrete-time Markov chains by saying that two states $i, j \in S$ *communicate* with each other if, for some $t > 0$, $p^{(t)}(i,j) > 0$ and $p^{(t)}(j,i) > 0$. A CTMC is *irreducible* if each pair of states communicates. The next theorem, found in [125] (p. 20), establishes sufficient conditions for the existence of a stationary distribution.

*** Theorem A.2.1.** ***(Ergodic theorem)***

Let $\{X(t) : t \geq 0\}$ be an irreducible continuous-time Markov chain, and suppose all states are positive recurrent. Then, for all states $i, j \in S$, the following limit exists:

$$\lim_{t\to\infty} p^{(t)}(i,j) =: \pi(j).$$

Furthermore, this limit is independent of the initial state $i \in S$. The probability distribution $\{\pi(i)\}_{i\in S}$ is given by the solution of the equations:

$$\sum_{i\in S} \pi(i)q(i,j) = 0 \qquad (j \in S),$$

subject to the normalising condition $\sum_{i\in S} \pi(i) = 1$.

It follows from the previous arguments that two processes with identical infinitesimal generators and initial distributions are also identical with respect to their probabilistic structures. This appendix will conclude by briefly describing an important technique known as *uniformisation*, which allows a CTMC to be modelled using an equivalent discrete-time Markov chain.

Resnick [142] provides the following intuitive description of uniformisation: "If a continuous-time Markov chain can be constructed by placing the transitions of a discrete-time Markov chain at times of Poisson jumps, then we call the continuous-time process *uniformisable*". In fact, any process in which the set of sojourn time parameters $\{\tau(i)\}_{i\in S}$ is bounded above can be uniformised (see [142], p. 438), and this implies that any CTMC with a finite state space S can be uniformised. The remainder of this appendix will provide details of the uniformisation procedure.

Assuming that $\sup_{i\in S} \tau(i) < \infty$, one can choose a constant $C < \infty$ satisfying:

$$\sup_{i\in S}(1 - \sigma(i,i))\tau(i) \leq C < \infty. \tag{A.2.11}$$

It is then possible to define a new process $\{\hat{X}(t) : t \geq 0\}$ with *state-independent* sojourn time parameters $\hat{\tau}(j) = C$ and transition probabilities $u(i,j)$ for $i, j \in S$ given by:

$$u(i,j) = \begin{cases} 1 - \dfrac{(1-\sigma(i,i))\tau(i)}{C}, & \text{if } j = i, \\ \dfrac{\sigma(i,j)\tau(i)}{C}, & \text{if } j \neq i. \end{cases} \tag{A.2.12}$$

The matrix U with elements $u(i,j)$ may be referred to as the *uniformised transition matrix*. By referring to (A.2.4) and using an analogous definition for the infinitesimal generators $\hat{q}(i,j)$ of the new process, one can verify that the generator matrix $\hat{Q}$ of $\hat{X}$ satisfies:

$$\hat{Q} = Q. \tag{A.2.13}$$

That is, the two processes are equal in distribution. The process $\hat{X}$ is referred to as the *uniformisation* of X because it has an identical sojourn time distribution in every state.

Puterman [141] (p. 562) notes that the uniformised CTMC may be regarded as an "equivalent process, in which the system state is observed at random times which are exponentially distributed with parameter C. Because of the Markov property, it begins anew at each observation point." He also points out: "As a result of the definition of C, we observe the system state more frequently in the uniformisation than in the original system and, as a result, increase the probability that the system occupies the same state at different observation times. Alternatively, this transformation may be viewed as inducing extra or 'fictitious' transitions from a state to itself."

After applying uniformisation to a CTMC, it is then possible to apply results from Appendix A.1 to examine its limiting behaviour by analysing the equivalent discrete-time MC.

A.3 Proofs for results in Chapter 3

Proof of Theorem 3.4.4.

This result is proved by Ross [145] (p. 60), but the proof is given in the context of a cost minimisation problem in which all costs are assumed to be non-negative. The uniformised MDP Φ can be transformed into a process $\Phi^{\dagger}$ which is identical to Φ except that single-step (discounted) rewards are given for state-action pairs $((\mathbf{x},\omega),a) \in \mathcal{S} \times \mathcal{A}_{(\mathbf{x},\omega)}$ by the *non-negative* function:

$$\mathcal{R}^{\dagger}_{\phi}((\mathbf{x},\omega),a) := \alpha^{*} - \mathcal{R}_{\phi}((\mathbf{x},\omega),a), \tag{A.3.1}$$

where $\alpha^* = \max(\alpha_1, \alpha_2, ..., \alpha_N)$. It can be checked using (3.3.12) that $\mathcal{R}^\dagger_\phi((\mathbf{x}, \omega), a) \geq 0$ for all pairs $((\mathbf{x}, \omega), a) \in \mathcal{S} \times \mathcal{A}_{(\mathbf{x},\omega)}$. Let θ be an arbitrary policy. It is clear from (A.3.1) that the expected total discounted reward $v^\dagger_{\phi,\theta}((\mathbf{x}, \omega))$ earned by θ in the transformed process $\Phi^\dagger$, given an initial state $(\mathbf{x}, \omega) \in \mathcal{S}$, is related to the corresponding value $v_{\phi,\theta}((\mathbf{x}, \omega))$ as follows:

$$v^\dagger_{\phi,\theta}((\mathbf{x}, \omega)) = \frac{\alpha^*}{1 - \phi} - v_{\phi,\theta}((\mathbf{x}, \omega)). \tag{A.3.2}$$

Thus, any policy θ which maximises $v_{\phi,\theta}((\mathbf{x}, \omega))$ over all admissible policies must also *minimise* $v^\dagger_{\phi,\theta}((\mathbf{x}, \omega))$; that is, maximising the expected total discounted reward for the original process Φ is equivalent to *minimising* the corresponding value for the new process $\Phi^\dagger$. Next, let $v^{\dagger(0)}_\phi((\mathbf{x}, \omega)) = 0$ for each $(\mathbf{x}, \omega) \in \mathcal{S}$ and define $v^{\dagger(n+1)}_\phi((\mathbf{x}, \omega))$ for integers $n \geq 0$ as follows:

$$v^{\dagger(n+1)}_\phi((\mathbf{x}, \omega)) = \min_{a \in \mathcal{A}_{(\mathbf{x},\omega)}} \left\{ \mathcal{R}^\dagger_\phi((\mathbf{x}, \omega), a) + \phi \sum_{(\mathbf{x}', \omega') \in \mathcal{S}} \mathcal{P}((\mathbf{x}, \omega), a, (\mathbf{x}', \omega'))\, v^{\dagger(n)}_\phi((\mathbf{x}', \omega')) \right\}. \tag{A.3.3}$$

Inductively, it can be shown that for all $n \geq 0$, the finite-stage values $v^{\dagger(n)}_\phi((\mathbf{x}, \omega))$ are related to the values $v^{(n)}_\phi((\mathbf{x}, \omega))$ defined in the statement of the theorem as follows:

$$v^{\dagger(n)}_\phi((\mathbf{x}, \omega)) = \alpha^* \left(\frac{1 - \phi^n}{1 - \phi} \right) - v^{(n)}_\phi((\mathbf{x}, \omega)). \tag{A.3.4}$$

Since the rewards $\mathcal{R}^\dagger_\phi((\mathbf{x}, \omega), a)$ are non-negative, the proof in [145] (p. 60) can be applied in order to show that $\lim_{n\to\infty} v^{\dagger(n)}_\phi((\mathbf{x}, \omega)) = v^\dagger_\phi((\mathbf{x}, \omega))$ for each $(\mathbf{x}, \omega) \in \mathcal{S}$, where $v^\dagger_\phi((\mathbf{x}, \omega)) = \inf_\theta v^\dagger_{\phi,\theta}((\mathbf{x}, \omega))$ is the optimal (minimal) expected total discounted reward for the transformed process $\Phi^\dagger$. Hence, by taking limits as $n \to \infty$ in (A.3.4), one finds that for all states $(\mathbf{x}, \omega) \in \mathcal{S}$:

$$\lim_{n\to\infty} v^{(n)}_\phi((\mathbf{x}, \omega)) = \frac{\alpha^*}{1 - \phi} - v^\dagger_\phi((\mathbf{x}, \omega)) \tag{A.3.5}$$

However, the right-hand side of (A.3.5) is equal to the optimal expected total discounted reward $v_\phi((\mathbf{x}, \omega))$ for the *original* process Φ, due to (A.3.2) and the previously-noted fact that minimising $v^\dagger_{\phi,\theta}((\mathbf{x}, \omega))$ is equivalent to maximising $v_{\phi,\theta}((\mathbf{x}, \omega))$. This completes the proof. □

Proof of Lemma 3.4.6.

First, let the vector $\mathbf{x} \in \mathbb{N}_0^N$ and facilities $i, j \in \{1, 2, ..., N\}$ (with $i \neq j$) be arbitrary. The easiest way to prove (3.4.9) and (3.4.10) would involve arguing directly via the continuous-time optimality equations (3.4.1). However, for the purposes of later results, it will be useful to begin

by considering the discrete-time MDP Φ obtained from Ψ via uniformisation, and prove using the method of successive approximations (Theorem 3.4.4) that for all stages $n \in \mathbb{N}$:

$$\begin{aligned} &v_\phi^{(n)}((\mathbf{x}^{a_n^*+}, M_i)) - v_\phi^{(n)}((\mathbf{x}, \Lambda)) \leq \alpha_i, \\ &v_\phi^{(n)}((\mathbf{x}, M_i)) - v_\phi^{(n)}((\mathbf{x}, M_j)) \leq \max(\alpha_i - \alpha_j,\, 0), \end{aligned} \tag{A.3.6}$$

where $\phi = 1/(1+\gamma\Delta)$ is the discount factor for the process Φ, and $a_n^* \in \{0, 1, ..., N\}$ is an action which attains the maximum on the right-hand side of (3.4.5) for the state $(\mathbf{x}, \Lambda)$ with n replaced by $n-1$; that is, a_n^* is an optimal action at $(\mathbf{x}, \Lambda)$ in a finite-stage problem with n stages. Let $q_\phi^{(n)}((\mathbf{x}, \omega), a)$ be defined for $(\mathbf{x}, \omega) \in \mathcal{S}$, $a \in \{0, 1, ..., N\}$ and $n \in \mathbb{N}$ as follows:

$$q_\phi^{(n)}((\mathbf{x}, \omega), a) := \mathcal{R}_\phi((\mathbf{x}, \omega), a) + \phi \sum_{(\mathbf{x}', \omega') \in \mathcal{S}} \mathcal{P}((\mathbf{x}, \omega), a, (\mathbf{x}', \omega'))\, v_\phi^{(n-1)}((\mathbf{x}', \omega')). \tag{A.3.7}$$

Hence, (3.4.5) implies that $v_\phi^{(n)}((\mathbf{x}, \omega)) = \max_a q_\phi^{(n)}((\mathbf{x}, \omega), a)$ for all $n \in \mathbb{N}$. It will therefore be sufficient to show that for all actions $a \in \{0, 1, ..., N\}$ and integers $n \in \mathbb{N}$:

$$\begin{aligned} &v_\phi^{(n)}((\mathbf{x}^{a+}, M_i)) - q_\phi^{(n)}((\mathbf{x}, \Lambda), a) \leq \alpha_i, \\ &v_\phi^{(n)}((\mathbf{x}, M_i)) - v_\phi^{(n)}((\mathbf{x}, M_j)) \leq \max(\alpha_i - \alpha_j,\, 0). \end{aligned} \tag{A.3.8}$$

In the successive approximations method one assumes that $v_\phi^{(0)}((\mathbf{x}, \omega)) = 0$ for all $(\mathbf{x}, \omega) \in \mathcal{S}$, so in order to show that the properties (A.3.8) hold when $n = 1$ it suffices to show:

$$\begin{aligned} &\mathcal{R}_\phi((\mathbf{x}^{a+}, M_i), 0) - \mathcal{R}_\phi((\mathbf{x}, \Lambda), a) \leq \alpha_i, \\ &\mathcal{R}_\phi((\mathbf{x}, M_i), 0) - \mathcal{R}_\phi((\mathbf{x}, M_j), 0) \leq \max(\alpha_i - \alpha_j,\, 0), \end{aligned} \tag{A.3.9}$$

where $a \in \{0, 1, ..., N\}$ is arbitrary. Indeed, using the definition of $\mathcal{R}_\phi(\cdot)$ in (3.3.12):

$$\begin{aligned} \mathcal{R}_\phi((\mathbf{x}^{a+}, M_i), 0) - \mathcal{R}_\phi((\mathbf{x}, \Lambda), a) &= \alpha_i \left(1 - \phi \left(1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j^{a+}, c_j)\mu_j\Delta\right)\right) \\ &\leq \alpha_i, \end{aligned} \tag{A.3.10}$$

where the inequality is due to the definition of Δ in (3.3.9), which implies that the expression $1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j^{a+}, c_j)\mu_j\Delta$ is non-negative. Similarly, using (3.3.12):

$$\mathcal{R}_\phi((\mathbf{x}, M_i), 0) - \mathcal{R}_\phi((\mathbf{x}, M_j), 0) = (\alpha_i - \alpha_j) \left(1 - \phi \left(1 - \lambda\Delta - \sum_{m=1}^{N} \min(x_m, c_m)\mu_m\Delta\right)\right). \tag{A.3.11}$$

The expression on the right-hand side of (A.3.11) is bounded above by zero if $\alpha_i < \alpha_j$, and by $\alpha_i - \alpha_j$ otherwise. So, both of the properties (A.3.8) hold when $n = 1$. Assume, as an inductive hypothesis, that they also hold when $n = k$ for arbitrary $k \in \mathbb{N}_0$. It can be verified using (3.3.8) and (3.3.10) that the transition probabilities from states $(\mathbf{x}^{a+}, M_i)$ and $(\mathbf{x}, \Lambda)$ (assuming, in the latter case, that action a is chosen) are identical, except that a 'self-transition' from state $(\mathbf{x}^{a+}, M_i)$ to itself occurs with probability $1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j^{a+}, c_j)\mu_j\Delta$, while a self-transition from $(\mathbf{x}, \Lambda)$ to itself also occurs with the same probability. Hence, following the cancellations of transition probabilities to other states, one finds that $v_\phi^{(k+1)}((\mathbf{x}^{a+}, M_i)) - q_\phi^{(k+1)}((\mathbf{x}, \Lambda), a)$ must satisfy:

$$v_\phi^{(k+1)}((\mathbf{x}^{a+}, M_i)) - q_\phi^{(k+1)}((\mathbf{x}, \Lambda), a) \leq \mathcal{R}_\phi((\mathbf{x}^{a+}, M_i), 0) - \mathcal{R}_\phi((\mathbf{x}, \Lambda), a) + \phi \left(1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j^{a+}, c_j)\mu_j\Delta\right) \left(v_\phi^{(k)}((\mathbf{x}^{a+}, M_i)) - v_\phi^{(k)}((\mathbf{x}, \Lambda))\right). \tag{A.3.12}$$

Then, by noting that $v_\phi^{(k)}((\mathbf{x}^{a+}, M_i)) - v_\phi^{(k)}((\mathbf{x}, \Lambda))$ is bounded above by $v_\phi^{(k)}((\mathbf{x}^{a+}, M_i)) - q_\phi^{(k)}((\mathbf{x}, \Lambda), a)$ and $v_\phi^{(k)}((\mathbf{x}^{a+}, M_i)) - q_\phi^{(k)}((\mathbf{x}, \Lambda), a) \leq \alpha_i$ by the inductive assumption, and also recalling the property (A.3.10), one infers from (A.3.12):

$$\begin{aligned} v_\phi^{(k+1)}((\mathbf{x}^{a+}, M_i)) - v_\phi^{(k+1)}((\mathbf{x}, \Lambda)) &\leq \alpha_i \left(1 - \phi \left(1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j^{a+}, c_j)\mu_j\Delta\right)\right) \\ &\quad + \alpha_i\phi \left(1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j^{a+}, c_j)\mu_j\Delta\right) \\ &= \alpha_i. \end{aligned} \tag{A.3.13}$$

This proves by induction that the first property in (A.3.8) holds for all $n \in \mathbb{N}$, which (as noted previously) is sufficient to show that the first property in (A.3.6) holds for all $n \in \mathbb{N}$. The arguments used to establish the second property are similar. Indeed, the transition probabilities from states $(\mathbf{x}, M_i)$ and $(\mathbf{x}, M_j)$ are identical except when one considers self-transitions, which occur (in both cases) with probability $1 - \lambda\Delta - \sum_{m=1}^{N} \min(x_m, c_m)\mu_m\Delta$. Hence, one finds:

$$v_\phi^{(k+1)}((\mathbf{x}, M_i)) - v_\phi^{(k+1)}((\mathbf{x}, M_j)) \leq \mathcal{R}_\phi((\mathbf{x}, M_i), 0) - \mathcal{R}_\phi((\mathbf{x}, M_j), 0) + \phi \left(1 - \lambda\Delta - \sum_{m=1}^{N} \min(x_m, c_m)\mu_m\Delta\right) \left(v_\phi^{(k)}((\mathbf{x}, M_i)) - v_\phi^{(k)}((\mathbf{x}, M_j))\right). \tag{A.3.14}$$

By the inductive assumption, $v_\phi^{(k)}((\mathbf{x}, M_i)) - v_\phi^{(k)}((\mathbf{x}, M_j))$ is bounded above by $\max(\alpha_i - \alpha_j,\ 0)$. Hence, by also using (A.3.11), one obtains the following from (A.3.14):

$$v_\phi^{(k+1)}((\mathbf{x}, M_i)) - v_\phi^{(k+1)}((\mathbf{x}, M_j)) \leq \max(\alpha_i - \alpha_j,\ 0),$$

which confirms that the second property in (A.3.6) also holds for all $n \in \mathbb{N}$. By applying the result of Theorem 3.4.4 and recalling the equivalence between v_ϕ and v_γ, one then obtains:

$$\begin{aligned} &v_\gamma((\mathbf{x}^{a^*+}, M_i)) - v_\gamma((\mathbf{x}, \Lambda)) \leq \alpha_i, \\ &v_\gamma((\mathbf{x}, M_i)) - v_\gamma((\mathbf{x}, M_j)) \leq \max(\alpha_i - \alpha_j,\ 0). \end{aligned} \tag{A.3.15}$$

To prove the lemma, it remains to be shown that the equalities (3.4.9) and (3.4.10) hold. As mentioned earlier, it is easiest to argue using the continuous-time optimality equations (3.4.1). As before, assume that action $a^* \in \{0, 1, ..., N\}$ is chosen at state $(\mathbf{x}, \Lambda)$ by the γ-discount optimal policy θ_γ^*. It can be checked using (3.2.9) that the sojourn time parameters $\tau((\mathbf{x}^{a^*+}, M_i), 0)$ and $\tau((\mathbf{x}, \Lambda), a^*)$ are equal. Moreover, it can also be verified using (3.2.10)-(3.2.11) that, for any state $(\mathbf{x}', \omega') \in \mathcal{S}$, the transition probabilities $\sigma((\mathbf{x}^{a^*+}, M_i), 0, (\mathbf{x}', \omega'))$ and $\sigma((\mathbf{x}, \Lambda), a^*, (\mathbf{x}', \omega'))$ are equal. Therefore (3.4.1) implies that $v_\gamma((\mathbf{x}^{a^*+}, M_i)) - v_\gamma((\mathbf{x}, \Lambda))$ is simply equal to the difference in the reward terms, $\xi_\gamma((\mathbf{x}^{a^*+}, M_i), 0) - \xi_\gamma((\mathbf{x}, \Lambda), a^*)$. Hence, using (3.2.18), one finds:

$$v_\gamma((\mathbf{x}^{a^*+}, M_i)) - v_\gamma((\mathbf{x}, \Lambda)) = \xi_\gamma((\mathbf{x}^{a^*+}, M_i), 0) - \xi_\gamma((\mathbf{x}, \Lambda), a^*) = \alpha_i.$$

Similarly, for any N-vector $\mathbf{x} \in \mathbb{N}_0^N$ and facilities $i, j \in \{1, 2, ..., N\}$ with $i \neq j$, the sojourn time parameters and transition probabilities for the state-action pairs $((\mathbf{x}, M_i), 0)$ and $((\mathbf{x}, M_j), 0)$ are equal. Hence, using the same reasoning as above, the result follows:

$$v_\gamma((\mathbf{x}, M_i)) - v_\gamma((\mathbf{x}, M_j)) = \xi_\gamma((\mathbf{x}, M_i), 0) - \xi_\gamma((\mathbf{x}, M_j), 0) = \alpha_i - \alpha_j. \qquad \square$$

Proof of Lemma 3.4.7.

By Theorem 3.3.3, it is sufficient to consider the discrete-time MDP Φ obtained by uniformisation and prove that (3.4.12) holds with $v_\gamma(\cdot)$ replaced by $v_\phi(\cdot)$. Let the finite-stage approximation functions $v_\phi^{(n)}$ be defined for $n \in \mathbb{N}_0$ as in the statement of Theorem 3.4.4. Then it is sufficient to show that for all $(\mathbf{x}, \omega) \in \mathcal{S}$, $i \in \{1, 2, ..., N\}$, $\phi \in (0, 1)$ and integers $n \geq 0$:

$$v_\phi^{(n)}((\mathbf{x}^{i+}, \omega)) - v_\phi^{(n)}((\mathbf{x}, \omega)) \leq \alpha^* - \frac{\beta_i}{\mu_i}. \tag{A.3.16}$$

The property (A.3.16) may be proved using induction on the 'stage counter' variable n. It holds trivially when $n = 0$ due to the convention that $v_\phi^{(0)}((\mathbf{x},\omega)) = 0$ for all $(\mathbf{x},\omega) \in \mathcal{S}$, so one may assume (as an inductive hypothesis) that it also holds at some arbitrary stage $n = k \in \mathbb{N}_0$. For arbitrary $(\mathbf{x},\omega) \in \mathcal{S}$ and $i \in \{1,2,...,N\}$, let $a_0 \in \mathcal{A}_{(\mathbf{x},\omega)}$ and $a_1 \in \mathcal{A}_{(\mathbf{x}^{i+},\omega)}$ be actions attaining the maximum on the right-hand side of the finite-stage optimality equations (3.4.5) for states $(\mathbf{x},\omega)$ and $(\mathbf{x}^{i+},\omega)$ respectively at stage $n = k+1$. In other words, a_0 and a_1 satisfy:

$$a_0 \in \underset{a\in\mathcal{A}_{(\mathbf{x},\omega)}}{\arg\max} \left\{ \mathcal{R}_\phi((\mathbf{x},\omega),a) + \phi \sum_{(\mathbf{x}',\omega')\in\mathcal{S}} \mathcal{P}((\mathbf{x},\omega),a,(\mathbf{x}',\omega'))\, v_\phi^{(k)}((\mathbf{x}',\omega')) \right\}, \tag{A.3.17}$$

$$a_1 \in \underset{a\in\mathcal{A}_{(\mathbf{x}^{i+},\omega)}}{\arg\max} \left\{ \mathcal{R}_\phi((\mathbf{x}^{i+},\omega),a) + \phi \sum_{(\mathbf{x}',\omega')\in\mathcal{S}} \mathcal{P}((\mathbf{x}^{i+},\omega),a,(\mathbf{x}',\omega'))\, v_\phi^{(k)}((\mathbf{x}',\omega')) \right\}.$$

Then, the difference $v_\phi^{(k+1)}((\mathbf{x}^{i+},\omega)) - v_\phi^{(k+1)}((\mathbf{x},\omega))$ may be bounded above as follows:

$$\begin{aligned}
v_\phi^{(k+1)}((\mathbf{x}^{i+},\omega)) - v_\phi^{(k+1)}((\mathbf{x},\omega)) &= \mathcal{R}_\phi((\mathbf{x}^{i+},\omega),a_1) - \mathcal{R}_\phi((\mathbf{x},\omega),a_0) \\
&\quad + \phi \sum_{(\mathbf{x}',\omega')\in\mathcal{S}} \Big(\mathcal{P}((\mathbf{x}^{i+},\omega),a_1,(\mathbf{x}',\omega'))\, v_\phi^{(k)}((\mathbf{x}',\omega')) - \mathcal{P}((\mathbf{x},\omega),a_0,(\mathbf{x}',\omega'))\, v_\phi^{(k)}((\mathbf{x}',\omega')) \Big) \\
&\leq \mathcal{R}_\phi((\mathbf{x}^{i+},\omega),a_1) - \mathcal{R}_\phi((\mathbf{x},\omega),a_1) \\
&\quad + \phi \sum_{(\mathbf{x}',\omega')\in\mathcal{S}} \Big(\mathcal{P}((\mathbf{x}^{i+},\omega),a_1,(\mathbf{x}',\omega'))\, v_\phi^{(k)}((\mathbf{x}',\omega')) - \mathcal{P}((\mathbf{x},\omega),a_1,(\mathbf{x}',\omega'))\, v_\phi^{(k)}((\mathbf{x}',\omega')) \Big),
\end{aligned} \tag{A.3.18}$$

where the inequality is due to the superiority of action a_0 to a_1 at state $(\mathbf{x},\omega)$ in a finite-horizon problem with $k+1$ stages, as implied by (A.3.17). By writing out the transition probabilities

$\mathcal{P}((\mathbf{x}^{i+},\omega),a_1,(\mathbf{x}',\omega'))$ and $\mathcal{P}((\mathbf{x},\omega),a_1,(\mathbf{x}',\omega'))$ in full, one then obtains:

$$\begin{aligned}
v_\phi^{(k+1)}((\mathbf{x}^{i+},\omega)) - v_\phi^{(k+1)}((\mathbf{x},\omega)) &\leq \mathcal{R}_\phi((\mathbf{x}^{i+},\omega),a_1) - \mathcal{R}_\phi((\mathbf{x},\omega),a_1) \\
&+ \phi\lambda\Delta\left(v_\phi^{(k)}(((\mathbf{x}^{i+})^{a_1+},\Lambda)) - v_\phi^{(k)}((\mathbf{x}^{a_1+},\Lambda))\right) \\
&+ \phi\sum_{j=1}^{N}\min(x_j^{a_1+},c_j)\mu_j\Delta\left(v_\phi^{(k)}((((\mathbf{x}^{i+})^{a_1+})^{j-},M_j)) - v_\phi^{(k)}(((\mathbf{x}^{a_1+})^{j-},M_j))\right) \\
&+ \phi I(x_i<c_i)\mu_i\Delta\left(v_\phi^{(k)}((\mathbf{x}^{a_1+},M_i)) - v_\phi^{(k)}((\mathbf{x},\omega))\right) \\
&+ \phi\left[1-\lambda\Delta-\sum_{j=1}^{N}\min(x_j^{a_1+},c_j)\mu_j\Delta - I(x_i<c_i)\mu_i\Delta\right]\left(v_\phi^{(k)}((\mathbf{x}^{i+},\omega)) - v_\phi^{(k)}((\mathbf{x},\omega))\right),
\end{aligned} \tag{A.3.19}$$

where I is the indicator function, and the notational convention has been assumed whereby $\mathbf{x}^{0+}=\mathbf{x}$. Note that this convention ensures that (A.3.19) makes sense when $\omega = M_j$ for any $j\in\{1,2,...,N\}$, since in that case the action sets $A_{(\mathbf{x},\omega)}$ and $A_{(\mathbf{x}^{i+},\omega)}$ are both singletons containing only the 'dummy' action 0, so one must have $a_1=0$. The indicator term $I(x_i<c_i)\mu_i\Delta$ arises because the state $(\mathbf{x}^{i+},\omega)$ may (or may not) have one extra service in progress in comparison to $(\mathbf{x},\omega)$, depending on whether or not $x_i<c_i$. However, the condition for Δ in (3.3.9) ensures that the expression in square brackets in the last line of (A.3.19) is always non-negative. It is also worth noting that the states $(((\mathbf{x}^{i+})^{a_1+})^{j-},M_j)$ and $((\mathbf{x}^{a_1+})^{j-},M_j)$ may not exist if $x_j=0$, but in this case one has $\min(x_j,c_j)\mu_j\Delta=0$, so the line in which those terms appear may be ignored.

By the inductive assumption that (A.3.16) holds for all $(\mathbf{x},\omega)\in\mathcal{S}$ when $n=k$, the differences $v_\phi^{(k)}(((\mathbf{x}^{i+})^{a_1+},\Lambda)) - v_\phi^{(k)}((\mathbf{x}^{a_1+},\Lambda))$, $v_\phi^{(k)}((((\mathbf{x}^{i+})^{a_1+})^{j-},M_j)) - v_\phi^{(k)}(((\mathbf{x}^{a_1+})^{j-},M_j))$ and $v_\phi^{(k)}((\mathbf{x}^{i+},\omega)) - v_\phi^{(k)}((\mathbf{x},\omega))$ are all bounded above by $\alpha^*-\beta_i/\mu_i$. Hence, from (A.3.19):

$$\begin{aligned}
v_\phi^{(k+1)}((\mathbf{x}^{i+},\omega)) - v_\phi^{(k+1)}((\mathbf{x},\omega)) &\leq \mathcal{R}_\phi((\mathbf{x}^{i+},\omega),a_1) - \mathcal{R}_\phi((\mathbf{x},\omega),a_1) \\
&+ \phi I(x_i<c_i)\mu_i\Delta\left(v_\phi^{(k)}((\mathbf{x}^{a_1+},M_i)) - v_\phi^{(k)}((\mathbf{x},\omega))\right) \\
&+ \phi\big(1-I(x_i<c_i)\mu_i\Delta\big)\left(\alpha^*-\frac{\beta_i}{\mu_i}\right).
\end{aligned} \tag{A.3.20}$$

The objective is then to show that the right-hand side of (A.3.20) is also bounded above by $\alpha^*-\beta_i/\mu_i$. There are two possible cases: either $x_i<c_i$ or $x_i\geq c_i$. If $x_i<c_i$, then the indicator term $I(x_i<c_i)$ is equal to one. Here, it is necessary to consider three possible sub-cases: $\omega=\Lambda$, $\omega=M_j$

(where $j \neq i$) and $\omega = M_i$. Firstly, if $\omega = \Lambda$ then (3.3.12) implies the following:

$$\mathcal{R}_\phi((\mathbf{x}^{i+}, \omega), a_1) - \mathcal{R}_\phi((\mathbf{x}, \omega), a_1) = -\phi\Delta\beta_i. \tag{A.3.21}$$

Additionally, the expression $v_\phi^{(k)}((\mathbf{x}^{a_1+}, M_i)) - v_\phi^{(k)}((\mathbf{x}, \Lambda))$ appears on the right-hand side of (A.3.20). It is necessary to find an upper bound for this quantity. Using the same reasoning as discussed in the proof of Lemma 3.4.6, it is possible to assume that the action a_1 is chosen at state $(\mathbf{x}, \Lambda)$ in a k-stage problem, since this assumption will effectively under-estimate the value of $v_\phi^{(k)}((\mathbf{x}, \Lambda))$ and thereby yield an upper bound for $v_\phi^{(k)}((\mathbf{x}^{a_1+}, M_i)) - v_\phi^{(k)}((\mathbf{x}, \Lambda))$ (more formally, one relies upon the principle that $v_\phi^{(k)}((\mathbf{x}^{a_1+}, M_i)) - v_\phi^{(k)}((\mathbf{x}, \Lambda)) \leq v_\phi^{(k)}((\mathbf{x}^{a_1+}, M_i)) - q_\phi^{(k)}((\mathbf{x}, \Lambda), a_1)$, where $q_\phi^{(k)}(\cdot)$ is as defined in (A.3.7)). Hence, using the first of the two properties in (A.3.6):

$$v_\phi^{(k)}((\mathbf{x}^{a_1+}, M_i)) - v_\phi^{(k)}((\mathbf{x}, \Lambda)) \leq \alpha_i \leq \alpha^*. \tag{A.3.22}$$

Therefore, by combining (A.3.21) and (A.3.22), one obtains from (A.3.20):

$$\begin{aligned} v_\phi^{(k+1)}((\mathbf{x}^{i+}, \Lambda)) - v_\phi^{(k+1)}((\mathbf{x}, \Lambda)) &\leq -\phi\Delta\beta_i + \phi\mu_i\Delta\alpha^* + \phi(1 - \mu_i\Delta)\left(\alpha^* - \frac{\beta_i}{\mu_i}\right) \\ &= \phi\left(\alpha^* - \frac{\beta_i}{\mu_i}\right), \end{aligned}$$

and the upper bound $v_\phi^{(k+1)}((\mathbf{x}^{i+}, \Lambda)) - v_\phi^{(k+1)}((\mathbf{x}, \Lambda)) \leq \alpha^* - \beta_i/\mu_i$ follows since $\phi \in (0, 1)$. Next, consider the case where $\omega = M_j$ for some $j \neq i$ (still assuming that $x_i < c_i$). In this case, the action a_1 must be equal to zero since $\mathcal{A}_{(\mathbf{x}, M_j)} = \mathcal{A}_{(\mathbf{x}^{i+}, M_j)} = \{0\}$. By (3.3.12), $\mathcal{R}_\phi((\mathbf{x}^{i+}, M_j), 0) - \mathcal{R}_\phi((\mathbf{x}, M_j), 0) = \phi\Delta(\alpha_j\mu_i - \beta_i)$. Also, given $a_1 = 0$, the difference $v_\phi^{(k)}((\mathbf{x}^{a_1+}, M_i)) - v_\phi^{(k)}((\mathbf{x}, M_j))$ is equal to $v_\phi^{(k)}((\mathbf{x}, M_i)) - v_\phi^{(k)}((\mathbf{x}, M_j))$ (recall the convention that $\mathbf{x}^{0+} = \mathbf{x}$), which is bounded above by $\max(\alpha_i - \alpha_j,\ 0)$ by the second property in (A.3.6). Hence, from (A.3.20):

$$\begin{aligned} v_\phi^{(k+1)}((\mathbf{x}^{i+}, M_j)) - v_\phi^{(k+1)}((\mathbf{x}, M_j)) &\leq \phi\Delta(\alpha_j\mu_i - \beta_i) + \phi\mu_i\Delta\max(\alpha_i - \alpha_j,\ 0) + \phi(1 - \mu_i\Delta)\left(\alpha^* - \frac{\beta_i}{\mu_i}\right) \\ &\leq \phi\mu_i\Delta\left(\alpha^* - \frac{\beta_i}{\mu_i}\right) + \phi(1 - \mu_i\Delta)\left(\alpha^* - \frac{\beta_i}{\mu_i}\right) \\ &\leq \alpha^* - \frac{\beta_i}{\mu_i}. \end{aligned}$$

Finally, suppose $\omega = M_i$. Since $a_1 = 0$, one simply has $v_\phi^{(k)}((\mathbf{x}^{a_1+}, M_i)) - v_\phi^{(k)}((\mathbf{x}, M_i)) = 0$. Hence, since $\mathcal{R}_\phi((\mathbf{x}^{i+}, M_j), 0) - \mathcal{R}_\phi((\mathbf{x}, M_j), 0) = \phi\Delta(\alpha_i\mu_i - \beta_i)$, (A.3.20) implies:

$$\begin{aligned} v_\phi^{(k+1)}((\mathbf{x}^{i+}, M_i)) - v_\phi^{(k+1)}((\mathbf{x}, M_i)) &\leq \phi\Delta(\alpha_i\mu_i - \beta_i) + \phi(1 - \mu_i\Delta)\left(\alpha^* - \frac{\beta_i}{\mu_i}\right) \\ &\leq \alpha^* - \frac{\beta_i}{\mu_i}. \end{aligned}$$

This confirms that, in the case $x_i < c_i$, $v_\phi^{(k+1)}((\mathbf{x}^{i+},\omega)) - v_\phi^{(k+1)}((\mathbf{x},\omega))$ is bounded above by $\alpha^* - \beta_i/\mu_i$ for all possible values of ω. The case $x_i \geq c_i$ is much easier to check, since in this case one has $I(x_i < c_i) = 0$. If $x_i \geq c_i$ then (3.4.7) and (3.4.8) imply that $\mathcal{R}_\phi((\mathbf{x}^{i+},\omega),a_1) - \mathcal{R}_\phi((\mathbf{x},\omega),a_1) = -\phi\Delta\beta_i$, which enables the inequality in (A.3.20) to be tightened as follows:

$$\begin{aligned} v_\phi^{(k+1)}((\mathbf{x}^{i+},\omega)) - v_\phi^{(k+1)}((\mathbf{x},\omega)) &\leq -\phi\Delta\beta_i + \phi\big(1 - I(x_i < c_i)\mu_i\Delta\big)\left(\alpha^* - \frac{\beta_i}{\mu_i}\right) \\ &= -\phi\Delta\beta_i + \phi\left(\alpha^* - \frac{\beta_i}{\mu_i}\right) \\ &\leq \alpha^* - \frac{\beta_i}{\mu_i}, \end{aligned} \tag{A.3.23}$$

so the required upper bound holds for $v_\phi^{(k+1)}((\mathbf{x}^{i+},\omega)) - v_\phi^{(k+1)}((\mathbf{x},\omega))$ in this case also. This completes the inductive proof that (A.3.16) holds for all $(\mathbf{x},\omega) \in \mathcal{S}$, $i \in \{1,2,...,N\}$, $\phi \in (0,1)$ and $n \in \mathbb{N}_0$. In view of Theorem 3.4.4, the result follows by taking limits as $n \to \infty$. $\square$

Proof of Lemma 3.5.1.

Consider a stationary policy θ satisfying the conditions stated by the lemma. As stated by the lemma, the 'aggregate set' $H_{\mathbf{x}}$ is defined for vectors $\mathbf{x} \in S$ as follows:

$$H_{\mathbf{x}} := F_{\mathbf{x}} \cup G_{\mathbf{x}},$$

where the disjoint sets $F_{\mathbf{x}}$ and $G_{\mathbf{x}}$ are given by:

$$\begin{aligned} F_{\mathbf{x}} &:= \big\{(\mathbf{y},\omega) \in \mathcal{S} : \mathbf{y} = \mathbf{x}^{i-} \text{ for some } i \in \{1,2,...,N\},\ \omega = \Lambda \text{ and } \theta((\mathbf{y},\omega)) = i\big\}, \\ G_{\mathbf{x}} &:= \big\{(\mathbf{y},\omega) \in \mathcal{S} : \mathbf{y} = \mathbf{x} \text{ and } \omega = M_i \text{ for some } i \in \{1,2,...,N\}\big\}. \end{aligned} \tag{A.3.24}$$

where $\theta((\mathbf{y},\omega))$ is the action chosen by θ at state $(\mathbf{y},\omega) \in \mathcal{S}$. The dependence of $F_{\mathbf{x}}$ and $H_{\mathbf{x}}$ on θ has been suppressed in order to simplify notation. It should also be noted that $F_{\mathbf{x}}$ may be empty for a given vector $\mathbf{x}$, but $G_{\mathbf{x}}$ (and hence $H_{\mathbf{x}}$) cannot be empty. Let $(\mathbf{x}_n,\omega_n)$ denote the state of the process Φ at time step $n \geq 0$. Using probabilistic arguments which exploit the fact that $F_{\mathbf{x}^{i+}}$ and $G_{\mathbf{x}^{i+}}$ form a partition of $H_{\mathbf{x}^{i+}}$, the conditional probability $P\Big((\mathbf{x}_{n+1},\omega_{n+1}) \in H_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n,\omega_n) \in H_{\mathbf{x}}\Big)$

(for an arbitrary facility $i \in \{1, 2, ..., N\}$) may be decomposed as follows:

$$\begin{aligned}
&P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in H_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big) \\
&= P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in F_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big) + P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in G_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big) \\
&= P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in F_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in F_{\mathbf{x}}\Big) P\Big((\mathbf{x}_n, \omega_n) \in F_{\mathbf{x}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big) \\
&\quad + P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in G_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in F_{\mathbf{x}}\Big) P\Big((\mathbf{x}_n, \omega_n) \in F_{\mathbf{x}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big) \\
&\quad + P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in F_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in G_{\mathbf{x}}\Big) P\Big((\mathbf{x}_n, \omega_n) \in G_{\mathbf{x}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big) \\
&\quad + P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in G_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in G_{\mathbf{x}}\Big) P\Big((\mathbf{x}_n, \omega_n) \in G_{\mathbf{x}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big). \qquad \text{(A.3.25)}
\end{aligned}$$

Here, the conditional probabilities $P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in F_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in F_{\mathbf{x}}\Big)$, $P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in F_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in G_{\mathbf{x}}\Big)$ etc. may be written explicitly using the transition probabilities $\mathcal{P}((\mathbf{x}, \omega), a, (\mathbf{x}', \omega'))$ defined in (3.3.8)-(3.3.10). The resulting expressions are:

$$\begin{aligned}
&P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in F_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in F_{\mathbf{x}}\Big) \\
&= \sum_{\substack{j \in \{1,2,...,N\} \\ \theta((\mathbf{x}^{j-}, \Lambda)) = j}} P\Big((\mathbf{x}_n, \omega_n) = (\mathbf{x}^{j-}, \Lambda) \Big| (\mathbf{x}_n, \omega_n) \in F_{\mathbf{x}}\Big) \sum_{\substack{k \in \{1,2,...,N\} \\ \theta(((\mathbf{x}^{i+})^{k-}, \Lambda)) = k}} \mathcal{P}((\mathbf{x}^{j-}, \Lambda), j, ((\mathbf{x}^{i+})^{k-}, \Lambda)), \qquad \text{(A.3.26)}
\end{aligned}$$

$$\begin{aligned}
&P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in G_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in F_{\mathbf{x}}\Big) \\
&= \sum_{\substack{j \in \{1,2,...,N\} \\ \theta((\mathbf{x}^{j-}, \Lambda)) = j}} P\Big((\mathbf{x}_n, \omega_n) = (\mathbf{x}^{j-}, \Lambda) \Big| (\mathbf{x}_n, \omega_n) \in F_{\mathbf{x}}\Big) \sum_{k=1}^{N} \mathcal{P}((\mathbf{x}^{j-}, \Lambda), j, (\mathbf{x}^{i+}, M_k)), \qquad \text{(A.3.27)}
\end{aligned}$$

$$\begin{aligned}
&P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in F_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in G_{\mathbf{x}}\Big) \\
&= \sum_{j=1}^{N} P\Big((\mathbf{x}_n, \omega_n) = (\mathbf{x}, M_j) \Big| (\mathbf{x}_n, \omega_n) \in G_{\mathbf{x}}\Big) \sum_{\substack{k \in \{1,2,...,N\} \\ \theta(((\mathbf{x}^{i+})^{k-}, \Lambda)) = k}} \mathcal{P}((\mathbf{x}, M_j), 0, ((\mathbf{x}^{i+})^{k-}, \Lambda)), \qquad \text{(A.3.28)}
\end{aligned}$$

$$\begin{aligned}
&P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in G_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in G_{\mathbf{x}}\Big) \\
&= \sum_{j=1}^{N} P\Big((\mathbf{x}_n, \omega_n) = (\mathbf{x}, M_j) \Big| (\mathbf{x}_n, \omega_n) \in G_{\mathbf{x}}\Big) \sum_{k=1}^{N} \mathcal{P}((\mathbf{x}, M_j), 0, (\mathbf{x}^{i+}, M_k)). \qquad \text{(A.3.29)}
\end{aligned}$$

Naturally, any probabilities involving states which are not elements of $\mathcal{S}$ (i.e. states $(\mathbf{x}, \omega)$ for which the vector $\mathbf{x}$ includes negative components) may be omitted from the relevant summations in (A.3.26)-(A.3.29). The expressions (A.3.26)-(A.3.29) can be simplified as follows:

- In (A.3.26), the transition probability $\mathcal{P}((\mathbf{x}^{j-},\Lambda),j,((\mathbf{x}^{i+})^{k-},\Lambda))$ can only be non-zero if $k=i$. In this case, it takes a value of $\lambda\Delta$, but one should also note that it is included in the second summation if and only if $\theta((\mathbf{x},\Lambda))=i$. Since the states $(\mathbf{x}^{j-},\Lambda)$ with $\theta((\mathbf{x}^{j-},\Lambda))=j$ form a partition of the set $F_{\mathbf{x}}$, it follows from these arguments that:

$$P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in F_{\mathbf{x}^{i+}}\Big|(\mathbf{x}_n,\omega_n)\in F_{\mathbf{x}}\Big)=\begin{cases}\lambda\Delta, & \text{if } \theta((\mathbf{x},\Lambda))=i,\\ 0, & \text{otherwise.}\end{cases} \tag{A.3.30}$$

- In (A.3.27), the transition probability $\mathcal{P}((\mathbf{x}^{j-},\Lambda),j,(\mathbf{x}^{i+},M_k))$ is always zero. Hence:

$$P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in G_{\mathbf{x}^{i+}}\Big|(\mathbf{x}_n,\omega_n)\in F_{\mathbf{x}}\Big)=0. \tag{A.3.31}$$

- In (A.3.28), the transition probability $\mathcal{P}((\mathbf{x},M_j),0,((\mathbf{x}^{i+})^{k-},\Lambda))$ can only be non-zero if $k=i$, in which case it is equal to $\lambda\Delta$ but is included in the second summation only if $\theta((\mathbf{x},\Lambda))=i$. Hence, by the same reasoning as in the case (A.3.26), one has:

$$P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in F_{\mathbf{x}^{i+}}\Big|(\mathbf{x}_n,\omega_n)\in G_{\mathbf{x}}\Big)=\begin{cases}\lambda\Delta, & \text{if } \theta((\mathbf{x},\Lambda))=i,\\ 0, & \text{otherwise.}\end{cases} \tag{A.3.32}$$

- Finally, in (A.3.29), the probability $\mathcal{P}((\mathbf{x},M_j),0,(\mathbf{x}^{i+},M_k))$ is always zero. Hence:

$$P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in G_{\mathbf{x}^{i+}}\Big|(\mathbf{x}_n,\omega_n)\in G_{\mathbf{x}}\Big)=0. \tag{A.3.33}$$

Hence, by substituting the expressions in (A.3.30)-(A.3.33) back into (A.3.25) and again using the fact that $F_{\mathbf{x}}$ and $G_{\mathbf{x}}$ form a partition of $H_{\mathbf{x}}$, one obtains (for $i\in\{1,2,...,N\}$):

$$P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in H_{\mathbf{x}^{i+}}\Big|(\mathbf{x}_n,\omega_n)\in H_{\mathbf{x}}\Big)=\begin{cases}\lambda\Delta, & \text{if } \theta((\mathbf{x},\Lambda))=i,\\ 0, & \text{otherwise.}\end{cases} \tag{A.3.34}$$

Importantly, (A.3.34) remains valid if one conditions on $(\mathbf{x}_n,\omega_n)$ being a *particular state* belonging to $H_{\mathbf{x}}$, as opposed to $(\mathbf{x}_n,\omega_n)$ simply being an element of $H_{\mathbf{x}}$. In other words, if the system is in *any* state belonging to $H_{\mathbf{x}}$ then the probability of transferring to $H_{\mathbf{x}^{i+}}$ at the next time step is given by (A.3.34). This follows from the decomposition arguments in (A.3.26)-(A.3.29) and (A.3.30)-(A.3.33) which show that if $(\mathbf{x}_n,\omega_n)$ is any state belonging to either $F_{\mathbf{x}}$ or $G_{\mathbf{x}}$ then the probability of transferring to $F_{\mathbf{x}^{i+}}$ (hence, $H_{\mathbf{x}^{i+}}$) is $\lambda\Delta$ if $\theta((\mathbf{x},\Lambda))=i$, and 0 otherwise.

Next, consider the conditional probability $P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in H_{\mathbf{x}^{i-}}\Big|(\mathbf{x}_n,\omega_n)\in H_{\mathbf{x}}\Big)$ for arbitrary $i\in\{1,2,...,N\}$, assuming that the i^{th} component of $\mathbf{x}$ is at least one. Using arguments similar to those given previously, this probability can be decomposed as follows:

$$\begin{aligned}
&P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in H_{\mathbf{x}^{i-}}\Big|(\mathbf{x}_n,\omega_n)\in H_{\mathbf{x}}\Big)\\
&=P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in F_{\mathbf{x}^{i-}}\Big|(\mathbf{x}_n,\omega_n)\in F_{\mathbf{x}}\Big)P\Big((\mathbf{x}_n,\omega_n)\in F_{\mathbf{x}}\Big|(\mathbf{x}_n,\omega_n)\in H_{\mathbf{x}}\Big)\\
&\quad+P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in G_{\mathbf{x}^{i-}}\Big|(\mathbf{x}_n,\omega_n)\in F_{\mathbf{x}}\Big)P\Big((\mathbf{x}_n,\omega_n)\in F_{\mathbf{x}}\Big|(\mathbf{x}_n,\omega_n)\in H_{\mathbf{x}}\Big)\\
&\quad+P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in F_{\mathbf{x}^{i-}}\Big|(\mathbf{x}_n,\omega_n)\in G_{\mathbf{x}}\Big)P\Big((\mathbf{x}_n,\omega_n)\in G_{\mathbf{x}}\Big|(\mathbf{x}_n,\omega_n)\in H_{\mathbf{x}}\Big)\\
&\quad+P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in G_{\mathbf{x}^{i-}}\Big|(\mathbf{x}_n,\omega_n)\in G_{\mathbf{x}}\Big)P\Big((\mathbf{x}_n,\omega_n)\in G_{\mathbf{x}}\Big|(\mathbf{x}_n,\omega_n)\in H_{\mathbf{x}}\Big).
\end{aligned}\tag{A.3.35}$$

Then, analogously to (A.3.26)-(A.3.29), one finds:

$$\begin{aligned}
&P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in F_{\mathbf{x}^{i-}}\Big|(\mathbf{x}_n,\omega_n)\in F_{\mathbf{x}}\Big)\\
&=\sum_{\substack{j\in\{1,2,...,N\}\\ \theta((\mathbf{x}^{j-},\Lambda))=j}}P\Big((\mathbf{x}_n,\omega_n)=(\mathbf{x}^{j-},\Lambda)\Big|(\mathbf{x}_n,\omega_n)\in F_{\mathbf{x}}\Big)\sum_{\substack{k\in\{1,2,...,N\}\\ \theta(((\mathbf{x}^{i-})^{k-},\Lambda))=k}}\mathcal{P}((\mathbf{x}^{j-},\Lambda),j,((\mathbf{x}^{i-})^{k-},\Lambda)),\\
&=0,
\end{aligned}\tag{A.3.36}$$

$$\begin{aligned}
&P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in G_{\mathbf{x}^{i-}}\Big|(\mathbf{x}_n,\omega_n)\in F_{\mathbf{x}}\Big)\\
&=\sum_{\substack{j\in\{1,2,...,N\}\\ \theta((\mathbf{x}^{j-},\Lambda))=j}}P\Big((\mathbf{x}_n,\omega_n)=(\mathbf{x}^{j-},\Lambda)\Big|(\mathbf{x}_n,\omega_n)\in F_{\mathbf{x}}\Big)\sum_{k=1}^{N}\mathcal{P}((\mathbf{x}^{j-},\Lambda),j,(\mathbf{x}^{i-},M_k)),\\
&=\min(x_i,c_i)\mu_i\Delta,
\end{aligned}\tag{A.3.37}$$

$$\begin{aligned}
&P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in F_{\mathbf{x}^{i-}}\Big|(\mathbf{x}_n,\omega_n)\in G_{\mathbf{x}}\Big)\\
&=\sum_{j=1}^{N}P\Big((\mathbf{x}_n,\omega_n)=(\mathbf{x},M_j)\Big|(\mathbf{x}_n,\omega_n)\in G_{\mathbf{x}}\Big)\sum_{\substack{k\in\{1,2,...,N\}\\ \theta(((\mathbf{x}^{i-})^{k-},\Lambda))=k}}\mathcal{P}((\mathbf{x},M_j),0,((\mathbf{x}^{i-})^{k-},\Lambda)),\\
&=0,
\end{aligned}\tag{A.3.38}$$

$$\begin{aligned}
&P\Big((\mathbf{x}_{n+1},\omega_{n+1})\in G_{\mathbf{x}^{i-}}\Big|(\mathbf{x}_n,\omega_n)\in G_{\mathbf{x}}\Big)\\
&=\sum_{j=1}^{N}P\Big((\mathbf{x}_n,\omega_n)=(\mathbf{x},M_j)\Big|(\mathbf{x}_n,\omega_n)\in G_{\mathbf{x}}\Big)\sum_{k=1}^{N}\mathcal{P}((\mathbf{x},M_j),0,(\mathbf{x}^{i-},M_k))\\
&=\min(x_i,c_i)\mu_i\Delta.
\end{aligned}\tag{A.3.39}$$

Note that in (A.3.37) and (A.3.39), the transition probability $\mathcal{P}((\mathbf{x}, M_j), 0, (\mathbf{x}^{i-}, M_k))$ is non-zero if and only if $k = i$. Again using partitioning arguments, (A.3.36)-(A.3.39) imply:

$$P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in H_{\mathbf{x}^{i-}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big) = \min(x_i, c_i)\mu_i\Delta. \tag{A.3.40}$$

It can easily be checked using the transition probabilities (3.3.8)-(3.3.10) and the definition of $H_{\mathbf{x}}$ that $P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in H_{\mathbf{y}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big) = 0$ for any vector $\mathbf{y}$ which is not of the form $\mathbf{x}^{i+}$ or $\mathbf{x}^{i-}$ (for some $i \in \{1, 2, ..., N\}$), or equal to $\mathbf{x}$. Hence, (A.3.34) and (A.3.40) imply:

$$P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in H_{\mathbf{y}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big) = \begin{cases} \lambda\Delta, & \text{if } \mathbf{y} = \mathbf{x}^{i+} \text{ and } \theta((\mathbf{x}, \Lambda)) = i \\ & \text{for some } i \in \{1, 2, ..., N\}, \\ \min(x_i, c_i)\mu_i\Delta, & \text{if } \mathbf{y} = \mathbf{x}^{i-} \\ & \text{for some } i \in \{1, 2, ..., N\}, \\ 1 - I(\theta((\mathbf{x}, \Lambda)) \neq 0)\lambda\Delta - \displaystyle\sum_{i=1}^{N} \min(x_i, c_i)\mu_i\Delta, & \text{if } \mathbf{y} = \mathbf{x}, \\ 0, & \text{otherwise.} \end{cases}$$

The remarks made earlier concerning the lack of dependence of $P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in H_{\mathbf{x}^{i+}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big)$ on the particular state $(\mathbf{x}_n, \omega_n)$ also apply to $P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in H_{\mathbf{x}^{i-}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big)$. That is, if $(\mathbf{x}_n, \omega_n)$ is *any* state belonging to the set $H_{\mathbf{x}}$ then the probability of transferring to $H_{\mathbf{x}^{i-}}$ at the next time step is $\min(x_i, c_i)\mu_i\Delta$. It follows that in general, $P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in H_{\mathbf{y}} \Big| (\mathbf{x}_n, \omega_n) \in H_{\mathbf{x}}\Big)$ remains the same if one conditions on the entire history of the process; that is:

$$\begin{aligned} &P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in H_{n+1} \Big| (\mathbf{x}_n, \omega_n) \in H_n, (\mathbf{x}_{n-1}, \omega_{n-1}) \in H_{n-1}, ..., (\mathbf{x}_0, \omega_0) \in H_0\Big) \\ &= P\Big((\mathbf{x}_{n+1}, \omega_{n+1}) \in H_{n+1} \Big| (\mathbf{x}_n, \omega_n) \in H_n\Big), \end{aligned} \tag{A.3.41}$$

where H_n is the 'aggregate set' of the process Φ (operating under θ) at time $n \in \mathbb{N}_0$. The stochastic process $(H_0, H_1, H_2, ...)$ is therefore itself a Markov chain with stationary distribution $\{\pi_\theta(H_{\mathbf{x}})\}_{\mathbf{x} \in S}$, where S is as defined in (3.5.1) and the stationary probabilities are given by:

$$\pi_\theta(H_{\mathbf{x}}) = \sum_{(\mathbf{x}, \omega) \in H_{\mathbf{x}}} \pi_\theta((\mathbf{x}, \omega)) \qquad \forall\, \mathbf{x} \in S.$$

Suppose the process Υ operates under a policy which (as described in the statement of the lemma) chooses the same action at state $\mathbf{x} \in S$ as that chosen by θ at state $(\mathbf{x}, \Lambda) \in \mathcal{S}$ in the process Φ.

Then, by comparing (A.3.41) with (3.5.3), one finds that Υ has exactly the same set of transition probabilities as the stochastic process $(H_0, H_1, ...)$ described above. Thus, the two processes have equivalent balance equations and stationary distributions. In a slight abuse of notation, let $\{\pi_\theta(\mathbf{x})\}_{\mathbf{x}\in S}$ denote the stationary distribution for the process Υ operating in the manner described. Then, by the previous arguments, one has the equivalence relationship:

$$\pi_\theta(\mathbf{x}) = \pi_\theta(H_\mathbf{x}) = \sum_{(\mathbf{x},\omega)\in H_\mathbf{x}} \pi_\theta((\mathbf{x},\omega)) \qquad \forall\ \mathbf{x} \in S. \tag{A.3.42}$$

Thus, Φ and Υ have analogous steady-state distributions, which completes the proof. $\square$

A.4 Proofs for results in Chapter 4

Proof of Lemma 4.2.1.

Let the discount factor $\phi \in (0,1)$ be fixed, and let $\hat{v}_\phi^{(0)}(\mathbf{x}) = 0$ for all $\mathbf{x} \in S$. Then, for states $\mathbf{x} \in S$ and integers $n \geq 0$, the finite-stage approximations $\hat{v}_\phi^{(n+1)}(\mathbf{x})$ are given by:

$$\hat{v}_\phi^{(n+1)} = \max_a \left\{ \hat{r}(\mathbf{x}, a) + \phi \sum_{\mathbf{y}\in S} p(\mathbf{x}, a, \mathbf{y}) \hat{v}_\phi^{(n)}(\mathbf{y}) \right\}. \tag{A.4.1}$$

Hence, by referring to (3.5.3) and enumerating transition probabilities in (A.4.1):

$$\begin{aligned}\hat{v}_\phi^{(n+1)}(\mathbf{x}) = &\max_a \left\{ \hat{r}(\mathbf{x}, a) + \phi\lambda\Delta \hat{v}_\phi^{(n)}(\mathbf{x}^{a+}) \right\} \\ &+ \phi \left[\sum_{i=1}^N \min(x_i, c_i)\mu_i\Delta \hat{v}_\phi^{(n)}(\mathbf{x}^{i-}) + \left(1 - \lambda\Delta - \sum_{i=1}^N \min(x_i, c_i)\mu_i\Delta\right) \hat{v}_\phi^{(n)}(\mathbf{x}) \right]. \end{aligned} \tag{A.4.2}$$

It is sufficient to show that the monotonicity property holds at each stage $n \in \mathbb{N}$. That is, for each state $\mathbf{x} \in S$, discount factor $\phi \in (0,1)$, facility $i \in \{1, 2, ..., N\}$ and $n \geq 1$:

$$\hat{v}_\phi^{(n)}(\mathbf{x}^{i+}) \leq \hat{v}_\phi^{(n)}(\mathbf{x}). \tag{A.4.3}$$

Given that $\hat{v}_\phi^{(0)}(\mathbf{x}) = 0$ for all $\mathbf{x} \in S$, the one-stage return for choosing action a under state $\mathbf{x}$ is simply $\hat{r}(\mathbf{x}, a)$, so in order to verify (A.4.3) when $n = 1$, it suffices to show:

$$\max_a \hat{r}(\mathbf{x}^{i+}, a) \leq \max_b \hat{r}(\mathbf{x}, b),$$

for each facility i. Indeed, let $a^* \in \arg\max_a \hat{r}(\mathbf{x}^{i+}, a)$. It follows from the definition of $\hat{r}(\mathbf{x}, a)$ in (3.5.15) that $\hat{r}(\mathbf{x}^{i+}, a) \leq \hat{r}(\mathbf{x}, a)$ for any fixed action a and facility i. Hence:

$$\max_a \hat{r}(\mathbf{x}^{i+}, a) = \hat{r}(\mathbf{x}^{i+}, a^*) \leq \hat{r}(\mathbf{x}, a^*) \leq \max_b \hat{r}(\mathbf{x}, b),$$

so (A.4.3) holds for $n = 1$. Let it be assumed (as an inductive hypothesis) that (A.4.3) also holds for $n = k$, where $k \geq 1$ is arbitrary. Then, at stage $n = k + 1$:

$$\begin{aligned}
&\hat{v}_\phi^{(k+1)}(\mathbf{x}^{i+}) - \hat{v}_\phi^{(k+1)}(\mathbf{x}) \\
&\quad = \max_a \left\{ \hat{r}(\mathbf{x}^{i+}, a) + \phi\lambda\Delta \hat{v}_\phi^{(k)}\big((\mathbf{x}^{i+})^{a+}\big) \right\} - \max_b \left\{ \hat{r}(\mathbf{x}, b) + \phi\lambda\Delta \hat{v}_\phi^{(k)}(\mathbf{x}^{b+}) \right\} \\
&\qquad + \phi \sum_{j=1}^{N} \min(x_j, c_j)\mu_j\Delta \left(\hat{v}_\phi^{(k)}\big((\mathbf{x}^{i+})^{j-}\big) - \hat{v}_\phi^{(k)}(\mathbf{x}^{j-}) \right) \\
&\qquad + \phi \left(1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j, c_j)\mu_j\Delta \right) \left(\hat{v}_\phi^{(k)}(\mathbf{x}^{i+}) - \hat{v}_\phi^{(k)}(\mathbf{x}) \right) \\
&\qquad - \phi\, I(x_i < c_i)\mu_i\Delta \left(\hat{v}_\phi^{(k)}(\mathbf{x}^{i+}) - \hat{v}_\phi^{(k)}(\mathbf{x}) \right),
\end{aligned} \tag{A.4.4}$$

where the indicator term in (A.4.4) arises because, under state $\mathbf{x}^{i+}$, there may (or may not) be one extra service in progress at facility i, depending on whether or not $x_i < c_i$. Recall that $\lambda\Delta + \sum_{i=1}^{N} c_i\mu_i\Delta \leq 1$ by definition of Δ, and hence $\left(1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j, c_j)\mu_j\Delta - I(x_i < c_i)\mu_i\Delta\right)$ must always be non-negative. Also, the inequalities $\hat{v}_\phi^{(k)}(\mathbf{x}^{i+}) \leq \hat{v}_\phi^{(k)}(\mathbf{x})$ and $\hat{v}_\phi^{(k)}\big((\mathbf{x}^{i+})^{j-}\big) \leq \hat{v}_\phi^{(k)}(\mathbf{x}^{j-})$ (for facilities $j = 1, 2, ..., N$) follow by the inductive assumption of monotonicity at stage k. Hence, in order to verify that (A.4.4) is non-positive, it suffices to show:

$$\max_a \left\{ \hat{r}(\mathbf{x}^{i+}, a) + \phi\lambda\Delta \hat{v}_\phi^{(k)}\big((\mathbf{x}^{i+})^{a+}\big) \right\} \leq \max_b \left\{ \hat{r}(\mathbf{x}, b) + \phi\lambda\Delta \hat{v}_\phi^{(k)}(\mathbf{x}^{b+}) \right\}. \tag{A.4.5}$$

Here, let a^* be a maximizing action on the left-hand side of (A.4.5). That is:

$$a^* \in \arg\max_a \left\{ \hat{r}(\mathbf{x}^{i+}, a) + \phi\lambda\Delta \hat{v}_\phi^{(k)}\big((\mathbf{x}^{i+})^{a+}\big) \right\}.$$

Then, by the monotonicity of $\hat{r}$ and the inductive assumption:

$$\hat{r}(\mathbf{x}^{i+}, a^*) \leq \hat{r}(\mathbf{x}, a^*),$$

$$\hat{v}_\phi^{(k)}\big((\mathbf{x}^{a^*+})^{i+}\big) \leq \hat{v}_\phi^{(k)}(\mathbf{x}^{a^*+}).$$

Hence the left-hand side of (A.4.5) is bounded above by $\hat{r}(\mathbf{x}, a^*) + \phi\lambda\Delta\hat{v}_\phi^{(k)}\left(\mathbf{x}^{a^*+}\right)$, which in turn is bounded above by $\max_b \left\{\hat{r}(\mathbf{x}, b) + \phi\lambda\Delta\hat{v}_\phi^{(k)}(\mathbf{x}^{b+})\right\}$. This shows that $\hat{v}_\phi^{(k+1)}(\mathbf{x}^{i+}) \leq \hat{v}_\phi^{(k+1)}(\mathbf{x})$, which completes the inductive proof that (A.4.3) holds for all $n \in \mathbb{N}$. Since $\lim_{n\to\infty} \hat{v}_\phi^{(n)}(\mathbf{x}) = \hat{v}_\phi(\mathbf{x})$ for all $\mathbf{x} \in S$ by Theorem 3.4.4, the conclusion follows that $\hat{v}_\phi(\mathbf{x}^{i+}) \leq \hat{v}_\phi(\mathbf{x})$. □

Proof of Lemma 4.2.2.

Let $\alpha^* = \max_{i\in\{1,2,\ldots,N\}} \alpha_i$, i.e. α^* is the maximum value of service across all facilities. For each discount factor $\phi \in (0, 1)$ and policy θ, define a new function $\hat{u}_{\phi,\theta}$ by:

$$\hat{u}_{\phi,\theta}(\mathbf{x}) := E_\theta\left[\sum_{n=0}^{\infty} \phi^n \left(\hat{r}(\mathbf{x}_n, a_n) - \lambda\alpha^*\right) \middle| \mathbf{x}_0 = \mathbf{x}\right] \qquad (\mathbf{x} \in S).$$

By comparison with the definition of $\hat{v}_{\phi,\theta}$ in (4.2.3), it may be seen that:

$$\hat{u}_{\phi,\theta}(\mathbf{x}) = \hat{v}_{\phi,\theta}(\mathbf{x}) - \frac{\lambda\alpha^*}{1-\phi}.$$

Furthermore, since the subtraction of the constant α^* from each single-step reward does not affect the discounted reward optimality criterion, it also follows that:

$$\hat{u}_\phi(\mathbf{x}) = \hat{v}_\phi(\mathbf{x}) - \frac{\lambda\alpha^*}{1-\phi}, \tag{A.4.6}$$

where $\hat{u}_\phi(\mathbf{x}) = \sup_\theta \hat{u}_{\phi,\theta}(\mathbf{x})$. By the definition of $\hat{r}$ in (3.5.15) it can be checked that $\hat{r}(\mathbf{x}, a) \leq \lambda\alpha^*$ for all state-action pairs $(\mathbf{x}, a)$. Therefore $\hat{u}_\phi(\mathbf{x})$ is a sum of non-positive terms and must be non-positive itself. Furthermore, $\hat{u}_\phi$ is the expected total discounted reward function for a new MDP $\hat{\Upsilon}$ which is identical to the original MDP except that each reward $\hat{r}(\mathbf{x}_n, a_n)$ (for $n = 0, 1, 2, ...$) is replaced by $\hat{r}(\mathbf{x}_n, a_n) - \lambda\alpha^*$. Thus, $\hat{u}_\phi$ satisfies the discount optimality equations:

$$\hat{u}_\phi(\mathbf{x}) = \max_a \left\{\hat{r}(\mathbf{x}, a) - \lambda\alpha^* + \phi\sum_{\mathbf{y}\in S} p(\mathbf{x}, a, \mathbf{y})\hat{u}_\phi(\mathbf{y})\right\} \quad (\mathbf{x} \in S). \tag{A.4.7}$$

Consider $\mathbf{x} = \mathbf{0}^{i+}$, for an arbitrary $i \in \{1, 2, ..., N\}$. For all actions a, by (A.4.7):

$$\hat{u}_\phi(\mathbf{0}^{i+}) \geq \hat{r}(\mathbf{0}^{i+}, a) - \lambda\alpha^* + \phi\sum_{\mathbf{y}\in S} p(\mathbf{0}^{i+}, a, \mathbf{y})\hat{u}_\phi(\mathbf{y}).$$

In particular, if the action $a = 0$ is to balk then $\hat{r}(\mathbf{0}^{i+}, a) = 0$ and the only possible transitions are to states $\mathbf{0}$ or $\mathbf{0}^{i+}$. Hence, by the previous arguments:

$$\hat{u}_\phi(\mathbf{0}^{i+}) \geq -\lambda\alpha^* + \phi\mu_i\Delta\hat{u}_\phi(\mathbf{0}) + \phi(1 - \mu_i\Delta)\hat{u}_\phi(\mathbf{0}^{i+}).$$

Then, since $0 < \phi < 1$ and by the non-positivity of $\hat{u}_\phi(\mathbf{0})$ and $\hat{u}_\phi(\mathbf{0}^{i+})$:

$$\hat{u}_\phi(\mathbf{0}^{i+}) \geq -\lambda\alpha^* + \mu_i\Delta\hat{u}_\phi(\mathbf{0}) + (1-\mu_i\Delta)\hat{u}_\phi(\mathbf{0}^{i+}). \tag{A.4.8}$$

So, using (A.4.6) and (A.4.8) a lower bound for $\hat{v}_\phi(\mathbf{0}^{i+}) - \hat{v}_\phi(\mathbf{0})$ may be derived:

$$\hat{v}_\phi(\mathbf{0}^{i+}) - \hat{v}_\phi(\mathbf{0}) = \hat{u}_\phi(\mathbf{0}^{i+}) - \hat{u}_\phi(\mathbf{0}) \geq -\frac{\lambda\alpha^*}{\mu_i\Delta}, \tag{A.4.9}$$

and this lower bound is independent of ϕ as required. It remains to be shown that for each $\mathbf{x} \in S$, a lower bound exists for $\hat{v}_\phi(\mathbf{x}) - \hat{v}_\phi(\mathbf{0})$. Consider an arbitrary state $\mathbf{x} \in S$ and let an inductive hypothesis be formed as follows: for all states $\mathbf{y} \in S$ which satisfy the componentwise inequality $\mathbf{y} < \mathbf{x}$, there exists a non-negative value $f(\mathbf{y})$ such that for all $\phi \in (0,1)$:

$$\hat{v}_\phi(\mathbf{y}) - \hat{v}_\phi(\mathbf{0}) \geq -\lambda\alpha^* f(\mathbf{y}). \tag{A.4.10}$$

The values $f(\mathbf{0}) = 0$ and $f(\mathbf{0}^{i+}) = 1/(\mu_i\Delta)$ (for $i = 1, 2, ..., N$) will clearly suffice for states with at most one customer present. The next step is to show that the same property holds for an arbitrary state $\mathbf{x} \neq \mathbf{0}$, under the inductive assumption that for each $j \in \{1, 2, ..., N\}$ with $x_j \geq 1$, (A.4.10) holds with $\mathbf{y} = \mathbf{x}^{j-}$. Using similar steps to those described earlier, one finds:

$$\hat{u}_\phi(\mathbf{x}) \geq -\lambda\alpha^* + \phi\sum_{j=1}^{N}\min(x_j, c_j)\mu_j\Delta\hat{u}_\phi(\mathbf{x}^{j-}) + \phi\left(1 - \sum_{j=1}^{N}\min(x_j, c_j)\mu_j\Delta\right)\hat{u}_\phi(\mathbf{x}),$$

and hence:

$$\sum_{j=1}^{N}\min(x_j, c_j)\mu_j\Delta\left(\hat{u}_\phi(\mathbf{x}) - \hat{u}_\phi(\mathbf{x}^{j-})\right) \geq -\lambda\alpha^*.$$

Then, using the inductive assumption that, for each facility $j \in \{1, 2, ..., N\}$, $\hat{u}_\phi(\mathbf{x}^{j-}) - \hat{u}_\phi(\mathbf{0})$ is bounded below by $-\lambda\alpha^* f(\mathbf{x}^{j-})$, it follows after simple manipulations that:

$$\hat{u}_\phi(\mathbf{x}) - \hat{u}_\phi(\mathbf{0}) \geq -\lambda\alpha^*\left(\frac{1 + \sum_{j=1}^{N}\min(x_j, c_j)\mu_j\Delta f(\mathbf{x}^{j-})}{\sum_{j=1}^{N}\min(x_j, c_j)\mu_j\Delta}\right). \tag{A.4.11}$$

By (A.4.6), $\hat{u}_\phi(\mathbf{x}) - \hat{u}_\phi(\mathbf{0}) = \hat{v}_\phi(\mathbf{x}) - \hat{v}_\phi(\mathbf{0})$, so one may define:

$$f(\mathbf{x}) := \frac{1 + \sum_{j=1}^{N}\min(x_j, c_j)\mu_j\Delta f(\mathbf{x}^{j-})}{\sum_{j=1}^{N}\min(x_j, c_j)\mu_j\Delta},$$

with the result that $\hat{v}_\phi(\mathbf{x}) - \hat{v}_\phi(\mathbf{0})$ is bounded below by an expression which depends only on the system input parameters λ, α^* and the service rates $\mu_1, \mu_2, ..., \mu_N$ as required. Using an inductive procedure, a lower bound of this form can be derived for every state $\mathbf{x} \in S$. $\square$

Proof of Lemma 4.3.2.

Let $\mathbf{x} \in \tilde{S}$ be a state with $x_i < c_i$ for some $i \in \{1, 2, ..., N\}$. Then, for all $n \geq 0$:

$$\begin{aligned} h_{n+1}(\mathbf{x}^{i+}) - h_{n+1}(\mathbf{x}) = r(\mathbf{x}^{i+}) - r(\mathbf{x}) \\ + \lambda\Delta\left(h_n((\mathbf{x}^{i+})^{a_1+}) - h_n(\mathbf{x}^{a_0+})\right) \\ + \sum_{j=1}^{N} \min(x_j, c_j)\mu_j\Delta\left(h_n((\mathbf{x}^{i+})^{j-}) - h_n(\mathbf{x}^{j-})\right) \\ + \left(1 - \lambda\Delta - \sum_{j=1}^{N} \min(x_j, c_j)\mu_j\Delta - \mu_i\Delta\right)\left(h_n(\mathbf{x}^{i+}) - h_n(\mathbf{x})\right), \end{aligned} \quad \text{(A.4.12)}$$

where the actions a_0 and a_1 satisfy:

$$\begin{aligned} a_0 &\in \arg\max_{a \in A_{\mathbf{x}}} h_n(\mathbf{x}^{a+}), \\ a_1 &\in \arg\max_{a \in A_{\mathbf{x}^{i+}}} h_n((\mathbf{x}^{i+})^{a+}). \end{aligned} \quad \text{(A.4.13)}$$

It is assumed that $h_0(\mathbf{x}) = 0$ for all $\mathbf{x} \in \tilde{S}$. Hence, (A.4.12) implies:

$$h_1(\mathbf{x}^{i+}) - h_1(\mathbf{x}) = r(\mathbf{x}^{i+}) - r(\mathbf{x}) = \alpha_i\mu_i - \beta_i > 0,$$

where the positivity of $\alpha_i\mu_i - \beta_i$ follows by the assumption in Section 3.1. It will be sufficient to show that for all $n \geq 1$, the following holds for states $\mathbf{x} \in \tilde{S}$ with $x_i < c_i$:

$$h_n(\mathbf{x}^{i+}) - h_n(\mathbf{x}) \geq \alpha_i\mu_i - \beta_i. \quad \text{(A.4.14)}$$

Proceeding by induction, let it be assumed that for all $\mathbf{x} \in \tilde{S}$ and $i \in \{1, 2, ..., N\}$:

$$x_i < c_i \quad \Rightarrow \quad h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x}) \geq \alpha_i\mu_i - \beta_i, \quad \text{(A.4.15)}$$

where $k \geq 1$ is arbitrary. It must then be shown that (A.4.15) also holds with k replaced by $k + 1$. Given that $r(\mathbf{x}^{i+}) - r(\mathbf{x}) = \alpha_i\mu_i - \beta_i$ when $x_i < c_i$, it will be sufficient to consider an arbitrary state $\mathbf{x} \in \tilde{S}$ with $x_i < c_i$ for some $i \in \{1, 2, ..., N\}$ and show that the second, third and fourth lines on the right-hand side of (A.4.12) are non-negative when $n = k$. The inductive assumption (A.4.15) implies that $h_k((\mathbf{x}^{i+})^{j-}) - h_k(\mathbf{x}^{j-})$ and $h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x})$ are both bounded below $\alpha_i\mu_i - \beta_i$ (noting, in the former case, that if $\mathbf{x}$ is a state with fewer than c_i customers present at facility i then the same applies to any state of the form $\mathbf{x}^{j-}$), so the third and fourth lines of (A.4.12) are positive when $n = k$. The fact that $h_k((\mathbf{x}^{i+})^{a_1+}) - h_k(\mathbf{x}^{a_0+}) \geq 0$ can be shown as follows:

- If $a_0 = i$, then it is sufficient to note:

$$h_k((\mathbf{x}^{i+})^{a_1+}) - h_k(\mathbf{x}^{a_0+}) \geq h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x}^{i+}) = 0,$$

  where the inequality follows from the definition of action a_1 in (A.4.13); specifically, the fact that $h_k((\mathbf{x}^{i+})^{a_1+}) \geq h_k((\mathbf{x}^{i+})^{0+}) = h_k(\mathbf{x}^{i+})$.

- if a_0 is any *other* action, i.e. $a_0 \in \{0, 1, 2, ..., N\} \setminus \{i\}$, then:

$$h_k((\mathbf{x}^{i+})^{a_1+}) - h_k(\mathbf{x}^{a_0+}) \geq h_k((\mathbf{x}^{i+})^{a_0+}) - h_k(\mathbf{x}^{a_0+}) > 0,$$

  where the first inequality again follows from the definition of a_1 in (A.4.13), and the second inequality is due to the inductive assumption, since (given that $a_0 \neq i$) $\mathbf{x}^{a_0+}$ is a state with fewer than c_i customers present at facility i.

These arguments show that the expression on the second line of (A.4.12) is non-negative when $n = k$, which confirms that $h_{k+1}(\mathbf{x}^{i+}) - h_{k+1}(\mathbf{x}) \geq \alpha_i\mu_i - \beta_i$ and hence (by induction) $h_n(\mathbf{x}^{i+}) - h_n(\mathbf{x}) \geq \alpha_i\mu_i - \beta_i > 0$ for all integers $n \geq 0$. By Theorem 3.7.4, $\lim_{n\to\infty} h_n(\mathbf{x}) = h(\mathbf{x})$ for all states $\mathbf{x} \in \tilde{S}$, so it follows that $h(\mathbf{x}^{i+}) > h(\mathbf{x})$. This completes the proof of the lemma. □

A.5 Proofs for results in Chapter 5

Proof of Lemma 5.1.2.

As stated in the introductory remarks, $h(\cdot)$ is assumed to be defined as in the statement of Theorem 3.7.4. That is, $h(x) = \lim_{n\to\infty} h_n(x)$ for $x \in \{0, 1, ..., \tilde{T}\}$, where (for integers $n \geq 0$):

$$h_{n+1}(x) := \max_{a\in A_x} \left\{ r(x) + \sum_{y\leq\tilde{T}} p(x, a, y)h_n(y) \right\} - \max_{a\in A_x} \left\{ r(0) + \sum_{y\leq\tilde{T}} p(0, a, y)h_n(y) \right\},$$

and $h_0(x) = 0$ for all $x \in \{0, 1, ..., \tilde{T}\}$. Obviously, the analogous definition for $\hat{h}_{n+1}(x)$ (with $r(x)$ and $h_n(y)$ replaced by $\hat{r}(x, a)$ and $\hat{h}_n(y)$ respectively, and $\hat{h}_0(x) = 0$ for all states $x \in \{0, 1, ..., \tilde{T}\}$) is also valid. Thus, it is sufficient to show that for all integers $n \geq 0$:

$$D(x, h_n) \leq (\alpha - \beta/\mu)/\Delta \qquad \forall\, x \in \{0, 1, ..., \tilde{T} - 1\}, \tag{A.5.1}$$

$$D(x + 1, h_n) \leq D(x, h_n) \qquad \forall\, x \in \{0, 1, ..., \tilde{T} - 2\}, \tag{A.5.2}$$

$$D(x, \hat{h}_n) \leq 0 \qquad \forall\, x \in \{0, 1, ..., \tilde{T} - 1\}, \tag{A.5.3}$$

$$\hat{r}(x+1, 1) + \lambda\Delta D(x+1, \hat{h}_n) \leq \hat{r}(x, 1) + \lambda\Delta D(x, \hat{h}_n) \qquad \forall\, x \in \{0, 1, ..., \tilde{T} - 2\}. \tag{A.5.4}$$

All of these properties hold trivially when $n = 0$ since $h_0(x) = \hat{h}_0(x) = 0$ for all x, $\alpha - \beta/\mu$ is assumed to be positive (as stated in Section 3.1) and $\hat{r}(x+1, 1) \leq \hat{r}(x, 1)$ by definition of $\hat{r}$. Proceeding by induction, assume that (A.5.1)-(A.5.4) hold when $n = k$, where $k \in \mathbb{N}_0$ is arbitrary. First, it will be shown that (A.5.1) holds with $n = k + 1$. Indeed, for $x \in \{0, 1, ..., \tilde{T} - 1\}$:

$$\begin{aligned} D(x, h_{k+1}) = D(x, r) &+ \lambda\Delta\big(h_k(x + 1 + a_1) - h_k(x + a_0)\big) \\ &+ \min(x, c)\mu\Delta D(x - 1, h_k) \\ &+ \left(1 - \lambda\Delta - \min(x, c)\mu\Delta - I(x < c)\mu\Delta\right) D(x, h_k), \end{aligned} \tag{A.5.5}$$

where I denotes the indicator function and the actions a_0 and a_1 satisfy $a_0 \in \arg\max_a h_k(x + a)$ and $a_1 \in \arg\max_a h_k(x + 1 + a)$. Next, consider the possibilities for actions a_0 and a_1. First, note that it is not possible to have $a_0 = 0$ and $a_1 = 1$, since this would imply:

$$D(x, h_k) < 0, \qquad D(x + 1, h_k) \geq 0,$$

which contradicts the second inductive assumption that $D(x + 1, h_k) \leq D(x, h_k)$. If $a_0 = a_1 = 0$, then $h_k(x + 1 + a_1) - h_k(x + a_0)$ is equal to $D(x, h_k)$, which is bounded above by $(\alpha - \beta/\mu)/\Delta$ by the first inductive assumption. Similarly, if $a_0 = a_1 = 1$ then $h_k(x + 1 + a_1) - h_k(x + a_0)$ is equal to $D_k(x + 1, h_k)$, which is also bounded above by $(\alpha - \beta/\mu)/\Delta$. Finally, if $a_0 = 1$ and $a_1 = 0$, then $h_k(x + 1 + a_1) - h_k(x + a_0) = 0$. So, in all possible cases for a_0 and a_1:

$$h_k(x + 1 + a_1) - h_k(x + a_0) \leq (\alpha - \beta/\mu)/\Delta. \tag{A.5.6}$$

It will be convenient to consider the cases $x < c$ and $x \geq c$ separately. If $x < c$, then:

$$\begin{aligned} D(x, h_{k+1}) = D(x, r) &+ \lambda\Delta\big(h_k(x + 1 + a_1) - h_k(x + a_0)\big) \\ &+ x\mu\Delta D(x - 1, h_k) + \left(1 - \lambda\Delta - (x + 1)\mu\Delta\right) D(x, h_k). \end{aligned} \tag{A.5.7}$$

Due to the first inductive assumption, $D(x - 1, h_k)$ (when it exists, i.e. when $x \geq 1$) is bounded above by $(\alpha - \beta/\mu)/\Delta$, and the same applies to $D(x, h_k)$ and $h_k(x + 1 + a_1) - h_k(x + a_0)$, where the latter is due to (A.5.6). Hence, after simplifications, (A.5.7) implies:

$$D(x, h_{k+1}) \leq D(x, r) + (1 - \mu\Delta)(\alpha - \beta/\mu)/\Delta.$$

Noting that $D(x,r) = \alpha\mu - \beta$ for $x < c$, this yields:

$$D(x, h_{k+1}) \leq \alpha\mu - \beta + (1 - \mu\Delta)(\alpha - \beta/\mu)/\Delta = (\alpha - \beta/\mu)/\Delta,$$

as required. Next, consider the case $x \geq c$. In this case:

$$\begin{aligned} D(x, h_{k+1}) = D(x,r) \\ &+ \lambda\Delta\big(h_k(x+1+a_1) - h_k(x+a_0)\big) + c\mu\Delta D(x-1, h_k) \\ &+ (1 - \lambda\Delta - c\mu\Delta)\, D(x, h_k). \end{aligned} \tag{A.5.8}$$

The terms in the last two lines of (A.5.8) represent a convex combination of terms bounded above by $(\alpha - \beta/\mu)/\Delta$. Therefore, since $r(x+1) - r(x) = -\beta < 0$ when $x \geq c$:

$$D(x, h_{k+1}) \leq (\alpha - \beta/\mu)/\Delta,$$

as required. This establishes that (A.5.1) holds when $n = k+1$. The next task is to show that the same is true for (A.5.2). At stage $k+1$, one has (after some simplifications):

$$\begin{aligned} D(x+1, h_{k+1}) - D(x, h_{k+1}) &= D(x+1, r) - D(x, r) \\ &+ \lambda\Delta\big(h_k(x+2+a_2) - h_k(x+1+a_1) - h_k(x+1+a_1) + h_k(x+a_0)\big) \\ &+ \min(x,c)\mu\Delta\big(D(x, h_k) - D(x-1, h_k)\big) \\ &+ I(x = c-1)\mu\Delta D(x, h_k) \\ &+ \big(1 - \lambda\Delta - \min(x,c)\mu\Delta - I(x < c-1)\mu\Delta - I(x < c)\mu\Delta\big)\big(D(x+1, h_k) - D(x, h_k)\big), \end{aligned} \tag{A.5.9}$$

where $a_0, a_1, a_2 \in \{0, 1\}$ are actions which maximise $h_k(x + a_0)$, $h_k(x + 1 + a_1)$ and $h_k(x + 2 + a_2)$ respectively. The reward function $r(x)$ increases linearly (with gradient $\alpha\mu - \beta > 0$) when $x < c$, and decreases linearly (with gradient $-\beta < 0$) when $x \geq c$. Hence, for $x \in \{0, 1, ..., \tilde{T} - 2\}$:

$$D(x+1, r) - D(x, r) = \begin{cases} -\alpha\mu, & \text{if } x = c-1, \\ 0, & \text{otherwise,} \end{cases} \tag{A.5.10}$$

so $D(x+1, r) - D(x, r)$ is non-positive. Next, it will be shown that:

$$h_k(x+2+a_2) - h_k(x+1+a_1) - h_k(x+1+a_1) + h_k(x+a_0) \leq 0. \tag{A.5.11}$$

This will be done by considering the possible combinations for the actions a_0 and a_2 and applying the AS principle (Lemma 5.1.1). First, note that it is not possible to have $a_0 = 0$ and $a_2 = 1$, as this would imply $D(x, h_k) < 0$ and $D(x + 2, h_k) \geq 0$, thereby contradicting the inductive assumption that $D(x + 2, h_k) \leq D(x + 1, h_k) \leq D(x, h_k)$. Hence, there are 3 possible cases for the pair $(a_0, a_2) \in \{0, 1\} \times \{0, 1\}$. These cases are considered one-by-one below:

- If $a_0 = a_2 = 0$, then (A.5.11) holds with $a_1 = 0$ due to the inductive assumption. By Lemma 5.1.1, it must also hold with a_1 defined as the action which maximises $h_k(x + 1 + a_1)$.
- If $a_0 = a_2 = 1$, one may choose $a_1 = 1$ and apply the same reasoning as above.
- If $a_0 = 1$ and $a_2 = 0$, then it is sufficient to note:

$$h_k(x + 2) - h_k(x + 2) - h_k(x + 1) + h_k(x + 1) = 0. \tag{A.5.12}$$

  Since a_1 is the action which maximises $h_k(x+1+a_1)$, the left-hand side of (A.5.12) is bounded below by the left-hand side of (A.5.11), so the latter must be non-positive.

Due to the inductive assumption, the differences $D(x, h_k) - D(x-1, h_k)$ and $D(x+1, h_k) - D(x, h_k)$ are both non-positive. Thus, all terms on the right-hand side of (A.5.9) have been shown to be non-positive except for $I(x = c - 1)\mu\Delta D(x, h_k)$. This term is zero if $x \neq c - 1$ by definition of the indicator function, whereas if $x = c-1$ then it can be bounded above by $\mu\Delta(\alpha - \beta/\mu)/\Delta = \alpha\mu - \beta$ due to the first inductive assumption. However, (A.5.10) states that $D(x + 1, r) - D(x, r) = -\alpha\mu$ when $x = c - 1$, so in conclusion the following holds for $x \in \{0, 1, ..., \tilde{T} - 2\}$:

$$D(x + 1, r) - D(x, r) + I(x = c - 1)\mu\Delta D(x, h_k) \leq 0.$$

This establishes that $D(x+1, h_{k+1}) - D(x, h_{k+1})$ is non-positive for $x \in \{0, 1, ..., \tilde{T} - 2\}$ as required. At this point, the properties (A.5.1) and (A.5.2) have been shown to hold for $n = k + 1$ under the assumption that they hold for $n = k$. It remains to show that the properties (A.5.3) and (A.5.4) are propagated in a similar manner. Indeed, for states $x \in \{0, 1, ..., \tilde{T} - 1\}$ one has:

$$\begin{aligned} D(x, \hat{h}_{k+1}) = {} & \hat{r}(x + 1, a_1) - \hat{r}(x, a_0) + \lambda\Delta\big(\hat{h}_k(x + 1 + a_1) - \hat{h}_k(x + a_0)\big) \\ & + \min(x, c)\mu\Delta D(x - 1, \hat{h}_k) \\ & + (1 - \lambda\Delta - \min(x, c)\mu\Delta - I(x < c)\mu\Delta)\, D(x, \hat{h}_k), \end{aligned} \tag{A.5.13}$$

where a_0 and a_1 are actions maximising $\hat{r}(x, a_0) + \lambda\Delta\hat{h}_k(x + a_0)$ and $\hat{r}(x+1, a_1) + \lambda\Delta\hat{h}_k(x+1+a_1)$ respectively. First, it will be shown that the following holds:

$$\hat{r}(x+1, a_1) - \hat{r}(x, a_0) + \lambda\Delta\big(\hat{h}_k(x+1+a_1) - \hat{h}_k(x+a_0)\big) \leq 0. \tag{A.5.14}$$

It is not possible to have $a_0 = 0$ and $a_1 = 1$, since this would contradict the inductive assumption that $\hat{r}(x+1, 1) + \lambda\Delta D(x+1, \hat{h}_k) \leq \hat{r}(x, 1) + \lambda\Delta D(x, \hat{h}_k)$. Thus, if $a_0 = 0$ then $a_1 = 0$ also and the left-hand side of (A.5.14) reduces to $\lambda\Delta D(x, \hat{h}_k)$, which is non-positive by the third inductive assumption. On the other hand, if $a_0 = 1$ then there are two possible cases:

- If $a_0 = a_1 = 1$, the left-hand side of (A.5.14) reduces to $\hat{r}(x+1, 1) - \hat{r}(x, 1) + \lambda\Delta\big(\hat{h}_k(x+2) - \hat{h}_k(x+1)\big)$, which is non-positive due to the third inductive assumption.
- If $a_0 = 1$ and $a_1 = 0$, the left-hand side of (A.5.14) reduces to $-\hat{r}(x, 1)$. Given that $a_0 = 1$, it must be the case that $\hat{r}(x, 1) + \lambda\Delta D(x, \hat{h}_k) \geq 0$ and hence $\hat{r}(x, 1) \geq 0$ since $D(x, \hat{h}_k) \leq 0$ by the third inductive assumption. Hence, the inequality (A.5.14) again holds.

So, (A.5.14) holds in all possible cases. Given that $D(x-1, \hat{h}_k)$ and $D(x, \hat{h}_k)$ are both non-positive by the third inductive assumption, it follows from (A.5.13) that $D(x, \hat{h}_{k+1})$ is non-positive as required. The only remaining property to check is (A.5.4). This property can be written:

$$D(x+1, \hat{h}_n) - D(x, \hat{h}_n) \leq \frac{\hat{r}(x, 1) - \hat{r}(x+1, 1)}{\lambda\Delta}, \tag{A.5.15}$$

which is assumed to be true for all $x \in \{0, 1, ..., \tilde{T} - 2\}$ when $n = k$. For $n = k + 1$, one has:

$$\begin{aligned} D(x+1, \hat{h}_{k+1}) - D(x, \hat{h}_{k+1}) &= \hat{r}(x+2, a_2) - \hat{r}(x+1, a_1) - \hat{r}(x+1, a_1) + \hat{r}(x, a_0) \\ &+ \lambda\Delta\big(\hat{h}_k(x+2+a_2) - \hat{h}_k(x+1+a_1) - \hat{h}_k(x+1+a_1) + \hat{h}_k(x+a_0)\big) \\ &+ \min(x, c)\mu\Delta\big(D(x, \hat{h}_k) - D(x-1, \hat{h}_k)\big) \\ &+ I(x = c-1)\mu\Delta D(x, \hat{h}_k) \\ &+ \big(1 - \lambda\Delta - \min(x, c)\mu\Delta - I(x < c)\mu\Delta - I(x < c-1)\mu\Delta\big)\big(D(x+1, \hat{h}_k) - D(x, \hat{h}_k)\big), \end{aligned} \tag{A.5.16}$$

where $a_0, a_1, a_2 \in \{0, 1\}$ are actions which maximise $\hat{r}(x, a_0) + \lambda\Delta\hat{h}_k(x + a_0)$, $\hat{r}(x+1, a_1) + \lambda\Delta\hat{h}_k(x+1+a_1)$ and $\hat{r}(x+2, a_2) + \lambda\Delta\hat{h}_k(x+2+a_2)$ respectively. Referring to (A.5.15), it must then be shown that, for each $x \in \{0, 1, ..., \tilde{T} - 2\}$, the right-hand side of (A.5.16)

is bounded above by $(\hat{r}(x,1) - \hat{r}(x+1,1))/(\lambda\Delta)$. It will be convenient to begin by showing that, for all relevant states x and actions $a_0, a_1, a_2 \in \{0,1\}$, the following holds:

$$\begin{aligned}&\hat{r}(x+2,a_2) - \hat{r}(x+1,a_1) - \hat{r}(x+1,a_1) + \hat{r}(x,a_0) \\ &+ \lambda\Delta\big(\hat{h}_k(x+2+a_2) - \hat{h}_k(x+1+a_1) - \hat{h}_k(x+1+a_1) + \hat{h}_k(x+a_0)\big) \leq \hat{r}(x,1) - \hat{r}(x+1,1).\end{aligned} \tag{A.5.17}$$

It can be checked using (3.5.15) that the reward terms satisfy:

$$\hat{r}(x+1,1) - \hat{r}(x,1) = \begin{cases} -\dfrac{\lambda\beta}{c\mu}, & \text{if } x \geq c-1, \\ 0, & \text{otherwise.} \end{cases} \tag{A.5.18}$$

$$\hat{r}(x+2,1) - \hat{r}(x+1,1) - \hat{r}(x+1,1) + \hat{r}(x,1) = \begin{cases} -\dfrac{\lambda\beta}{c\mu}, & \text{if } x = c-2, \\ 0, & \text{otherwise.} \end{cases} \tag{A.5.19}$$

It is not possible to have $a_0 = 0$ and $a_2 = 1$, since this would contradict the inductive assumption that $\hat{r}(x+2,1) + \lambda\Delta D(x+2,\hat{h}_k) \leq \hat{r}(x+1,1) + \lambda\Delta D(x+1,\hat{h}_k) \leq \hat{r}(x,1) + \lambda\Delta D(x,\hat{h}_k)$. Hence, there are 3 possible cases for the pair $(a_0,a_2) \in \{0,1\} \times \{0,1\}$, considered below:

- If $a_0 = a_2 = 0$, then one may use Lemma 5.1.1 and choose $a_1 = 0$. Then the inequality (A.5.17) holds because the reward terms on the left-hand side are all zero, the expression $\hat{h}_k(x+2) - \hat{h}_k(x+1) - \hat{h}_k(x+1) + \hat{h}_k(x)$ is bounded above by $\big(\hat{r}(x,1) - \hat{r}(x+1,1)\big)/(\lambda\Delta)$ and hence the left-hand side is bounded above by $\hat{r}(x,1) - \hat{r}(x+1,1)$.

- If $a_0 = a_2 = 1$, then one may again use Lemma 5.1.1 and choose $a_1 = 1$. Here, it is easiest to consider the three sub-cases $x < c-2$, $x = c-2$ and $x \geq c-1$ separately:

  - If $x < c-2$, then $\hat{r}(x+2,1) - \hat{r}(x+1,1) - \hat{r}(x+1,1) + \hat{r}(x,1) = 0$ and $\hat{h}_k(x+3) - \hat{h}_k(x+2) - \hat{h}_k(x+2) + \hat{h}_k(x+1)$ is bounded above by $\big(\hat{r}(x+1,1) - \hat{r}(x+2,1)\big)/(\lambda\Delta)$ due to the inductive assumption, but this quantity is equal to zero since $x+1 < c-1$. The right-hand side of (A.5.17) is equal to zero, so the inequality holds.
  - If $x = c-2$, then $\hat{r}(x+2,1) - \hat{r}(x+1,1) - \hat{r}(x+1,1) + \hat{r}(x,1) = -\lambda\beta/(c\mu)$ and $\hat{h}_k(x+3) - \hat{h}_k(x+2) - \hat{h}_k(x+2) + \hat{h}_k(x+1)$ is bounded above by $\big(\hat{r}(x+1,1) - \hat{r}(x+2,1)\big)/(\lambda\Delta)$ due to the inductive assumption, which is equal to $[\lambda\beta/(c\mu)]/(\lambda\Delta) = \beta/(c\mu\Delta)$ since $x+1 \geq$

$c-1$. Hence, the left-hand side of (A.5.17) is bounded above by $-\lambda\beta/(c\mu)+\lambda\beta/(c\mu)=0$. The right-hand side of (A.5.17) is equal to zero, so the inequality holds.

– If $x \geq c-1$, then $\hat{r}(x+2,1)-\hat{r}(x+1,1)-\hat{r}(x+1,1)+\hat{r}(x,1)=0$ and $\hat{h}_k(x+3)-\hat{h}_k(x+2)-\hat{h}_k(x+2)+\hat{h}_k(x+1)$ is bounded above by $\big(\hat{r}(x+1,1)-\hat{r}(x+2,1)\big)/(\lambda\Delta)$ due to the inductive assumption, which is equal to $[\lambda\beta/(c\mu)]/(\lambda\Delta)$. So, the left-hand side of (A.5.17) is bounded above by $\lambda\beta/(c\mu)$. Since $x \geq c-1$, the right-hand side of (A.5.17) is equal to the same quantity, $\lambda\beta/(c\mu)$. So the inequality holds.

- Finally, if $a_0=1$ and $a_1=0$ then it is sufficient to observe:

$$\begin{aligned}
&\hat{r}(x+2,a_2)-\hat{r}(x+1,a_1)-\hat{r}(x+1,a_1)+\hat{r}(x,a_0)\\
&\quad+\lambda\Delta\big(\hat{h}_k(x+2+a_2)-\hat{h}_k(x+1+a_1)-\hat{h}_k(x+1+a_1)+\hat{h}_k(x+a_0)\big)\\
&\leq \hat{r}(x+2,0)-\hat{r}(x+1,0)-\hat{r}(x+1,1)+\hat{r}(x,1)\\
&\quad+\lambda\Delta\big(\hat{h}_k(x+2)-\hat{h}_k(x+1)-\hat{h}_k(x+2)+\hat{h}_k(x+1)\big)\\
&=\hat{r}(x,1)-\hat{r}(x+1,1).
\end{aligned}$$

So the inequality (A.5.17) again holds.

In conclusion, (A.5.17) holds for all $x \in \{0,1,...,\tilde{T}-2\}$ and all possible actions $a_0,a_1,a_2 \in \{0,1\}$. Next, using the inductive assumption and also referring to (A.5.19), one can write:

$$D(x,\hat{h}_k)-D(x-1,\hat{h}_k) \leq \frac{\hat{r}(x-1,1)-\hat{r}(x,1)}{\lambda\Delta} \leq \frac{\hat{r}(x,1)-\hat{r}(x+1,1)}{\lambda\Delta},$$

$$D(x+1,\hat{h}_k)-D(x,\hat{h}_k) \leq \frac{\hat{r}(x,1)-\hat{r}(x+1,1)}{\lambda\Delta}.$$

Then, noting that $D(x,\hat{h}_k) \leq 0$ by the third inductive assumption, it can be seen that the sum of the last three lines in (A.5.16) is bounded above by $(1-\lambda\Delta)\big(\hat{r}(x,1)-\hat{r}(x+1,1)\big)/(\lambda\Delta)$. So, recalling (A.5.17), it follows that $D(x+1,\hat{h}_{k+1})-D(x,\hat{h}_{k+1})$ satisfies:

$$\begin{aligned}
D(x+1,\hat{h}_{k+1})-D(x,\hat{h}_{k+1}) &\leq \hat{r}(x,1)-\hat{r}(x+1,1)+(1-\lambda\Delta)\frac{\hat{r}(x,1)-\hat{r}(x+1,1)}{\lambda\Delta}\\
&=\frac{\hat{r}(x,1)-\hat{r}(x+1,1)}{\lambda\Delta},
\end{aligned}$$

which establishes that $\hat{r}(x+1,1)-\hat{r}(x,1)+\lambda\Delta\big(D(x+1,\hat{h}_{k+1})-D(x,\hat{h}_{k+1})\big) \leq 0$ as required. This completes the inductive proof that all four of the properties (A.5.1)-(A.5.4) hold for all $n \in \mathbb{N}_0$.

Since it is known that $h(x) = \lim_{n\to\infty} h_n(x)$ and $\hat{h}(x) = \lim_{n\to\infty} \hat{h}_n(x)$ by Theorem 3.7.4, this completes the proof of the four properties in the statement of the lemma. □

Proof of Lemma 5.1.5.

The result for $\hat{h}_n$ will be shown first. When $n = 0$, the right-hand side of (5.1.18) is trivially equal to zero since it is assumed that $\hat{h}_0(x) = 0$ for all $x \in \{0, 1, ..., \tilde{T}\}$. By definition of the finite-stage relative value function in (3.7.7), the left-hand side of (5.1.18) reduces to:

$$\max_a \hat{r}(x+1, a) - \max_a \hat{r}(x, a).$$

Let $a^* \in \{0, 1\}$ be a maximising action at state $x + 1$, i.e. $a^* \in \arg\max_a \hat{r}(x + 1, a)$. Then, using the definition of the anticipatory reward function $\hat{r}$ in (5.1.4):

$$\begin{aligned}\max_a \hat{r}(x+1, a) - \max_a \hat{r}(x, a) &= \hat{r}(x+1, a^*) - \max_a \hat{r}(x, a) \\ &\leq \hat{r}(x+1, a^*) - \hat{r}(x, a^*) \\ &\leq 0,\end{aligned}$$

so (5.1.18) holds when $n = 0$. Proceeding by induction again, assume that it also holds at an arbitrary stage $k \in \mathbb{N}_0$. Then, for $x \in \{0, 1, ..., \tilde{T} - 1\}$, one finds (after simplifications):

$$\begin{aligned}D(x, \hat{h}_{k+2}) - D(x, \hat{h}_{k+1}) = \hat{r}(x+1, a_1) - \hat{r}(x, a_0) - \hat{r}(x+1, b_1) + \hat{r}(x, b_0) \\ + \lambda\Delta\left(\hat{h}_{k+1}(x+1+a_1) - \hat{h}_{k+1}(x+a_0) - \hat{h}_k(x+1+b_1) + \hat{h}_k(x+b_0)\right) \\ + \min(x, c)\mu\Delta\left(D(x-1, \hat{h}_{k+1}) - D(x-1, \hat{h}_k)\right) \\ + (1 - \lambda\Delta - \min(x, c)\mu\Delta - I(x < c)\mu\Delta)\left(D(x, \hat{h}_{k+1}) - D(x, \hat{h}_k)\right),\end{aligned}$$

where $a_0, a_1, b_0, b_1 \in \{0, 1\}$ are actions which maximise $\hat{r}(x, a_0) + \lambda\Delta\hat{h}_{k+1}(x + a_0)$, $\hat{r}(x + 1, a_1) + \lambda\Delta\hat{h}_{k+1}(x + 1 + a_1)$, $\hat{r}(x, b_0) + \lambda\Delta\hat{h}_k(x + b_0)$ and $\hat{r}(x + 1, b_1) + \lambda\Delta\hat{h}_k(x + 1 + b_1)$ respectively. By the inductive assumption, the differences $D(x - 1, \hat{h}_{k+1}) - D(x - 1, \hat{h}_k)$ and $D(x, \hat{h}_{k+1}) - D(x, \hat{h}_k)$ are both non-positive. Therefore it will be sufficient to show that:

$$\begin{aligned}&\hat{r}(x+1, a_1) - \hat{r}(x, a_0) - \hat{r}(x+1, b_1) + \hat{r}(x, b_0) \\ &+ \lambda\Delta\left(\hat{h}_{k+1}(x+1+a_1) - \hat{h}_{k+1}(x+a_0) - \hat{h}_k(x+1+b_1) + \hat{h}_k(x+b_0)\right) \leq 0. \qquad \text{(A.5.20)}\end{aligned}$$

It is not possible to have $a_1 = 1$ and $b_0 = 0$. Indeed, if $b_0 = 0$ then $\hat{r}(x, 1) + \lambda\Delta D(x, \hat{h}_k) < 0$ by definition of b_0. Then, by the property (5.1.17) proved in Lemma 5.1.2, it must also be the case

that $\hat{r}(x+1,1)+\lambda\Delta D(x+1,\hat{h}_k)<0$. However, if $a_1=1$ then $\hat{r}(x+1,1)+\lambda\Delta D(x+1,\hat{h}_{k+1})\geq 0$, thereby contradicting the inductive assumption that $D(x+1,\hat{h}_{k+1})\leq D(x+1,\hat{h}_k)$. It follows that there are only 3 possible cases for the pair (a_1,b_0), considered below.

- If $a_1=b_0=0$ then one may apply Lemma 5.1.1 and choose $a_0=0$ and $b_1=0$ in (A.5.20), in which case the inequality holds due to the inductive assumption.
- If $a_1=b_0=1$ then, similarly, one may choose $a_0=1$ and $b_1=1$, in which case the inequality (A.5.20) holds by the same reasoning as above.
- If $a_1=0$ and $b_0=1$ then, by Lemma 5.1.1 again, it is sufficient to choose $a_0=1$ and $b_1=0$ in (A.5.20), in which case the left-hand side is identically equal to zero.

Thus, it is established that $D(x,\hat{h}_{k+2})\leq D(x,\hat{h}_{k+1})$, and therefore (5.1.18) holds for all $n\in\mathbb{N}_0$ as required. Next, the result will be proved for the alternative relative value function h. In this case, the property to be proved is slightly different (in fact, slightly weaker) than the corresponding property for $\hat{h}$. It will be convenient to show, for $x\in\{0,1,...,\tilde{T}-1\}$ and $n\in\mathbb{N}_0$:

$$D(x,h_{n+1})\geq 0 \;\;\Rightarrow\;\; D(x,h_{n+2})\geq D(x,h_{n+1}).$$

Clearly, this will be sufficient to imply (5.1.19). When $n=0$, one has:

$$\begin{aligned} D(x,h_2) = D(x,r) &+ \lambda\Delta\big(h_1(x+1+a_1)-h_1(x+a_0)\big) \\ &+ \min(x,c)\mu\Delta D(x-1,h_1) \\ &+ \left(1-\lambda\Delta-\min(x,c)\mu\Delta-I(x<c)\mu\Delta\right)D(x,h_1), \end{aligned} \tag{A.5.21}$$

where a_0 and a_1 are actions maximising $h_1(x+a_0)$ and $h_1(x+1+a_1)$ respectively. Assume that $D(x,h_1)\geq 0$. Since $h_0(x)=0$ for all $x\in\{0,1,...,\tilde{T}\}$, this simply states that $r(x+1)-r(x)\geq 0$. Hence, in order to show that $D(x,h_2)\geq D(x,h_1)\geq 0$, it suffices to show that the terms added to $r(x+1)-r(x)$ on the right-hand side of (A.5.21) are all non-negative.

Given that $D(x,h_1)\geq 0$ by assumption, it follows by the concavity property (5.1.15) shown in the proof of Lemma 5.1.2 that $D(x-1,h_1)\geq 0$, so it remains only to show:

$$h_1(x+1+a_1)-h_1(x+a_0)\geq 0. \tag{A.5.22}$$

This can be easily done by again taking advantage of Lemma 5.1.1. If $a_0 = 0$, one may choose $a_1 = 0$, in which case the left-hand side of (A.5.22) is non-negative by assumption. On the other hand, if $a_0 = 1$ then one may again choose $a_1 = 0$, in which case the left-hand side of (A.5.22) is equal to zero. This confirms that $D(x, h_1) \geq 0$ implies $D(x, h_2) \geq D(x, h_1)$ as required. As an inductive assumption, suppose that for all $x \in \{0, 1, ..., \tilde{T} - 1\}$ and arbitrary $k \geq 1$:

$$D(x, h_k) \geq 0 \quad \Rightarrow \quad D(x, h_{k+1}) \geq D(x, h_k). \tag{A.5.23}$$

After simplifications, one has the following expression for $D(x, h_{k+2}) - D(x, h_{k+1})$:

$$\begin{aligned} D(x, h_{k+2}) - D(x, h_{k+1}) = \lambda\Delta\big(h_{k+1}(x + 1 + a_1) - h_{k+1}(x + a_0) - h_k(x + 1 + b_1) + h_k(x + b_0)\big) \\ + \min(x, c)\mu\Delta \left(D(x - 1, h_{k+1}) - D(x - 1, h_k)\right) \\ + (1 - \lambda\Delta - \min(x, c)\mu\Delta - I(x < c)\mu\Delta) \left(D(x, h_{k+1}) - D(x, h_k)\right), \end{aligned} \tag{A.5.24}$$

where a_0, a_1, b_0 and b_1 are actions maximising $h_{k+1}(x + a_0)$, $h_{k+1}(x + 1 + a_1)$, $h_k(x + b_0)$ and $h_k(x + 1 + b_1)$ respectively. It must then be shown that the expression on the right-hand side of (A.5.24) is non-negative, under the assumption that $D(x, h_{k+1}) \geq 0$. Indeed, if $D(x, h_{k+1}) \geq 0$ then it can be deduced that $D(x-1, h_{k+1}) \geq 0$ by the concavity property of h_{k+1} proved in Lemma 5.1.2. Hence, the second and third lines on the right-hand side of (A.5.24) must be non-negative; indeed, note that if $D(x, h_k)$ is positive then the inductive assumption implies that $D(x, h_{k+1}) - D(x, h_k)$ is non-negative, whereas if $D(x, h_k) \leq 0$ then $D(x, h_{k+1}) - D(x, h_k)$ is automatically non-negative since $D(x, h_{k+1}) \geq 0$. Similar reasoning can be used in order to show that $D(x - 1, h_{k+1}) - D(x - 1, h_k)$ must also be non-negative. In light of these arguments, it only remains to show:

$$h_{k+1}(x + 1 + a_1) - h_{k+1}(x + a_0) - h_k(x + 1 + b_1) + h_k(x + b_0) \geq 0. \tag{A.5.25}$$

Note that it is not possible to have $a_0 = 0$ and $b_1 = 1$. Indeed, if $b_1 = 1$ then this implies $D(x + 1, h_k) \geq 0$ by definition of b_1, and hence $D(x, h_k) \geq 0$ due to the concavity of h_k proved in Lemma 5.1.2. However, in this case the inductive assumption implies $D(x, h_{k+1}) \geq 0$, which implies $a_0 = 1$ by definition of a_0 and thereby gives a contradiction. So there are three possible cases for the pair $(a_0, b_1) \in \{0, 1\} \times \{0, 1\}$. These cases are considered below.

- If $a_0 = b_1 = 0$, then one may apply Lemma 5.1.1 and choose $a_1 = b_0 = 0$ in (A.5.25), in which

case it is sufficient to show that the following holds:

$$D(x, h_{k+1}) - D(x, h_k) \geq 0. \tag{A.5.26}$$

If $D(x, h_k) \geq 0$, then (A.5.26) holds due to the inductive assumption. On the other hand, if $D(x, h_k) < 0$ then (A.5.26) holds due to the assumption that $D(x, h_{k+1}) \geq 0$.

- If $a_0 = b_1 = 1$ then similarly, one may apply 5.1.1 and choose $b_0 = 1$ in (A.5.25), in which case it is sufficient to show the following:

$$h_{k+1}(x + 1 + a_1) - h_{k+1}(x + 1) - h_k(x + 2) + h_k(x + 1) \geq 0. \tag{A.5.27}$$

If $D(x + 1, h_k) < 0$, then choosing $a_1 = 0$ in (A.5.27) causes the left-hand side to be non-negative. On the other hand, if $D(x + 1, h_k) \geq 0$ then choosing $a_1 = 1$ achieves the same result, since $D(x + 1, h_{k+1}) \geq D(x + 1, h_k)$ due to the inductive assumption.

- If $a_0 = 1$ and $b_1 = 0$ then one may again apply Lemma 5.1.1 and choose $a_1 = 0$ and $b_0 = 1$, in which the left-hand side of (A.5.26) is identically equal to zero.

These arguments show that (A.5.25) holds in all possible cases for the actions a_0 and b_1, so the expression for $D(x, h_{k+2}) - D(x, h_{k+1})$ in (A.5.24) is non-negative as required. This completes the inductive proof that (5.1.19) holds for all $n \in \mathbb{N}_0$ as stated by the lemma. □

Proof of Lemma 5.1.7.

Consider two different demand rates λ_0 and λ_1, with $0 < \lambda_0 < \lambda_1$. For each state $x \in \{0, 1, ..., \tilde{T}\}$, the relative values $h(x, \lambda_1)$ and $h(x, \lambda_2)$ are obtained as limits of the finite-stage function values $h_n(x, \lambda_1)$ and $h_n(x, \lambda_2)$ respectively as $n \to \infty$. These finite-stage values are obtained by relative value iteration and depend on the transition probabilities defined in (3.5.3), and these transition probabilities depend upon the discrete-time step size parameter Δ. Given an arbitrary demand rate $\lambda > 0$, the parameter Δ may take any positive value *smaller* than $(\lambda + c\mu)^{-1}$; therefore, $(\lambda_1 + c\mu)^{-1}$ is a valid choice under either of the demand rates λ_0 and λ_1. In this proof, Δ will be held at a *fixed* value $\Delta = (\lambda_1 + c\mu)^{-1}$ as opposed to being assigned different values for the two respective demand rates; note that, due to Lemma 3.8.4, changing the value of Δ affects the relative values $h(\cdot)$ only by a positive multiplicative factor, so any such change cannot cause the first-order difference in

(5.1.25) to go from positive to negative, or vice versa. It should also be noted that the concavity property (5.1.15) proved in Lemma 5.1.2 is not affected by altering the value of Δ.

The aim of the proof is to show that, for $x \in \{0, 1, ..., \tilde{T} - 1\}$ and integers $n \geq 0$:

$$D(x, \lambda_0, h_n) \geq D(x, \lambda_1, h_n), \tag{A.5.28}$$

with $\Delta = (\lambda_1 + c\mu)^{-1}$ used in the computation of *all* of the terms in (A.5.28), including those based on the demand rate λ_0. When $n = 0$, both sides of (A.5.28) are equal to zero. Assume that the inequality holds at an arbitrary stage $k \in \mathbb{N}_0$. Then, after simplifications:

$$\begin{aligned}
&D(x, \lambda_0, h_{k+1}) - D(x, \lambda_1, h_{k+1}) \\
&= \lambda_0 \Delta \left(h_k(x + 1 + a_1, \lambda_0) - h_k(x + a_0, \lambda_0) - h_k(x + 1 + b_1, \lambda_1) + h_k(x + b_0, \lambda_1) \right) \\
&\quad + (\lambda_1 - \lambda_0)\Delta \left(D(x, \lambda_0, h_k) - h_k(x + 1 + b_1, \lambda_1) + h_k(x + b_0, \lambda_1) \right) \\
&\quad + \min(x, c)\mu\Delta \left(D(x - 1, \lambda_0, h_k) - D(x - 1, \lambda_1, h_k) \right) \\
&\quad + (1 - \lambda_1\Delta - \min(x, c)\mu\Delta - I(x < c)\mu\Delta) \left(D(x, \lambda_0, h_k) - D(x, \lambda_1, h_k) \right),
\end{aligned} \tag{A.5.29}$$

where a_0, a_1, b_0 and b_1 are actions maximising $h_k(x + a_0, \lambda_0)$, $h_k(x + 1 + a_1, \lambda_0)$, $h_k(x + b_0, \lambda_1)$ and $h_k(x + 1 + b_1, \lambda_1)$ respectively. Due to the inductive assumption, the differences $D(x - 1, \lambda_0, h_k) - D(x - 1, \lambda_1, h_k)$ and $D(x, \lambda_0, h_k) - D(x, \lambda_1, h_k)$ are both non-negative, so only the terms on the second and third lines in (A.5.29) remain to be checked for non-negativity. By Lemma 5.1.1, it suffices to consider all possible combinations for the actions a_0 and b_1 and show that, for each combination, there exists some choice of a_1 and b_0 such that the following holds:

$$\begin{aligned}
&\lambda_0 \Delta \left(h_k(x + 1 + a_1, \lambda_0) - h_k(x + a_0, \lambda_0) - h_k(x + 1 + b_1, \lambda_1) + h_k(x + b_0, \lambda_1) \right) \\
&+ (\lambda_1 - \lambda_0)\Delta \left(D(x, \lambda_0, h_k) - h_k(x + 1 + b_1, \lambda_1) + h_k(x + b_0, \lambda_1) \right) \geq 0.
\end{aligned} \tag{A.5.30}$$

It is not possible to have $a_0 = 0$ and $b_1 = 1$, since this would imply $D(x, \lambda_0, h_k) < 0$ and $D(x + 1, \lambda_1, h_k) \geq 0$, and the latter of these inequalities would then imply $D(x, \lambda_1, h_k) \geq 0$ (due to Lemma 5.1.2), thereby contradicting the inductive assumption that $D(x, \lambda_0, h_k) \geq D(x, \lambda_1, h_k)$. Hence, there are only three possibilities remaining for the pair $(a_0, b_1) \in \{0, 1\}^2$:

- If $a_0 = b_1 = 0$, then one may apply Lemma 5.1.1 and choose $a_1 = 0$ and $b_0 = 0$ in (A.5.30), in which case the left-hand side is non-negative by inductive assumption.

- If $a_0 = b_1 = 1$, one may choose $a_1 = 1$ and $b_0 = 1$. Then the first line on the left-hand side of (A.5.30) is non-negative by the inductive assumption, and the second line gives:

$$(\lambda_1 - \lambda_0)\Delta \left(D(x, \lambda_0, h_k) - D(x+1, \lambda_1, h_k)\right). \tag{A.5.31}$$

  Due to the concavity of $h_k(\cdot)$ proved in Lemma 5.1.2, (A.5.31) is bounded below by:

$$(\lambda_1 - \lambda_0)\Delta \left(D(x, \lambda_0, h_k) - D(x, \lambda_1, h_k)\right),$$

  which is non-negative due to the inductive assumption.

- If $a_0 = 1$ and $b_1 = 0$, one may again apply Lemma 5.1.1 and choose $a_1 = 0$ and $b_0 = 1$. Then the first line on the left-hand side of (A.5.30) is equal to zero, and the second line simplifies to $(\lambda_1 - \lambda_0)\Delta\, D(x, \lambda_0, h_k)$. Given that $a_0 = 1$, it must be the case that $D(x, \lambda_0, h_k) \geq 0$. So, again, the left-hand side of (A.5.30) is non-negative as required.

These arguments confirm that the sum of the terms on the right-hand side of (A.5.29) is non-negative, so $D(x, \lambda_0, h_{k+1}) \geq D(x, \lambda_1, h_{k+1})$ as required. This completes the inductive proof that (A.5.28) holds for all $n \in \mathbb{N}_0$, and the result follows by taking limits as $n \to \infty$. □

Proof of Lemma 5.1.9.

Consider an arbitrary threshold policy θ with threshold $T \in \mathbb{N}$, and for each state $x \in \{0, 1, ..., T\}$ let $\pi_\theta(x)$ denote the stationary probability of the process being in state x. Then the expected long-run average reward under θ, given a demand rate $\lambda > 0$, is given by:

$$g_\theta(\lambda) = \lambda\alpha(1 - \pi_\theta(T)) - \beta \sum_{i=0}^{T} i\, \pi_\theta(i).$$

Let $\rho = \lambda/(c\mu)$. Then the stationary probabilities are given by (see, e.g. [67]):

$$\pi_\theta(0) = \begin{cases} \left(\displaystyle\sum_{x=0}^{c-1} \frac{(c\rho)^x}{x!} + \frac{(c\rho)^c}{c!} \frac{1 - \rho^{T-c+1}}{1 - \rho} \right)^{-1}, & \text{if } \rho \neq 1, \\ \left(\displaystyle\sum_{x=0}^{c-1} \frac{c^x}{x!} + \frac{c^c}{c!} (T - c + 1) \right)^{-1}, & \text{if } \rho = 1, \end{cases}$$

$$\pi_\theta(x) = \begin{cases} \dfrac{(c\rho)^x}{x!}\, \pi_\theta(0), & \text{if } 1 \leq x < c, \\ \dfrac{(c\rho)^x}{c^{x-c} c!}\, \pi_\theta(0), & \text{if } c \leq x \leq T. \end{cases}$$

Using elementary results for sums and products of continuous functions, it follows that $g_\theta(\lambda)$ is a continuous function of λ. By Theorem 5.1.4, the selfishly optimal policy $\tilde{\theta}$ is a threshold policy with fixed threshold $\tilde{T} = \lfloor \alpha c\mu/\beta \rfloor$, so by the above, $\tilde{g}(\lambda)$ is continuous in λ. Next, consider the socially optimal case. Given any demand rate $\lambda > 0$, there exists a socially optimal policy with threshold $T^*(\lambda) \leq \tilde{T}$ due to Theorem 5.1.4. Also, due to Theorem 5.1.8, $T^*(\lambda)$ is monotonically decreasing with λ and $\lim_{\lambda\to\infty} T^*(\lambda) = c$, assuming that $T^*(\lambda)$ corresponds to the policy found by relative value iteration on $\{0, 1, ..., \tilde{T}\}$. It follows that there exist m demand rates $\lambda_1 < \lambda_2 < ... < \lambda_m$ and $m+1$ thresholds $T_0^* > T_1^* > ... > T_{m-1}^* > T_m^* = c$, such that:

$$T^*(\lambda) = T_i^* \ \ \forall\ \lambda \in (\lambda_i, \lambda_{i+1}] \qquad (i = 0, 1, ..., m-1), \tag{A.5.32}$$

and $T^*(\lambda) = c$ for all $\lambda > \lambda_m$. One may define $\lambda_0 := 0$ so that (A.5.32) makes sense when $i = 0$ (obviously, λ_0 is not a real demand rate since demand rates should be positive). Note that it is possible to have $m = 0$; this represents the case where $\tilde{T} = c$, in which case there is nothing to prove because previous results imply $T^*(\lambda) = c$ for all demand rates $\lambda > 0$. Essentially, the proof is accomplished by using the fact that a maximum of continuous functions is itself continuous. To be more specific, it is necessary to show that for each $i \in \{1, 2, ..., m\}$:

$$\lim_{\lambda\to\lambda_i^+} g^*(\lambda) = g^*(\lambda_i). \tag{A.5.33}$$

That is, $g^*(\lambda)$ has right-sided continuity at λ_i. Note that the left-sided continuity (for $i \geq 1$) is automatic due to the definition of λ_i in (A.5.32), which implies that $g^*(\lambda)$ is equal to the average reward of a fixed threshold policy for $\lambda \in (\lambda_{i-1}, \lambda_i]$. Using similar arguments, the limit on the left-hand side of (A.5.33) is known to exist since $g^*(\lambda)$ is equal to the average reward of the policy with threshold T_i^* as λ approaches λ_i from the right. Suppose, for a contradiction, that $\lim_{\lambda\to\lambda_i^+} g^*(\lambda) \neq g^*(\lambda_i)$ for some $i \in \{1, 2, ..., m\}$. Equivalently, for some $\epsilon > 0$:

$$\left| \lim_{\lambda\to\lambda_i^+} g^*(\lambda) - g^*(\lambda_i) \right| = \epsilon. \tag{A.5.34}$$

Let θ_L and θ_U denote the threshold policies with thresholds T_{i-1}^* and T_i^* respectively. Note that $g^*(\lambda) = g_{\theta_L}(\lambda)$ for demand rates $\lambda \in (\lambda_{i-1}, \lambda_i]$, and $g^*(\lambda) = g_{\theta_U}(\lambda)$ for demand rates $\lambda \in (\lambda_i, \lambda_{i+1}]$ (or $\lambda \in (\lambda_i, \infty)$ if $i = m$). So, (A.5.34) can be written equivalently as follows:

$$\left| \lim_{\lambda\to\lambda_i^+} g_{\theta_U}(\lambda) - g_{\theta_L}(\lambda_i) \right| = \epsilon > 0. \tag{A.5.35}$$

Also, $\lim_{\lambda \to \lambda_i^+} g_{\theta_U}(\lambda) = g_{\theta_U}(\lambda_i)$ by continuity of $g_{\theta_U}(\lambda)$, so (A.5.35) implies:

$$\left| g_{\theta_U}(\lambda_i) - g_{\theta_L}(\lambda_i) \right| = \epsilon > 0. \tag{A.5.36}$$

There are two possible cases to consider in (A.5.36); first, consider the case where $g_{\theta_U}(\lambda_i) > g_{\theta_L}(\lambda_i)$. By continuity of $g_{\theta_U}(\lambda)$ and $g_{\theta_L}(\lambda)$, there exist two values $\delta_L, \delta_U > 0$ such that:

$$\left| g_{\theta_L}(\lambda) - g_{\theta_L}(\lambda_i) \right| < \frac{\epsilon}{2} \qquad \forall\ \lambda \in (\lambda_i - \delta_L, \lambda_i + \delta_L), \tag{A.5.37}$$

$$\left| g_{\theta_U}(\lambda) - g_{\theta_U}(\lambda_i) \right| < \frac{\epsilon}{2} \qquad \forall\ \lambda \in (\lambda_i - \delta_U, \lambda_i + \delta_U). \tag{A.5.38}$$

Let $\delta := \min(\delta_L, \delta_U)$. Then, for all demand rates $\lambda \in (\lambda_i - \delta, \lambda_i)$:

$$\begin{aligned} g_{\theta_U}(\lambda) - g_{\theta_L}(\lambda) &> g_{\theta_U}(\lambda_i) - \frac{\epsilon}{2} - \left(g_{\theta_L}(\lambda_i) + \frac{\epsilon}{2} \right) \\ &= \epsilon - \frac{\epsilon}{2} - \frac{\epsilon}{2} = 0, \end{aligned} \tag{A.5.39}$$

where the inequality is due to (A.5.37) and (A.5.38), and the equality in the second line is due to (A.5.36) and the assumption that $g_{\theta_U}(\lambda_i) > g_{\theta_L}(\lambda_i)$. However, (A.5.39) states that $g_{\theta_U}(\lambda) > g_{\theta_L}(\lambda)$ for $\lambda \in (\lambda_i - \delta, \lambda_i)$, which contradicts the fact that the policy θ_L with threshold T_{i-1}^* is optimal for λ in the interval $(\lambda_{i-1}, \lambda_i]$. Similarly, consider the case where $g_{\theta_U}(\lambda_i) < g_{\theta_L}(\lambda_i)$. In this case, due to (A.5.37) and (A.5.38) again, the following holds for $\lambda \in (\lambda_i, \lambda_i + \delta)$:

$$\begin{aligned} g_{\theta_L}(\lambda) - g_{\theta_U}(\lambda) &> g_{\theta_L}(\lambda_i) - \frac{\epsilon}{2} - \left(g_{\theta_U}(\lambda_i) + \frac{\epsilon}{2} \right) \\ &= \epsilon - \frac{\epsilon}{2} - \frac{\epsilon}{2} = 0, \end{aligned}$$

which states that $g_{\theta_L}(\lambda) > g_{\theta_U}(\lambda)$ for $\lambda \in (\lambda_i, \lambda_i + \delta)$, thereby contradicting the fact that the policy θ_U with threshold T_i^* is optimal for λ in the interval $(\lambda_i, \lambda_{i+1}]$ (or (λ_i, ∞) if $i = m$). It follows that (A.5.33) holds for each $i \in \{1, 2, ..., m\}$, which completes the proof. □

Proof of Lemma 5.2.9.

The proof is accomplished using induction on the iterates $h_n(\mathbf{x})$ obtained by relative value iteration. It will be shown that for any $\mathbf{x} \in \tilde{S}$ and $j \in \{1, 2, ..., N\}$ such that $\mathbf{x}^{j+} \in \tilde{S}$:

$$D_j(\mathbf{x}, h_n) \leq \frac{\alpha_j \mu_j - \beta_j}{\mu_j \Delta} \qquad \forall\ n \in \mathbb{N}_0. \tag{A.5.40}$$

It is assumed that $h_0(\mathbf{x}) = 0$ for all $\mathbf{x} \in \tilde{S}$, so (A.5.40) holds trivially when $n = 0$. Assume that it also holds when $n = k$, where $k \in \mathbb{N}_0$ is arbitrary. Then, at stage $k+1$:

$$D_j(\mathbf{x}, h_{k+1}) = D_j(\mathbf{x}, r) + \lambda\Delta\left(h_k((\mathbf{x}^{j+})^{a_1+}) - h_k(\mathbf{x}^{a_0+})\right) + \sum_{i=1}^{N} \min(x_i, c_i)\mu_i\Delta D_j(\mathbf{x}^{i-}, h_k)$$
$$+ \left(1 - \lambda\Delta - \sum_{i=1}^{N} \min(x_i, c_i)\mu_i\Delta - I(x_j < c_j)\mu_j\Delta\right) D_j(\mathbf{x}, h_k), \tag{A.5.41}$$

where I is the indicator function and a_0 and a_1 are optimal actions at states $\mathbf{x}$ and $\mathbf{x}^{j+}$ respectively in a $(k+1)$-stage problem; that is, $a_0 \in \arg\max_a h_k(\mathbf{x}^{a+})$ and $a_1 \in \arg\max_a h_k((\mathbf{x}^{j+})^{a+})$. First, it may be shown that the difference $h_k((\mathbf{x}^{j+})^{a_1+}) - h_k(\mathbf{x}^{a_0+})$ is bounded above by $(\alpha_j\mu_j - \beta_j)/(\mu_j\Delta)$. Indeed, one may apply the following argument (based on Lemma 5.1.1):

$$h_k((\mathbf{x}^{j+})^{a_1+}) - h_k(\mathbf{x}^{a_0+}) \leq h_k((\mathbf{x}^{j+})^{a_1+}) - h_k(\mathbf{x}^{a_1+}) = D_j(\mathbf{x}^{a_1+}, h_k) \leq \frac{\alpha_j\mu_j - \beta_j}{\mu_j\Delta},$$

where the last inequality follows from the inductive assumption. Similarly, $D_j(\mathbf{x}^{i-}, h_k)$ is bounded above by $(\alpha_j\mu_j - \beta_j)/(\mu_j\Delta)$ for any $i \in \{1, 2, ..., N\}$, again due to the inductive assumption. To complete the proof, it will be convenient to consider the cases $x_j \geq c_j$ and $x_j < c_j$ separately. If $x_j \geq c_j$, then $D_j(\mathbf{x}, r) = -\beta_j < 0$ and $I(x_j < c_j) = 0$. Hence, due to the previous arguments, the right-hand side of (A.5.41) may be bounded above by a convex combination of terms bounded above by $(\alpha_j\mu_j - \beta_j)/(\mu_j\Delta)$, which establishes that $D_j(\mathbf{x}, h_{k+1}) \leq (\alpha_j\mu_j - \beta_j)/(\mu_j\Delta)$ as required. On the other hand, if $x_j < c_j$, then $D_j(\mathbf{x}, r) = \alpha_j\mu_j - \beta_j > 0$. However, since $I(x_j < c_j) = 1$ in this case, (A.5.41) implies (again making use of the inductive assumption):

$$D_j(\mathbf{x}, h_{k+1}) \leq \alpha_j\mu_j - \beta_j + (1 - \mu_j\Delta)\frac{\alpha_j\mu_j - \beta_j}{\mu_j\Delta} = \frac{\alpha_j\mu_j - \beta_j}{\mu_j\Delta}.$$

In conclusion, $D_j(\mathbf{x}, h_{k+1})$ is bounded above by $(\alpha_j\mu_j - \beta_j)/(\mu_j\Delta)$ in all possible cases, which completes the inductive proof that (A.5.40) holds for all $n \in \mathbb{N}_0$. Since $\lim_{n\to\infty} h_n(\mathbf{x}) = h(\mathbf{x})$ for all states $\mathbf{x} \in \tilde{S}$ by Theorem 3.7.4, the required upper bound (5.2.12) follows. □

Proof of Lemma 5.2.10.

It is sufficient to show that for states $\mathbf{x} \in \tilde{S}$, $i, j \in \{1, 2\}$ with $i \neq j$ and $n \in \mathbb{N}_0$:

$$D_{ij}(\mathbf{x}, h_n) \leq 0, \tag{A.5.42}$$

$$D_{jj}(\mathbf{x}, h_n) \leq 0, \tag{A.5.43}$$

$$D_{jj}(\mathbf{x}, h_n) - D_{ij}(\mathbf{x}, h_n) \leq 0. \tag{A.5.44}$$

Since the finite state space $\tilde{S}$ is being considered, it should be noted that (A.5.42) needs to be proved only in the case of states $\mathbf{x} \in \tilde{S}$ with $x_i \leq \tilde{B}_i - 1$ and $x_j \leq \tilde{B}_j - 1$. Similarly, (A.5.43) applies to states with $x_j \leq \tilde{B}_j - 2$ and (A.5.44) applies to states with both $x_j \leq \tilde{B}_j - 2$ and $x_i \leq \tilde{B}_i - 1$. All three properties hold trivially when $n = 0$ since $h_0(\mathbf{x}) = 0$ for all $\mathbf{x} \in \tilde{S}$. Assume that they also hold when $n = k$, where $k \in \mathbb{N}_0$ is arbitrary. The submodularity property (A.5.42) will be considered first. At stage $n = k + 1$, one has (after simplifications):

$$\begin{aligned} D_{ij}(\mathbf{x}, h_{k+1}) = D_{ij}(\mathbf{x}, r) &+ \lambda\Delta \left(h_k(((\mathbf{x}^{i+})^{j+})^{a_1+}) - h_k((\mathbf{x}^{i+})^{a_0+}) - h_k((\mathbf{x}^{j+})^{b_1+}) + h_k(\mathbf{x}^{b_0+}) \right) \\ &+ \sum_{m=1}^{2} \min(x_m, 1)\mu_m \Delta D_{ij}(\mathbf{x}^{m-}, h_k) \\ &+ \left(1 - \lambda\Delta - \sum_{m=1}^{2} \min(x_m, 1)\mu_m\Delta - I(x_i = 0)\mu_i\Delta - I(x_j = 0)\mu_j\Delta \right) D_{ij}(\mathbf{x}, h_k), \end{aligned}$$

where a_0, a_1, b_0, b_1 are optimal actions at states $\mathbf{x}^{i+}$, $(\mathbf{x}^{i+})^{j+}$, $\mathbf{x}$ and $\mathbf{x}^{j+}$ respectively in a $(k+1)$-stage problem. It is known that $D_{ij}(\mathbf{x}, r) = 0$, and $D_{ij}(\mathbf{x}, h_k)$ and $D_{ij}(\mathbf{x}^{m-}, h_k)$ (for $m \in \{1, 2\}$) are non-positive by the inductive assumption. Hence, it suffices to show:

$$h_k(((\mathbf{x}^{i+})^{j+})^{a_1+}) - h_k((\mathbf{x}^{i+})^{a_0+}) - h_k((\mathbf{x}^{j+})^{b_1+}) + h_k(\mathbf{x}^{b_0+}) \ \leq 0. \tag{A.5.45}$$

One may proceed by considering all possible cases for the actions $a_1, b_0 \in \{0, 1, 2\}$. First, note that if $b_0 = 0$ then it is not possible to have $a_1 = 1$ or $a_1 = 2$. Indeed, if $a_1 = j$ for some $j \in \{1, 2\}$ then this implies $D_j((\mathbf{x}^{i+})^{j+}, h_k) \geq 0$ and hence (by the assumed concavity of h_k) $D_j(\mathbf{x}^{i+}, h_k) \geq 0$, in which case one cannot have $b_0 = 0$ since this implies $D_j(\mathbf{x}, h_k) < 0$ and thereby contradicts the inductive assumption that $D_{ij}(\mathbf{x}, h_k) \leq 0$. The number of cases to consider may be further reduced by noting that the expression for $D_{ij}(\mathbf{x}, h_k)$ is symmetric in i and j (that is, $D_{ij}(\mathbf{x}, h_k)$=$D_{ji}(\mathbf{x}, h_k)$), so (for example) it is not necessary to treat $a_1 = b_0 = i$ and $a_1 = b_0 = j$ as separate cases. After reductions, four possibilities remain for a_1 and b_0. These are considered below.

- If $a_1 = b_0 = 0$, one may apply the AS principle (Lemma 5.1.1) and choose $a_0 = b_1 = 0$, then (A.5.45) holds since the left-hand side is equal to $D_{ij}(\mathbf{x}, h_k) \leq 0$.
- If $a_1 = b_0 = j$ for some $j \in \{1, 2\}$, then similarly one may choose $a_0 = b_1 = j$, in which case (A.5.45) holds since the left-hand side is equal to $D_{ij}(\mathbf{x}^{j+}, h_k) \leq 0$.
- If $a_1 = 0$ and $b_0 = j$ for some $j \in \{1, 2\}$, then one may choose $a_0 = j$ and $b_1 = 0$, in which case (A.5.45) holds since the left-hand side is equal to zero.

- If $a_1 = j$ and $b_0 = i$ for some $i, j \in \{1, 2\}$ with $i \neq j$, one may choose $a_0 = j$ and $b_1 = i$, in which case (A.5.45) holds since the left-hand side is equal to $D_{jj}(\mathbf{x}^{i+}, h_k) \leq 0$.

The conclusion is that $D_{ij}(\mathbf{x}, h_{k+1})$ is non-positive in all possible cases, as required. Note that the inductive assumptions of submodularity *and* concavity for the function h_k are required in order to verify that h_{k+1} is submodular. It is a general theme of this proof that the three structural properties of interest (submodularity, concavity and diagonal submissiveness) depend on each other, which is why they are not proved using separate lemmas. Next, the diagonal submissiveness property will be considered. One may derive the following expression for $D_{jj}(\mathbf{x}, h_{k+1})$:

$$\begin{aligned}
&D_{jj}(\mathbf{x}, h_{k+1}) - D_{ij}(\mathbf{x}, h_{k+1}) = D_{jj}(\mathbf{x}, r) - D_{ij}(\mathbf{x}, r) \\
&\quad + \lambda\Delta\left(h_k(((\mathbf{x}^{j+})^{j+})^{a_1+}) - h_k(((\mathbf{x}^{i+})^{j+})^{a_0+}) - h_k((\mathbf{x}^{j+})^{b_1+}) + h_k((\mathbf{x}^{i+})^{b_0+})\right) \\
&\quad + \sum_{m=1}^{2} \min(x_m, 1)\mu_m\Delta\big(D_{jj}(\mathbf{x}^{m-}, h_k) - D_{ij}(\mathbf{x}^{m-}, h_k)\big) \\
&\quad + I(x_j = 0)\mu_j\Delta D_j(\mathbf{x}, h_k) + I(x_i = 0)\mu_i\Delta D_{jj}(\mathbf{x}, h_k) \\
&\quad + \left(1 - \lambda\Delta - \sum_{m=1}^{2} \min(x_m, 1)\mu_m\Delta - I(x_j = 0)\mu_j\Delta - I(x_i = 0)\mu_i\Delta\right)\big(D_{jj}(\mathbf{x}, h_k) - D_{ij}(\mathbf{x}, h_k)\big),
\end{aligned} \tag{A.5.46}$$

where a_0, a_1, b_0 and b_1 are optimal actions at states $(\mathbf{x}^{i+})^{j+}$, $(\mathbf{x}^{j+})^{j+}$, $\mathbf{x}^{i+}$ and $\mathbf{x}^{j+}$ respectively in a $(k+1)$-stage problem. Due to Lemma 5.2.8, $D_{jj}(\mathbf{x}, r) - D_{ij}(\mathbf{x}, r) \leq 0$. In addition, $D_{jj}(\mathbf{x}, h_k) - D_{ij}(\mathbf{x}, h_k) \leq 0$ and $D_{jj}(\mathbf{x}^{m-}, h_k) - D_{ij}(\mathbf{x}^{m-}, h_k) \leq 0$ (for $m \in \{1, 2\}$) due to the inductive assumption. To continue the proof, it will be useful to show:

$$h_k(((\mathbf{x}^{j+})^{j+})^{a_1+}) - h_k(((\mathbf{x}^{i+})^{j+})^{a_0+}) - h_k((\mathbf{x}^{j+})^{b_1+}) + h_k((\mathbf{x}^{i+})^{b_0+}) \leq 0. \tag{A.5.47}$$

Again, this may be done by considering the possible cases for the actions $a_1, b_0 \in \{0, 1, 2\}$. This time, there are nine possibilities. These are considered below.

- If $a_1 = b_0 = 0$, then one may apply Lemma 5.1.1 and choose $a_0 = b_1 = 0$. Then (A.5.47) holds because the left-hand side is equal to $D_{jj}(\mathbf{x}, h_k) - D_{ij}(\mathbf{x}, h_k) \leq 0$.

- If $a_1 = b_0 = j$, one may again apply Lemma 5.1.1 and choose $a_0 = b_1 = j$. Then (A.5.47) holds because the left-hand side is equal to $D_{jj}(\mathbf{x}^{j+}, h_k) - D_{ij}(\mathbf{x}^{j+}, h_k) \leq 0$.

- If $a_1 = b_0 = i$, then similarly one may choose $a_0 = b_1 = i$. Then (A.5.47) holds because the left-hand side is equal to $D_{jj}(\mathbf{x}^{i+}, h_k) - D_{ij}(\mathbf{x}^{i+}, h_k) \leq 0$.

- If $a_1 = i$ and $b_0 = j$, then one may choose $a_0 = j$ and $b_1 = i$, in which case (A.5.47) holds trivially because the left-hand side is equal to zero.

- If $a_1 = 0$ and $b_0 = j$, then one may choose $a_0 = 0$ and $b_1 = j$, in which case (A.5.47) again holds trivially since the left-hand side is equal to zero.

- If $a_1 = 0$ and $b_0 = i$, one may choose $a_0 = 0$ and $b_1 = i$ and aim to show:

$$h_k((\mathbf{x}^{j+})^{j+}) - h_k((\mathbf{x}^{i+})^{j+}) - h_k((\mathbf{x}^{i+})^{j+}) + h_k((\mathbf{x}^{i+})^{i+}) \leq 0. \tag{A.5.48}$$

  Due to the inductive assumption, the left-hand side of (A.5.48) is bounded above by:

$$\begin{aligned} &h_k(\mathbf{x}^{j+}) - h_k(\mathbf{x}^{i+}) - h_k((\mathbf{x}^{i+})^{j+}) + h_k((\mathbf{x}^{i+})^{i+}) \\ &= h_k((\mathbf{x}^{i+})^{i+}) - h_k((\mathbf{x}^{i+})^{j+}) - h_k(\mathbf{x}^{i+}) + h_k(\mathbf{x}^{j+}), \end{aligned}$$

  which is equal to $D_{ii}(\mathbf{x}, h_k) - D_{ij}(\mathbf{x}, h_k)$ and is therefore non-positive due to the same inductive assumption (with components i and j interchanged).

- If $a_1 = j$ and $b_0 = 0$, one may choose $a_0 = j$ and $b_1 = 0$. Then, due to the inductive assumption of diagonal submissiveness again, one may write:

$$\begin{aligned} &h_k(((\mathbf{x}^{j+})^{j+})^{j+}) - h_k(((\mathbf{x}^{i+})^{j+})^{j+}) - h_k(\mathbf{x}^{j+}) + h_k(\mathbf{x}^{i+}) \\ &\leq h_k((\mathbf{x}^{j+})^{j+}) - h_k((\mathbf{x}^{i+})^{j+}) - h_k(\mathbf{x}^{j+}) + h_k(\mathbf{x}^{i+}), \end{aligned}$$

  which is equal to $D_{jj}(\mathbf{x}, h_k) - D_{ij}(\mathbf{x}, h_k) \leq 0$.

- If $a_1 = i$ and $b_0 = 0$, one may choose $a_0 = 0$ and $b_1 = i$. Then the left-hand side of (A.5.47) is equal to $D_{jj}(\mathbf{x}^{i+}, h_k)$, which is non-positive due to concavity of h_k.

- Finally, if $a_1 = j$ and $b_0 = i$, then $D_j((\mathbf{x}^{j+})^{j+}, h_k) \geq D_i((\mathbf{x}^{j+})^{j+}, h_k)$ and $D_j(\mathbf{x}^{i+}, h_k) \leq D_i(\mathbf{x}^{i+}, h_k)$ by definition of a_1 and b_0. The former inequality implies $D_j(\mathbf{x}^{j+}, h_k) \geq D_i(\mathbf{x}^{j+}, h_k)$ due to the inductive assumption of diagonal submissiveness at stage k, which in turn implies $D_j(\mathbf{x}, h_k) \geq D_i(\mathbf{x}, h_k)$. By the same reasoning, the inequality $D_j(\mathbf{x}^{i+}, h_k) \leq D_i(\mathbf{x}^{i+}, h_k)$ implies $D_j(\mathbf{x}, h_k) \leq D_i(\mathbf{x}, h_k)$. Hence, $D_j(\mathbf{x}, h_k) \leq D_i(\mathbf{x}, h_k)$ and $D_j(\mathbf{x}, h_k) \geq D_i(\mathbf{x}, h_k)$ are

both true, implying $D_j(\mathbf{x}, h_k) = D_i(\mathbf{x}, h_k) \geq 0$; that is, the actions i and j are both optimal at state $\mathbf{x}$. It has already been shown that (A.5.47) holds when $a_1 = b_0 = j$, so it must also hold when $a_1 = j$ and $b_0 = i$ given that $h_k(\mathbf{x}^{j+})$ and $h_k(\mathbf{x}^{i+})$ are equal.

These arguments confirm that (A.5.47) holds in all possible cases for the actions a_1 and b_0. It follows that the sum of the second, third and fifth lines on the right-hand side of (A.5.46) is non-positive. Hence, $D_{jj}(\mathbf{x}, h_{k+1}) - D_{ij}(\mathbf{x}, h_{k+1})$ may be bounded above as follows:

$$\begin{aligned} D_{jj}(\mathbf{x}, h_{k+1}) - D_{ij}(\mathbf{x}, h_{k+1}) &\leq D_{jj}(\mathbf{x}, r) - D_{ij}(\mathbf{x}, r) \\ &\quad + I(x_j = 0)\mu_j \Delta D_j(\mathbf{x}, h_k) + I(x_i = 0)\mu_i \Delta D_{jj}(\mathbf{x}, h_k). \end{aligned} \tag{A.5.49}$$

The term $I(x_i = 0)\mu_i \Delta D_{jj}(\mathbf{x}, h_k)$ is always non-positive due to the inductive assumption. If $x_j \geq 1$, then $I(x_j = 0) = 0$ and $D_{jj}(\mathbf{x}, r) - D_{ij}(\mathbf{x}, r) = 0$, so the right-hand side of (A.5.49) is non-positive as required. On the other hand, if $x_j = 0$ then $I(x_j = 0) = 1$ and $D_{jj}(\mathbf{x}, r) - D_{ij}(\mathbf{x}, r) = -\alpha_j \mu_j$. In this case, one may apply Lemma 5.2.9 to establish the following:

$$D_{jj}(\mathbf{x}, h_{k+1}) - D_{ij}(\mathbf{x}, h_{k+1}) \leq -\alpha_j \mu_j + \mu_j \Delta \left(\frac{\alpha_j \mu_j - \beta_j}{\mu_j \Delta} \right) = -\beta_j < 0. \tag{A.5.50}$$

This confirms that $D_{jj}(\mathbf{x}, h_{k+1}) - D_{ij}(\mathbf{x}, h_{k+1}) \leq 0$ for relevant states $\mathbf{x} \in \tilde{S}$. Since it was shown earlier that $D_{ij}(\mathbf{x}, h_{k+1}) \leq 0$, it follows automatically that $D_{jj}(\mathbf{x}, h_{k+1}) \leq 0$, which completes the inductive proof that all three of the properties (A.5.42)-(A.5.44) hold for integers $n \in \mathbb{N}_0$. Since $h(\mathbf{x}) = \lim_{n \to \infty} h_n(\mathbf{x})$ for all $\mathbf{x} \in \tilde{S}$, this completes the proof of the lemma. □

Proof of Lemma 5.3.1.

Consider the evolution of relative value iteration on the finite state space $\tilde{S}$, with the reward formulation (3.5.4) used. Let $\mathbf{x} \in \tilde{S}$ and $\mathbf{y} \in \tilde{S}$ be similar states; that is, $(y_1, y_2, ..., y_N)$ is simply a re-ordering of $(x_1, x_2, ..., x_N)$. Then it is sufficient to show, for all $n \in \mathbb{N}_0$:

$$h_n(\mathbf{x}) = h_n(\mathbf{y}). \tag{A.5.51}$$

The equation (A.5.51) holds trivially when $n = 0$ since $h_0(\mathbf{x}) = h_0(\mathbf{y}) = 0$, so assume that it holds

at an arbitrary stage $n = k \in \mathbb{N}_0$. Then, at stage $k+1$, one obtains:

$$\begin{aligned} h_{k+1}(\mathbf{x}) - h_{k+1}(\mathbf{y}) = r(\mathbf{x}) - r(\mathbf{y}) + \lambda\Delta\big(h_k(\mathbf{x}^{a_1+}) - h_k(\mathbf{y}^{a_2+})\big) \\ + \sum_{i=1}^{N} \min(x_i, c)\mu\Delta h_k(\mathbf{x}^{i-}) - \sum_{j=1}^{N} \min(y_j, c)\mu\Delta h_k(\mathbf{y}^{j-}) \\ + \left(1 - \lambda\Delta - \sum_{i=1}^{N} \min(x_i, c)\mu\Delta\right)\left(h_k(\mathbf{x}) - h_k(\mathbf{y})\right), \end{aligned} \tag{A.5.52}$$

where $a_1, a_2 \in \{0, 1, ..., N\}$ are optimal actions at states $\mathbf{x}$ and $\mathbf{y}$ respectively in a $(k+1)$-stage problem. From (3.5.4), it can easily be seen that $r(\mathbf{x}) = r(\mathbf{y})$ when $\mathbf{x}$ and $\mathbf{y}$ are similar. Furthermore, let $(p_1, p_2, ..., p_N)$ be a permutation of the set of integers $\{1, 2, ..., N\}$ such that $x_{p_i} \leq x_{p_{i+1}}$ for each $i \in \{1, 2, ..., N-1\}$, and similarly let $(q_1, q_2, ..., q_N)$ be a permutation of $\{1, 2, ..., N\}$ such that $y_{q_j} \leq y_{q_{j+1}}$ for $j \in \{1, 2, ..., N-1\}$; for example, given $\mathbf{x} = (5, 7, 3)$ and $\mathbf{y} = (7, 3, 5)$, one would choose $(p_1, p_2, p_3) = (3, 1, 2)$ and $(q_1, q_2, q_3) = (2, 3, 1)$. Then, clearly:

$$\begin{aligned} &\sum_{i=1}^{N} \min(x_i, c)\mu\Delta h_k(\mathbf{x}^{i-}) - \sum_{j=1}^{N} \min(y_j, c)\mu\Delta h_k(\mathbf{y}^{j-}) \\ &= \sum_{i=1}^{N} \min(x_{p_i}, c)\mu\Delta\big(h_k(\mathbf{x}^{p_i-}) - h_k(\mathbf{y}^{q_i-})\big), \end{aligned} \tag{A.5.53}$$

and the series on the right-hand of (A.5.53) is equal to zero by the inductive assumption, since $\mathbf{x}^{p_i-}$ and $\mathbf{y}^{q_i-}$ are similar states for each $i \in \{1, 2, ..., N\}$. By the inductive assumption, $h_k(\mathbf{x}) - h_k(\mathbf{y}) = 0$, so in order to show that $h_{k+1}(\mathbf{x}) - h_{k+1}(\mathbf{y}) = 0$ it only remains to show:

$$h_k(\mathbf{x}^{a_1+}) - h_k(\mathbf{y}^{a_2+}) = 0.$$

For notational convenience, let m denote the number of components of the vector $\mathbf{x}$ which are equal to the selfish threshold $\tilde{B}$ (recall that if $x_i = \tilde{B}$ for any component $i \in \{1, 2, ..., N\}$, then joining facility i is not permitted at $\mathbf{x}$; hence, $N + 1 - m$ is the number of permissible actions at $\mathbf{x}$). Then, using the permutations $(p_1, p_2, ..., p_N)$ and $(q_1, q_2, ..., q_N)$ again, one may observe:

$$\begin{aligned} h_k(\mathbf{x}^{a_1+}) &= \max\big(h_k(\mathbf{x}), h_k(\mathbf{x}^{p_1+}), h_k(\mathbf{x}^{p_2+}), ..., h_k(\mathbf{x}^{p_{N-m}+})\big), \\ h_k(\mathbf{y}^{a_2+}) &= \max\big(h_k(\mathbf{y}), h_k(\mathbf{y}^{q_1+}), h_k(\mathbf{y}^{q_2+}), ..., h_k(\mathbf{y}^{q_{N-m}+})\big), \end{aligned}$$

from which it is clear that $h_k(\mathbf{x}^{a_1+}) = h_k(\mathbf{y}^{a_2+})$ since $h_k(\mathbf{x}) = h_k(\mathbf{y})$ and $h_k(\mathbf{x}^{p_i+}) = h_k(\mathbf{y}^{q_i+})$ for each $i \in \{1, 2, ..., N\}$ due to the inductive assumption. It follows that $h_{k+1}(\mathbf{x}) - h_{k+1}(\mathbf{y}) = 0$,

which completes the inductive proof that $h_n(\mathbf{x}) = h_n(\mathbf{y})$ for all $n \in \mathbb{N}_0$. The result then follows by applying Theorem 3.7.4 which states that $h(\mathbf{x}) = \lim_{n\to\infty} h_n(\mathbf{x})$ for each $\mathbf{x} \in \tilde{S}$. $\square$

Proof of Lemma 5.3.2.

Consider an arbitrary state $\mathbf{x} \in \tilde{S}$ with $x_i \leq x_j < \tilde{B}$ for some $i, j \in \{1, 2, ..., N\}$ with $i \neq j$. Using induction again, it will be sufficient to show that for all integers $n \geq 0$:

$$h_n(\mathbf{x}^{i+}) \geq h_n(\mathbf{x}^{j+}), \tag{A.5.54}$$

where it is again assumed that the real-time reward formulation (3.5.4) is used. If $x_i = x_j$ then $\mathbf{x}^{i+}$ and $\mathbf{x}^{j+}$ are similar states, and hence (A.5.54) holds with equality for all $n \in \mathbb{N}_0$ due to Lemma 5.3.1. Therefore it remains only to consider the case where x_i is strictly smaller than x_j. The case $n = 0$ is trivial, so assume that (A.5.54) holds with $n = k \in \mathbb{N}_0$. Then:

$$\begin{aligned} h_{k+1}(\mathbf{x}^{i+}) - h_{k+1}(\mathbf{x}^{j+}) &= r(\mathbf{x}^{i+}) - r(\mathbf{x}^{j+}) + \lambda\Delta\big(h_k((\mathbf{x}^{i+})^{a_1+}) - h_k((\mathbf{x}^{j+})^{a_2+})\big) \\ &\quad + \sum_{m=1}^{N} \min(x_m, c)\mu\Delta\big(h_k((\mathbf{x}^{i+})^{m-}) - h_k((\mathbf{x}^{j+})^{m-})\big) \\ &\quad + I(x_i < c)\mu\Delta\big(h_k(\mathbf{x}) - h_k(\mathbf{x}^{j+})\big) + I(x_j < c)\mu\Delta\big(h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x})\big) \\ &\quad + \left(1 - \lambda\Delta - \sum_{m=1}^{N} \min(x_m, c)\mu\Delta - I(x_i < c)\mu\Delta - I(x_j < c)\mu\Delta\right)\big(h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x}^{j+})\big), \end{aligned} \tag{A.5.55}$$

where $a_1, a_2 \in \{0, 1, ..., N\}$ are optimal actions at states $\mathbf{x}^{i+}$ and $\mathbf{x}^{j+}$ respectively in a $(k+1)$-stage problem. Given that $x_i < x_j$, it can be seen from (3.5.4) that:

$$r(\mathbf{x}^{i+}) - r(\mathbf{x}^{j+}) = \begin{cases} \alpha\mu, & \text{if } x_i < c \text{ and } x_j \geq c, \\ 0, & \text{otherwise.} \end{cases} \tag{A.5.56}$$

Hence, $r(\mathbf{x}^{i+}) \geq r(\mathbf{x}^{j+})$. Next, consider the series on the second line of (A.5.55). For $m \neq j$, $\mathbf{x}^{m-}$ is a state with fewer customers at facility i than at facility j, in which case $h_k((\mathbf{x}^{i+})^{m-}) \geq h_k((\mathbf{x}^{j+})^{m-})$ by the inductive assumption. For $m = j$, the state $\mathbf{x}^{m-} = \mathbf{x}^{j-}$ has a j^{th} component at least equal to its i^{th} component due to the *strict* inequality $x_i < x_j$, so again the term $h_k((\mathbf{x}^{i+})^{m-}) - h_k((\mathbf{x}^{j+})^{m-})$ is non-negative due to the inductive assumption. Hence, all terms in the series $\sum_{m=1}^{N} \min(x_m, c)\mu\Delta\big(h_k((\mathbf{x}^{i+})^{m-}) - h_k((\mathbf{x}^{j+})^{m-})\big)$ are non-negative.

Next, consider the indicator terms in the third line of (A.5.55). If $x_i \geq c$, then both of these terms are equal to zero. If $x_j < c$ (implying $x_i < c$), then the third line in (A.5.55) yields:

$$\mu\Delta\big(h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x}^{j+})\big),$$

which is non-negative by the inductive assumption. The remaining possibility is that $x_i < c$ and $x_j \geq c$. In this case, the third line of (A.5.55) yields a term $\mu\Delta\big(h_k(\mathbf{x}) - h_k(\mathbf{x}^{j+})\big)$, which may be negative. However, due to the upper boundedness property of $D_j(\mathbf{x}, h_k)$ proved by Lemma 5.2.9 (which applies to any general system of N facilities with multiple servers), this term is bounded *below* by $-(\alpha\mu - \beta)$. In addition, referring to (A.5.56), one obtains a positive term $\alpha\mu$ from $r(\mathbf{x}^{i+}) - r(\mathbf{x}^{j+})$ in this case. The positive term $\alpha\mu$ is greater in absolute value than the negative lower bound $-(\alpha\mu - \beta)$, so one may conclude that in all possible cases with $x_i < x_j$:

$$\begin{aligned}
&r(\mathbf{x}^{i+}) - r(\mathbf{x}^{j+}) + \sum_{m=1}^{N} \min(x_m, c)\mu\Delta\big(h_k((\mathbf{x}^{i+})^{m-}) - h_k((\mathbf{x}^{j+})^{m-})\big) \\
&\quad + I(x_i < c)\mu\Delta\big(h_k(\mathbf{x}) - h_k(\mathbf{x}^{j+})\big) + I(x_j < c)\mu\Delta\big(h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x})\big) \\
&\quad + \left(1 - \lambda\Delta - \sum_{m=1}^{N} \min(x_m, c)\mu\Delta - I(x_i < c)\mu\Delta - I(x_j < c)\mu\Delta\right)\big(h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x}^{j+})\big) \\
&\geq 0.
\end{aligned}$$

Thus, referring to (A.5.55), it remains only to show:

$$h_k((\mathbf{x}^{i+})^{a_1+}) - h_k((\mathbf{x}^{j+})^{a_2+}) \geq 0.$$

Here, using an argument based on Lemma 5.1.1, one may simply write:

$$h_k((\mathbf{x}^{i+})^{a_1+}) - h_k((\mathbf{x}^{j+})^{a_2+}) \geq h_k((\mathbf{x}^{i+})^{a_2+}) - h_k((\mathbf{x}^{j+})^{a_2+}), \tag{A.5.57}$$

and the expression on the right-hand side of (A.5.57) is non-negative due to the inductive assumption, since the strict inequality $x_i < x_j$ ensures that $\mathbf{x}^{a_2+}$ is a state with a j^{th} component at least equal to its i^{th} component, even in the case where $a_2 = i$. From (A.5.55), it follows that $h_{k+1}(\mathbf{x}^{i+}) \geq h_{k+1}(\mathbf{x}^{j+})$, which completes the inductive proof that (A.5.54) holds for all $n \in \mathbb{N}_0$. As in previous inductive proofs, the result then follows by taking limits as $n \to \infty$. $\square$

A.6 Proofs for results in Chapter 6

Proof of Lemma 6.1.9.

In this proof it is assumed that the real-time reward formulation (3.5.4) is used to compute θ^* using relative value iteration on the finite state space $\tilde{S}$, although the result can also be established under the alternative formulation in (3.5.15) or a 'hybrid' formulation such as (3.5.26). Given that balking is chosen at some state $\mathbf{x} \in \tilde{S}$ by the *optimal* policy θ^*, one has $h(\mathbf{x}^{i+}) < h(\mathbf{x})$ for all $i \in \{1, 2, ..., N\}$ with $\mathbf{x}^{i+} \in \tilde{S}$, where h is the relative value function satisfying (3.7.6). In order to show that $\theta^{[W]}$ chooses to balk at the same state, it is then necessary to show that $W_i(x_i) < 0$ for all i. By definition of the Whittle indices $W_i(x_i)$, this is equivalent to showing that for each facility i, the optimal threshold T_i^* found by applying relative value iteration to the corresponding single-facility problem with state space $\{0, 1, ..., \tilde{T}_i\}$ (where $\tilde{T}_i = \lfloor \alpha_i c_i \mu_i / \beta_i \rfloor$) is less than or equal to x_i; or equivalently, balking would be chosen at state $x_i \in \mathbb{N}_0$ by the policy θ_i^*.

For ease of notation, let the facility $i \in \{1, 2, ..., N\}$ be fixed, let $R(\cdot)$ be the real-time reward function in a single-facility problem with parameters corresponding to those of i, and let $H(\cdot)$ be the associated relative value function defined on the set $\{0, 1, ..., \tilde{T}_i\}$. That is:

$$R(x) = \min(x, c_i)\alpha_i \mu_i - \beta_i x \qquad \forall\, x \in \{0, 1, ..., \tilde{T}_i\}, \tag{A.6.1}$$

and the values $H(x)$, together with a constant G^*, satisfy the optimality equations:

$$G^* + H(x) = \max_{a \in A_x} \left\{ R(x) + \sum_{y \leq \tilde{T}_i} P(x, a, y) H(y) \right\} \qquad (x \in \{0, 1, ..., \tilde{T}_i\}). \tag{A.6.2}$$

Further, suppose that the transition probabilities $P(x, a, y)$ in (A.6.2) are obtained by using a discrete-time step size $\Delta = (\lambda + \sum_j c_j \mu_j)^{-1}$ as opposed to the larger value $(\lambda + c_i \mu_i)^{-1}$; that is, the maximal value of the discrete-time step size used for the N-facility problem is also used as the step size for the single-facility problem. Recall from the discussion in Section 3.8 (page 119) that reducing the value of Δ in this way affects the relative values $H(x)$ only by a positive multiplicative constant, so this cannot alter the sign of a quantity such as $H(x + 1) - H(x)$. The transition

probabilities $P(x, a, y)$ for the single-facility problem are then given by:

$$P(x,a,y) = \begin{cases} \lambda\Delta, & \text{if } y = x+1 \text{ and } a = 1, \\ \min(x, c_i)\mu_i\Delta, & \text{if } y = x-1, \\ 1 - I(a=1)\lambda\Delta - \min(x,c_i)\mu_i\Delta, & \text{if } y = x, \\ 0, & \text{otherwise,} \end{cases}$$

with $\Delta = (\lambda + \sum_j c_j\mu_j)^{-1}$. Meanwhile, for the N-facility problem, let $r(\mathbf{x})$, $h(\mathbf{x})$ and $p(\mathbf{x}, a, \mathbf{y})$ be as defined in (3.5.4), (3.7.6) and (3.5.3) respectively, again assuming that $\Delta = (\lambda + \sum_j c_j\mu_j)^{-1}$. By the previous arguments, it is sufficient to show that for all $\mathbf{x} \in \tilde{S}$ with $x_i < \tilde{T}_i$:

$$h(\mathbf{x}^{i+}) < h(\mathbf{x}) \;\; \Rightarrow \;\; H(x_i + 1) < H(x_i).$$

In fact, it will be possible to establish a slightly stronger property. Let $h_n(\cdot)$ be the finite-stage relative value function defined in (3.7.7), and let $H_n(\cdot)$ be the analogous function for the single-facility problem involving facility i. The aim of this proof will be to show that, given any $\mathbf{x} \in \tilde{S}$ with $x_i < \tilde{T}_i$, the following property holds for all integers y satisfying $x_i \leq y < \tilde{T}_i$:

$$h_n(\mathbf{x}^{i+}) - h_n(\mathbf{x}) \geq H_n(y+1) - H_n(y) \qquad \forall\, n \in \mathbb{N}_0. \tag{A.6.3}$$

One assumes that $h_0(\mathbf{x}) = 0$ for all $\mathbf{x} \in \tilde{S}$ and $H_0(x) = 0$ for all $x \in \{0, 1, ..., \tilde{T}_i\}$, so (A.6.3) holds trivially when $n = 0$. As an inductive hypothesis, suppose that it also holds at an arbitrary stage

$k \in \mathbb{N}_0$. Then, by definition of Δ, one can write the following at stage $k+1$:

$$
\begin{aligned}
& h_{k+1}(\mathbf{x}^{i+}) - h_{k+1}(\mathbf{x}) - H_{k+1}(y+1) + H_{k+1}(y) \\
& = r(\mathbf{x}^{i+}) - r(\mathbf{x}) - R(y+1) + R(y) && \text{(A.6.4)} \\
& \quad + \lambda\Delta\left(h_k((\mathbf{x}^{i+})^{a_1+}) - h_k(\mathbf{x}^{a_0+}) - H_k(y+1+b_1) + H_k(y+b_0)\right) && \text{(A.6.5)} \\
& \quad + \sum_{j\neq i} \min(x_j, c_j)\mu_j\Delta\left(h_k((\mathbf{x}^{i+})^{j-}) - h_k(\mathbf{x}^{j-}) - H_k(y+1) + H_k(y)\right) && \text{(A.6.6)} \\
& \quad + \min(x_i, c_i)\mu_i\Delta\left(h_k(\mathbf{x}) - h_k(\mathbf{x}^{i-}) - H_k(y) + H_k(y-1)\right) && \text{(A.6.7)} \\
& \quad + I(x_i < c_i \text{ and } x_i < y)\mu_i\Delta\left(h_k(\mathbf{x}) - h_k(\mathbf{x}) - H_k(y) + H_k(y-1)\right) && \text{(A.6.8)} \\
& \quad + I(x_i = y < c_i)\mu_i\Delta\left(h_k(\mathbf{x}) - h_k(\mathbf{x}) - H_k(y) + H_k(y)\right) && \text{(A.6.9)} \\
& \quad + I(x_i < y-1 \text{ and } x_i < c_i - 1)(\min(y, c_i) - (x_i+1))\mu_i\Delta \\
& \qquad \times \left(h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x}) - H_k(y) + H_k(y-1)\right) && \text{(A.6.10)} \\
& \quad + I(x_i < y < c_i)\mu_i\Delta\left(h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x}) - H_k(y) + H_k(y)\right) && \text{(A.6.11)} \\
& \quad + \left(\max(c_i - (y+1), 0)\mu_i\Delta + \sum_{j\neq i}\max(c_j - x_j, 0)\mu_j\Delta\right) \\
& \qquad \times \left(h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x}) - H_k(y+1) + H_k(y)\right), && \text{(A.6.12)}
\end{aligned}
$$

where $a_0, a_1 \in \{0, 1, ..., N\}$ are optimal actions at states $\mathbf{x}$ and $\mathbf{x}^{i+}$ in a $(k+1)$-stage problem with N facilities, and similarly $b_0, b_1 \in \{0, 1\}$ are optimal actions at states y and $y+1$ in the single-facility problem with $k+1$ stages. In order to make sense of the terms in lines (A.6.5)-(A.6.12) it is helpful to use the analogy of a sample path argument as discussed in Section 3.8. In doing so, one imagines four separate processes Υ_1, Υ_2, Υ_3 and Υ_4 following the same sequence of random events; Υ_1 and Υ_2 are initialised at states $\mathbf{x}^{i+}$ and $\mathbf{x}$ respectively in a system with N facilities, while Υ_3 and Υ_4 are initialised in states $y+1$ and y respectively in a single-facility system. Line (A.6.5) obviously represents all four processes receiving an arrival, while line (A.6.6) represents service completions at facilities $j \neq i$ seen by Υ_1 and Υ_2, but not seen by Υ_3 or Υ_4 (because the facilities in question do not exist in the single-facility system). Line (A.6.7) represents all four processes seeing a service completion at facility i (which, in the case of Υ_3 and Υ_4, is the only facility). Line (A.6.8) represents a service completion seen by all processes except Υ_2, which is possible if $x_i < c_i$ and $x_i < y$ (in which case Υ_2 has the smallest number of services in progress at i). Line (A.6.9) represents a service completion at i seen by Υ_1 and Υ_3, but not by Υ_2 or Υ_4 (which is possible

only if $x_i = y < c_i$, so that Υ_1 and Υ_3 jointly have the greatest number of services in progress at i). Line (A.6.10) represents service completions at i seen by Υ_3 and Υ_4 but not by Υ_1 or Υ_2, which requires $x_i + 1$ to be smaller than both y and c_i. Line (A.6.11) represents a service completion seen by Υ_3 only, which requires $x_i < y < c_i$ (so that Υ_3 has the greatest number of services in progress at i). Line (A.6.12) represents service completions seen by none of the four processes.

The objective is to show that the sum of the terms in lines (A.6.4)-(A.6.12) is non-negative. Using (3.5.4) and (A.6.1), the reward terms in line (A.6.4) reduce to:

$$r(\mathbf{x}^{i+}) - r(\mathbf{x}) - R(y+1) + R(y) = \begin{cases} \alpha_i \mu_i, & \text{if } x_i < c_i \leq y, \\ 0, & \text{otherwise.} \end{cases} \tag{A.6.13}$$

Hence, line (A.6.4) is non-negative. Since $\mathbf{x}$ is a state whose i^{th} component is bounded above by y, the same also applies to the state $\mathbf{x}^{j-}$ (where $j \in \{1, 2, ..., N\} \setminus \{i\}$), and hence line (A.6.6) is non-negative due to the inductive assumption. Using similar reasoning, $\mathbf{x}^{i-}$ is a state whose i^{th} component is bounded above by $y - 1$ (due to the inequality $x_i \leq y$) and therefore line (A.6.7) is non-negative by the inductive assumption. Line (A.6.9) is trivially equal to zero. The indicator term $I(x_i < y - 1 \text{ and } x_i < c_i - 1)$ can only be non-zero only if $x_i < y$, in which case the inequality $x_i \leq y - 1$ holds, implying that line (A.6.10) is also non-negative by the inductive assumption. Line (A.6.12) is obviously also non-negative, again due to the inductive assumption.

Summarising the arguments given thus far, in order to complete the inductive proof that (A.6.3) holds at stage $n = k + 1$ it remains only to show that lines (A.6.5), (A.6.8) and (A.6.11) cannot cause the sum of the terms in lines (A.6.4)-(A.6.12) to be negative. Consider line (A.6.11) first. The indicator term $I(x_i < y < c_i)$ can only be non-zero if the indicator $I(x_i < c_i \text{ and } x_i < y)$ in line (A.6.8) is also non-zero, in which case the sum of lines (A.6.11) and (A.6.8) is equal to $\mu_i \Delta \left(h_k(\mathbf{x}^{i+}) - h_k(\mathbf{x}) - H_k(y) + H_k(y-1) \right)$, which is again non-negative by the inductive assumption due to the fact that (in this particular case) the inequality $x_i \leq y - 1$ holds.

Next, consider line (A.6.8) and assume $x_i < c_i$ and $x_i < y$ to avoid triviality. If $y < c_i$ then, as explained in the paragraph above, the sum of lines (A.6.8) and (A.6.11) is non-negative. On the other hand, suppose $y \geq c_i$. In this case (A.6.13) implies that the reward terms in line (A.6.4) yield

a positive term $\alpha_i\mu_i$. Hence, summing lines (A.6.8) and (A.6.4) gives:

$$\alpha_i\mu_i + \mu_i\Delta\left(H_k(y-1) - H_k(y)\right). \tag{A.6.14}$$

However, the upper boundedness property proved in Lemma 5.2.9 (which applies to any system of N facilities) implies that the difference $H_k(y) - H_k(y-1)$ satisfies:

$$H_k(y) - H_k(y-1) \leq \frac{\alpha_i\mu_i - \beta_i}{\mu_i\Delta},$$

which immediately implies that the expression in (A.6.14) is bounded *below* by the positive quantity β_i. These arguments show that the sum of the terms in lines (A.6.4), (A.6.8) and (A.6.11) is always non-negative. In order to complete the proof, it remains only to show that the 'arrival terms' in line (A.6.5) sum to a non-negative value. Using the same approach as in previous proofs, this may be done by considering all possibilities for the actions a_0 and b_1. Firstly, note that one cannot have $a_0 = 0$ and $b_1 = 1$, since this would imply $H_k(y+2) \geq H_k(y+1) \geq H_k(y)$ (where the second inequality is due to the concavity property proved in Lemma 5.1.2) and $h_k(\mathbf{x}^{i+}) < h_k(\mathbf{x})$, thereby contradicting the inductive assumption. The remaining possibilities for the actions $a_0 \in \{0, 1, ..., N\}$ and $b_1 \in \{0, 1\}$ are considered one-by-one below. Note that the arguments given take advantage of the AS principle (see Lemma 5.1.1), which essentially enables 'convenient' actions to be chosen for a_1 and b_0, so that it is not necessary to consider all possibilities for a_1 and b_0.

- If $a_0 = b_1 = 0$, one may use an argument based on Lemma 5.1.1 and choose $a_1 = b_0 = 0$, in which case line (A.6.5) is non-negative by the inductive assumption.

- Similarly, if $a_0 = i$ and $b_1 = 1$ then one may choose $a_1 = i$ and $b_0 = 1$, in which case line (A.6.5) is non-negative by the inductive assumption since $x_i + 1 \leq y + 1$.

- If $a_0 = i$ and $b_1 = 0$ then one may apply Lemma 5.1.1 again and choose $a_1 = 0$ and $b_0 = 1$, in which case line (A.6.5) is trivially equal to zero.

- If $a_0 = j$ for some $j \in \{1, 2, ..., N\} \setminus \{i\}$ and $b_1 = 0$ then one may choose $a_1 = j$ and $b_0 = 0$, in which case line (A.6.5) is again non-negative by the inductive assumption again since $\mathbf{x}^{j+}$ is a state whose i^{th} component is bounded above by y.

- Finally, if $a_0 = j$ for some $j \in \{1, 2, ..., N\} \setminus \{i\}$ and $b_1 = 1$ then one may choose $a_1 = j$ and $b_0 = 1$, with the result that line (A.6.5) is non-negative by the inductive assumption since $\mathbf{x}^{j+}$ is a state whose i^{th} component is bounded above by $y + 1$.

Thus, it has been verified that line (A.6.5) is non-negative in all possible cases. In view of the previous arguments, this establishes that the sum of the terms in lines (A.6.4)-(A.6.12) must be non-negative, which completes the inductive proof that (A.6.3) holds for all $n \in \mathbb{N}_0$. By the limiting properties of $h_n(\cdot)$ and $H_n(\cdot)$ as $n \to \infty$, this implies that $h(\mathbf{x}^{i+}) - h(\mathbf{x}) \geq H(y+1) - H(y)$ for all $\mathbf{x} \in \tilde{S}$ and $y \in \mathbb{N}_0$ with $x_i \leq y < \tilde{T}_i$, where $i \in \{1, 2, ..., N\}$ is arbitrary. By the arguments given earlier in the proof, this confirms that $\theta^*(\mathbf{x}) = 0 \Rightarrow \theta^{[W]}(\mathbf{x}) = 0$ as required. $\square$

Proof of uniqueness of optimal static policies (Theorem 6.2.1)

Let $(\lambda_1, \lambda_2, ..., \lambda_N)$ denote a static routing policy satisfying $\lambda_i \in [0, \lambda]$ for each $i \in \{1, 2, ..., N\}$ and $\sum_{i=1}^{N} \lambda_i \leq \lambda$, where $\lambda > 0$ is the system demand rate. As discussed in the proof of Theorem 6.2.1, the long-run average reward $g_i(\lambda_i)$ at any facility $i \in \{1, 2, ..., N\}$ is a strictly concave function of the Poisson demand rate λ_i under the policy $(\lambda_1, \lambda_2, ..., \lambda_N)$. As such, for each facility i there exists a unique demand rate $\lambda_i = y_i$ (depicted in Figure 6.3) which maximises $g_i(\lambda_i)$. If $\sum_{i=1}^{N} y_i \leq \lambda$, then the static policy with $\lambda_i = y_i$ for each $i \in \{1, 2, ..., N\}$ is obviously the unique optimal static policy. Therefore, in order to avoid triviality, one may assume that $\sum_{i=1}^{N} y_i > \lambda$.

Suppose, for a contradiction, that there exist two different optimal static policies $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ and $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$. Under the assumption that $\sum_{i=1}^{N} y_i > \lambda$, it must be the case that $\lambda_i^* < y_i$ for at least one facility i and similarly $\bar{\lambda}_j^* < y_j$ for at least one facility j. Hence, given that $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ and $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$ are both optimal static policies, it is possible to show that:

$$\sum_{i=1}^{N} \lambda_i^* = \sum_{i=1}^{N} \bar{\lambda}_i^* = \lambda. \tag{A.6.15}$$

Indeed, given that $\lambda_i^* < y_i$ for some facility $i \in \{1, 2, ..., N\}$, the function $g_i(\lambda_i)$ must be strictly increasing in the interval $[\lambda_i^*, y_i]$ due to concavity. Therefore one cannot have $\sum_{j=1}^{N} \lambda_j^* < \lambda$, otherwise one would be able to increase the proportion of traffic sent to facility i in such a way as to improve the overall average reward $g(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*) = \sum_{j=1}^{N} g_j(\lambda_j^*)$, thereby implying sub-optimality (among static policies) of the policy $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$. Applying the same argument to $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$ confirms that (A.6.15) must hold. Of course, the physical interpretation of (A.6.15) is that customers never balk under either of the static policies $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ and $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$.

Due to (A.6.15) and the assumption that $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ and $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$ are not identical, one

may define two non-empty subsets of the set of facilities $\{1, 2, ..., N\}$ as follows:

$$M := \left\{ i \in \{1, 2, ..., N\} : \lambda_i^* > \bar{\lambda}_i^* \right\},$$
$$R := \left\{ j \in \{1, 2, ..., N\} : \lambda_j^* < \bar{\lambda}_j^* \right\}.$$

Also, let ϵ_i (for each $i \in M$) and δ_j (for each $j \in R$) be defined as follows:

$$\epsilon_i := \lambda_i^* - \bar{\lambda}_i^* > 0 \quad (i \in M),$$
$$\delta_j := \bar{\lambda}_j^* - \lambda_j^* > 0 \quad (j \in R). \tag{A.6.16}$$

Of course, due to (A.6.15) again, $\sum_{i \in M} \epsilon_i = \sum_{j \in R} \delta_j$. Also, since $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ and $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$ achieve the same overall performance and $\lambda_i^* = \bar{\lambda}_i^*$ for all $i \notin M \cup R$:

$$\sum_{i \in M \cup R} g_i(\lambda_i^*) = \sum_{i \in M \cup R} g_i(\bar{\lambda}_i^*).$$

Hence, noting that $\bar{\lambda}_i^* = \lambda_i^* - \epsilon_i$ for each $i \in M$ and $\bar{\lambda}_j^* = \lambda_j^* + \delta_j$ for each $j \in R$:

$$\sum_{i \in M} g_i(\lambda_i^*) + \sum_{j \in R} g_j(\lambda_j^*) = \sum_{i \in M} g_i(\lambda_i^* - \epsilon_i) + \sum_{j \in R} g_j(\lambda_j^* + \delta_j). \tag{A.6.17}$$

It will be useful to write (A.6.17) equivalently as follows:

$$\sum_{i \in M} \Big(g_i(\lambda_i^*) - g_i(\lambda_i^* - \epsilon_i/2) + g_i(\lambda_i^* - \epsilon_i/2) - g_i(\lambda_i^* - \epsilon_i) \Big)$$
$$= \sum_{j \in R} \Big(g_j(\lambda_j^* + \delta_j/2) - g_j(\lambda_j^*) + g_j(\lambda_j^* + \delta_j) - g_j(\lambda_j^* + \delta_j/2) \Big). \tag{A.6.18}$$

Due to the strict concavity of the functions $g_i(\cdot)$, the following inequalities hold:

$$\sum_{i \in M} \Big(g_i(\lambda_i^*) - g_i(\lambda_i^* - \epsilon_i/2) \Big) < \sum_{i \in M} \Big(g_i(\lambda_i^* - \epsilon_i/2) - g_i(\lambda_i^* - \epsilon_i) \Big), \tag{A.6.19}$$
$$\sum_{j \in R} \Big(g_j(\lambda_j^* + \delta_j) - g_j(\lambda_j^* + \delta_j/2) \Big) < \sum_{j \in R} \Big(g_j(\lambda_j^* + \delta_j/2) - g_j(\lambda_j^*) \Big). \tag{A.6.20}$$

Then using (A.6.18), (A.6.19) and (A.6.20), one may write:

$$\sum_{j \in R} \Big(g_j(\lambda_j^* + \delta_j/2) - g_j(\lambda_j^*) \Big) > \sum_{j \in R} \Big(g_j(\lambda_j^* + \delta_j) - g_j(\lambda_j^* + \delta_j/2) \Big)$$
$$= \sum_{i \in M} \Big(g_i(\lambda_i^*) - g_i(\lambda_i^* - \epsilon_i/2) + g_i(\lambda_i^* - \epsilon_i/2) - g_i(\lambda_i^* - \epsilon_i) \Big) - \sum_{j \in R} \Big(g_j(\lambda_j^* + \delta_j/2) - g_j(\lambda_j^*) \Big)$$
$$> 2 \sum_{i \in M} \Big(g_i(\lambda_i^*) - g_i(\lambda_i^* - \epsilon_i/2) \Big) - \sum_{j \in R} \Big(g_j(\lambda_j^* + \delta_j/2) - g_j(\lambda_j^*) \Big). \tag{A.6.21}$$

Note that all of the differences $g_i(\lambda_i^*) - g_i(\lambda_i^* - \epsilon_i/2)$ and $g_i(\lambda_i^* - \epsilon_i/2) - g_i(\lambda_i^* - \epsilon_i)$ (for $i \in M$) must be strictly positive, since it must be the case that $\lambda_i^* \leq y_i$ for each $i \in M$ due to the optimality of $(\lambda_1^*, \lambda_2^*, ...\lambda_N^*)$, and hence the function $g_i(\cdot)$ is strictly increasing over the interval $[\lambda_i^* - \epsilon_i, \lambda_i^*]$. For the same reason, the differences $g_j(\lambda_j^* + \delta_j) - g_j(\lambda_j^* + \delta_j/2)$ and $g_j(\lambda_j^* + \delta_j/2) - g_j(\lambda_j^*)$ (for $j \in R$) are strictly positive, since $\bar{\lambda}_j^* = \lambda_j^* + \delta_j \leq y_j$ for $j \in R$. Hence, (A.6.21) implies:

$$\sum_{i \in M} \Big(g_i(\lambda_i^*) - g_i(\lambda_i^* - \epsilon_i/2) \Big) < \sum_{j \in R} \Big(g_j(\lambda_j^* + \delta_j/2) - g_j(\lambda_j^*) \Big). \tag{A.6.22}$$

However, this contradicts the optimality of $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$, since it implies:

$$\sum_{i \in M} g_i(\lambda_i^* - \epsilon_i/2) + \sum_{j \in R} g_j(\lambda_j^* + \delta_j/2) > \sum_{i \in M} g_i(\lambda_i^*) + \sum_{j \in R} g_j(\lambda_j^*). \tag{A.6.23}$$

The static policy $(\lambda_1^\dagger, \lambda_2^\dagger, ..., \lambda_N^\dagger)$ with $\lambda_i^\dagger = \lambda_i^* - \epsilon_i/2$ for each $i \in M$, $\lambda_j^\dagger = \lambda_j^* + \delta_j/2$ for each $j \in R$ and $\lambda_k^\dagger = \lambda_k^*$ for each $k \notin M \cup R$ is a feasible static policy under demand rate λ, since $\sum_{i \in M} \epsilon_i/2 = \sum_{j \in R} \delta_j/2$ by definition of ϵ_i and δ_j in (A.6.16), so the constraint $\sum_{i=1}^N \lambda_i^\dagger = \lambda$ holds. However, (A.6.23) implies that $(\lambda_1^\dagger, \lambda_2^\dagger, ..., \lambda_N^\dagger)$ yields a strictly greater average reward than $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$, which contradicts the assumption that $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ is an optimal static policy. In conclusion, it is not possible for $(\lambda_1^*, \lambda_2^*, ..., \lambda_N^*)$ and $(\bar{\lambda}_1^*, \bar{\lambda}_2^*, ..., \bar{\lambda}_N^*)$ to differ from each other; that is, the optimal static policy (known to exist by Theorem 6.2.1) is unique. □

Proof of Theorem 6.2.6.

Since the inductive arguments required in this proof are almost trivial, it will be desirable to keep them as brief as possible. Let $i \in \{1, 2, ..., N\}$ be an arbitrary facility and let $h_i^{(n)}(\cdot)$ be the n-stage relative value function for facility i, defined for $x \in \mathbb{N}_0$ and $n \in \mathbb{N}_0$ as follows:

$$\begin{aligned} h_i^{(n+1)}(x, \lambda_i^*) &= V_i^{(n+1)}(x, \lambda_i^*) - V_i^{(n+1)}(0, \lambda_i^*) \\ &= \hat{r}_i(x, \lambda_i^*) + \sum_{y \in \mathbb{N}_0} p_i(x, y, \lambda_i^*) V_i^{(n)}(y, \lambda_i^*) - \hat{r}_i(0, \lambda_i^*) - \sum_{y \in \mathbb{N}_0} p_i(0, y, \lambda_i^*) V_i^{(n)}(y, \lambda_i^*) \\ &= \hat{r}_i(x, \lambda_i^*) + \sum_{y \in \mathbb{N}_0} p_i(x, y, \lambda_i^*) h_i^{(n)}(y, \lambda_i^*) - \hat{r}_i(0, \lambda_i^*) - \sum_{y \in \mathbb{N}_0} p_i(0, y, \lambda_i^*) h_i^{(n)}(y, \lambda_i^*), \end{aligned}$$

where $V_i^{(n)}(x, \lambda_i^*)$ is the expected total reward earned at facility i over n stages under the optimal static policy, $\hat{r}_i(\cdot)$ and $p_i(\cdot)$ are as defined in (6.2.13) and (6.2.19) respectively, and $h_i^{(0)}(x, \lambda_i^*) = 0$

for all $x \in \mathbb{N}_0$. Similarly, let $D_i^{(n)}(x, \lambda_i^*)$ be defined for $x \in \mathbb{N}_0$ and $n \in \mathbb{N}_0$ as:

$$D_i^{(n)}(x, \lambda_i^*) = h_i^{(n)}(x+1, \lambda_i^*) - h_i^{(n)}(x, \lambda_i^*).$$

Thus, the definition of $h_i(x, \lambda_i^*)$ in (6.2.17) implies that for all $x \in \mathbb{N}_0$:

$$h_i(x, \lambda_i^*) = \lim_{n\to\infty} \Big(V_i^{(n)}(x, \lambda_i^*) - V_i^{(n)}(0, \lambda_i^*) \Big) = \lim_{n\to\infty} h_i^{(n)}(x, \lambda_i^*), \tag{A.6.24}$$

and it follows that structural properties involving the values $h_i(x, \lambda_i^*)$ can be proved by induction on the finite-stage values $h_i^{(n)}(x, \lambda_i^*)$. First, it will be useful to show that $D_i(x, \lambda_i^*)$ is non-positive for all $x \in \mathbb{N}_0$. At stage $n = 0$, one trivially has $D_i^{(0)}(x, \lambda_i^*) = 0$ for all states x. Let it be assumed that $D_i^{(k)}(x, \lambda_i^*) \leq 0$ for all x, where $k \in \mathbb{N}_0$ is arbitrary. Then, at stage $k+1$:

$$\begin{aligned} D_i^{(k+1)}(x, \lambda_i^*) = {} & \hat{r}_i(x+1, \lambda_i^*) - \hat{r}_i(x, \lambda_i^*) + \lambda_i^* \Delta D_i^{(k)}(x+1, \lambda_i^*) \\ & + \min(x, c_i)\mu_i \Delta D_i^{(k)}(x-1, \lambda_i^*) \\ & + \big(1 - \lambda_i^* \Delta - \min(x, c_i)\mu_i \Delta - I(x < c_i)\mu_i \Delta\big) D_i^{(k)}(x, \lambda_i^*), \end{aligned}$$

and this expression is non-positive since $D_i^{(k)}(x+1, \lambda_i^*)$, $D_i^{(k)}(x-1, \lambda_i^*)$ and $D_i^{(k)}(x, \lambda_i^*)$ are non-positive by the inductive assumption and $\hat{r}_i(x+1, \lambda_i^*) \leq \hat{r}_i(x, \lambda_i^*)$ due to (6.2.13). It follows from (A.6.24) that $D_i(x, \lambda_i^*) \leq 0$ for all $x \in \mathbb{N}_0$. In order to prove property (1) in the statement of the theorem (the containment property), it is sufficient to show that for any state $\mathbf{x} \in S$ with $x_i \geq \tilde{B}_i$ (with $\tilde{B}_i$ as defined in (4.1.2)), one must have $P_i(x_i, \lambda_i^*) < 0$. Equivalently:

$$w_i(x_i) + \Delta D_i(x_i \lambda_i^*) < 0.$$

Indeed, if $x_i \geq \tilde{B}_i$ then $w_i(x_i) < 0$ by definition of $\tilde{B}_i$, and since $D_i(x_i, \lambda_i^*) \leq 0$ the result follows. The three monotonicity properties, numbered (2)-(4) in the statement of the theorem, can be established by showing that the indices $P_i(x, \lambda_i^*)$ are monotonically decreasing in x. Equivalently, using the induction approach again, it is sufficient to show that for $x, n \in \mathbb{N}_0$:

$$w_i(x+1) + \Delta D_i^{(n)}(x+1, \lambda_i^*) \leq w_i(x) + \Delta D_i^{(n)}(x, \lambda_i^*).$$

It will be more convenient to write this in the form:

$$D_i^{(n)}(x+1, \lambda_i^*) - D_i^{(n)}(x, \lambda_i^*) \leq \frac{w_i(x) - w_i(x+1)}{\Delta}. \tag{A.6.25}$$

Since the right-hand side of (A.6.25) is non-negative, the inequality holds trivially when $n = 0$. Assume that it holds at an arbitrary stage $n = k \in \mathbb{N}_0$. Then, at stage $k+1$:

$$\begin{aligned}
&D_i^{(k+1)}(x+1,\lambda_i^*) - D_i^{(k+1)}(x,\lambda_i^*) \\
&= \hat{r}_i(x+1,\lambda_i^*) - \hat{r}_i(x,\lambda_i^*) + \lambda_i^*\Delta\big(D_i^{(k)}(x+2,\lambda_i^*) - D_i^{(k)}(x+1,\lambda_i^*)\big) \\
&\quad + \min(x,c_i)\mu_i\Delta\big(D_i^{(k)}(x,\lambda_i^*) - D_i^{(k)}(x-1,\lambda_i^*)\big) \\
&\quad + I(x<c_i)\mu_i\Delta D_i^{(k)}(x,\lambda_i^*) - I(x<c_i-1)\mu_i\Delta D_i^{(k)}(x,\lambda_i^*) \\
&\quad + \Big(1 - \lambda_i^*\Delta - \min(x,c_i)\mu_i\Delta - I(x<c_i)\mu_i\Delta - I(x<c_i-1)\mu_i\Delta\Big)\big(D_i^{(k)}(x+1,\lambda_i^*) - D_i^{(k)}(x,\lambda_i^*)\big).
\end{aligned} \tag{A.6.26}$$

There are three possible cases to check in (A.6.26): $x < c_i - 1$, $x = c_i - 1$ and $x \geq c_i$. In each case, it can easily be verified that the sum of the terms on the right-hand side of (A.6.26) is bounded above by $\big(w_i(x) - w_i(x+1)\big)/\Delta$ by using the inductive assumption that (A.6.25) holds for all $x \in \mathbb{N}_0$ when $n = k$, and also the fact that $\hat{r}_i(x+1,\lambda_i^*) - \hat{r}_i(x,\lambda_i^*) \leq 0$. In the case $x = c_i - 1$, one must also use the fact that $D_i^{(k)}(x,\lambda_i^*) \leq 0$ as shown in the earlier part of the proof. These arguments are sufficient to establish that the Bernoulli indices $P_i(x,\lambda_i^*)$ are monotonically decreasing in x, from which the three monotonicity properties (2)-(4) follow using trivial arguments.

Property (5) (the sink property) is directly implied by the fact that $P_i(x,\lambda_i^*)$ is decreasing with x. Indeed, one may refer to the proof of the corresponding property for the Whittle policy $\theta^{[W]}$ (property (5) in Theorem 6.1.7) and use identical arguments, merely replacing $W_i(x)$, $\theta^{[W]}$ and S_W with $P_i(x,\lambda_i^*)$, $\theta^{[B]}$ and S_B respectively in order to show that $\theta^{[B]}$ is a sink policy.

Finally, in order to prove property (6), it is necessary to invoke the result of Theorem 6.2.2, which states that the demand rate λ_i^* associated with the optimal static policy is monotonically increasing with the system demand rate λ. Let $\lambda, \bar{\lambda} > 0$ be two system demand rates with $\lambda < \bar{\lambda}$, and let λ_i^* and $\bar{\lambda}_i^*$ be the corresponding demand rates for facility i under the two respective optimal static policies. Then, by Theorem 6.2.2, $\lambda_i^* \leq \bar{\lambda}_i^*$. One may also apply the same uniformisation 'trick' used in previous proofs, and define $\Delta = (\bar{\lambda} + \sum_{j=1}^N c_j\mu_j)^{-1}$ regardless of whether the system is operating under demand rate λ or $\bar{\lambda}$. Then, since $w_i(x)$ is independent of the demand rate and Δ is held at a constant value, it is sufficient to show that for all $x \in \mathbb{N}_0$ and $n \in \mathbb{N}_0$:

$$D_i^{(n)}(x,\bar{\lambda}_i^*) \leq D_i^{(n)}(x,\lambda_i^*). \tag{A.6.27}$$

The property again holds trivially when $n = 0$, so assume that it holds at an arbitrary stage $n = k \in \mathbb{N}_0$. Then the definition of Δ implies that at stage $k+1$:

$$\begin{aligned} D_i^{(k+1)}(x, \bar{\lambda}_i^*) - D_i^{(k+1)}(x, \lambda_i^*) = \hat{r}_i(x+1, \bar{\lambda}_i^*) - \hat{r}_i(x, \bar{\lambda}_i^*) - \hat{r}_i(x+1, \lambda_i^*) + \hat{r}_i(x, \lambda_i^*) \\ + \lambda_i^* \Delta\big(D_i^{(k)}(x+1, \bar{\lambda}_i^*) - D_i^{(k)}(x+1, \lambda_i^*)\big) \\ + (\bar{\lambda}_i^* - \lambda_i^*)\Delta\big(D_i^{(k)}(x+1, \bar{\lambda}_i^*) - D_i^{(k)}(x, \lambda_i^*)\big) \\ + \min(x, c_i)\mu_i \Delta\big(D_i^{(k)}(x-1, \bar{\lambda}_i^*) - D_i^{(k)}(x-1, \lambda_i^*)\big) \\ + \Big(1 - \bar{\lambda}_i^* \Delta - \min(x, c_i)\mu_i \Delta\Big)\big(D_i^{(k)}(x, \bar{\lambda}_i^*) - D_i^{(k)}(x, \lambda_i^*)\big). \end{aligned} \tag{A.6.28}$$

Then, using properties established earlier in the proof and the inductive assumption that (A.6.27) holds for $x \in \mathbb{N}_0$ at stage $n = k$, one may infer the following from (A.6.28):

$$\begin{aligned} & D_i^{(k+1)}(x, \bar{\lambda}_i^*) - D_i^{(k+1)}(x, \lambda_i^*) \\ & \leq \hat{r}_i(x+1, \bar{\lambda}_i^*) - \hat{r}_i(x, \bar{\lambda}_i^*) - \hat{r}_i(x+1, \lambda_i^*) + \hat{r}_i(x, \lambda_i^*) + (\bar{\lambda}_i^* - \lambda_i^*)\Delta\big(D_i^{(k)}(x+1, \bar{\lambda}_i^*) - D_i^{(k)}(x, \lambda_i^*)\big) \\ & \leq \hat{r}_i(x+1, \bar{\lambda}_i^*) - \hat{r}_i(x, \bar{\lambda}_i^*) - \hat{r}_i(x+1, \lambda_i^*) + \hat{r}_i(x, \lambda_i^*) \\ & \quad + (\bar{\lambda}_i^* - \lambda_i^*)\Delta\left(D_i^{(k)}(x, \bar{\lambda}_i^*) - D_i^{(k)}(x, \lambda_i^*) + \frac{w_i(x) - w_i(x+1)}{\Delta}\right) \\ & \leq \hat{r}_i(x+1, \bar{\lambda}_i^*) - \hat{r}_i(x, \bar{\lambda}_i^*) - \hat{r}_i(x+1, \lambda_i^*) + \hat{r}_i(x, \lambda_i^*) + (\bar{\lambda}_i^* - \lambda_i^*)\big(w_i(x) - w_i(x+1)\big) \\ & = (\bar{\lambda}_i^* - \lambda_i^*)\big(w_i(x+1) - w_i(x)\big) + (\bar{\lambda}_i^* - \lambda_i^*)\big(w_i(x) - w_i(x+1)\big) \\ & = 0. \end{aligned} \tag{A.6.29}$$

This completes the inductive proof that (A.6.27) holds for all $n \in \mathbb{N}_0$, which implies that the indices $P_i(x, \lambda_i^*)$ are monotonically decreasing with the system demand rate λ. By property (5), $\theta^{[B]}$ is a sink policy with a sink state $\mathbf{z} \in S$ satisfying $z_i = \min\{x \in \mathbb{N}_0 : P_i(x, \lambda_i^*) < 0\}$ for all $i \in \{1, 2, ..., N\}$. Hence, in order for $S_B(\bar{\lambda})$ *not* to be contained in $S_B(\lambda)$, some index value $P_i(x, \lambda_i^*)$ would have to be non-negative under the system demand rate $\bar{\lambda}$ and negative under λ. Since this has been shown to be impossible, property (6) follows, which completes the proof. □

A.7 Example: Jackson networks

A *Jackson network* is essentially a network of queues, in which customers may follow complicated (possibly never-ending) paths through the system as opposed to simply being directed to a particular

queue and then departing from the system after their service has been completed. Consider a queueing system with N service facilities. In a typical formulation of a Jackson network *without* any routing or admission control (see, for example, [67] p. 165), one assumes that customers arrive from 'outside' the system via a Poisson process with rate $\lambda > 0$, which is then split (via a probability distribution) into N separate Poisson arrival streams; one for each facility $i \in \{1, 2, ..., N\}$. Upon completing service at facility $i \in \{1, 2, ..., N\}$, customers are then routed to facility $j \in \{1, 2, ..., N\}$ with probability $q(i, j)$, where $\sum_{j=1}^{N} q(i, j) \leq 1$ and $q(i, 0) = 1 - \sum_{j=1}^{N} q(i, j)$ is the probability that a customer will permanently depart from the system after being served at facility i. Service times at any server of facility i have an exponential distribution with parameter $\mu_i > 0$.

Jackson networks were originally introduced by Jackson [91], and various other authors have studied their equilibrium behaviour and steady-state performance measures (see, for example, [61, 92, 97, 187]). The natural way to alter the classical formulation of a Jackson network so that it becomes a problem involving admission and routing control is to allow state-dependent routing decisions (effectively replacing the routing probabilities) to be made when customers arrive, and also when they complete service. The system considered in this example is described below.

1. Customers arrive from outside the system via a Poisson process with rate $\lambda > 0$ and either join some facility $i \in \{1, 2, ..., N\}$ or balk, in which case they leave the system permanently. These decisions are made deterministically according to the state of the system.

2. Each facility $i \in \{1, 2, ..., N\}$ has its own queue (with a first-come-first-served queue discipline), number of servers c_i and exponential service rate $\mu_i > 0$. Customers who complete service at facility i then either join facility $j \in \{1, 2, ..., N\}$ (with $j = i$ allowed) or balk. Again, decisions are made deterministically according to the system state.

A diagrammatic representation of the system is given in Figure A.1. Essentially, the system differs from the system described in Section 3.1 only by allowing customers to be re-routed to other facilities (or the same facility) after they have completed service, as opposed to being immediately lost from the system. It will also be assumed that the cost and reward system described in Section 3.1 is implemented, so that a holding cost $\beta_i > 0$ per customer per unit time is incurred at facility $i \in \{1, 2, ..., N\}$ and a fixed reward $\alpha_i > 0$ is earned after each service completion.

At first sight, it appears that the characterisation of socially optimal policies for this Jackson

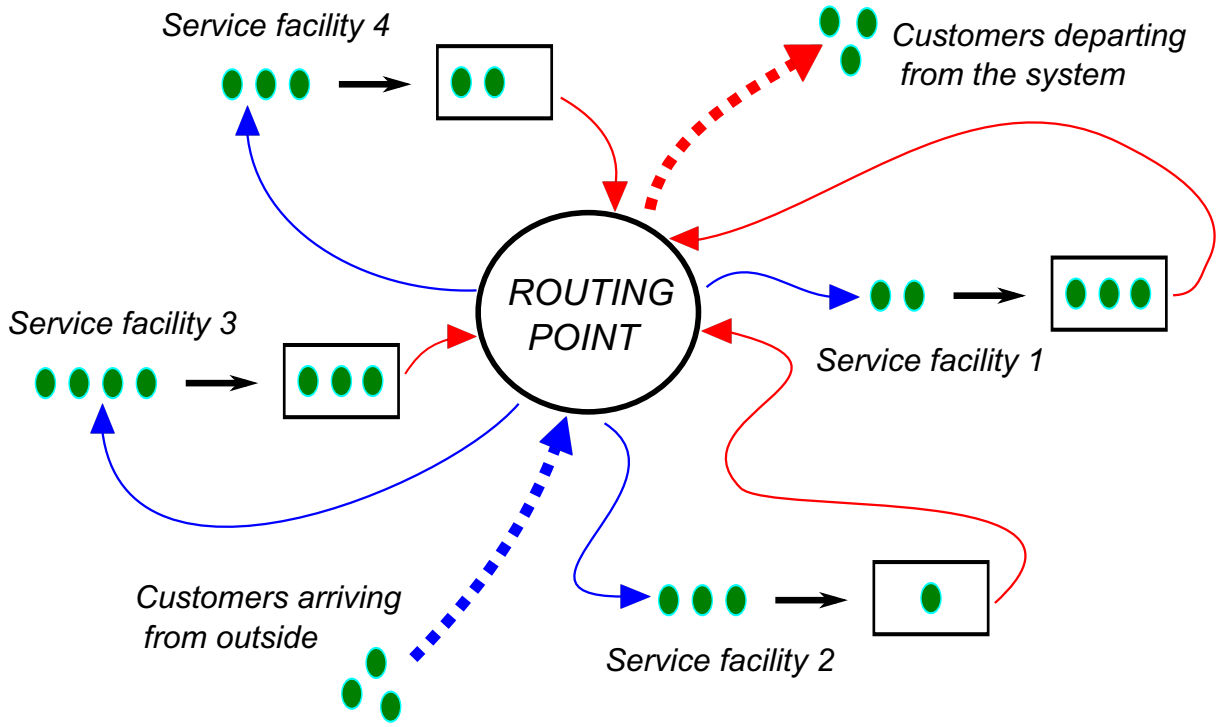


Figure A.1: A Jackson network with 4 service facilities.

network problem will be considerably more difficult than for the system described in Section 3.1; after all, decisions are required to be made much more frequently (after each service completion). Moreover, there are extra complications to consider; for example, suppose the state of the system is simply represented by a vector $\mathbf{x} = (x_1, x_2, ..., x_N)$, i.e. a vector of head counts at the various facilities. Then the next customer to be 'routed' to some destination may be a customer who arrives from outside, or (if the system is non-empty) a customer who completes service at one of the N facilities. This raises of the question of whether routing decisions should be made independently of where customers 'arrive from' (i.e. whether they arrive at the routing point from outside the system, or from within the system), or whether routing decisions must be allowed to depend on customers' origins in order to optimise the economic performance of the system.

In fact, the answer to the above question is that routing decisions *should* be allowed to depend on where customers have arrived from in order for socially optimal behaviour to be achieved. In an MDP formulation, this will generally mean that the action chosen at some state $\mathbf{x} \in S$ will actually be a vector $\mathbf{a} = (a_0, a_1, ..., a_N)$, where $a_0 \in \{0, 1, ..., N\}$ denotes the decision for a customer who

arrives from 'outside', and a_i is the decision for a customer who has just completed service at facility i (for $i = 1, 2, ..., N$). However, it transpires that the characterisation of a socially optimal policy for this problem is remarkably simple, and (even more surprisingly) socially optimal policies are unaffected by the demand rate λ and also unaffected by the values of *all other system parameters*, assuming that the usual assumption $\alpha_i\mu_i - \beta_i > 0$ for all $i \in \{1, 2, ..., N\}$ holds!

To see this, suppose the Jackson network is formulated as an MDP with the state space S defined in (3.5.1) and the real-time formulation (3.5.4) used for the single-stage rewards. Then each single-stage reward $r(\mathbf{x})$ is bounded above by $\sum_{i=1}^{N} c_i(\alpha_i\mu_i - \beta_i)$, and it follows that the expected long-run average reward $g_\theta(\mathbf{x})$ under any policy θ is also bounded above by the same value (given any initial state $\mathbf{x} \in S$). In the proof of Theorem 4.4.7 it was shown that the optimal average reward $g^*(\lambda)$ for the process Υ tends to $\sum_{i=1}^{N} c_i(\alpha_i\mu_i - \beta_i)$ as $\lambda \to \infty$, but it can never actually attain this value if λ is finite (since the steady-state probability of being in state $(c_1, c_2, ..., c_N)$ cannot be equal to one). However, with a little thought, one realises that in the case of the Jackson network, it is possible for the expected long-run average reward to actually attain a value $\sum_{i=1}^{N} c_i(\alpha_i\mu_i - \beta_i)$ by following a very simple policy. Specifically, consider the policy which operates as follows:

- Suppose a customer enters the system from 'outside' under state $\mathbf{x} \in S$. If $x_i < c_i$ for at least one facility $i \in \{1, 2, ..., N\}$, then they join a facility at which they don't have to wait in the queue. If $x_i \geq c_i$ for all $i \in \{1, 2, ..., N\}$, then they balk immediately.

- Any customer who completes service at some facility $i \in \{1, 2, ..., N\}$ immediately re-joins the same facility. This rule is always applied, regardless of the system state.

It should be noted (as a minor detail) that the description of the policy given above appears to suggest that, in fact, all customers (regardless of whether they arrive from outside, or arrive from a service facility) *are* adhering to the same rule and balking from the system if and only if all servers are occupied, since a service completion at facility $i \in \{1, 2, ..., N\}$ will imply that the customer who has just completed service will observe state $\mathbf{x}^{i-}$ before their new decision is made. While this is true in a physical sense, it is also important to recall (from the discussion in Section 3.5) that when the simplified state space formulation S is used, decisions are made anticipatively without knowledge of whether the next event will be an arrival or a service completion. Hence, the comment made earlier (which stated that routing decisions must depend on where customers have arrived

from) is valid, in the sense that an action vector $\mathbf{a} = (a_0, a_1, ..., a_N)$ must be used to differentiate between customers who arrive from outside and those who complete service.

Under the above policy, no customers are ever lost from any service facility, since they immediately re-join (and re-enter service) after their service has been completed. Regardless of the demand rate $\lambda > 0$, the system eventually 'fills up', so that each facility has all of its service channels occupied. At this point, all subsequent customers who arrive from outside the system balk, and each facility permanently has all of its service channels in use and no customers waiting in the queue. Hence, after a finite amount of time, the system is able to remain permanently in state $(c_1, c_2, ..., c_N)$ and therefore the expected long-run average reward is equal to the optimal value $\sum_{i=1}^{N} c_i(\alpha_i\mu_i - \beta_i)$. Figure A.2 illustrates the system operating under the policy described above.

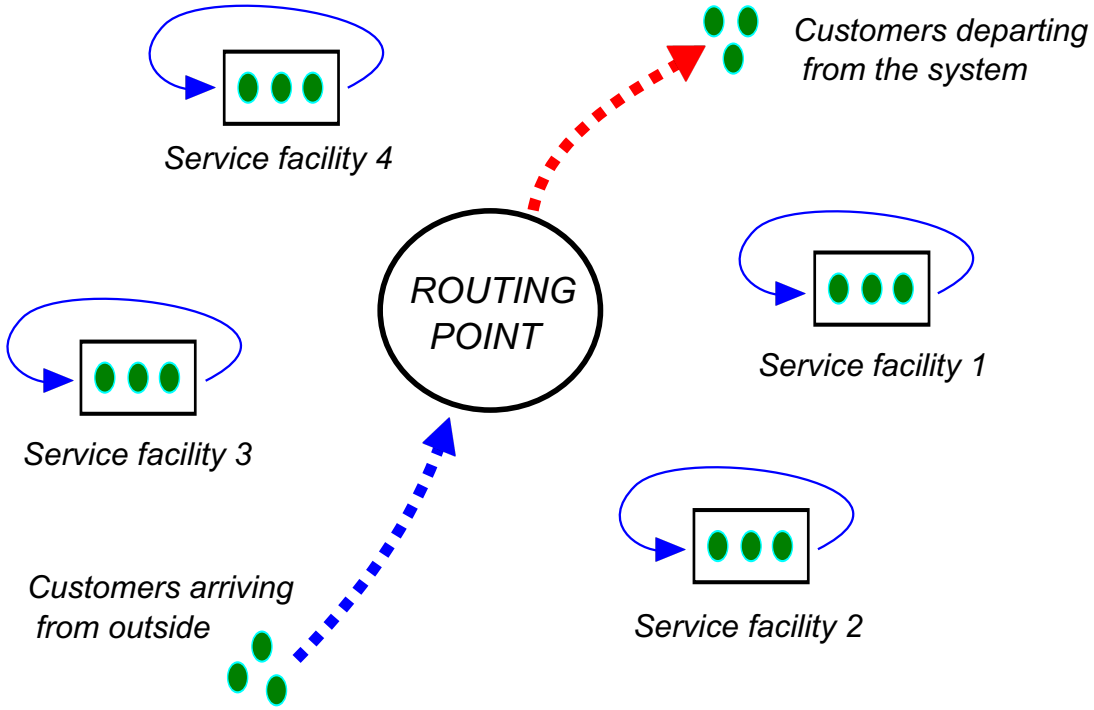


Figure A.2: The equilibrium behaviour of the Jackson network operating under the socially optimal 'customer recycling' policy described in the text, with 4 facilities and 3 servers at each facility.

This example has shown how a problem which at first appears to be more complicated than other problems considered in this thesis thus far in fact permits a much simpler optimal solution. The most interesting aspect of the problem is how the real-time reward formulation (3.5.4) enables

an optimal policy to be recognised easily. If the formulation (3.5.15) was used instead, it would perhaps be less obvious that an optimal policy would have such a simple form.

Of course, when one realises that the policy illustrated in Figure A.2 is optimal, it becomes clear that the problem considered in this example is trivial. There are various ways of altering the Jackson network control problem so that it is no longer trivial. For example, one might incorporate a 'drop-out' probability $q_i \in [0,1]$ for each service facility $i \in \{1,2,...,N\}$, so that any customer who completes service at facility i is permanently lost to the system with probability q_i and available for re-routing with probability $1-q_i$. The case $q_i = 1$ (for each i) would then correspond to the system described in Section 3.1, whereas the case $q_i = 0$ would represent the problem considered in this example. One might also restrict the set of actions $\mathbf{a} = (a_0, a_1, ..., a_N)$ available in various different ways in order to limit the routing possibilities. For example, customers might be prevented from immediately re-joining the same facility (by imposing the constraint $a_i \neq i$ for $i \in \{1,2,...,N\}$) or alternatively one might set up a system of queues in series by imposing the constraint $a_i < i$ for each facility $i \in \{1,2,...,N\}$ (without any restriction on a_0), so that customers are required to visit the facilities in descending numerical order (e.g. $N, N-1, N-2, ...$) but also have the option to 'skip' any number of queues when they arrive or complete service at any facility. Non-trivial problems involving Jackson networks are not discussed any further in this thesis. ⊠

A.8 Various counter-examples

As discussed in the introduction to Chapter 5, structural properties of average reward optimal policies (e.g. monotonicity) may be extremely difficult to prove in systems with an arbitrary number of heterogeneous service facilities. Indeed, certain properties which ostensibly seem 'sensible' may fail to hold. This appendix will present a series of counter-examples in order to show that various results that were proved in Sections 5.1, 5.2 and 5.3 (all of which were subject to restrictive assumptions on the complexity of the system) do not hold in greater generality.

Example A.8.1. ***(Lack of monotonicity in a two-facility system)***

Recall the two-facility system considered in Example 4.1.1. The demand rate used in that example

was $\lambda = 10$, and the parameters for the two facilities were given as follows:

$$c_1 = 2, \qquad \mu_1 = 8, \qquad \beta_1 = 10, \qquad \alpha_1 = 2,$$
$$c_2 = 2, \qquad \mu_2 = 2, \qquad \beta_2 = 10, \qquad \alpha_2 = 6.$$

Now suppose the demand rate is increased from $\lambda = 10$ to $\lambda = 12$. Using relative value iteration to find the optimal policy θ^*, one finds that it has the form shown in Table A.1.

	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	2	1	1
$x_1 = 1$	1	1	1
$x_1 = 2$	2	2	0
$x_1 = 3$	2	2	0

Table A.1: The optimal policy θ^* found by the RVIA, with a demand rate $\lambda = 12$.

Comparing Table A.1 with Table 4.1, one finds that increasing the demand rate from 10 to 12 causes the decision at state $(1, 0)$ to change from 2 to 1. However, this looks slightly odd, as the decision at state $(0, 0)$ is to join facility 2. So, comparing the decisions at states $(0, 0)$ and $(1, 0)$, one finds that increasing the number of customers at facility 1 actually causes facility 1 to become a 'more attractive' option to the policy θ^*. Further experimentation shows that changing the decision at state $(0, 0)$ from 2 to 1 causes the average reward to decrease by about 0.06%, and changing the decision at state $(1, 0)$ from 1 to 2 causes the average reward to decrease by 0.02%.

In Section 5.2 it was proved that, in a $2DSS$ system (i.e. a system with two *single-server* facilities), the policy θ^* found by the RVIA has the property that, if joining facility $i \in \{1, 2\}$ is chosen at some state $\mathbf{x} \in \tilde{S}$ with $x_i \geq 1$, then facility i must also be chosen at state $\mathbf{x}^{i-}$. This example has shown that the same property need not hold if the facilities have multiple servers. ⊠

Example A.8.2. ***(Conservativity of demand may not hold in a 2DSS system)***

Consider a system with two single-server facilities, with the following parameters:

$$c_1 = 1, \qquad \mu_1 = 14, \qquad \beta_1 = 5, \qquad \alpha_1 = 9,$$
$$c_2 = 1, \qquad \mu_2 = 5, \qquad \beta_2 = 3, \qquad \alpha_2 = 20.$$

Consider two different demand rates, $\lambda_1 = 9.8$ and $\lambda_2 = 10$. Under the smaller demand rate λ_1, the optimal policy θ_1^* found by the RVIA is a sink policy (see Definition 5.2.14) with sink state $(11, 13)$. However, under the demand rate λ_2, the RVIA policy θ_2^* is a sink policy with sink state $(10, 14)$. Table A.2 shows the decisions made under the policies θ_1^* and θ_2^* at states (x_1, x_2) belonging to a selected subset of $\tilde{S}$; specifically, the set $\big\{(x_1, x_2) \in \mathbb{N}_0^2 : 9 \leq x_1 \leq 11,\ 12 \leq x_2 \leq 14\big\}$.

Policy θ_1^*			
	$x_2 = 12$	$x_2 = 13$	$x_2 = 14$
$x_1 = 9$	1	1	1
$x_1 = 10$	2	1	0
$x_1 = 11$	2	0	0

Policy θ_2^*			
	$x_2 = 12$	$x_2 = 13$	$x_2 = 14$
$x_1 = 9$	1	1	1
$x_1 = 10$	2	2	0
$x_1 = 11$	2	0	0

Table A.2: The optimal policies θ_1^* and θ_2^* under demand rates $\lambda_1 = 9.8$ and $\lambda_2 = 10$ respectively.

The highlighted states in Table A.2 are those which are *not* positive recurrent under the relevant policies. Let S_1 and S_2 denote the sets of states which are positive recurrent under policies θ_1^* and θ_2^* respectively. As stated above, θ_1^* is a sink policy with sink state $(11, 13)$, and therefore S_1 does not include any states with $x_2 \geq 14$. This implies that S_2 is not contained in S_1.

Theorem 5.1.8 established that, in a single-facility system with multiple servers, optimal policies become increasingly *conservative* as the demand rate λ increases. This means that, given any two demand rates $\lambda_1, \lambda_2 > 0$ with $\lambda_1 < \lambda_2$, one can say that the set of positive recurrent states under the optimal policy associated with demand rate λ_2 is contained within the corresponding set under λ_1. This example has shown that the same property need not hold in a system with more than one facility, even if the system in question consists of only two single-server facilities. ⊠

Example A.8.3. ***(Non-monotonic progression of finite-stage optimal policies)***

Consider a $2DSS$ system with the same parameters as in Example A.8.2, and suppose the demand rate is $\lambda = 10$. This example concerns the progression of finite-stage optimal policies during relative value iteration. When the *real-time* reward formulation (3.5.1) is used, the $(n+1)$-stage optimal

policy θ_{n+1} chooses, for each state $\mathbf{x} \in \tilde{S}$, an action $\theta_{n+1}(\mathbf{x}) \in A_{\mathbf{x}}$ which satisfies:

$$\theta_{n+1}(\mathbf{x}) \in \arg\max_{a \in A_{\mathbf{x}}} \left\{ r(\mathbf{x}) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) h_n(\mathbf{y}) \right\}. \tag{A.8.1}$$

Here, the relative values $h_n(\mathbf{x})$ are computed recursively using (3.7.7) (with $h_0(\mathbf{x}) = 0$ for all states $\mathbf{x} \in \tilde{S}$). It will also be assumed in this example that the uniformisation parameter $\Delta = (\lambda + \sum_{i=1}^{N} c_i \mu_i)^{-1} = 1/29$ is used to calculate the transition probabilities $p(\mathbf{x}, a, \mathbf{y})$ (see (3.5.3)). As noted in previous sections, the condition (A.8.1) can actually be simplified to:

$$\theta_{n+1}(\mathbf{x}) \in \arg\max_{a \in A_{\mathbf{x}}} h_n(\mathbf{x}^{a+}).$$

On the other hand, when the *anticipatory* formulation (3.5.15) is used, the $(n+1)$-stage optimal policy $\hat{\theta}_{n+1}$ chooses, for each state $\mathbf{x} \in \tilde{S}$, an action $\hat{\theta}_{n+1}(\mathbf{x}) \in A_{\mathbf{x}}$ satisfying:

$$\hat{\theta}_{n+1}(\mathbf{x}) \in \arg\max_{a \in A_{\mathbf{x}}} \left\{ \hat{r}(\mathbf{x}, a) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) \hat{h}_n(\mathbf{y}) \right\},$$

where the values $\hat{h}(\mathbf{x})$ are computed recursively by using (3.7.7) and replacing $r(\mathbf{x})$ and $r(\mathbf{0})$ with $\hat{r}(\mathbf{x}, a)$ and $\hat{r}(\mathbf{0}, a)$. As discussed in previous sections, it is assumed that in the case of ties between two or more different actions, the finite-stage optimal policies choose actions according to a fixed ranking order of the N facilities. When the RVIA is applied to the system in Example A.8.2, with $\lambda = 10$ and the *real-time* formulation (3.5.1) used, one finds that the general trend is for the finite-stage optimal policies θ_n to become *less conservative* as n increases. For example, the policy θ_{20} is a sink policy with sink state $(3, 2)$, but θ_{60} is a sink policy with sink state $(5, 3)$.

However, after 272 iterations, the monotonic progression of the finite-stage optimal policies is broken. Indeed, the policy θ_{271} is a sink policy with sink state $(10, 6)$, but θ_{272} is a sink policy with sink state $(9, 7)$. Table A.3 shows the decisions made under the finite-stage policies θ_{271} and θ_{272} at states (x_1, x_2) belonging to the set $\left\{(x_1, x_2) \in \mathbb{N}_0^2 : 8 \leq x_1 \leq 10,\ 5 \leq x_2 \leq 7\right\} \subseteq \tilde{S}$.

The highlighted states in Table A.3 are those which are not positive recurrent under the relevant policies. Let S_{271} and S_{272} denote the sets of positive recurrent states under policies θ_{271} and θ_{272} respectively. Since θ_{272} is a sink policy with sink state $(9, 7)$, S_{272} does not include any states with $x_1 \geq 10$ and therefore S_{271} is not contained in S_{272}. Lemma 5.1.5 established that the progression

Policy θ_{271}			
	$x_2 = 5$	$x_2 = 6$	$x_2 = 7$
$x_1 = 8$	2	1	1
$x_1 = 9$	2	1	0
$x_1 = 10$	2	0	0

Policy θ_{272}			
	$x_2 = 5$	$x_2 = 6$	$x_2 = 7$
$x_1 = 8$	2	1	1
$x_1 = 9$	2	2	0
$x_1 = 10$	2	0	0

Table A.3: The finite-stage optimal policies θ_{271} and θ_{272} after iterations 271 and 272 of the RVIA.

of finite-horizon of finite-stage optimal policies in a *single-facility* system is monotonic when the real-time reward formulation is used, in the sense that the finite-stage optimal thresholds T_n^* are monotonically increasing with n. Thus, this example has shown that the same property need not hold in a system with more than one facility, even if the system in question is $2DSS$.

Similarly, it can be shown that the progression of finite-stage optimal policies need not be monotonic under the *anticipatory* reward formulation (3.5.15). Indeed, applying the RVIA to the same $2DSS$ system with the anticipatory formulation used, one finds that the general trend is for the finite-stage optimal policies $\hat{\theta}_n$ to become *more* conservative with n. However, this trend is broken after the 40^{th} iteration. Indeed, the policy $\hat{\theta}_{39}$ is a sink policy with sink state $(24, 32)$, but the policy $\hat{\theta}_{40}$ is a sink policy with sink state $(23, 33)$. Thus, the recurrent state space $\hat{S}_{40}$ associated with the policy $\hat{\theta}_{40}$ is *not* contained within the corresponding set $\hat{S}_{39}$ associated with $\hat{\theta}_{39}$, which confirms that the monotonic progression of the thresholds $\hat{T}_n^*$ in a single-facility system problem does not imply a similar monotonic progression in a system with more than one facility.

This example has important implications for the Shrinking Box algorithm discussed in Section 5.4 (see page 210). Indeed, the Shrinking Box algorithm relies upon the premise that the finite-stage optimal policies $\hat{\theta}_n$ become more conservative with n when the anticipatory formulation is used; more specifically, it assumes that any state $\mathbf{x} \in \tilde{S}$ which is excluded from the recurrent state space $\hat{S}_n$ under some finite-stage optimal policy $\hat{\theta}_n$ must also be excluded from the sets $\hat{S}_m$ on all subsequent iterations $m > n$. Since this property does not hold in general, the Shrinking Box algorithm may not succeed in converging to an average reward optimal policy. ⊠

Example A.8.4. ***(The optimal policy found by the RVIA may not be a sink policy)***

Theorem 5.2.15 established that, in a $2DSS$ system, the optimal policy θ^* found by relative value iteration is always a sink policy (see Definition 5.2.14). Evidence gathered from numerical experiments appears to indicate that in systems with $N \leq 5$, average reward optimal policies are *usually* sink policies; indeed, counter-examples are difficult to find. The system described in this example was found using a numerical search involving randomly-generated parameters. Consider a system with demand rate $\lambda = 21.57$ and three facilities, with the following parameters:

$$\begin{array}{llll} c_1 = 2, & \mu_1 = 15.17, & \beta_1 = 12.01, & \alpha_1 = 5.65, \\ c_2 = 4, & \mu_2 = 10.09, & \beta_2 = 22.4, & \alpha_2 = 9.07, \\ c_3 = 3, & \mu_3 = 6.36, & \beta_3 = 7.16, & \alpha_3 = 5.46. \end{array}$$

Of course, the *selfishly* optimal policy $\tilde{\theta}$ is always a sink policy, and in this example the sink state of $\tilde{\theta}$ is $(14, 16, 14)$. However, it transpires that the socially optimal policy θ^* found by the RVIA is *not* a sink policy. In fact, balking is chosen by θ^* at the states $(13, 10, 14)$ and $(12, 11, 14)$, both of which are positive recurrent under θ^*. In order to show how this is possible, it will be useful to examine the decisions made by θ^* at some particular states within $\tilde{S}$. Given that $\tilde{S}$ may be represented geometrically as a 3-dimensional cuboid of size $14 \times 16 \times 14$, one may consider some fixed value $x_3 = n \in \{0, 1, ..., 14\}$ and refer to the subset $\{(x_1, x_2, n) \in \mathbb{N}_0^3 : x_1 \leq 14, x_2 \leq 16\}$ as the n^{th} (x_1, x_2)-slice of $\tilde{S}$. Table A.4 compares the 13^{th} and 14^{th} (x_1, x_2)-slices of $\tilde{S}$ with respect to the decisions made by the optimal policy θ^* for certain pairs (x_1, x_2) (in other words, the comparison is between states of the form $(x_1, x_2, 13)$ and states of the form $(x_1, x_2, 14)$).

Essentially, the reason for the unusual structure of θ^* in this example is that facility 2 is chosen at state $(12, 10, 13)$, but facility 1 is chosen at state $(12, 10, 14)$. This means that the presence of an extra customer at the third facility causes θ^* to alter its preference between the first and second facilities. The state $(12, 10, 13)$ is positive recurrent under θ^*, and it can be seen from Table A.4 that two consecutive arrivals (without any service completions) will cause the system to transfer from state $(12, 10, 13)$ to $(12, 11, 14)$, at which balking is chosen. However, if a service completion then occurs at facility 2, the system finds itself in state $(12, 10, 14)$, from which the *other* balking state $(13, 10, 14)$ becomes accessible via an arrival. Thus, the process operating under policy θ^* is able to access two different 'balking states', and therefore θ^* is not a sink policy. ⊠

Decisions at states $(x_1, x_2, 13)$			
	$x_2 = 9$	$x_2 = 10$	$x_2 = 11$
$x_1 = 11$	2	1	1
$x_1 = 12$	2	2	3
$x_1 = 13$	2	3	0

Decisions at states $(x_1, x_2, 14)$			
	$x_2 = 9$	$x_2 = 10$	$x_2 = 11$
$x_1 = 11$	2	1	1
$x_1 = 12$	2	1	0
$x_1 = 13$	2	0	0

Table A.4: Decisions made by θ^* at selected states belonging to the 13^{th} and 14^{th} (x_1, x_2)-slices of $\tilde{S}$.

Example A.8.5. ***(The optimal policy found by the RVIA in a system with homogeneous facilities may not be cube-shaped)***

In Section 5.3 it was proved that the optimal policy θ^* found by the RVIA in a system with homogeneous facilities is always a 'Join the Shortest Queue' (JSQ) policy. This means that, at any state $\mathbf{x} \in \tilde{S}$, θ^* either chooses to join a facility $j \in \{1, 2, ..., N\}$ which has the lowest (or joint-lowest) number of customers present, or chooses to balk. In cases where two or more facilities are tied for the smallest number of customers present, one may assume that the facility with the smallest index j is chosen. Hence, as discussed in Section 5.3, a complete characterisation of the policy θ^* requires only the identification of a subset of states $S_B \subseteq \tilde{S}$ at which balking is chosen by θ^*.

Let S_{θ^*} denote the set of states in $\tilde{S}$ which are positive recurrent under θ^*, and let T_j^* denote the maximum number of customers that may be present at facility $j \in \{1, 2, ..., N\}$ when the process operates under θ^* (assuming that the initial state is contained in S_{θ^*}). That is:

$$T_j^* = \max\{n \in \mathbb{N}_0 : \exists\, \mathbf{x} \in S_{\theta^*} \text{ such that } x_j = n\}.$$

In a system with homogeneous facilities, it would seem logical to suppose that the values T_j^* for all facilities j should be equal to some common value $T \in \mathbb{N}$, since there is no obvious reason why the optimal policy θ^* should not treat all facilities equally. If this is indeed the case, then by considering a sequence of NT consecutive customer arrivals starting from state $\mathbf{0}$, one can easily see that θ^* (being a JSQ policy) must be a sink policy with sink state $(T, T, ..., T)$. Informally speaking, one might describe such a policy as 'cube-shaped', since its set of positive recurrent states is representable geometrically as an N-dimensional cube. However, this example will show that the RVIA policy θ^* need not be cube-shaped, even if the system is $2DSS$.

Consider a system with demand rate $\lambda = 15$ and two single-server facilities. Suppose also that the facilities are *homogeneous* and share a common set of parameters, given by:

$$c = 1, \qquad \mu = 4, \qquad \beta = 1, \qquad \alpha = 5.$$

With these parameters, it transpires that the set of positive recurrent states S_{θ^*} associated with the RVIA policy θ^* is of dimension 3×2. The policy is shown in Table A.5.

	$x_2 = 0$	$x_2 = 1$	$x_2 = 2$
$x_1 = 0$	1	1	1
$x_1 = 1$	2	1	1
$x_1 = 2$	2	2	1
$x_1 = 3$	2	2	0

Table A.5: The optimal policy θ^* found by the RVIA in Example A.8.5.

Of course, the fact that the policy θ^* shown in Table A.5 chooses facility 1 rather than facility 2 at states $(0,0)$, $(1,1)$ and $(2,2)$ is due to the convention assumed for tie-breaking, whereby the facility with the lowest index is chosen. Indeed, Lemma 5.3.1 implies that the relative values $h((x,y))$ and $h((y,x))$ are equal for any pair $(x,y) \in \mathbb{N}_0^2$, and therefore the 2×3 policy that would be obtained by transposing Table A.5 and changing each '1' to '2' (and vice versa) would also be average reward optimal for this system. Thus, the important characteristic of θ^* in this example is that it allows *one* of the two service facilities to have up to three customers present, but not both.

Further experimentation shows that the sink policies with sink states $(2,2)$ and $(3,3)$ (and all routing decisions made according to the JSQ rule) are both sub-optimal. Indeed, it can also be verified that there do not exist any average reward optimal 'cube-shaped' policies whose positive recurrent state spaces are contained within the selfish space $\tilde{S}$. In view of Theorem 4.2.5, this is sufficient to show that the 'non-cube' policy θ^* given in Table A.5 performs better than any cube-shaped policy, despite its lack of even-handed treatment of the two facilities. ⊠

Before concluding this appendix, it will be appropriate to mention some further properties for which no proofs or counter-examples have yet been found. These are stated below.

- Let θ^* denote the optimal policy found by relative value iteration. In Section 5.2 it was proved

(see Theorem 5.2.11) that if balking is chosen by θ^* at some state $\mathbf{x} \in \tilde{S}$ in a $2DSS$ system, then balking is also chosen at states of the form $\mathbf{x}^{j+}$ (where $j \in \{1, 2\}$). Whether or not this property also holds in a more general N-facility system remains unknown.

- Let $g^*(\lambda)$ denote the optimal value of the average reward given a demand rate $\lambda > 0$, and let $\tilde{g}(\lambda)$ denote the corresponding average reward given by the selfishly optimal policy $\tilde{\theta}$ defined in Section 4.1. In Section 5.1 it was proved that, in an $M/M/1$ system, the *loss per unit time* $g^*(\lambda) - \tilde{g}(\lambda)$ is monotonically increasing with λ. Whether or not this property also holds in the case of multiple servers and/or multiple service facilities is also unknown.

The challenge of establishing whether or not the two properties referred to above can indeed be shown to hold in greater generality remains the subject of ongoing work.

A.9 Convergence of RL algorithms

This appendix discusses the convergence properties of RL algorithms, *assuming that no restriction is placed on the number of iterations to be performed*; in other words, the behaviour of the Q-factor estimates as the number of iterations completed tends to infinity is now of interest. Throughout this appendix, the term "convergence" should be understood to refer to the limiting behaviour of the Q-factor estimates $\hat{Q}(\mathbf{x}, a)$ which are updated during the evolution of an RL algorithm. However, due to the uniqueness of solutions to the optimality equations (7.2.2) and the evaluation equations (7.2.3), the successful convergence of the Q-factor estimates to their true theoretical values directly implies the attainment of a policy which attains the optimal average reward g^* (in the case of the optimality equations) or the policy-specific value g_θ (in the case of the evaluation equations). Thus, the convergence of the Q-factor estimates to a set of values satisfying (7.2.2) may equivalently be described as the convergence of an RL algorithm to an average reward optimal policy.

Before proceeding any further, it should be noted that even if an RL algorithm possesses convergence properties *in theory*, it may well be the case that this convergence is much too slow to be attainable in practice. Indeed, Example 7.2.3 in Section 7.2 has shown that the R-learning algorithm may perform tens of millions of iterations without visiting more than 5% of state-action pairs in the finite set $\tilde{S} \times A_{\mathbf{x}}$. In order for accurate estimates $\hat{Q}(\mathbf{x}, a)$ to be obtained for *all* state-action pairs

$(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$, it is necessary for each individual pair to be visited many times (as will become clear from later results). However, it is apparent from Example 7.2.3 that the number of iterations required for R-learning (or similar algorithms) to cover a state-action space consisting of tens of thousands of elements in such a comprehensive fashion is likely to be astronomical.

Since an essential premise throughout Chapter 7 is that RL algorithms are much *faster* to run than DP algorithms in systems with a large number of states, there is a certain conflict of interest here; it appears that the convergence of an RL algorithm to a complete and accurate set of Q-factor estimates may not be achievable unless the algorithm is allowed to run for as long as (or even longer than) a DP algorithm. Due to issues of practicality, as well as the highly technical nature of the subject in general, convergence properties of RL algorithms will not be discussed here in rigorous detail; instead, a brief survey of important results from the literature will be given.

The RL algorithms discussed in Section 7.2 are applicable to *average reward* optimisation problems, in which the performance measure of interest under a particular policy is the expected long-run average reward per unit time. However, a significant proportion of the RL literature addresses *discounted reward* problems. As discussed in Chapter 3, a *discounted reward* problem (in a discrete-time context) is one in which a reward earned n time steps into the future is discounted by a factor ϕ^n relative to an immediate reward, where $\phi \in (0, 1)$. Discounted reward problems tend to be somewhat more mathematically tractable than average reward problems, and it is therefore unsurprising that theoretical guarantees have been developed for the performance of RL algorithms in the discounted reward case which do not extend easily to the average reward case.

This appendix will begin by outlining certain results from the literature regarding the convergence properties of RL algorithms in discounted reward problems, and discuss (where possible) the generalisation of these results to average reward problems. Although this thesis is not expressly concerned with discounted reward problems, it will be useful to provide an insight into the nature of the technical challenges involved in proving RL convergence properties in the discounted case, if only to show why the average reward case presents an even greater level of difficulty.

As stated at the end of Section 7.2, there does not exist any formal proof that the R-learning algorithm presented on page 301 converges to an average reward optimal policy, although its strong performance has been demonstrated in experiments (see [123, 154, 162]). The TD learning algorithm

presented on page 292, which evaluates a fixed stationary policy θ, is essentially a special case of the R-learning algorithm with only one action allowed at each state, and therefore a proof of its convergence to the theoretical value g_θ is not attainable either. However, the picture is not entirely bleak as far as average reward RL algorithms are concerned. There does exist an algorithm, known as *relative Q-learning*, which *has* been proven to converge to an average reward optimal policy, provided that certain conditions are met. This will be discussed later in this appendix.

As mentioned in Section 7.2, R-learning bears a close resemblance to the famous *Q-learning* algorithm developed by Watkins [191], which applies to discounted reward problems. In Q-learning, the update rule used for the estimated value $\hat{Q}_\phi(\mathbf{x}_n, a_n)$ (where $(\mathbf{x}_n, a_n) \in S \times A_{\mathbf{x}}$ is the state-action pair selected on the n^{th} iteration and $\phi \in (0,1)$ is the discount factor) is given by:

$$\hat{Q}_\phi(\mathbf{x}_n, a_n) \leftarrow (1-\delta_n)\hat{Q}_\phi(\mathbf{x}_n, a_n) + \delta_n \left[r_n + \phi \max_{a' \in A_{\mathbf{x}_{n+1}}} \hat{Q}_\phi(\mathbf{x}_{n+1}, a') \right], \tag{A.9.1}$$

where, as usual, δ_n is a learning parameter. The estimated average reward g_n does not feature in Q-learning at all, and as such step 4 of the R-learning algorithm is omitted entirely from Q-learning. This simplifies matters to a certain extent, since the question of how g^* should be estimated in R-learning and variants such as R-SMART (see [62, 64]) is not easy to resolve.

Like most RL algorithms, Q-learning operates by updating the estimate $\hat{Q}_\phi(\mathbf{x}, a)$ for any given state-action pair $(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$ as and when the state $\mathbf{x}$ is 'visited' by an agent following a path through the system, and the action a is chosen. It may be said, therefore, that Q-learning uses *asynchronous updates* to estimate the Q-factors which characterise an optimal policy. The difference between *asynchronous* and *synchronous* updates may be explained most easily by referring to the dynamic programming algorithms introduced in Chapter 3. DP algorithms such as the Relative Value Iteration Algorithm (page 101) use a *synchronous* updating style, which involves sweeping through *all* states (or, if the Q-factors $Q(\mathbf{x}, a)$ are being evaluated as opposed to the relative values $h(\mathbf{x})$, all state-action pairs) on every iteration and updating each finite-stage value $h_n(\mathbf{x})$ using the values $h_{n-1}(\mathbf{y})$ obtained on the *previous* iteration. Consequently, one can say that if the procedure is terminated at the end of the n^{th} iteration (for some $n \in \mathbb{N}$), the relative values for *all* states $\mathbf{x} \in S$ will have been updated exactly n times since initialisation of the algorithm.

On the other hand, it is obvious that in RL algorithms such as Q-learning, the order in which

the estimates $\hat{Q}(\mathbf{x}, a)$ are updated may be haphazard; that is, some state-action pairs may be 'sampled' much more frequently than others. In fact, the asynchronous style of updating used in RL is one of its great advantages, since it enables the Q-factors at states which are visited often to be updated more frequently (and, therefore, approximated more accurately) than those which are associated with seldom-visited states, which in turn causes an appropriate emphasis to be placed on obtaining accurate estimates for the Q-factors which are critical to the performance of the system; this has been shown by Example 7.2.3. However, as one might expect, convergence proofs are more difficult to establish for algorithms which use asynchronous updating than for those which update Q-factors uniformly in the style of a DP algorithm. In this appendix, the main result which implies convergence of *synchronous* algorithms which perform simulation-based updates similar to those of RL algorithms will be given first, followed by discussion of the *asynchronous* case.

The next result, which may be found in Gosavi [62] (p. 382) and is based on results from [22] and [111], uses the concept of a *stochastic approximation scheme*. Informally speaking, this refers to an algorithmic procedure which uses 'noisy' observations to estimate the characteristics of functions which cannot be computed directly; for example, the 'data pieces' $r_n - g_n + \hat{Q}_\theta(\mathbf{x}_{n+1})$ acquired during the TD learning algorithm (see page 292) are subject to the random 'noise' caused by simulation. The Robbins-Monro algorithm on page 294 is an example of a stochastic approximation scheme. Note that the theorem relies upon the assumption of *finite* state and action spaces.

*** Theorem A.9.1.** *Assume that the state space $\tilde{S}$ and action spaces $A_{\mathbf{x}}$ (for $\mathbf{x} \in \tilde{S}$) are finite, and let $\chi := \sum_{\mathbf{x} \in \tilde{S}} |A_{\mathbf{x}}|$ denote the total number of state-action pairs. For $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ and $n \in \mathbb{N}_0$, let $\hat{Q}_n(\mathbf{x}, a)$ be given by the following stochastic approximation scheme:*

$$\hat{Q}_n(\mathbf{x}, a) = \begin{cases} 0, & \text{if } n = 0, \\ (1 - \delta_n)\hat{Q}_{n-1}(\mathbf{x}, a) + \delta_n \left(H(\hat{\mathbf{Q}}_{n-1})(\mathbf{x}, a) + \sigma_n(\mathbf{x}, a) \right), & \text{if } n \geq 1. \end{cases} \tag{A.9.2}$$

where $\hat{\mathbf{Q}}_n$ is a vector of dimension χ whose elements are the estimated Q-factors $\hat{Q}_n(\mathbf{x}, a)$ (assuming that these have been arranged in some systematic order), H is a function which maps $\mathbb{R}^\chi$ to $\mathbb{R}^\chi$, $H(\hat{\mathbf{Q}}_n)(\mathbf{x}, a)$ denotes the component of the vector $H(\hat{\mathbf{Q}}_n)$ associated with the pair $(\mathbf{x}, a)$, $\sigma_n(\mathbf{x}, a)$ denotes the random noise generated on the n^{th} iteration when the Q-factor for the pair $(\mathbf{x}, a)$ is

updated, and the parameters δ_n satisfy the Robbins-Monro conditions:

$$\sum_{n=1}^{\infty} \delta_n = \infty, \qquad \sum_{n=1}^{\infty} (\delta_n)^2 < \infty. \tag{A.9.3}$$

Assume the approximation scheme in (A.9.2) is used synchronously*; that is, the Q-factor estimates for all state-action pairs are updated exactly once on every iteration, using the estimates from the previous iteration. Also assume that the following conditions hold:*

1. *The iterates $\hat{\mathbf{Q}}_n$ are bounded. That is, there exists a constant $K < \infty$ such that for all state-action pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ and $n \in \mathbb{N}_0$, the following holds:*

$$\hat{Q}_n(\mathbf{x}, a) \leq K. \tag{A.9.4}$$

2. *For each $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ and $n \in \mathbb{N}$, the noise term $\sigma_n(\mathbf{x}, a)$ satisfies:*

$$\begin{aligned} &E\left[\sigma_n(\mathbf{x}, a) \,\middle|\, \hat{\mathbf{Q}}_0, \hat{\mathbf{Q}}_1, ..., \hat{\mathbf{Q}}_{n-1}, \sigma_1(\mathbf{x}, a), \sigma_2(\mathbf{x}, a), ..., \sigma_{n-1}(\mathbf{x}, a)\right] = 0, \\ &E\left[(\sigma_n(\mathbf{x}, a))^2 \,\middle|\, \hat{\mathbf{Q}}_0, \hat{\mathbf{Q}}_1, ..., \hat{\mathbf{Q}}_{n-1}, \sigma_1(\mathbf{x}, a), \sigma_2(\mathbf{x}, a), ..., \sigma_{n-1}(\mathbf{x}, a)\right] \leq A_1 + A_2 ||\hat{\mathbf{Q}}_n||_\infty^2, \end{aligned} \tag{A.9.5}$$

   where A_1 and A_2 are constants, and $||\hat{\mathbf{Q}}_n||_\infty = \max_{(\mathbf{x},a) \in S \times A_{\mathbf{x}}} |\hat{Q}_n(\mathbf{x}, a)|$.

3. *The function $H : \mathbb{R}^\chi \to \mathbb{R}^\chi$ is a non-expansive mapping with respect to the max norm, $||\cdot||_\infty$. That is, for any two vectors $\mathbf{Q}, \mathbf{Q}' \in \mathbb{R}^\chi$, the following holds:*

$$||H(\mathbf{Q}) - H(\mathbf{Q}')||_\infty \leq ||\mathbf{Q} - \mathbf{Q}'||_\infty. \tag{A.9.6}$$

Then, with probability one, the sequence $(\hat{\mathbf{Q}}_0, \hat{\mathbf{Q}}_1, ...)$ converges to a vector $\mathbf{Q}$ such that:

$$\mathbf{Q} = H(\mathbf{Q}),$$

provided that such a vector (referred to as a fixed point *of H) exists.*

The fact that Theorem A.9.1 assumes a *finite* number of state-action pairs is not particularly a limitation in the context of the queueing systems considered in this thesis, since the results in Chapter 4 have established the existence of discount-optimal and average reward optimal stationary policies which cause the process to remain contained within the selfishly optimal state space $\tilde{S}$

defined in (4.1.3), and therefore it is reasonable to assume that convergence of an RL algorithm (if it occurs) will yield a policy of this type. A proper justification for this assertion will become apparent when the meaning of the transformation H (in an MDP context) is explained later in this appendix. As one might expect, convergence properties of RL algorithms are considerably more difficult to prove when the state space is countably infinite; see, for example, [20].

As discussed in [62] (p. 382), the proof of Theorem A.9.1 relies upon a rather more general result which is due to Kushner and Clark [111] (p. 39) and concerns the solution of a certain type of ordinary differential equation (ODE), which can be expressed in the form:

$$\frac{d}{dt}\mathbf{Q}(t) = H(\mathbf{Q}(t)) - \mathbf{Q}(t), \tag{A.9.7}$$

or (equivalently) $\dot{\mathbf{Q}} = H(\mathbf{Q}) - \mathbf{Q}$. After establishing the stability of the ODE (A.9.7) and its relationship to the discrete-time approximation schemes used by RL algorithms such as Q-learning, the proof of the theorem then makes use of a well-known convergence theorem for martingales which can be found in [24] (p. 89). Although the full details of these technical results are somewhat beyond the scope of this thesis, a short and informal explanation will be given here of the 'ODE method', which involves showing that a continuous-time formulation of (A.9.2) is possible. The explanation here will mainly follow the development in Bertsekas and Tsitsiklis [15] (p. 171).

Recall that the learning parameters δ_n are required to satisfy the Robbins-Monro conditions (A.9.3). In particular, the second condition in (A.9.3) ensures that $\delta_n \to 0$ as $n \to \infty$. Following Bertsekas and Tsitsiklis [15], let δ be a small, fixed positive number. Evidently, it is possible to choose a strictly increasing sequence of integers $(k_m)_{m\in\mathbb{N}}$ such that for $m \in \mathbb{N}$, one can say:

$$\sum_{n=k_m+1}^{k_{m+1}} \delta_n \approx \delta. \tag{A.9.8}$$

Intuitively, one may observe that as the δ_n values become small, it becomes possible to arrange them into a sequence of finite series such that each series closely approximates the constant δ. Next, it can be shown that under conditions (A.9.4) and (A.9.5), the sequence $\left(\sum_{m=1}^{n} \delta_m \sigma_m(\mathbf{x}, a)\right)_{n\in\mathbb{N}}$ is a martingale with bounded second moments, which implies that it is convergent with probability one due to the martingale convergence theorem (see [24], p. 89) and hence:

$$\lim_{n\to\infty} \delta_n \sigma_n(\mathbf{x}, a) = 0. \tag{A.9.9}$$

Let $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ be arbitrary. From (A.9.2) it can be inferred that, for $n \geq 1$:

$$\hat{Q}_n(\mathbf{x}, a) = \hat{Q}_{n-1}(\mathbf{x}, a) + O(\delta_n), \tag{A.9.10}$$

where $O(\delta_n)$ denotes a quantity which tends to zero as $\delta_n \to 0$. By iterating (A.9.10) and also using (A.9.8), one can then say that for integers $m \geq 1$ and $n \in \{k_m + 1, k_m + 2, ..., k_{m+1}\}$:

$$\hat{Q}_n(\mathbf{x}, a) = \hat{Q}_{k_m}(\mathbf{x}, a) + O(\delta). \tag{A.9.11}$$

From (A.9.2), it follows that for integers $m \geq 1$:

$$\hat{Q}_{k_{m+1}}(\mathbf{x}, a) = \hat{Q}_{k_m}(\mathbf{x}, a) + \sum_{n=k_m+1}^{k_{m+1}} \delta_n \left(H(\hat{\mathbf{Q}}_{n-1})(\mathbf{x}, a) - \hat{Q}_{n-1}(\mathbf{x}, a) \right) + \sum_{n=k_m+1}^{k_{m+1}} \delta_n \sigma_n(\mathbf{x}, a). \tag{A.9.12}$$

By using the non-expansiveness property (A.9.6) and the fact that (A.9.11) holds for *all* state-action pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$, one can show that for $n \in \{k_m + 1, k_m + 2, ..., k_{m+1}\}$:

$$H(\hat{\mathbf{Q}}_n)(\mathbf{x}, a) - H(\hat{\mathbf{Q}}_{k_m})(\mathbf{x}, a) \leq ||\hat{\mathbf{Q}}_n - \hat{\mathbf{Q}}_{k_m}||_\infty = O(\delta). \tag{A.9.13}$$

Note that (A.9.9) ensures that $\sum_{n=k_m+1}^{k_{m+1}} \delta_n \sigma_n(\mathbf{x}, a)$ tends to zero as $m \to \infty$. Hence, using (A.9.11), (A.9.12) and (A.9.13), one can show that for sufficiently large $m \in \mathbb{N}$:

$$\begin{aligned} \hat{Q}_{k_{m+1}}(\mathbf{x}, a) &\approx \hat{Q}_{k_m}(\mathbf{x}, a) + \sum_{n=k_m+1}^{k_{m+1}} \delta_n \left(H(\hat{\mathbf{Q}}_{k_m})(\mathbf{x}, a) - \hat{Q}_{k_m}(\mathbf{x}, a) + O(\delta) \right) \\ &\approx \hat{Q}_{k_m}(\mathbf{x}, a) + \delta \left(H(\hat{\mathbf{Q}}_{k_m})(\mathbf{x}, a) - \hat{Q}_{k_m}(\mathbf{x}, a) \right) + O(\delta^2). \end{aligned}$$

Hence, up to first order in δ, a reasonable approximation of (A.9.2) is:

$$\hat{Q}_{k_{m+1}}(\mathbf{x}, a) - \hat{Q}_{k_m}(\mathbf{x}, a) = \delta \left(H(\hat{\mathbf{Q}}_{k_m})(\mathbf{x}, a) - \hat{Q}_{k_m}(\mathbf{x}, a) \right). \tag{A.9.14}$$

As illustrated by Figure A.3, the Q-factor estimates $\hat{Q}_{k_m}(\mathbf{x}, a)$ and $\hat{Q}_{k_{m+1}}(\mathbf{x}, a)$ may be regarded as two estimates for $Q(\mathbf{x}, a)$ taken at two different points in time. Moreover, the amount of time separating these two points is $\delta_{k_m+1} + \delta_{k_m+2} + ... + \delta_{k_{m+1}} \approx \delta$. Hence, after dividing by δ in (A.9.14) and considering the limit as $\delta \to 0$, it is clear that the behaviour of the stochastic approximation algorithm (A.9.2) is similar to that of the deterministic differential equation:

$$\frac{d}{dt}\mathbf{Q}(t) = H(\mathbf{Q}(t)) - \mathbf{Q}(t). \tag{A.9.15}$$

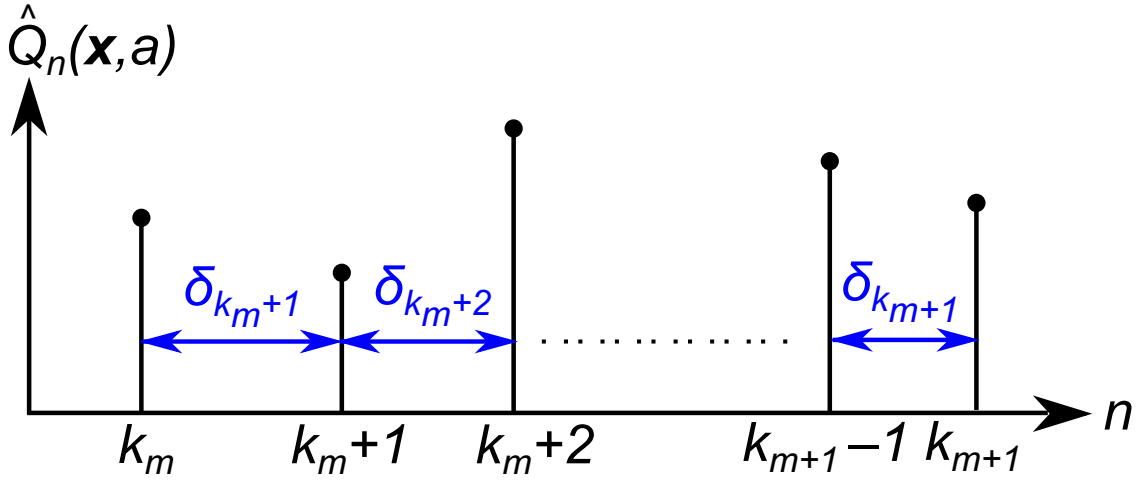


Figure A.3: The distance between $n = k_m$ and $n = k_{m+1}$ on the 'time axis' is $\delta_{k_m+1} + ... + \delta_{k_{m+1}} \approx \delta$.

As remarked by Bertsekas and Tsitsiklis [15] (p. 173), the arguments given in the preceding two pages are somewhat non-rigorous, and one should refer to a source such as Borkar and Meyn [21] for a fully rigorous approach. After the analogy between (A.9.2) and the ODE (A.9.15) has been properly established, the proof of Theorem A.9.1 follows using the arguments given by Abounadi et. al. [1]. However, it is important to note that the theorem applies to stochastic approximation schemes which operate in a *synchronous* manner. As discussed previously, all of the RL algorithms considered in this chapter rely upon *asynchronous* updates. The generalisation of Theorem A.9.1 to asynchronous approximation schemes will be discussed later in this appendix, but first it will be useful to illustrate the result of the theorem by considering some specific examples.

Consider a synchronous version of the Q-learning algorithm discussed earlier, which operates by updating the estimate $\hat{Q}_n(\mathbf{x}, a)$ for *all* state-action pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ on the n^{th} iteration ($n = 1, 2, ...$) using the update rule (A.9.1). For clarity, the algorithm which operates in this manner will be referred to in this appendix as *synchronous Q-learning*, whereas the shorter name 'Q-learning' will always refer to the *asynchronous* version of the same algorithm (in other words, Q-learning should be assumed to be asynchronous unless otherwise stated). In the case of *synchronous* Q-learning, the use of the notation $\mathbf{x}_n$ to denote the state 'visited' on the n^{th} iteration (and, similarly, a_n to denote the action chosen) becomes somewhat inappropriate, as the Q-factor estimates for the various state-action pairs are updated even-handedly on each iteration, as opposed to being based on the simulated path of an agent exploring the system. The update rule for *synchronous*

Q-learning can be expressed, for $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ and integers $n \geq 1$, in the form:

$$\hat{Q}_n(\mathbf{x}, a) = (1 - \delta_n)\hat{Q}_{n-1}(\mathbf{x}, a) + \delta_n \left[r(\mathbf{x}, a, \mathbf{Y}) + \phi \max_{a' \in A_{\mathbf{Y}}} \hat{Q}_{n-1}(\mathbf{Y}, a') \right], \tag{A.9.16}$$

where $\hat{Q}_n(\mathbf{x}, a)$ is the estimate of $Q(\mathbf{x}, a)$ obtained on the n^{th} iteration, and $\mathbf{Y}$ should be interpreted as a random variable which is generated (like the 'next state' $\mathbf{x}_{n+1}$ in the asynchronous case) using simulation; specifically, for each $\mathbf{y} \in \tilde{S}$, $\mathbf{Y} = \mathbf{y}$ with probability $p(\mathbf{x}, a, \mathbf{y})$. As such, the reward $r(\mathbf{x}, a, \mathbf{Y})$ is also random, due to its dependence on $\mathbf{Y}$. The randomly-sampled state $\mathbf{Y}$ is used only to update the Q-factor estimate for the pair $(\mathbf{x}, a)$; this contrasts with the asynchronous case, in which the state $\mathbf{x}_{n+1}$ sampled using simulation becomes the next state to be 'visited'. It should also be understood that on the n^{th} iteration, the same learning parameter δ_n is used to update each state-action pair. Next, define the transformation $J : \mathbb{R}^{\chi} \to \mathbb{R}^{\chi}$ as follows:

$$J(\hat{\mathbf{Q}}_n)(\mathbf{x}, a) := r(\mathbf{x}, a, \mathbf{Y}) + \phi \max_{a' \in A_{\mathbf{Y}}} \hat{Q}_n(\mathbf{Y}, a'),$$

where $\hat{\mathbf{Q}}_n$ is the vector with components $\hat{Q}_n(\mathbf{x}, a)$. Also, define $H : \mathbb{R}^{\chi} \to \mathbb{R}^{\chi}$ by:

$$H(\hat{\mathbf{Q}}_n)(\mathbf{x}, a) := E[J(\hat{\mathbf{Q}}_n)(\mathbf{x}, a)] = \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) \left(r(\mathbf{x}, a, \mathbf{y}) + \phi \max_{a' \in A_{\mathbf{y}}} \hat{Q}_n(\mathbf{y}, a') \right), \tag{A.9.17}$$

where the expectation is with respect to the distribution of $\mathbf{Y}$. The 'random noise' $\sigma_n(\mathbf{x}, a)$ generated when the pair $(\mathbf{x}, a)$ is updated on the n^{th} iteration is then given by:

$$\sigma_n(\mathbf{x}, a) = J(\hat{\mathbf{Q}}_{n-1})(\mathbf{x}, a) - H(\hat{\mathbf{Q}}_{n-1})(\mathbf{x}, a). \tag{A.9.18}$$

Thus, the synchronous Q-learning update (A.9.16) can be re-written as follows:

$$\hat{Q}_n(\mathbf{x}, a) = (1 - \delta_n)\hat{Q}_{n-1}(\mathbf{x}, a) + \delta_n \left[H(\hat{\mathbf{Q}}_{n-1})(\mathbf{x}, a) + \sigma_n(\mathbf{x}, a) \right], \tag{A.9.19}$$

which is of the same form as (A.9.2). Therefore, in order to establish the convergence of synchronous Q-learning, it only remains to show that the assumptions of Theorem A.9.1 are satisfied, and also that a fixed point of H (in other words, a vector $\mathbf{Q} \in \mathbb{R}^{\chi}$ satisfying $\mathbf{Q} = H(\mathbf{Q})$) exists. The latter property can be established immediately using results from previous chapters. Indeed, the discount optimality equations for MDPs defined on a finite or countable state space S (see [141], p. 146) imply that there exists a set of values $\{Q(\mathbf{x}, a)\}$ satisfying, for all $(\mathbf{x}, a) \in S \times A_{\mathbf{x}}$:

$$Q(\mathbf{x}, a) = \sum_{\mathbf{y} \in S} p(\mathbf{x}, a, \mathbf{y}) \left(r(\mathbf{x}, a, \mathbf{y}) + \phi \max_{a' \in A_{\mathbf{y}}} Q(\mathbf{y}, a') \right),$$

or equivalently $\mathbf{Q} = H(\mathbf{Q})$, as required. The first assumption in Theorem A.9.1, which states that the iterates $\hat{Q}_n(\mathbf{x}, a)$ are bounded for each $n \in \mathbb{N}_0$, can be verified using an argument provided by Gosavi [62] (p. 394). Specifically, for each $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ and $n \geq 0$, one can show:

$$|Q_n(\mathbf{x}, a)| \leq \max(r_{\max}, 0) \left(1 + \phi + \phi^2 + ... + \phi^n\right),$$

where $r_{\max} = \max_{\mathbf{x},\mathbf{y} \in \tilde{S}, a \in A_{\mathbf{x}}} r(\mathbf{x}, a, \mathbf{y})$. Hence, $K = \max(r_{\max}, 0)/(1-\phi)$ can be taken as the upper bound in (A.9.4). The first of the two 'noise term' properties in (A.9.5) can be established easily by taking the expectation of both sides in equation (A.9.18) and using the fact that $E[J(\hat{\mathbf{Q}}_n)(\mathbf{x}, a)] = H(\hat{\mathbf{Q}}_n)(\mathbf{x}, a)$, while the second property follows from the fact that the (conditional) second moment of $\sigma_n(\mathbf{x}, a)$ is bounded by a function of the square of $||\hat{\mathbf{Q}}_{n-1}||_\infty$, which itself is bounded due to (A.9.4). Finally, the non-expansiveness property (A.9.6) is proved by Gosavi [62] (p. 396) using a simple argument. For any two vectors $\mathbf{Q}, \mathbf{Q}' \in \mathbb{R}^\chi$ one can write, for $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$:

$$\begin{aligned}
\left|H(\mathbf{Q})(\mathbf{x}, a) - H(\mathbf{Q}')(\mathbf{x}, a)\right| &= \phi \left| \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) \left(\max_{a' \in A_{\mathbf{y}}} Q(\mathbf{y}, a') - \max_{a' \in A_{\mathbf{y}}} Q'(\mathbf{y}, a') \right) \right| \\
&< \left| \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) \left(\max_{a' \in A_{\mathbf{y}}} Q(\mathbf{y}, a') - \max_{a' \in A_{\mathbf{y}}} Q'(\mathbf{y}, a') \right) \right| \\
&\leq \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) \max_{(\mathbf{z}, a') \in \tilde{S} \times A_{\mathbf{z}}} \left|Q(\mathbf{z}, a') - Q'(\mathbf{z}, a')\right| \\
&= ||\mathbf{Q} - \mathbf{Q}'||_\infty \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) \\
&= ||\mathbf{Q} - \mathbf{Q}'||_\infty. \qquad \text{(A.9.20)}
\end{aligned}$$

Since (A.9.20) applies to the particular state-action pair $(\mathbf{x}, a)$ which maximises $|H(\mathbf{Q})(\mathbf{x}, a) - H(\mathbf{Q}')(\mathbf{x}, a)|$, the result $||H(\mathbf{Q}) - H(\mathbf{Q}')||_\infty \leq ||\mathbf{Q} - \mathbf{Q}'||_\infty$ follows. Thus, the convergence of synchronous Q-learning is established by Theorem A.9.1. Of course, the synchronous Q-learning algorithm applies to *discounted* problems, and it would be desirable to establish similar convergence properties for an RL algorithm which could be applied to an *average* reward problem.

A simple, if rather crude, approach to solving an average reward problem using an RL algorithm with guaranteed convergence properties would be to apply the Q-learning algorithm and choose a discount factor ϕ very close to 1. This strategy, referred to as the *vanishing discount approach*, is discussed by Arapostathis et. al. [5] (see also [62, 63]). Of course, the policy obtained using this

approach would be ϕ-discount optimal rather than average reward optimal, but with $\phi \approx 1$, one might suppose that a ϕ-discount optimal policy would be a close approximation of an average reward optimal policy. However, this approach is clearly somewhat unsatisfactory from a mathematical point of view, and therefore it will not be considered any further in this appendix.

As mentioned earlier, there exists an RL algorithm known as *relative Q-learning* which possesses known convergence properties and applies to average reward problems. This algorithm operates in a similar manner to Q-learning, except that the update rule (A.9.1) is replaced by:

$$\hat{Q}(\mathbf{x}_n, a_n) \leftarrow (1-\delta_n)\hat{Q}(\mathbf{x}_n, a_n) + \delta_n \left[r_n + \max_{a' \in A_{\mathbf{x}_{n+1}}} \hat{Q}(\mathbf{x}_{n+1}, a') - \max_{a' \in A_{\mathbf{0}}} \hat{Q}(\mathbf{0}, a') \right]. \tag{A.9.21}$$

A theoretical justification for the update rule (A.9.21) is given by the next result, which is proved by Bertsekas and Tsitsiklis [15] using an argument based on stochastic shortest paths.

*** Lemma A.9.2.** *Assume that the state space $\tilde{S}$ and action spaces $A_{\mathbf{x}}$ (for $\mathbf{x} \in \tilde{S}$) are finite. For each $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$, let $Q_0(\mathbf{x}, a) = 0$ and let $Q_n(\mathbf{x}, a)$ be defined for $n \in \mathbb{N}$ by:*

$$Q_n(\mathbf{x}, a) = \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) \left(r(\mathbf{x}, a, \mathbf{y}) + \max_{a' \in A_{\mathbf{y}}} Q_{n-1}(\mathbf{y}, a') - \max_{a' \in A_{\mathbf{0}}} Q_{n-1}(\mathbf{0}, a') \right).$$

Then, for each $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$, the following limit exists:

$$Q(\mathbf{x}, a) := \lim_{n \to \infty} Q_n(\mathbf{x}, a).$$

Moreover, let $g^* := \max_{a \in A_{\mathbf{0}}} Q(\mathbf{0}, a)$. Then the constant g^*, together with the set of values $\{Q(\mathbf{x}, a)\}$, satisfies the Q-factor version of the average reward optimality equations:

$$g^* + Q(\mathbf{x}, a) = r(\mathbf{x}, a) + \sum_{\mathbf{y} \in \tilde{S}} p(\mathbf{x}, a, \mathbf{y}) \max_{a' \in A_{\mathbf{y}}} Q(\mathbf{y}, a'). \tag{A.9.22}$$

Hence, $\max_{a \in A_{\mathbf{0}}} Q(\mathbf{0}, a)$ is the optimal expected long-run average reward.

Proof. Refer to the derivation given by Bertsekas and Tsitsiklis [15] (pp. 392-400).

Lemma A.9.2 is analogous to Theorem 3.7.4 (which was used to establish the convergence of the Relative Value Iteration Algorithm); essentially, the only difference is that it is given in terms of the Q-factors $Q(\mathbf{x}, a)$ which characterise an average reward optimal policy, as opposed to the relative values $h(\mathbf{x})$. Note that the update rule (A.9.21) is given in the context of an *asynchronous*

algorithm, in which $((\mathbf{x}_0, a_0), (\mathbf{x}_1, a_1), ...)$ is the sequence of state-action pairs 'visited' by the agent. In a *synchronous* version of the algorithm, the update rule (cf. (A.9.16)) becomes:

$$\hat{Q}_n(\mathbf{x}, a) = (1 - \delta_n)\hat{Q}_{n-1}(\mathbf{x}, a) + \delta_n \left[r(\mathbf{x}, a, \mathbf{Y}) + \max_{a' \in A_{\mathbf{Y}}} \hat{Q}_{n-1}(\mathbf{Y}, a') - \max_{a' \in A_{\mathbf{0}}} \hat{Q}_{n-1}(\mathbf{0}, a') \right]. \quad \text{(A.9.23)}$$

The convergence properties of synchronous relative Q-learning can be established using Theorem A.9.1. In this case, one may define the transformation $L : \mathbb{R}^{\mathcal{X}} \to \mathbb{R}^{\mathcal{X}}$ by:

$$L(\hat{\mathbf{Q}}_n)(\mathbf{x}, a) := r(\mathbf{x}, a, \mathbf{Y}) + \max_{a' \in A_{\mathbf{Y}}} \hat{Q}_n(\mathbf{Y}, a') - \max_{a' \in A_{\mathbf{0}}} \hat{Q}_n(\mathbf{0}, a'),$$

whereupon it follows that the rule (A.9.23) used by synchronous relative Q-learning is of the required form (A.9.2), with $H(\hat{\mathbf{Q}}_n)(\mathbf{x}, a) = E[L(\hat{\mathbf{Q}}_n)(\mathbf{x}, a)]$ for each $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ and the noise term $\sigma_n(\mathbf{x}, a)$ again given by the difference between $L(\hat{\mathbf{Q}}_n)(\mathbf{x}, a)$ and its expected value $H(\hat{\mathbf{Q}}_n)(\mathbf{x}, a)$. The existence of a fixed point $\mathbf{Q}$ satisfying $\mathbf{Q} = H(\mathbf{Q})$ follows by Lemma A.9.2.

The fact that conditions (A.9.5) and (A.9.6) are satisfied by the synchronous relative Q-learning algorithm can be verified using arguments very similar to those given for the (discounted) Q-learning algorithm in the previous pages. The boundedness property (A.9.4) is rather more difficult to establish, but it can be done; details can be found in Abounadi et. al. [1] (p. 692). One may then conclude, by Theorem A.9.1, that the synchronous relative Q-learning algorithm possesses convergence properties similar to those of its discounted-reward counterpart, synchronous Q-learning. However, the problem remains of showing that the modified versions of these algorithms which rely upon *asynchronous* updates also achieve convergence. For this purpose, it will be easiest to discuss various results from the literature which apply to specific RL algorithms, as opposed to relying upon a general result for stochastic approximation schemes similar to Theorem A.9.1.

First, convergence properties of the asynchronous version of (discounted) Q-learning will be discussed. The primary reference on this topic is Tsitsiklis [184]. It will be useful to re-formulate the update rule (A.9.1) used by Q-learning in order to ensure compatibility with the results of Tsitsiklis and others in the literature. An asynchronous RL algorithm may be viewed as a variant of a synchronous algorithm in which only *some* state-action pairs are simulated on each iteration, as opposed to *all* pairs. Specifically, let X_n denote the set of pairs $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ to be updated on the n^{th} iteration. In the synchronous case, one would simply have $X_n = \tilde{S} \times A_{\mathbf{x}}$ for all $n \in \mathbb{N}$; on the other hand, if updates are to be performed based on the simulated path of an agent exploring

the system (as described in Section 7.1), then the set X_n is a singleton for each $n \in \mathbb{N}$ and is determined by the history of states visited, actions chosen and events witnessed during the process. One may then express the update rule used by Q-learning in the following form:

$$\begin{aligned}\hat{Q}_n(\mathbf{x}, a) &= \hat{Q}_{n-1}(\mathbf{x}, a) \\ &+ I((\mathbf{x}, a) \in X_n)\, \delta(\nu_n(\mathbf{x}, a)) \left[r(\mathbf{x}, a, \mathbf{Y}) + \phi \max_{a' \in A_{\mathbf{Y}}} \hat{Q}_{n-1}(\mathbf{Y}, a') - \hat{Q}_{n-1}(\mathbf{x}, a) \right],\end{aligned} \tag{A.9.24}$$

where $\nu_n(\mathbf{x}, a) = \sum_{k=1}^{n} I((\mathbf{x}, a) \in X_k)$ is the total number of updates made to the pair $(\mathbf{x}, a)$ up to and including time n, and the learning parameter δ is now a deterministic function of $\nu_n(\mathbf{x}, a)$. In order to be consistent with the definition of δ_n in (7.2.14), one would define:

$$\delta(n) := \frac{T}{T + n - 1} \qquad \forall\, n \in \mathbb{N}, \tag{A.9.25}$$

where $T \in \mathbb{N}$ is arbitrary. The results in [184] depend upon a slight strengthening of the Robbins-Monro conditions (A.9.3). It is necessary to assume that, *for all* $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$:

$$\sum_{n=1}^{\infty} I((\mathbf{x}, a) \in X_n)\, \delta(\nu(\mathbf{x}, a)) = \infty, \quad \sum_{n=1}^{\infty} (I((\mathbf{x}, a) \in X_n)\, \delta(\nu(\mathbf{x}, a)))^2 < \infty. \tag{A.9.26}$$

In particular, the first condition in (A.9.26) implies that all state-action pairs are simulated infinitely often during the course of the process. Note that, in the case of the queueing systems considered in this thesis, an ϵ-greedy rule for selecting actions will ensure that this condition is satisfied, since under this rule there is always a probability ϵ of choosing an action at random; as a result, every state is accessible via an appropriate sequence of decisions and customer arrivals. Tsitsiklis [184] shows that, provided the three assumptions stated in Theorem A.9.1 hold and the conditions (A.9.26) are also satisfied, the asynchronous version of Q-learning emulates the synchronous version by converging to a set of values $Q(\mathbf{x}, a)$ satisfying the optimality equations (7.1.4).

In the case of relative Q-learning, the update rule becomes:

$$\begin{aligned}\hat{Q}_n(\mathbf{x}, a) &= \hat{Q}_{n-1}(\mathbf{x}, a) \\ &+ I((\mathbf{x}, a) \in X_n)\, \delta(\nu_n(\mathbf{x}, a)) \left[r(\mathbf{x}, a, \mathbf{Y}) + \max_{a' \in A_{\mathbf{Y}}} \hat{Q}_{n-1}(\mathbf{Y}, a') - \max_{a' \in A_{\mathbf{0}}} \hat{Q}_{n-1}(\mathbf{0}, a') - \hat{Q}_{n-1}(\mathbf{x}, a) \right].\end{aligned} \tag{A.9.27}$$

The convergence of relative Q-learning has been proved by Abounadi et. al. [1] (see also [19, 21]), subject to some mild additional assumptions on the learning parameter function δ which are satisfied

for commonly-used functions such as $\delta(n) = 1/n$ and $\delta(n) = \log(n)/n$. Also, the sets X_n must be defined in such a way that, for all $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$ and some constant $C > 0$:

$$\liminf_{n\to\infty} \frac{\nu_n(\mathbf{x}, a)}{n} \geq C \text{ a.s.} \tag{A.9.28}$$

As explained in [21], this condition ensures that all state-action pairs are updated 'comparably often'. In the queueing systems considered in this thesis, the condition (A.9.28) holds if one adopts an asynchronous updating style based on the simulated path of an agent (as discussed in Section 7.1) and allows actions to be selected using an ϵ-greedy rule. This can be seen using a heuristic argument, without resorting to rigorous detail. Indeed, let $((\mathbf{x}_0, a_0), (\mathbf{x}_1, a_1), ...)$ denote the sequence of state-action pairs 'visited' by the agent, where the actions a_n are chosen using an ϵ-greedy rule. The sets X_n are given by $X_n = \{(\mathbf{x}_n, a_n)\}$ for $n \in \mathbb{N}$. Let P_0 be defined as follows:

$$P_0 := \liminf_{n\to\infty} \frac{\sum_{a\in A_{\mathbf{0}}} \nu_n(\mathbf{0}, a)}{n}.$$

That is, P_0 is the limit infimum of the proportion of time spent by the agent in state $\mathbf{0}$ during the evolution of the algorithm. Clearly, P_0 must be strictly positive; indeed, it is stochastically bounded below by the stationary probability of no customers being present in an infinite-capacity $M/M/1$ queue with a demand rate λ equal to that of the system under consideration and a service rate given by $\mu_{\min} = \min_{i\in\{1,2,...,N\}} \mu_i$, which itself is always greater than zero.

Next, consider an arbitrary state-action pair $(\mathbf{y}, a) \in \tilde{S} \times A_{\mathbf{y}}$ and let $Y := \sum_{i=1}^{N} y_i$ be the total number of customers present under state $\mathbf{y}$. Suppose the system is in state $\mathbf{0}$ on some iteration n of the algorithm (obviously, there must be infinitely many such iterations if the algorithm is not terminated). Then, under the ϵ-greedy rule, there is a positive probability that 'exploratory' actions will be chosen on all of the next $Y + 1$ iterations, and there is also a positive probability that the simulated random transitions resulting from choosing these actions will enable the agent to reach state $\mathbf{y}$ after $n+Y$ iterations and then choose action $a \in A_{\mathbf{y}}$ on the $(n+Y+1)^{th}$ iteration. Indeed, assuming that the simulated random transitions are determined according to (7.2.1) and noting the fact that the largest possible number of decision options at any state is $N + 1$, then the probability of the aforementioned sequence of events occurring (given that $\mathbf{x}_n = \mathbf{0}$) must be at least $C(\mathbf{y}, a) := (\epsilon\lambda\Delta/(N + 1))^{Y+1}$, from which it follows that $\liminf_{n\to\infty} \nu_n(\mathbf{y}, a)/n \geq P_0 C(\mathbf{y}, a)$. Since this argument applies to any state-action pair $(\mathbf{x}, a) \in \tilde{S} \times A_{\mathbf{x}}$, one can then take $C :=$

$P_0 \min_{(\mathbf{x},a)\in\tilde{S}\times A_{\mathbf{x}}} C(\mathbf{x},a)$ in order to show that the required condition (A.9.28) holds.

The conclusion of this appendix is that, provided the algorithm is configured in an appropriate way (e.g. with an ϵ-greedy rule for selecting actions and some mild conditions on the learning parameters), the assumptions of Theorem 3.5 in [1] are satisfied and therefore relative Q-learning will converge to a set of values $Q(\mathbf{x},a)$ satisfying the average reward optimality equations (A.9.27) in both the synchronous and asynchronous cases. The convergence of Q-learning (for discounted problems) has also been established. However, as discussed earlier, similar convergence properties for the R-learning algorithm given on page 301 have not been proved in the literature. It will be worthwhile to end this appendix with an example comparing the performances of the R-learning and relative Q-learning algorithms in the case of a queueing system with 3 facilities.

Example A.9.3. ***(Comparing the R-learning and relative Q-learning algorithms)***

Consider a system with a demand rate $\lambda = 12$ and 3 facilities, with parameters as follows:

$$\begin{array}{llll} c_1 = 2, & \mu_1 = 2, & \beta_1 = 4, & \alpha_1 = 10, \\ c_2 = 2, & \mu_2 = 3, & \beta_2 = 3, & \alpha_2 = 6, \\ c_3 = 1, & \mu_3 = 4, & \beta_3 = 7, & \alpha_3 = 20. \end{array}$$

Using relative value iteration, it can be shown that the optimal value for the expected long-run average reward is approximately 106.05. Figure A.9.3 shows the results of an experiment in which the behaviour of the R-learning algorithm during its first 700,000 iterations was compared with that of the relative Q-learning algorithm. For clarity, the relative Q-learning algorithm operates in the same way as the R-learning algorithm presented on page 301, except that the update rule used for $\hat{Q}(\mathbf{x}_n, a_n)$ in step 3 (reverting to the notation used in Section 7.2) is given by:

$$\hat{Q}(\mathbf{x}_n,a_n) \leftarrow (1-\delta_n)\hat{Q}(\mathbf{x}_n,a_n) + \delta_n \left[r_n - \max_{a'\in A_{\mathbf{0}}} \hat{Q}(\mathbf{0},a') + \max_{a'\in A_{\mathbf{x}_{n+1}}} \hat{Q}(\mathbf{x}_{n+1},a') \right], \qquad \text{(A.9.29)}$$

so that $\max_{a'\in A_{\mathbf{0}}} \hat{Q}(\mathbf{0},a')$ simply replaces g_n. The estimates g_n are not used by relative Q-learning at all, and accordingly step 4 of R-learning is omitted. The estimate of the optimal average reward obtained after the algorithm terminates is then given by $\max_{a'\in A_{\mathbf{0}}} \hat{Q}(\mathbf{0},a')$.

As in previous examples, an ϵ-greedy rule was used to select actions and the learning parameters δ_n and (in the R-learning case) ζ_n were given by (7.2.14) and $1/(n+1)$ respectively.

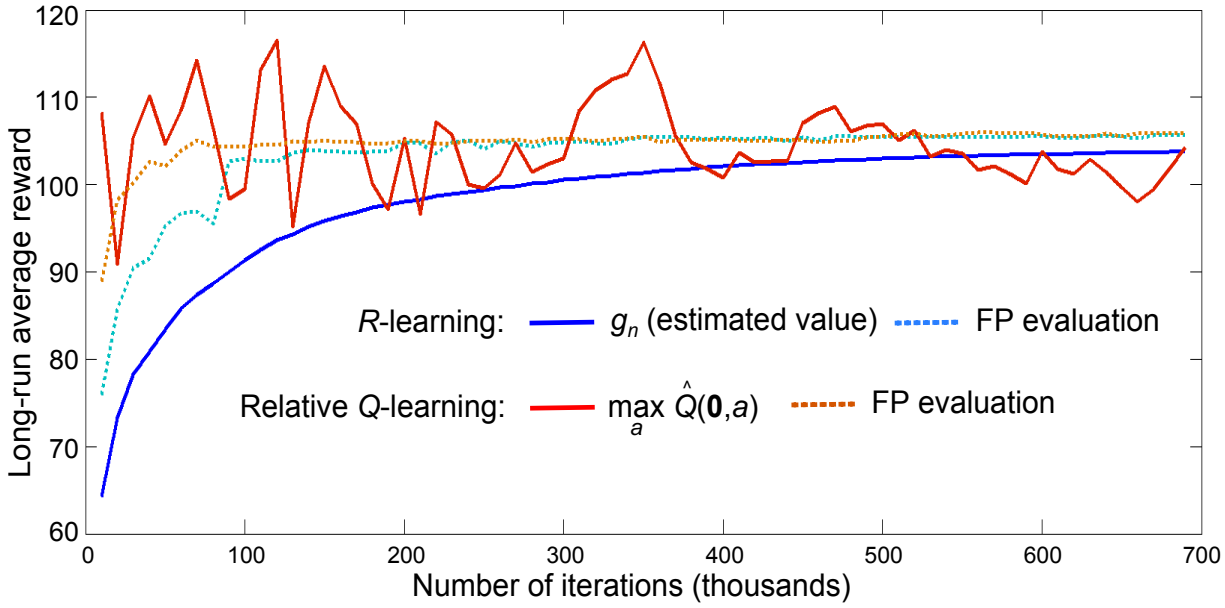


Figure A.4: Comparison between the performances of the R-learning and relative Q-learning algorithms.

One may inspect the evolution of the sequence (g_n), or $\max_{a' \in A_{\mathbf{0}}} \hat{Q}(\mathbf{0}, a')$ (depending on which algorithm is used) during the first 700,000 iterations in order to gain an idea of the relevant algorithm's stability during the early stages of its finite running time. Figure A.4 shows the progression of these values, and also shows (for each algorithm) the estimates of the long-run average reward obtained by performing Frozen Phase (FP) evaluations after every 10,000 iterations. As in Example 7.2.3, it is found that the estimates g_n used by R-learning converge towards the optimal value in a relatively smooth fashion. On the other hand, the values $\max_{a' \in A_{\mathbf{0}}} \hat{Q}(\mathbf{0}, a')$ obtained from relative Q-learning show much more erratic behaviour in the early stages of the algorithm. The FP estimates for the optimal average reward appear to converge rapidly towards $g^* \approx 106.05$ in both cases.

The fact that the values $\max_{a' \in A_{\mathbf{0}}} \hat{Q}(\mathbf{0}, a')$ used by relative Q-learning show more erratic behaviour than the sequence (g_n) used by R-learning does not appear to be unique to this particular example. Indeed, additional experiments performed with a wide range of different system parameters have shown that the general trends depicted in Figure A.4 are quite common. Moreover, when the number of facilities is increased, it appears that relative Q-learning may require a very large number of iterations before $\max_{a' \in A_{\mathbf{0}}} \hat{Q}(\mathbf{0}, a')$ begins to approach g^* (in the sense of being consistently within some tolerance limit of g^*); contrastingly, the 'smooth' convergence of (g_n) towards g^* seen using

R-learning appears to hold even in systems with higher-dimensional state spaces. ⊠

In summary, although the infinite-time convergence of $\max_{a' \in A_{\mathbf{0}}} \hat{Q}(\mathbf{0}, a')$ to g^* is theoretically guaranteed under relative Q-learning, numerical experiments show that this convergence may be considerably slower than the apparent convergence of g_n to g^* under R-learning.

References

[1] J. Abounadi, D.P. Bertsekas, and V.S. Borkar. Learning algorithms for Markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40:681–698, 2001.

[2] I. Adler and P. Naor. Social optimisation versus selfish optimisation in waiting lines, 1969.

[3] G. Allon, A. Bassamboo, and I. Gurvich. "We Will Be Right with You": Managing Customer Expectations with Vague Promises and Cheap Talk. *Operations Research*, 59(6):1382–1394, 2011.

[4] P.S. Ansell, K.D. Glazebrook, and C. Kirkbride. Generalised "join the shortest queue" policies for the dynamic routing of jobs to multi-class queues. *Journal of the Operational Research Society*, 54:379–389, 2003.

[5] A. Arapostathis, V.S. Borkar, E. Fernandez-Gaucherand, M.K. Ghosh, and S.I. Marcus. Discrete-time controlled Markov processes with average cost criterion: a survey. *SIAM Journal on Control and Optimization*, 31:282–344, 1993.

[6] N.T. Argon, L. Ding, K.D. Glazebrook, and S. Ziya. Dynamic routing of customers with general delay costs in a multiserver queuing system. *Probability in the Engineering and Informational Sciences*, 23(2):175–203, 2009.

[7] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Dover Publications Inc., 2003.

[8] K.R. Balachandran and M.E. Schaefer. Public and private optimization at a service facility with approximate information on congestion. *European Journal of Operational Research*, 4(3):195–202, 1980.

[9] C.E. Bell and S. Stidham. Individual Versus Social Optimization in the Allocation of Customers to Alternative Servers. *Management Science*, 29(7):831–839, 1983.

[10] R.E. Bellman. *Dynamic Programming*. Princeton University Press, New Jersey, 1957.

[11] V.E. Benes. *General Stochastic Processes in the Theory of Queues*. Addison-Wesley, 1963.

[12] D.P. Bertsekas. *Dynamic Programming and Stochastic Control.* Academic Press, New York, 1976.

[13] D.P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II.* Athena Scientific, Belmont, MA, 1995.

[14] D.P. Bertsekas and S. Shreve. *Stochastic Optimal Control: The Discrete Time Case.* Prentice-Hall, New Jersey, 1978.

[15] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[16] R.N. Bhattacharya and E.C. Waymire. *Stochastic Processes with Applications.* SIAM, Philadelphia, 2009.

[17] C.M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, 1995.

[18] V.S. Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29:291–294, 1997.

[19] V.S. Borkar. Asynchronous stochastic approximations. *SIAM Journal on Control and Optimization*, 36:840–851, 1998.

[20] V.S. Borkar. A learning algorithm for discrete-time stochastic control. *Probability in the Engineering and Informational Sciences*, 14:243–258, 2000.

[21] V.S. Borkar and S.P. Meyn. The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38:447–469, 2000.

[22] V.S. Borkar and K. Soumyanath. A new analog parallel scheme for fixed point computation, part I: Theory. *IEEE Transactions on Circuits and Systems I: Theory and Applications*, 44:351–355, 1997.

[23] J. Borwein and A.S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples.* Springer, 2000.

[24] L. Breiman. *Classics in Applied Mathematics: Probability Theory.* SIAM, Philadelphia, 1992.

[25] R. Cavazos-Cadena. Weak Conditions for the Existence of Optimal Stationary Policies in Average Markov Decision Chains with Unbounded Costs. *Kybernetika*, 25:145–156, 1989.

[26] R. Cavazos-Cadena. Recent Results on Conditions for the Existence of Average Optimal Stationary Policies. *Annals of Operations Research*, 28:3–28, 1991.

[27] R. Cavazos-Cadena. Solution to the Optimality Equation in a Class of Markov Decision Chains with the Average Cost Criterion. *Kybernetika*, 27:23–37, 1991.

[28] R. Cavazos-Cadena and L.I. Sennott. Comparing Recent Assumptions for the Existence of Optimal Stationary Policies. *Operations Research Letters*, 11:33–37, 1992.

[29] B. Chakraborty and E.E.M. Moodie. Statistical Reinforcement Learning. *Statistics for Biology and Health*, pages 31–52, 2013.

[30] B. Cheng and D.M. Titterington. Neural Networks: A Review from a Statistical Perspective. *Statistical Science*, 9:2–30, 1994.

[31] K.L. Chung. *Markov Chains with Stationary Transition Probabilities*. Springer, New York, 1967.

[32] E. Cinlar. *Introduction to Stochastic Processes*. Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[33] T.B. Crabill. Optimal Control of a Maintenance System with Variable Service Rates. *Operations Research*, 22:736–745, 1974.

[34] G.T. De Ghellinck. Les problemes de decisions sequentielles. *Cahiers de Centre de Recherche Operationelle*, pages 161–179, 1960.

[35] M. Delasay, B. Kolfal, and A. Ingolfsson. Maximizing throughput in finite-source parallel queue systems. *European Journal of Operational Research*, 217:554–559, 2012.

[36] F. d'Epenoux. A Probabilistic Production and Inventory Problem. *Management Science*, 10(1):98–108, 1963.

[37] C. Derman. *Finite State Markovian Decision Processes*. Academic Press, New York, 1970.

[38] C. Derman, G.J. Lieberman, and S.M. Ross. A stochastic sequential allocation model. *Operations Research*, 23:1120–1130, 1975.

[39] L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer, New York, 1996.

[40] A. Dixon, J. Appleby, R. Robertson, P. Burge, N. Devlin, and H. McGee. *Patient choice: how patients choose and how providers respond.* The King's Fund, London, 2010.

[41] E.B. Dynkin and A.A. Yushkevich. *Controlled Markov Processes.* Springer-Verlag, New York, 1979.

[42] J.E. Eckles. Optimum Maintenance with Incomplete Information. *Operations Research*, 16:1058–1067, 1968.

[43] A. Economou and S. Kanta. Optimal balking strategies and pricing for the single server Markovian queue with compartmented waiting space. *Queueing Systems*, 59:237–269, 2008.

[44] N.M. Edelson and D.K. Hildebrand. Congestion tolls for Poisson queuing processes. *Econometrica*, 43(1):81–92, 1975.

[45] M. El-Taha and S. Stidham. *Sample-Path Analysis of Queueing Systems.* Springer, 1999.

[46] A. Ephremides, P. Varaiya, and J. Walrand. A Simple Dynamic Routing Problem. *IEEE Transactions on Automatic Control*, 25(4):690–693, 1980.

[47] A. Federgruen, P.J. Schweitzer, and H.C. Tijms. Contraction mappings underlying undiscounted Markov decision processes. *Journal of Mathematical Analysis and Applications*, 65:711–730, 1977.

[48] A. Federgruen and H.C. Tijms. The Optimality Equation in Average Cost Denumerable State Semi-Markov Decision Problems, Recurrency Conditions and Algorithms. *Journal of Applied Probability*, 15:356–373, 1978.

[49] W. Feller. *An Introduction to Probability Theory and its Applications.* New York, 1950.

[50] J.A. Filar and O.J. Vrieze. *Competitive Markov Decision Processes.* Springer-Verlag, New York, 1997.

[51] D. Franke. Reinforcement learning in the El Farol model. *Journal of Economic Behavior & Organization*, 51:367–388, 2003.

[52] D.A. Freedman. Poisson Processes with Random Arrival Rate. *Annals of Mathematical Statistics*, 33(3):924–929, 1962.

[53] K.S. Fu. Learning control systems–Review and outlook. *IEEE Transactions on Automatic Control*, pages 210–221, 1970.

[54] G. Gilboa-Freedman and R. Hassin. The Price of Anarchy in the Markovian Single Server Queue. *IEEE Transactions on Automatic Control*, 59(2):455–459, 2014.

[55] J.C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society*, B41:148–177, 1979.

[56] P. Glasserman and D.D. Yao. Monotone optimal control of permutable GSMPs. *Mathematics of Operations Research*, 19(2):449–476, 1994.

[57] K.D. Glazebrook, D.J. Hodge, and C. Kirkbride. General notions of indexability for queueing control and asset management. *Annals of Applied Probability*, 21(3):876–907, 2011.

[58] K.D. Glazebrook, D.J. Hodge, and C. Kirkbride. Monotone policies and indexability for bidirectional restless bandits. *Advances in Applied Probability*, 45(1):51–85, 2013.

[59] K.D. Glazebrook, C. Kirkbride, and J. Ouenniche. Index Policies for the Admission Control and Routing of Impatient Customers to Heterogeneous Service Stations. *Operations Research*, 57(4):975–989, 2009.

[60] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[61] W.J. Gordon and G.F. Newell. Closed Queuing Systems with Exponential Servers. *Operations Research*, 15:254–265, 1967.

[62] A. Gosavi. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. Kluwer Academic Publishers, 2003.

[63] A. Gosavi. Reinforcement Learning: A Tutorial Survey and Recent Advances. *INFORMS Journal on Computing*, 21:178–192, 2009.

[64] A. Gosavi. A Tutorial for Reinforcement Learning. *Lecture notes*, 2011.

[65] W. Grassmann. The Convexity of the Mean Queue Size of the M/M/c Queue with Respect to the Traffic Intensity. *Journal of Applied Probability*, 20:916–919, 1983.

[66] G. Grimmett. *Probability and Random Processes*. Oxford University Press, USA, 2001.

[67] D. Gross and C. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, 1998.

[68] P. Guo and Q. Li. Strategic behavior and social optimization in partially-observable Markovian vacation queues. *Operations Research Letters*, 41:277–284, 2013.

[69] P. Guo and P. Zipkin. Analysis and Comparison of Queues with Different Levels of Delay Information. *Management Science*, 53:962–970, 2007.

[70] P. Guo and P. Zipkin. The Effects of Information on a Queue with Balking and Phase-type Service Times. *Naval Research of Logistics*, 55:406–411, 2008.

[71] X. Guo and O. Hernandez-Lerma. *Continuous-Time Markov Decision Processes: Theory and Applications*. Springer, New York, 2009.

[72] K. Gurney. *Introduction to Neural Networks*. CRC Press, 1997.

[73] A. Ha. Optimal Dynamic Scheduling Policy for a Make-To-Stock Production System. *Operations Research*, 45:42–53, 1997.

[74] B. Hajek. Optimal Control Of Two Interacting Service Stations. *IEEE Transactions on Automatic Control*, AC-29(6):491–499, 1984.

[75] R. Hassin. Information and Uncertainty in a Queueing System. *Probability in the Engineering and Informational Sciences*, 21:361–380, 2007.

[76] R. Hassin and M. Haviv. *To Queue or Not to Queue: Equilibrium Behavior in Queueing Systems*. Kluwer Academic Publishers, Norwell, MA, 2003.

[77] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

[78] M. Haviv and M. L. Puterman. Bias optimality in controlled queueing systems. *Journal of Applied Probability*, 35(1):136–150, 1998.

[79] M. Haviv and T. Roughgarden. The price of anarchy in an exponential multi-server. *Operations Research Letters*, 35(4):421–426, 2007.

[80] S.O. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

[81] O. Hernandez-Lerma. *Adaptive Markov Control Processes*. Springer-Verlag, New York, 1989.

[82] O. Hernandez-Lerma and J.B. Lasserre. *Further Topics on Discrete-time Markov Control Processes*. Springer-Verlag, New York, 1999.

[83] D.P. Heyman and M.J. Sobel. *Stochastic models in operations research, Vol. I*. McGraw-Hill, New York, 1982.

[84] H. Hinomoto. Sequential Control of Homogeneous Activities - Linear Programming of Semi Markovian Decisions. *Operations Research*, 19:1664–1674, 1971.

[85] D.J. Hodge and K.D. Glazebrook. On the asymptotic optimality of greedy index heuristics for multi-action restless bandits. *Advances in Applied Probability (to appear)*.

[86] P. Hokstad. The $G/M/m$ Queue with Finite Waiting Room. *Journal of Applied Probability*, 12(4):779–792, 1975.

[87] A. Hordijk and G. Koole. On the Optimality of the Generalised Shortest Queue Policy. *Probability in the Engineering and Informational Sciences*, 4(4):477–487, 1990.

[88] R. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

[89] J.M. Howie. *Real Analysis*. Springer, 2006.

[90] Q. Hu and W. Yue. *Markov Decision Processes with Their Applications*. Springer, New York, 2008.

[91] J.R. Jackson. Networks of Waiting Lines. *Operations Research*, 5:518–521, 1957.

[92] J.R. Jackson. Jobshop-like Queueing Systems. *Management Science*, 10:131–142, 1963.

[93] P.K. Johri. Optimality Of The Shortest Line Discipline With State-Dependent Service Times. *European Journal of Operational Research*, 41:157–161, 1990.

[94] L.C.M. Kallenberg. *Markov Decision Processes*. Lecture Notes, PhD Course, LNMB (Dutch Network on the Mathematics of Operations Research, 2007.

[95] S. Karlin. *A Finite Course in Stochastic Processes*. Academic, New York, 1969.

[96] J.H. Keisler. *Elementary calculus: an infinitesimal approach*. Prindle, Weber & Schmidt, 1986.

[97] F.P. Kelly. Networks of Queues. *Advances in Applied Probability*, 8:416–432, 1976.

[98] D.G. Kendall. Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. *Annals of Mathematical Statistics*, 24:338–354, 1953.

[99] B.G. Kingsman. Commodity Purchasing. *Operational Research Quarterly*, 20:59–79, 1969.

[100] L. Kleinrock. *Queueing Systems, Vol 1: Theory*. Wiley-Blackwell, 1975.

[101] V.A. Knight and P.R. Harper. Selfish routing in public services. *European Journal of Operational Research*, 230(1):122–132, 2013.

[102] V.A. Knight, J.E. Williams, and I. Reynolds. Modelling patient choice in healthcare systems: development and application of a discrete event simulation with agent-based decision making. *Journal of Simulation*, 6:92–102, 2012.

[103] N.C. Knudsen. Individual and Social Optimization in a Multiserver Queue with a General Cost-benefit Structure. *Econometrica*, 40(3):515–528, 1972.

[104] A.N. Kolmogorov. *Anfangsgrunde der Theorie der Markoffschen Ketten mit unendlichen vielen moglichen Zustanden*. University of Moscow, 1936.

[105] G. Koole. Structural Results for the Control of Queueing Systems Using Event-based Dynamic Programming. *Queueing Systems*, (30):323–339, 1998.

[106] G. Koole. Monotonicity in Markov reward and decision chains: Theory and applications. *Foundations and Trends in Stochastic Systems*, 1(1):1–76, 2006.

[107] G. Koole, P.D. Sparaggis, and D. Towsley. Minimizing response times and queue lengths in systems of parallel queues. *Journal of Applied Probability*, 36:1185–1193, 1999.

[108] K.R. Krishnan. Joining the right queue: A Markov decision-rule. *26th IEEE Conference on Decision and Control*, pages 1863–1868, 1987.

[109] K.R. Krishnan. Joining the right queue: A state-dependent decision rule. *IEEE Transactions on Automatic Control*, 35:104–108, 1990.

[110] H. Kushner. *Introduction to Stochastic Control*. Holt, Rineholt and Winston, New York, 1971.

[111] H.J. Kushner and D.S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer Verlag, New York, 1978.

[112] E. Kutschinski, T. Uthmann, and D. Polani. Learning competitive pricing strategies by multi-agent reinforcement learning. *Computing in economics and finance*, 27:2207–2218, 2003.

[113] A.M. Law. *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, 2006.

[114] H.L. Lee and A.M. Cohen. A Note on the Convexity of Performance Measures of M/M/c Queueing Systems. *Journal of Applied Probability*, 20:920–923, 1983.

[115] W.A. Leeman. Letter to the Editor - The Reduction of Queues Through the Use of Price. *Operations Research*, 12(5):783–785, 1964.

[116] M.E. Lewis and M.L. Puterman. A Note on Bias Optimality in Controlled Queueing Systems. *Journal of Applied Probability*, 37:300–305, 2000.

[117] M.E. Lewis and M.L. Puterman. *Bias Optimality,* in *The Handbook of Markov Decision Processes: Methods and Applications*. Kluwer Academic Publishers, 2001.

[118] W. Lin and P.R. Kumar. Optimal Control of a Queueing System with Two Heterogeneous Servers. *IEEE Transactions on Automatic Control*, 29:696–703, 1984.

[119] S.A. Lippman. Applying a new device in the optimisation of exponential queueing systems. *Operations Research*, 23(4):687–710, 1975.

[120] S.A. Lippman and S. Stidham. Individual Versus Social Optimization in Exponential Congestion Systems. *Operations Research*, 25(2):233–247, 1977.

[121] S.C. Littlechild. Optimal Arrival Rate in a Simple Queueing System. *International Journal of Production Research*, 12(3):391–397, 1974.

[122] J.B. MacQueen. A Modified Dynamic Programming Method for Markovian Decision Problems. *Journal of Mathematical Analysis and Applications*, 14:38–43, 1966.

[123] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22:159–195, 1996.

[124] A.S. Manne. Linear Programming and Sequential Decisions. *Management Science*, 6(3):259–267, 1960.

[125] J. Medhi. *Stochastic Models in Queueing Theory.* Academic Press, New York, 1991.

[126] B. Melamed and W. Whitt. On Arrivals That See Time Averages. *Operations Research*, 38:156–172, 1990.

[127] J.M. Mendel and R.W. McLaren. *Adaptive, Learning and Pattern Recognition Systems.* Academic Press, New York, 1970.

[128] R. Mendelssohn and M. Sobel. Capital Accumulation and the Optimisation of Renewable Models. *Journal of Economic Theory*, 23:243–260, 1980.

[129] R. Menich and R. Serfozo. Optimality of Shortest-Queue Routing for Dependent Service Stations. *Queueing Systems*, 9:403–418, 1991.

[130] H. Mine and S. Osaki. *Markov Decision Processes.* American Elsevier, New York, 1970.

[131] P. Naor. The Regulation of Queue Size by Levying Tolls. *Econometrica*, 37(1):15–24, 1969.

[132] U. Narayan Bhat. *An Introduction to Queueing Theory: Modelling and Analysis in Applications*. Birkhauser, 2008.

[133] P.S. Neelakanta and D. DeGroff. *Neural Network Modeling: Statistical Mechanics and Cybernetic Perspectives*. CRC Press, 1994.

[134] A. Ng. *Learning Theory (lecture notes)*. 2013.

[135] N. Nisan, T. Roughgarden, E. Tardos, and V.V. Vazirani (Editors). *Algorithmic Game Theory*. Cambridge University Press, New York, 2007.

[136] J.M. Norman and D.J. White. Control of Cash Reserves. *Operational Research Quarterly*, 16:309–328, 1965.

[137] A.R. Odoni. On Finding the Maximal Gain for Markov Decision Processes. *Operations Research*, 17:857–860, 1969.

[138] OpenOpt Home (last accessed September 2013). *http://openopt.org*.

[139] D. Pham and D. Karaboga. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer, 2011.

[140] W.B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley & Sons, New Jersey, 2007.

[141] M.L. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. Wiley & Sons, New York, 1994.

[142] S.I. Resnick. *Adventures in Stochastic Processes*. Birkhauser, Boston, 1992.

[143] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[144] S.M. Ross. *Applied Probability Models with Optimization Applications*. Holden-Day, San Francisco, 1970.

[145] S.M. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, New York, 1983.

[146] S.M. Ross. *Introduction to Probability Models, 9th edition.* Academic Press, 2007.

[147] T. Roughgarden. *Selfish routing and the price of anarchy.* MIT Press, 2005.

[148] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Pearson, 2013.

[149] V. Rykov. Monotone Control of Queueing Systems with Heterogeneous Servers. *Queueing Systems*, 37(4):391–403, 2001.

[150] V. Saario. Limiting Properties of the Discounted House-Selling Problem. *European Journal of Operational Research*, 20:206–210, 1985.

[151] T. Saaty. Burdens of queuing charges - comments on a letter by Leeman. *Operations Research*, 13(4):679–680, 1965.

[152] S.A.E. Sassen, H.C. Tijms, and R.D. Nobel. A heuristic rule for routing customers to parallel servers. *Statistica Neerlandica*, 51:107–121, 1997.

[153] M. Schal. On the Second Optimality Equation for Semi-Markov Decision Models. *Mathematics of Operations Research*, 17:470–486, 1992.

[154] A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards. *Proceedings of the Tenth Annual Conference on Machine Learning*, pages 298–305, 1993.

[155] L.I. Sennott. A new condition for the existence of optimum stationary policies in average cost Markov decision processes - unbounded cost case. *Proceedings of the 25th IEEE Conference on Decision and Control*, pages 1719–1721, 1986.

[156] L.I. Sennott. Average Cost Optimal Stationary Policies in Infinite State Markov Decision Processes with Unbounded Costs. *Operations Research*, 37(4):626–633, 1989.

[157] L.I. Sennott. Average cost semi-Markov decision processes and the control of queueing systems. *Probability in the Engineering and Informational Sciences*, 3(2):247–272, 1989.

[158] L.I. Sennott. The computation of average optimal policies in denumerable state Markov decision chains. *Advances in Applied Probability*, 29(1):114–137, 1997.

[159] L.I. Sennott. *Stochastic Dynamic Programming and the Control of Queueing Systems*. Wiley & Sons, New York, 1999.

[160] R. Serfozo. An equivalence between continuous and discrete time Markov decision processes. *Operations Research*, 27(3):616–620, 1979.

[161] R. Shone, V.A. Knight, and J.E. Williams. Comparisons between observable and unobservable $M/M/1$ queues with respect to optimal customer behavior. *European Journal of Operational Research*, 227:133–141, 2013.

[162] O. Sigaud and O. Buffet. *Markov Decision Processes in Artificial Intelligence*. Wiley & Sons, New York, 2013.

[163] R. Smallwood. Optimum Policy Regions for Markov Processes with Discounting. *Operations Research*, 14:658–669, 1966.

[164] M.J. Sobel. Making Short-Run Changes in Production When the Employment Level is Fixed. *Operations Research*, 18:35–51, 1970.

[165] D. Stengos and L.C. Thomas. The Blast Furnaces Problem. *European Journal of Operational Research*, 4:330–336, 1980.

[166] J. Stewart. *Multivariable calculus*. Cengage Learning, 2008.

[167] W.J. Stewart. *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*. Princeton University Press, 2009.

[168] S. Stidham. Socially and Individually Optimal Control of Arrivals to a GI/M/1 Queue. *Management Science*, 24(15):1598–1610, 1978.

[169] S. Stidham. Optimality Control of Admission to a Queueing System. *IEEE Transactions on Automatic Control*, 30:705–713, 1985.

[170] S. Stidham and R.R. Weber. A Survey of Markov Decision Models for Control of Networks of Queues. *Queueing Systems*, 13:291–314, 1993.

[171] D. Stirzaker. *Stochastic Processes and Models*. Oxford University Press, USA, 2005.

[172] R. Strauch. Negative Dynamic Programming. *Annals of Mathematical Statistics*, 37:871–890, 1966.

[173] A.L. Strehl and M.L. Littman. Online linear regression and its application to model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 2008.

[174] W. Sun, P. Guo, and N. Tian. Equilibrium Threshold Strategies in Observable Queueing Systems with Setup/Closedown Times. *Central European Journal of Operations Research*, 18:241–268, 2010.

[175] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[176] G.H. Symonds. Solution Method for a Class of Stochastic Scheduling Problems for the Production of a Single Commodity. *Operations Research*, 19:1459–1466, 1971.

[177] P. Tadepalli and D. Ok. Model-based average reward reinforcement learning. *Artificial Intelligence*, 100:177–224, 1998.

[178] L. Takacs. *Introduction to the Theory of Queues*. Oxford University Press, 1962.

[179] L.J. Thomas. Price Production Decisions with Random Demand. *Operations Research*, 22:513–518, 1974.

[180] P.J. Thomas. *Markov Decision Processes - Discrete Stochastic Dynamic Programming*. Wiley & Sons, New York, 1994.

[181] H. Thorisson. *Coupling, Stationarity and Regeneration*. Springer, New York, 2000.

[182] H.C. Tijms. *A First Course in Stochastic Models*. Wiley & Sons, Chichester, 2003.

[183] D.M. Topkis. *Supermodularity and Complementarity*. Princeton University Press, 1998.

[184] J.N. Tsitsiklis. Asynchronous Stochastic Approximation and Q-Learning. *Machine Learning*, 16:185–202, 1994.

[185] J.N. Tsitsiklis and B. Van Roy. Feature-Based Methods for Large Scale Dynamic Programming. *Machine Learning*, 22:59–94, 1996.

[186] E.A. van Doorn. Conditional PASTA. *Operations Research Letters*, 7(5):229–232, 1988.

[187] J. Walrand and P. Varaiya. Sojourn Times and the Overtaking Condition in Jacksonian Networks. *Advances in Applied Probability*, 12:1000–1018, 1980.

[188] M.D. Waltz and K.S. Fu. A heuristic approach to reinforcement learning control systems. *IEEE Transactions on Automatic Control*, 10:390–390, 1965.

[189] J. Wang and F. Zhang. Equilibrium analysis of the observable queues with balking and delayed repairs. *Applied Mathematics and Computation*, 218:2716–2729, 2011.

[190] P.D. Wasserman. *Advanced Methods in Neural Computing*. Van Nostrand Reinhold, 1993.

[191] C.J. Watkins. *Learning from Delayed Rewards*. PhD thesis, Kings College, Cambridge, 1989.

[192] R.R. Weber. On the Optimal Assignment of Customers to Parallel Servers. *Journal of Applied Probability*, 15:406–413, 1978.

[193] R.R. Weber and G. Weiss. On an index policy for restless bandits. *Journal of Applied Probability*, 27(3):637–648, 1990.

[194] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis of Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, 1974.

[195] D.J. White. Dynamic Programming, Markov Chains and the Method of Successive Approximations. *Journal of Mathematical Analysis and Applications*, 6:373–376, 1963.

[196] D.J. White. An Example of Loosely Coupled Stages in Dynamic Programming. *Management Science*, 19:739–746, 1973.

[197] D.J. White. *Markov Decision Processes*. Wiley & Sons, Chichester, 1993.

[198] P. Whittle. Restless Bandits: Activity Allocation in a Changing World. *Journal of Applied Probability*, 25:287–298, 1988.

[199] D. Widder. *The Laplace Transform*. Princeton University Press, New Jersey, 1941.

[200] B. Widrow and M.E. Hoff. Adaptive Switching Circuits. *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record*, 4:96–104, 1960.

[201] W. Winston. Optimality of the Shortest Line Discipline. *Journal of Applied Probability*, 14:181–189, 1977.

[202] R.W. Wolff. Poisson Arrivals See Time Averages. *Operations Research*, 30:223–231, 1982.

[203] U. Yechiali. On Optimal Balking Rules and Toll Charges in the GI/M/1 Queuing Process. *Operations Research*, 19(2):349–370, 1971.

[204] U. Yechiali. Customers' Optimal Joining Rules for the GI/M/s Queue. *Management Science*, 18(7):434–443, 1972.

[205] H. Zijm. The Optimality Equations in Multichain Denumerable State Markov Decision Proceses with the Average Cost Criterion: The Bounded Cost Case. *Statistics and Decisions*, (3):143–165, 1985.