Metaheuristics for Designing Efficient Routes & Schedules For Urban Transportation Networks

A thesis submitted in partial fulfilment

of the requirement for the degree of Doctor of Philosophy

Matthew P. John

August 2016

Cardiff University
School of Computer Science & Informatics
School of Mathematics

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 ..... (candidate) Date .....

Statement 1

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

Signed ..... (candidate) Date .....

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated, and the thesis has not been edited by a third party beyond what is permitted by Cardiff University's Policy on the Use of Third Party Editors by Research Degree Students. Other sources are acknowledged by explicit references. The views expressed are my own.

Signed ..... (candidate) Date .....

Statement 3

I hereby give consent for my thesis, if accepted, to be available online in the University's Open Access repository and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate) Date .....

To Mum, Dad & Alisha
for their patience and support.

Abstract

This thesis tackles the Urban Transit Network Design Problem (UTNDP) which involves determining an efficient set of routes and schedules for public transit networks. The UTNDP can be divided into five subproblems as identified by Ceder and Wilson [24]: i) network design, ii) frequency setting, iii) timetable development, iv) bus scheduling, and v) driver scheduling, with each problem requiring the output of the previous. In this thesis we focus on the first two stages, network design and frequency setting. We identify that evaluation is a major bottleneck for the network design problem and propose alternative approaches with the aim of decreasing the computation time. A multi-objective evolutionary algorithm (MOEA) for the network design problem is then presented that trades-off the passenger and operator costs. A passenger wishes to travel from their origin to destination in the shortest possible time, whereas the network operator must provide an adequate level of service whilst balancing the operational costs i.e. number of drivers and vehicles. The proposed MOEA combines a heuristically seeded population, using a novel construction algorithm, with several genetic operators to produce improved results compared with the state of the art from the literature. We provide an evaluation of the effectiveness of the genetic operators showing that improved performance, in terms of the number of dominating and nondominating solutions, is achieved as the size of the problem instance increases. Four surrogate models are proposed and an empirical evaluation is performed to assess the solution quality versus run trade-off in each case. It is found that surrogate models perform well on large problem instances producing improved Pareto sets compared with the original

algorithm due to the increased amount of evolution that is allowed to occur under fixed time limits. Finally we empirically evaluate three multi-objective approaches for the frequency setting problem utilising the route networks produced during our network design procedure. It is shown that a MOEA based on the NSGAII framework provides the best quality solutions due to the cost of evaluation when using a neighbourhood based approach such as multi-objective tabu search. Constraints on vehicle capacity and fleet size are then introduced. It is shown that such constraints vastly reduce the number of solutions from network design that can successfully undergo frequency setting. A discussion is then presented highlighting the limitations of conducting network design and frequency setting separately along with alternative approaches that could be used in the future. We conclude this thesis by summarising our findings and presenting topics for future works.

Acknowledgements

I would like to express my sincere gratitude to my supervisors Dr Christine Mumford and Dr Rhyd Lewis, for their continued support, guidance and endless encouragement throughout my PhD research. Without their constructive comments, insightful discussion and challenging questioning the results of this thesis would have differed greatly.

I would also like to thank my fellow PhD students for their support, feedback and company at the bar. For their continued support, endless feedback and being a sounding board for my sometimes wild ideas an explicit mention must be made for: Liam, Will, Chris, Lowri and Jonny.

For providing friendship and support I would like to thank Andrew, Matthew and Emlyn. From putting up with my antics on a night out to providing encouragement to complete this thesis, your friendship and support has been invaluable for more years than I care to remember.

Finally I would like to thank my parents and sister who have provided continued support throughout my studies. If it was not for their belief and determination to push me to achieve all that I can my path through life may have been vastly different.

Contents

List of Publications

Some of the work introduced in this thesis is based on the following publications.

- John, M. P., Mumford, C. L. and Lewis, R. An improved multi-objective algorithm for the urban transit routing problem. 14th European conference on Evolutionary Computation in Combinatorial Optimization. EvoCOP'14, Berlin, Heidelberg, Springer.

- Cooper, I. M., John, M. P., Lewis, R., Mumford, C. L and Olden, A. Optimising large scale public transport network design problems using mixed-mode parallel multi-objective evolutionary algorithms. 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, China. Evolutionary Computation (CEC), 2014 IEEE Congress on.

List of Figures

List of Tables

List of Algorithms

Acronyms

ACO Ant Colony Optimisation

APSP All Pairs Shortest Path

BCO Bee Colony Optimisation

CVRP Capacitated Vehicle Routing Problem

DARP Dial-a-Ride Problem

EA Evolutionary Algorithm

FSLC Fixed String Length Coding

GA Genetic Algorithm

GPU Graphics Processing Unit

GPU Graphics Processing Unit

MOEA Multi-objective Evolutionary Algorithm

MOFD Multi-objective First Descent

MOTS Multi-objective Tabu Search

NDP Network Design Problem

NSGAII Nondominated Sorting Genetic Algorithm II

PGA Parallel Genetic Algorithm

PLS Pareto Local Search

SA Simulated Annealing

SEAMO Simple Evolutionary Algorithm for Multi-objective Optimisation

TS Tabu Search

TSP Travelling Salesman Problem

TSP Travelling Salesman Problem

UTNDP Urban Transit Network Design Problem

VRP Vehicle Routing Problem

VRPB Vehicle Routing Problem with Backhauls

VRPPD Vehicle Routing Problem with Pick-up and Delivery

VRPTW Vehicle Routing Problem with Time Windows

VSLC Variable String Length Coding

Chapter 1

Introduction

1.1 Background

Public transportation is a key aspect of urban transport infrastructure providing economic, social and environmental benefits. In the UK five billion bus journeys are made each year, with one billion constituting travel to and from work, making up over two thirds of all public transport usage[1]. This compares well with the US where a total of ten billion journeys are made each year on public transport with 5 billion by bus[2]. Ridership in the US has increased by over a billion trips in each of the last two decades highlighting the importance of public transportation. With the increasing pressure upon global carbon dioxide emissions the role of public transport has been given a renewed focus. Five percent of CO_2 is produced by bus compared with sixty percent from passenger car trips for the UK road transport sector[1]. The design of efficient public transport systems is a matter of urgency to encourage commuters away from the private car and onto public transport.

An urban transportation system comprises several components ranging from private transport such as cars, motorcycles and bicycles, to commercial logistics that arrange for the collection and delivery of goods to council vehicles that maintain the infrastructure of our towns and cities.

[1] http://www.greenerjourneys.com/2012/06/did-you-know/
[2] http://www.apta.com/resources/statistics/Documents/FactBook/2015-APTA-Fact-Book.pdf

In most urban transportation systems the private car has become the dominant means of transportation for several reasons:

- It is difficult to predict the exact journey time when using public transport due to unforeseen delays (i.e. buses running late). [136]
- Journey time via car is generally shorter than a similar journey by public transport. [67]
- Ability to have increased control over the journey (i.e. departure and arrival times), physical and social environments. [67]
- Safety concerns when waiting for and using public transport (i.e. waiting at bus stops, travelling at night, transferring in unfamiliar places). [136, 67, 13]
- The need to make transfers when using public transport resulting in extra waiting time and a less comfortable journey. [13]
- A relatively small number of access points to public transport (e.g. bus stops) results in passengers having to walk a greater distance than their private car counterparts. [13]

Despite these obstacles the importance of public transportation cannot be overlooked. The use of public transportation could reduce many of the issues faced by the modern urban transport system. An increase in public transport patronage would reduce the number of vehicles on congested roads, reduce pollution and possibly lead to shorter travel times due to the reduction in congestion.

Public transportation in an urban area can be composed of several different modes including train, tram, bus and underground services. A co-ordinated approach using the different modes available can result in a reduction in many of the negative effects posed by the use of the private car. Well designed and co-ordinated operations could reduce the waiting time incurred when transferring, reducing the overall transportation time.

The usage of public transport can clearly result in a number of benefits. However, a reduction of funding for public transport in the UK has reduced patronage with many preferring to have a comfortable more convenient journey in their car.

Some modes of public transport are more flexible than others, and can adapt more easily to changes in the community they serve. For example, a set of bus routes are far easier to adapt or change compared with the fixed infrastructure of an underground system. As such, buses should form a core part of the urban transport system that must provide frequent services, minimise the waiting and in-vehicle travel time for passengers and avoid the need to transfer between vehicles. This must be balanced with the cost of operation for the network operator. In practice many passengers face infrequent or unreliable services that make no attempt at minimising the number of transfers required.

In the UK, public transport was deregulated by the 1985 Transport Act resulting in the vast majority of bus routes and schedules now being designed by bus companies (an exception being London where Transport for London decides the routes). Under this approach the bus companies will want to maximise their profits whilst maintaining an adequate service to the public.

This is not a consistent approach all over the world with local authorities deciding the routes and associated schedules in some areas. The local authority will place emphasis on the passenger requirements. Nevertheless, local transport policies and regulations must be observed [150] to ensure a satisfactory service is provided, otherwise it will not be used.

In the UK local authorities are held accountable by the local community who subsidise the operation of the routes and may therefore find it hard to justify under-utilised services. In some areas local authorities may subsidise a low demand service to maintain a minimum level of service on a non-profit route, such those in rural areas. On the other hand routes that do not receive public subsidies must be operated to provide a service that is commercially viable, meaning that some services may be limited in terms of route length or number of buses operating.

In the UK central government cuts due to the recession of 2008 have resulted in many local authorities cutting local bus routes, this is being felt especially in rural areas such as Wales. Recent BBC News articles have highlighted that from 2011 to 2014 nearly 100 subsidised bus routes have been discontinued by local authorities in Wales. Accounting for nearly one in seven bus routes across nineteen councils[3,4]. Effective bus service planning has been given an increased focus in the public domain in part to the cuts to bus services. The Welsh government has set aside £100,000 to analyse how bus companies and councils can work together to cut costs, plan routes and co-ordinate their timetables to meet passenger demand[5].

It is our belief that a software tool to help in the design of transport networks is warranted. Given the economic climate the need for cost efficient services is paramount. Furthermore, with the discontinuation of services by some bus companies, a software tool to help with the incorporation of these routes into other networks may be advantageous.

Compared with road and rail investment bus services are particularly undervalued. In June 2013 the UK Treasury announced that it would be investing £15.1 billion in the UK's strategic roads by 2021 to counter the effects of past under investment [47]. A further £12 billion has been set aside for maintaining the current network. Similar investment has also been earmarked for Britain's rail network with Network Rail, the body for maintaining the network, investing £38 billion in tracks and stations[6]. Investment on this scale is not seen for bus transportation with bus companies themselves having to provide the funds for new vehicles in most cases. An exception to this is the Green Bus Fund worth £87 million to provide 1200 low carbon buses in England[7].

Greener Journeys[8] is a national campaign with the backing from several major bus companies in the UK. Their aim is to encourage a shift from the private car to bus

[3]http://www.bbc.co.uk/news/uk-wales-26276565
[4]http://www.bbc.co.uk/news/uk-wales-26262972
[5]http://www.bbc.co.uk/news/uk-wales-27102732
[6]http://www.bbc.co.uk/news/business-26810369
[7]https://www.gov.uk/government/news/12-million-boost-for-greener-bus-journeys
[8]http://www.greenerjourneys.com

and coach travel. In 2012 Greener Journeys commissioned a report and found that bus commuters generate £45 billion of economic output with retail spend by bus users estimated at £21 billion [103]. Nearly a third of businesses would like to see improved transport links with other cities, with a quarter wishing to see improvements in public transport [147].

1.2 The Urban Transit Network Design Problem

The problem of designing an urban transportation network is commonly referred to as the Urban Transit Network Design Problem (UTNDP). The UTNDP can be divided into five main stages identified by Ceder and Wilson [24], these are: 1) network design, 2) frequency setting, 3) timetable development, 4) bus scheduling and 5) driver scheduling. These stages were grouped by Chakroborty [25] to produce the Urban Transit Routing Problem, consisting of solely network design, and the Urban Transit Scheduling Problem containing the remaining four stages.

The aim of the UTNDP is to construct an efficient set of routes and associated schedules that balance the cost to the operator whilst providing an adequate level service to the public. Passengers would like frequent services that provide direct travel between their origin and destination in an acceptable time. However, the network operator has constrained resources (i.e. vehicles, drivers) that must be used to provide services for the entire public transit network. Any attempt to design a public transport network must trade-off the cost to the operator with the service level provided to passengers.

Historically, route planners have used a combination of local knowledge and simple guidelines to produce route sets [114]. Several major studies (see [118, 154]) have identified the need for automated computer based tools for the design and evaluation of public transport networks. Automation is, however, highly complex and computationally expensive due to the large search space and multiple constraints involved in public transportation planning. To allow for the (re)design of transit networks there is a need

for alternative approaches that may only evaluate solutions approximately (referred to as surrogate models) but provide significant reductions in computation time.

The increase in congestion, pollution, greenhouse gas emissions and dwindling oil resources have placed emphasis on the use of public transport in recent years in an attempt to reduce the reliance of the private car. Achieving an increase in public transportation usage is clearly desirable but is also an extremely complex issue. However frequent and reliable cost-effective services are clearly key attributes.

Bagloee and Ceder [7] have recently pointed out that many public transit networks have not been reappraised from anywhere between 20 to 50 years. Land use patterns have changed considerably in this time period with the migration away from town centres into surrounding suburban areas; however public transport has been relatively slow to respond. It is our view that the development of automated tools to aid public transport networks is timely.

As mentioned previously the UTNDP concerns the design of a set of routes with a corresponding schedule to meet the demands of an urban transit system – this can be rail or bus, for example. The routes themselves must not be created independently, as they function as a collective allowing passengers to reach their destinations by using more than a single route via transfers. The design of routes must balance several conflicting objectives such as a minimum operating cost and a minimum passenger cost.

Network design attempts to determine an efficient set of routes (assume bus routes for simplicity) on an underlying transport network with a set of pre-determined pickup and drop off locations (e.g. bus stops). Note that determining the location of stops might also be part of the UTNDP although we assume that these have already been selected. The aim of this thesis is to redesign existing bus networks, to improve the efficiency for both operator and passenger. To achieve this, we must obey the problem constraints that are introduced in Chapter 2. In this thesis, we are concerned with redesigning the route network, whilst maintaining the existing infrastructure, in terms of of the physical location of bus stops and roads used.

Frequency setting sets the number of buses that will then operate on each of the routes created in the previous process while obeying constraints such as the fleet size and the capacity of buses operating the routes.

Timetable development assigns the departure and arrival times of buses at the stops specified by the routes and frequencies set earlier. Bus and driver schedules are then created to assign buses to the routes and the drivers to buses.

Clearly the optimum goal would be to optimise all aspects of the UTNDP simultaneously as they are heavily dependent upon one another. However, as stated by Chakroborty [25, p. 185], "given the complexity of the transit routing and scheduling problems, this is not felt to be possible, especially when the routing and scheduling problems are formulated realistically".

1.3 Main Contributions

The main contributions of this thesis are:

- A novel heuristic construction algorithm for creating route sets to be used during network design.
- A multi-objective approach to network design combining several mutation operators and crossover operators from the literature with an evolutionary framework that is able to outperform the state of the art.
- Three mutation operators for the network design problem.
- An assessment of alternative evaluation methods and graph topologies highlighting their suitability and issues for calculating the passenger travel time in the network design problem.
- An analysis of the effectiveness of the genetic operators used for network design.

- The presentation of four surrogate models for use with the network design problem and provide an empirical evaluation demonstrating their effectiveness and ability to produce improved approximate Pareto sets under constrained running times.
- The empirical evaluation of the effectiveness of three metaheuristic approaches to frequency setting.
- A discussion of the difficulties integrating network design and frequency setting together with alternative strategies for tackling the two problems.

1.4 Thesis Structure

Problem definition and formulation: this chapter first introduces concepts from optimisation and graph theory that are fundamental for the UTNDP. The subproblems of network design and frequency setting are then formulated and covered in greater detail. Problem instances used for the experimental work in this thesis are then defined and their methods of generation are described. Methods for evaluating the passenger cost are then discussed and empirically evaluated. It is found that a specialised graph, referred to as the transit network, is the least computationally expensive method for the identification of passenger transfers and the calculation of the passenger objective. Finally we cover the underlying concepts of computational complexity before providing a proof that the network design problem is $\mathcal{NP}$-Complete.

Literature review: a brief overview of vehicle routing problems is first presented and similarities with the UTNDP are highlighted. Approaches for tackling the network design problem are then discussed in three stages: 1) mathematical approaches, 2) heuristic approaches, and 3) metaheuristic approaches. Methods for solving the frequency setting problem are then reviewed before discussing the limitations of previous research for the network design and frequency setting problems. Commercial software for the design of public transportation networks is then presented.

An improved approach to network design: this chapter introduces our proposed approach to network design based upon our work in [88]. Firstly a novel heuristic procedure for generating initial solutions is detailed followed by an overview of the nondominated sorting genetic algorithm II (NSGAII) framework. The genetic operators used in this thesis are then introduced and described in detail. Our experimental approach and results are then presented. It is shown that the incorporation of our heuristic seeding and the combination of several mutation operators leads to improved results. An analysis of the performance of our genetic operators is then undertaken showing that improved performance, in terms of the number of dominating and nondominating solutions produced, is achieved as the problem size increases. A comparison is then made between our proposed method and the state of the art from the literature. Our proposed approach is able to achieve the best objective value from the passenger perspective in six out of the seven instances and for all the instances from the operator perspective.

Surrogate models for network design: to reduce the computation time of our network design algorithms we present several approximations for evaluating the passenger objective. These approximations, more formally defined as surrogate models, are empirically evaluated with each other and also with an alternative approach that uses a GPU (GPU)). We demonstrate that removing the reliance on the original objective function allows for improved results in terms of metric value. It is also shown that by incorporating more information into the mathematical model improved results are achieved.

Frequency setting: this chapter presents our approach to frequency setting. A heuristic approach is first described that is used to show that route sets from network design can be improved, from the passenger perspective, by augmenting the frequency on each route. An assignment model from the literature is then presented for calculating passenger travel paths and assigning passengers to routes. Three metaheuristic approaches are then discussed and evaluated empirically for optimising the frequencies on a route set. Constraints are then introduced upon the available fleet size and capacity of vehicles

where it is shown that route sets produced during network design are generally unsuitable for frequency setting. Finally the limitations of our approach to frequency setting and network design are summarised together with alternative strategies for tackling the two problems.

Conclusions & future work: provides a summary of our findings and several opportunities for future work on network design, frequency setting and UTNDP as a whole.

1.5 Summary

This chapter has introduced the background to the UTNDP and the need for algorithms to aid in the designing of public transportation networks. The five stages of the UTNDP were summarised with a focus on the network design and frequency setting stages – the focus of this thesis. Finally the main contributions of this thesis were listed before providing an overview of the structure of this thesis. In the next chapter we introduce the terminology used throughout this thesis and the formulations we use for the network design and frequency setting problems.

Chapter 2

Problem Definition and Formulation

In the previous chapter we noted that there are many stages involved in solving the UTNDP. In this chapter we define the two stages that are the focus of this thesis, the network design and frequency setting problems. We first provide background into the optimisation and graph theory that is used to provide the problem formulations used throughout this thesis. Network evaluation is then presented from the passenger perspective and shown to be a time consuming task that must be conducted on a specially constructed graph. A number of alternative approaches for the evaluation of the passenger objective are presented and shown to be unsuitable due to the need to include passenger transfers. This is followed by a discussion of the complexities of the UTNDP in general. Finally we introduce the theory of computational complexity and a proof that the network design problem is $\mathcal{NP}$-Complete.

2.1 Optimisation

In mathematics and computer science *optimisation* is the selection of the best element from a set of alternatives given some criteria on which to base the decision. Generally the determination of the best element will be made via a function that takes a feasible solution from the search space and maps it to the set of real numbers. More formally we can state an *optimisation problem* as follows, given a function $f : S \rightarrow \mathbb{R}$ from some set S to the set of real numbers, the aim is to determine an element $\bar{x} \in S$

such that $f(\bar{x}) \leq f(x)\ \forall x \in S$ in the case of a minimisation problem, or such that $f(\bar{x}) \geq f(x)\ \forall x \in S$ for a maximisation problem.

The domain S of f is called the solution space, the elements of S are referred to as candidate solutions and f is called the objective function. A solution that optimises the objective function subject to any problem specific constraints is called a globally optimal solution. [22, 33]

Problems that have a structure such as that described above are referred to as single-objective optimisation problems. Many problems can have several conflicting criteria that need to be optimised simultaneously such as the UTNDP where the cost to the network operator must be balanced with the level of service offered to passengers. Problems that share such a structure are referred to as *multi-objective optimisation problems* and can require the minimisation or maximisation of a set of objective functions or a combination of both. Without loss of generality a multi-objective minimisation problem can be formally defined as:

$$\text{minimise } \{f_1(x), f_2(x), \ldots, f_k(x)\} \text{ subject to } x \in S \tag{2.1}$$

with $k \geq 2$ objective functions $f_i : S \rightarrow \mathbb{R}$. Members of S are no longer solutions but decision vectors with the vector of objective values denoted by $f(x) = \{f_1(x), f_2(x), \ldots, f_k(x)\}^T$. The decision vector $x = \{x_1, x_2, \ldots, x_k\}^T$ belongs to S.

In multi-objective optimisation there is no longer the distinction of a globally optimal solution. The goal is to produce good compromises, often referred to as trade-offs, rather than a single solution.

Pareto optimality was put forward by Vilfredo Pareto [123] and refers to the situation where it is impossible to make an improvement to the value of one of the objectives in a solution without simultaneously degrading the value of one or more of the other objectives. Formally, a solution $x \in S$ is said to be Pareto optimal with respect to S if

and only if there is no $x' \in S$ for which $f(x') = \{f_1(x'), f_2(x'), \ldots, f_k(x')\}$ dominates $f(x) = \{f_1(x), f_2(x), \ldots, f_k(x)\}$. [33]

A vector $u = \{u_1, u_2, \ldots, u_k\}^T$ is said to dominate another vector $v = \{v_1, v_2, \ldots, v_k\}^T$, denoted by $u \prec v$, if and only if $\forall\, i \in \{1, \ldots, k\}$, $u_i \leq v_i \wedge \exists\, i \in \{1, \ldots, k\} : u_i < v_i$, in the case of a minimisation problem. Given a set of decision vectors P, the nondominated set or approximate *Pareto set* of decision vectors P' are those that are not dominated by any member of P. When P is the entire search space the resulting nondominated set P' is referred to as the *Pareto optimal set*. [43]

Evolutionary algorithms use the concept of Pareto optimality to find a set of solutions in a single optimisation run [33]. This is in contrast to classical approaches to multi-objective optimisation, such as weighted sum and ϵ-constraint, that produce a single optimised solution by modifying a single solution in each iteration [43]. The weighted sum approach optimises a weighted sum of objective functions allowing any single objective optimisation algorithm to be used. Each objective function $f_i(x)$ is assigned a weight $w_i > 0$ determined by the user and the goal is to minimise $\sum_{i=1}^{k} w_i f_i$. Although the weighted sum method is simple to implement it is highly sensitive to the assigned weights.

The ϵ-constraint method minimises one objective whilst assigning constraints to the worst value the remaining objectives are allowed to take. We therefore optimise $f_i(x)$ subject to the constraint that $f_j(x) \leq \epsilon_j$ for all $i \neq j$ where ϵ_j is the worst value $f_j(x)$ is allowed to take. A difficulty with this approach is the need to preselect which objective function will be optimised and the value of ϵ_j.

2.1.1 Multi-objective Performance Metrics

Evaluating the performance of multi-objective algorithms is key for enabling accurate comparisons of different algorithms. However, Deb and Jain [45] state that "the comparison of two nondominated set of solutions is not a straightforward matter, because of

the dimensionality involved in the sets" [45, p. 3] [94]. In multi-objective optimisation there are two distinct goals: 1) to find solutions as close as possible to the Pareto optimal set, and 2) discover solutions as diverse as possible in the nondominated set [43]. For measuring the achievement of these goals Deb [43] classified performance metrics into three categories: 1) metrics for convergence, 2) metrics for diversity, and 3) metrics for both convergence and diversity.

Metrics for convergence compute a measure of the closeness of a set of solutions from a known set of Pareto optimal solutions or equation where the relationship between decision variables is known. If no Pareto optimal set exists then a set of solutions that offer a good approximation are often used. As there are no published Pareto sets for any publicly available benchmark instances for the UTNDP we refrain from using convergence metrics. Metrics for diversity measure the range of solutions in a nondominated set and under Deb's categorisation also analyse the spread of solutions. Metrics for both convergence and diversity "can only provide a qualitative measure of convergence and the diversity, nevertheless they can be used along with one of the above metrics to get a better overall evaluation" [43, p. 332].

Schott [131] introduced a spacing metric, classified by Deb as a measure for diversity, to determine how evenly solutions in the Pareto set, $\mathcal{P}$, are distributed in the objective space. The Schott spacing metric, $\mathcal{D}$, is calculated as given by:

$$\mathcal{D} = \frac{1}{|\mathcal{P}| - 1} \sum_{i=1}^{|\mathcal{P}|} (\bar{d} - d_i)^2 \tag{2.2}$$

where $d_i = \min_j(|f_1(x_i) - f_1(x_j)| + |f_2(x_i) - f_2(x_j)|) \; \forall \; j \in \{1, \ldots, |\mathcal{P}|\}$ and $\bar{d}$ is the average of all d_i. A value of zero for this metric would indicate a perfect spacing of solutions i.e. all solutions in the Pareto set are equally spaced from one another.

A metric for convergence and diversity is the $\mathcal{S}$ metric or "hypervolume indicator" proposed by Zitzler and Thiele [156] and later improved in [155]. The $\mathcal{S}$ metric measures how much of the objective space is dominated by a given nondominated set,

$\mathcal{P}$. In the case of a minimisation problem we define a reference point $r = (r_1, \ldots, r_k)$ where k is the number of objectives. The reference point selected should be "worse" than the "good" solutions i.e. the reference point should be greater than the maximum objective value for each dimension. We can now formally define the $\mathcal{S}$ metric for a Pareto set $\mathcal{P}$ as:

$$\mathcal{S}(\mathcal{P}) = \text{Vol}(\bigcup_{u \in \mathcal{P}} E(u, r)) \tag{2.3}$$

where $E(u, r) = \{v \mid u_i \leq v_i \leq r_i,\ i = 1, \ldots, m\}$ [157].

The $\mathcal{S}$ metric gives the volume of the union of the polytopes $p_1, p_2, \ldots, p_t$ where each p_i is formed by the hyperplane perpendicular to the reference point and passing through the point $(f_1(x), f_2(x), \ldots, f_k(x))$. Figure 2.1 shows the hypervolume denoted by the hatched area between a reference point r and set of nondominated solutions.

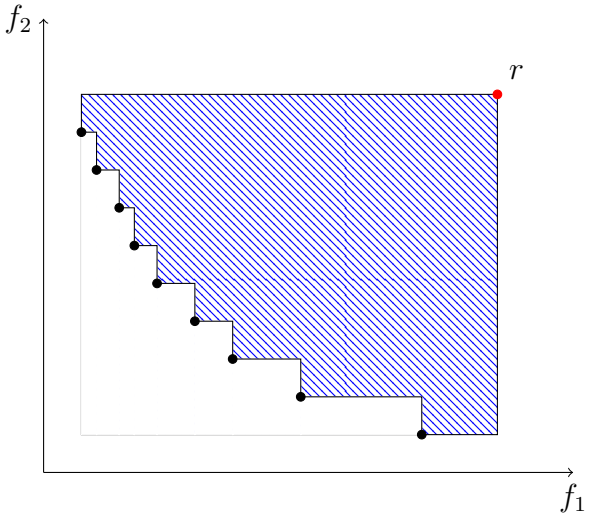


Figure 2.1: Illustration of the $\mathcal{S}$ metric for a bi-objective minimisation problem.

2.1.2 Combinatorial Optimisation

Combinatorial optimisation is a special type of optimisation that attempts to find an optimum solution from a finite set of solutions. The set of finite solutions usually has a

concise representation such as a graph or a permutation, but the number of solutions in the set can be very large. In fact the number of solutions grows exponentially as the size of the problem increases. This makes methods based around exhaustive search impractical resulting in the need for more efficient methods. [132]

2.2 Graph Theory

A graph $G = (V, E)$ consists of a finite nonempty set V of vertices and a set E, disjoint from V, of edges. An incidence function, ψ_G, is also associated with a graph G that associates each edge in G with an unordered pair of vertices in G. If e is an edge and u and v are vertices such that $\psi_G(e) = \{u, v\}$ then e is said to join u and v. The vertices u and v are said to be incident to the edge e and are known as the end points. A graph such as this is known as an undirected graph as shown in Figure 2.2.

$G = (V, E)$
$V = \{v_0, v_1, v_2, v_3\}$
$E = \{e_0, e_1, e_2, e_3, e_4, e_5\}$
$\psi(e_0) = \{v_0, v_1\}$ $\psi(e_1) = \{v_0, v_2\}$
$\psi(e_2) = \{v_0, v_3\}$ $\psi(e_3) = \{v_1, v_2\}$
$\psi(e_4) = \{v_1, v_3\}$ $\psi(e_5) = \{v_2, v_3\}$

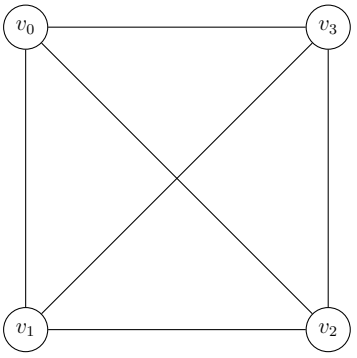


Figure 2.2: Undirected graph G

A pair of vertices u and v are adjacent if and only if there exists an edge whose ends are $\{u, v\}$. The set of all adjacent vertices to $v_i \in V$ is known as its neighbourhood, $\Gamma_G(v_i) = v_j \in V : \{v_i, v_j\} \in E$. The degree of a vertex v in a graph G, denoted by $\deg_G(v)$, is the number of edges incident with v. The minimum and maximum degrees of a graph G are denoted by $\delta(G)$ and $\Delta(G)$ respectively.

An edge with two identical ends is called a loop and an edge with distinct ends is called a link. A graph is "simple" if it is undirected, unweighted and there are no loops or

parallel edges. A graph is "complete" if every pair of vertices are adjacent. Removal of one or more vertices $V' \subseteq V$ and/or edges $E' \subseteq E$ from a graph G results in a subgraph $G'(V', E')$. We refer to a spanning subgraph as a graph $G' = (V, E')$, in other words a graph that contains all the vertices in G but a subset of its edges.

A walk in a graph G is a sequence $W = v_0 e_1 v_1 e_2 \dots e_n v_n$, whose terms are alternating vertices and edges of G, such that v_{i-1} and v_i are the end of e_i, $1 \leq i \leq n$. If a walk contains no repeated edges it is called a trail. A trail with no repeated vertices is called a path. A path that has identical vertices at each end ($v_0 = v_n$) is called a cycle.

A graph that contains a walk between every pair of vertices is said to be connected. A graph without a walk between every pair of vertices is said to be unconnected with a vertex having degree zero termed as "isolated", see Figure 2.3.

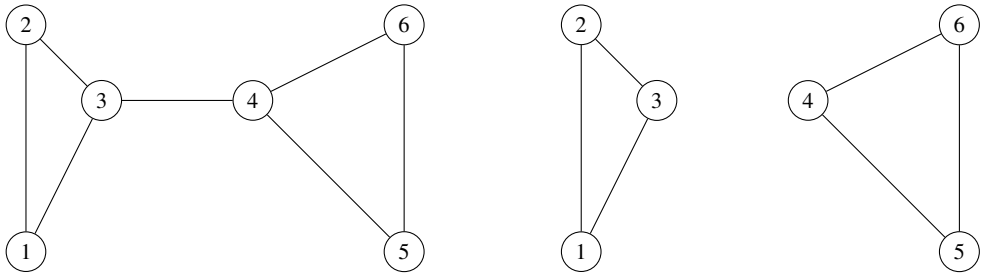


Figure 2.3: A connected (a) and unconnected graph (b)

In some situations a simple graph does not suffice. Consider modelling a road network. Here, not all streets are bi-directional; therefore we need to indicate in which way traffic is allowed to flow. To accomplish this each link must be assigned an orientation. A graph with such a structure is called a directed graph. A directed graph D consists of a finite non-empty set of vertices V and a set E, disjoint from V of arcs, together with an incidence function ψ_D that associates each arc of D to an ordered pair of vertices i.e. $(u, v) \neq (v, u)$. Let a be an arc and $\psi_D(a) = (u, v)$, the vertex u is said to be the tail of a and vertex v is its head. The vertices which dominate a vertex v are its in-neighbours, those which are dominated by the vertex its out-neighbours, denoted $N_D^-(v)$ and $N_D^+(v)$ respectively.

When modelling practical problems there is often the need to associate a cost with an edge. Consider again a road network, if we want to find the shortest path between two vertices each edge must have an associated cost to indicate the length of the road or travel time required to traverse the edge. With each edge $e \in E$ we associate a real number $w(e)$ called the weight. The graph G, together with these edge weights, is called a "weighted graph". A weight can be assigned to an edge or arc resulting in a weighted undirected graph or weighted directed graph respectively. An example weighted directed graph is given in Figure 2.4.

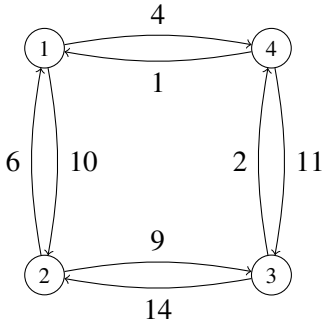


Figure 2.4: A directed weighted graph.

2.3 Network Design Problem

As mentioned previously network design is concerned with the determination of an efficient set of routes that balance the cost to both passenger and network operator. In a transport network adjacent vertices (e.g. bus stops) are connected with an edge. A route is the concatenation of a series of adjacent vertices that when combined together form a path. A route set is formed by collating a number of routes together.

In our problem formulation a route set should contain all the vertices present in the transport network but may not contain all of the edges i.e. the union of all the routes in the route set should form a spanning subgraph of the transport network. However, if several bus companies are servicing an area, the transport network may contain a subset

of vertices i.e. vertices may be removed from the transport network if they are already serviced by another network operator. The routes should also be connected (each route should share at least one vertex in common with at least one other route) allowing all vertices in the network to be reached by means of transfers if necessary.

Demand is the volume of passengers who wish to travel from one point in the network to another. Accurate demand figures are inherently hard to obtain and also vary over the time of day. It can be estimated in one of several ways: examining ticket sales, a survey of the local population, or undertaking a public and private vehicle analysis [9]. It is also difficult to estimate due to its variability and reliability on the current configuration of routes. It is important to note that demand is point to point and does not give a flow of passengers between an origin and destination. By this we mean the demand should not be assigned such that it shows the path through the network passengers take.

Point to point demand is an important criterion for the network design problem. If instead demand gave the passenger flow, i.e. the paths passengers take through the network, then the effects of a redesign would be limited. Network design requires the ability to change the path a passenger takes through the network dependent upon the current route set configuration.

Passenger flows have a greater importance when considering vehicle capacity. Given the path(s) passengers take through the network, passenger volume can be assigned to each leg of a route. Each route also has an associated frequency. We can therefore determine if the volume of passengers can be accommodated on the route given the capacity of vehicles operating upon it. As such, it is key that passenger flows are only assigned to the network after a redesign has taken place.

We adopt the convention of Chakroborty [25] and Yu et al. [152] similar to Fan [53] in his doctoral thesis to define an "efficient route set" as follows:

1. The entire demand is served, as such, all passengers can reach their destinations within an allowed number of transfers. It is assumed that a passenger will not use

public transport it they are required to make more than two transfers to reach their destination.

2. The majority of demand is served directly – passengers do not need to make a transfer to reach their destination.

3. The average travel time per passenger is low.

4. High network efficiency, i.e. prioritising the layout of those transit routes with the highest demand.

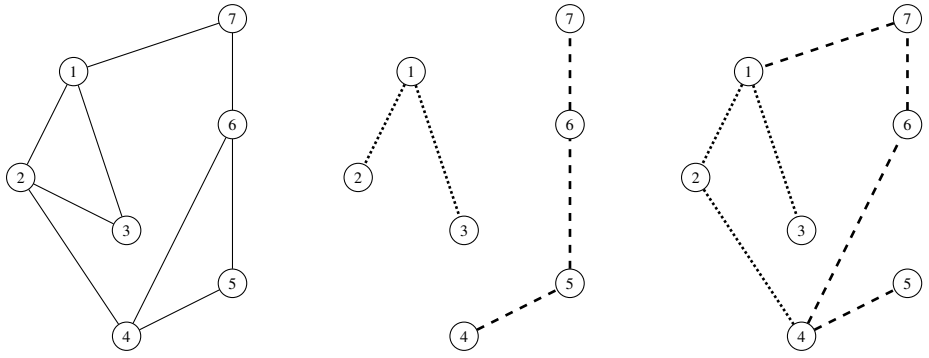


Figure 2.5: A sample representation of a transport network (a) with examples of an invalid (b) and valid (c) route network.

Our formulation of the network design problem which forms part of the UTNDP can be formally stated as follows. We are given a graph $G = (V, E, W)$ where $V = \{v_1, \ldots, v_n\}$ is a set of vertices, $E = \{e_1, \ldots, e_m\}$ is a set of edges and $W = \{w_1, \ldots, w_m\}$ a set of weights that define the cost to traverse edge e_i. We are also given a matrix $\mathbf{D}_{n \times n}$ where D_{v_i, v_j} gives the passenger demand between a pair of vertices v_i and v_j.

A route R_i is defined as a simple path (i.e. no loops/repeated vertices) through the graph G. Let $G_{R_i} = (V_{R_i}, E_{R_i})$ be the subgraph induced by a route R_i. A solution is defined as a set of overlapping routes $\mathcal{R} = \{R_1, \ldots, R_r\}$ where the number of routes, r, and

the minimum, m_1, and maximum, m_2, number of vertices in a route are specified by the user. In order for $\mathcal{R}$ to be valid the following conditions must hold:

$$\bigcup_{i=1}^{|\mathcal{R}|} V_{R_i} = V \tag{2.4}$$

$$m_1 \leq |V_{R_i}| \leq m_2 \; \forall R_i \in \mathcal{R} \tag{2.5}$$

$$\forall R_i \in \mathcal{R} \; \exists R_j \in \mathcal{R} \text{ s.t. } R_i \cap R_j \neq \varnothing \tag{2.6}$$

$$G_{\mathcal{R}} = (\bigcup_{i=1}^{|\mathcal{R}|} V_{R_i}, \bigcup_{i=1}^{|\mathcal{R}|} E_{R_i}) \text{ is connected} \tag{2.7}$$

$$|\mathcal{R}| = r \tag{2.8}$$

Constraint (2.4) ensures that all vertices in V are covered by at least one route in $\mathcal{R}$, while Constraint (2.5) specifies that each route should contain between m_1 and m_2 vertices (these values are based on considerations such as driver fatigue and the difficulty of maintaining the schedule [154]). Constraint (2.6) ensures that each route shares at least one vertex in common with another route, therefore allowing passenger transfers. Next, Constraint (2.7) specifies that at least one path should exist between each pair of vertices in $G_{\mathcal{R}}$. If Constraint (2.4) is satisfied then $G_{\mathcal{R}} = (V, \bigcup_{i=1}^{|\mathcal{R}|} E_{R_i})$. Finally, Constraint (2.8) ensures that the solution contains the correct number of routes r.

For this problem formulation, the following assumptions are made:

1. A vehicle travels back and forth along the same route, reversing its direction each time it reaches its terminal vertices. The physical road segments that are traversed

may differ between the outbound and inbound journeys. For example, a vehicle may travel a one-way street on the outbound journey requiring an alternative for the inbound journey. It is important to note that although physical road segments may differ, the stops visited on outbound and inbound journeys are identical.

2. A passenger's choice of routes between any two vertices is based only on shortest travel time (which includes transfer penalties) – see below.

When designing route sets in this way, the frequency of services is not yet determined. Thus two further assumptions also need to be made:

3. There will always be sufficient vehicles on each route $R_i \in \mathcal{R}$ to ensure that the demand between every pair of vertices on the route is satisfied.

4. The transfer penalty (representing the inconvenience of moving from one vehicle to another) is set as a fixed constant. In this study a fixed value of 5 minutes is used in line with previous works [26, 133, 55, 114, 88].

For our problem formulation both the *passenger cost* and *operator cost* are considered. In general, passengers would like to travel to their destination in the shortest possible time, whilst avoiding the inconvenience of making too many transfers. We denote the shortest time to travel between two vertices in the route set $\mathcal{R}$ as $\alpha_{v_i,v_j}(\mathcal{R})$. A path may include both transport edges and transfer edges (a transfer edge facilitates the changing from one vehicle to another with the associated time penalty). This is shown in Figure 2.6 with an example network expanded to include transfer vertices and transfer edges. The shortest path evaluation is thus completed on the transit network in Figure 2.6(b). The minimum journey time, $\alpha_{v_i,v_j}(\mathcal{R})$, from any given pair of vertices is thus made up of the sum of two components: in vehicle travel time and transfer penalty. We define the *passenger cost* for a route set $\mathcal{R}$ to be the mean journey time over all

passengers:

$$F_1(\mathcal{R}) = \frac{\sum_{i,j=1}^{n} D_{v_i,v_j} \alpha_{v_i,v_j}(\mathcal{R})}{\sum_{i,j=1}^{n} D_{v_i,v_j}} \tag{2.9}$$

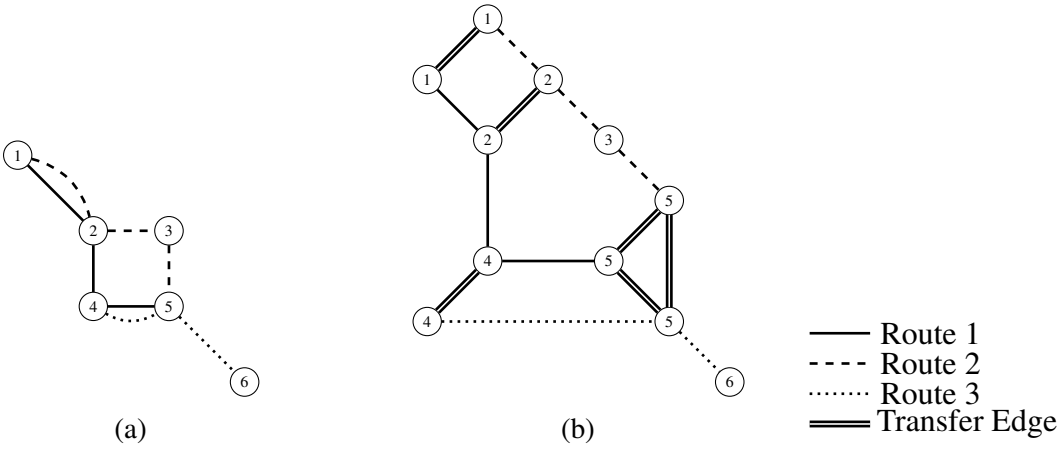


Figure 2.6: (a) Route network – road network with three routes overlaid (b) Transit network – network used for evaluation.

Operator costs, on the other hand, depend on many factors, such as the number of vehicles needed to maintain the required level of service, the daily distance travelled by the vehicles and the costs of employing sufficient drivers. Operators tend to favour shorter routes due to the ease of timetabling and scheduling, whereas passengers favour longer routes. It is obvious that longer routes include more vertices, providing an increase in the number of passengers who can reach their destination without making a transfer, thus resulting in a lower average passenger travel time. We use a simple proxy for operator costs: the sum of the costs (in time) for traversing all the routes in one direction, as suggested by Mumford [114]:

$$F_2(\mathcal{R}) = \sum_{\forall R_i \in \mathcal{R}} \sum_{\forall e_j \in E_{R_i}} w_j \tag{2.10}$$

Although F_2 may seem overly simplistic to accurately reflect the operator cost it, in fact, provides a good model. The sum of the route lengths can be used to determine

the number of vehicles required to maintain a given level of service, providing that we assume a constant frequency on all routes. In our case we assume a frequency of one vehicle every ten minutes giving an average waiting time of five minutes (i.e. the transfer penalty).

2.4 Frequency Setting Problem

As discussed previously frequency setting determines the number of vehicles required to operate the routes. The number of vehicles is determined by the vehicle *headway*, defined as the separation between two vehicles operating the same route. The frequency of the route is given by one over the headway. Frequency setting directly impacts the schedule to which vehicles operate on the route network. The schedule defines the arrival and departure times of vehicles (buses) at predefined pick-up and drop off points (e.g. bus stops). Frequency setting aims to minimise the passenger waiting time at each stop whilst ensuring that the operational cost is not excessive. The time that a passenger has to wait is composed of two components:

- Initial waiting time – this is the time from arrival at the point of origin to boarding the first vehicle.
- Transfer waiting time – the time a passenger must wait for a successful transfer to occur.

The passenger waiting time has to be balanced against the cost of operation and any operational constraints:

- Limited bus capacity – capacity on buses is limited and a fleet may contain buses of varying capacity.
- Limited fleet size – the operator has a finite number of vehicles available to operate the routes.

- Minimum level of service – given a route, a minimum frequency should be maintained at all times.
- Minimum and maximum stopping times – vehicles should not stop at predefined locations for a very short or very long period of time.
- Minimum and maximum transfer time – transfer times should not be excessively long.

The creation of effective frequencies is complex due to the non-deterministic nature of the real world. People arrive at bus stops stochastically and buses themselves can arrive early or late due to traffic conditions or the time required for boarding and embarking of passengers. When setting frequencies these issues need to be incorporated such that excessive queues do not form, thereby disadvantaging passengers (although some queues are expected at stops otherwise the service would not be cost effective). On the other hand, buses should not be empty whilst on their route as this is uneconomical to the operator. Well defined frequencies should minimise if not prevent these situations from occurring.

Transfers play a crucial role in any public transport system with research conducted into coordinated and uncoordinated operations (see [133]). In the coordinated transit system the frequency of vehicles and the time for which they remain at a stop is carefully controlled. Controlling the frequencies in this manner allows for vehicles to arrive at a transfer hub and passengers to board a waiting vehicle for the next leg of their journey, without incurring a significant waiting time.

Transport for London [61] guidelines for bus service planning suggest that where there is adequate demand, frequencies should allow for "turn-up-and-go" services. Services that run reliably every twelve minutes or less are considered to be "turn-up-and-go", with passengers consulting a timetable for services with a frequency less than this.

The frequency of a route dictates the number of vehicles and hence its capacity – the number of passengers that can be transported. A frequent service will produce a higher

capacity as more vehicles are traversing the route. Route frequency should be set to provide an adequate capacity at the busiest times and places [61]. Transport for London highlight general guidelines for service frequency. When the headway between vehicles is ten minutes or more passengers should be able to board the first vehicle that arrives at their stop. When the headway between vehicles is less than ten minutes passengers should not wait longer than ten minutes before being able to board a vehicle.

Operation of a reliable service in terms of maintaining a constant frequency is challenging due to delays faced when passengers embark and disembark a vehicle together with congestion and other road related issues. The impact of such delays can be minimised by allocating recovery time to routes at terminal vertices. The recovery time required is dependent upon the route length – longer routes may encounter more delays requiring a greater recovery time.

To allow passengers to easily remember timetables the headways of vehicles should be constant allowing for the construction of clock-face timetables. Clock-face timetables are timetables where a vehicle departs when the minute hand is at the same place every hour (i.e. 7:30, 8:30 etc.). [61]

In our definition of the frequency setting problem we are given a graph $G = (V, E)$ where $V = \{v_1, \ldots, v_n\}$ is a set of vertices and $E = \{e_1, \ldots, e_m\}$ is a set of edges. We are given:

- A weight for each edge, W_{e_i}, which defines the time it takes to traverse edge e_i;
- A matrix $\mathcal{D}_{n \times n}$ where $D_{i,j}$ gives the passenger demand between a pair of vertices v_i and v_j.

We are also given a route set $\mathcal{R}$ containing r routes. A frequency, f_i, must be defined for each $R_i \in \mathcal{R}$ and a solution is a set of frequencies $\mathcal{F} = \{f_1, \ldots, f_r\}$ such that the average travel time for passengers is minimised along with the number of vehicles required to operate the routes.

For this problem formulation, the following assumption is made:

1. There will always be sufficient vehicles on each route $R_i \in \mathcal{R}$ to ensure that the demand between every pair of vertices on the route is satisfied

For this problem both the *passenger cost* and *operator cost* are considered. Passengers do not travel from their origin to destination on a single path but may use several paths depending upon the frequency of service. The *optimal strategies* assignment model proposed by Spiess and Florian [135] is used to obtain the expected travel time, with frequencies considered, between an origin vertex, v_i, and a destination vertex, v_j, denoted $u_{i,j}$. We define the *passenger cost* for a route set $\mathcal{R}$ with frequencies $\mathcal{F}$ to be the mean journey time over all passengers:

$$F_3(\mathcal{R}, \mathcal{F}) = \frac{\sum_{i,j=1}^{n} D_{v_i,v_j} u_{i,j}}{\sum_{i,j=1}^{n} D_{v_i,v_j}} \tag{2.11}$$

For the operator cost we use the required fleet size. We assume that the network is operated by a homogeneous fleet of vehicles each with a maximum capacity $C_{\max}$. The operator cost is given by the number of vehicles that will be required to operate the network given the set of frequencies $\mathcal{F}$:

$$F_4(\mathcal{R}, \mathcal{F}) = \sum_{\forall f_i \in \mathcal{F}} 2 f_i \sum_{\forall e_j \in R_i} W_{e_j} \tag{2.12}$$

To reduce the size of the search space we discretise the set of allowed frequencies similar to Martínez et al. [107]. For our case we use the following set of frequencies $\theta = \{\frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}, \frac{1}{18}, \frac{1}{20}, \frac{1}{20}, \frac{1}{25}, \frac{1}{30}\}$ corresponding to twelve buses an hour to two buses an hour.

2.5 Problem Instances

For this work we use seven problem instances: Mandl's Swiss road network [105], four instances produced by Mumford [114] with two based upon real world cities and finally two instances based loosely upon the cities of Edinburgh and Nottingham in the UK. Table 2.1 provides an overview of the instances used along with their respective parameters for network design.

Instance	Number of Vertices and Edges	Number of Routes	Vertices /Route	Reference Point
Mandl	15 & 21	4 - 8	2 - 8	(60, 400)
Mumford0	30 & 90	12	2 - 15	(100, 2000)
Mumford1	70 & 210	15	10 - 30	(150, 6000)
Mumford2	110 & 385	56	10 - 22	(300, 20000)
Mumford3	127 & 425	60	12 - 25	(600, 30000)
Nottingham100	100 & 187	40	10 - 25	(100, 10000)
Edinburgh200	200 & 362	90	5 - 25	(600,20000)

Table 2.1: Problem Instances.

Each problem instance has two associated matrices: 1) a travel times matrix that defines the travel time in minutes between each pair of vertices, and 2) a demand matrix that gives the passenger demand between each pair of vertices over a twenty four hour period. All instances used in this thesis have symmetrical travel times and demand matrices. Furthermore all the networks are connected ensuring that any vertex can be reached from any other vertex. This is an important feature of all the instances used for the UTNDP enabling passengers to travel from an origin to any destination with the aid of transfers if necessary.

The instances used in this thesis are symmetrical due to their creation methods. The algorithms presented in later chapters are designed to work with both symmetrical and asymmetrical travel time matrices. In the real world we would expect travel times to not be symmetrical. For example, a bus may traverse a one-way street on part of its route. On the reverse leg an alternative path must be taken producing different travel times

between the stops depending on the direction of travel.

Mandl's instance has become the defacto benchmark instance for the UTNDP yet it contains only 15 vertices, which is very small given that real world public transport systems may have hundreds if not thousands of vertices (i.e. bus stops). Recently, Mumford [114] produced four benchmark instances making them publicly available to researchers. These instances have since been utilised in several other works [115, 92, 88, 37]. The Mumford instances are generated based upon user defined parameters governing the number of vertices, edges and an upper and lower bound on demand. The coordinates of vertices are generated from a uniform random distribution within an enclosing square region with a side length proportional to the square root of the number of vertices. Travel time between vertices is taken as the Euclidean distance. Demand between vertex pairs is generated at random between an upper and lower bound specified by the user. The selection of edges in the transit network is carefully controlled to ensure connectivity resembles a real road network. A minimum spanning tree (MST) is first constructed using Kruskal's algorithm [96]. Remaining edges are then selected by taking a vertex at random and adding the shortest unused edge out of this vertex to the network. This process is repeated until the required number of edges are obtained.

The Nottingham100 and Edinburgh200 instances were generated in a similar fashion to the Mumford instances. Vertices are selected at random from the location of physical bus stops identified by their latitude and longitude from the National Public Transport Data Repository [3]. Travel times between vertices are then obtained using the Google distance matrix API [2] taking the average of the travel time in both directions. Edges in the network are selected in a similar manner to the Mumford instances. Demand information was computed by relating the number of passengers to the population density in a 1×1Km square surrounding each bus stop (i.e. vertex).

The Nottingham100 and Edinburgh200 instances have a number advantages and disadvantages over the Mumford instances. The location of bus stops is based on the actual longitude and latitude of physical bus stops. However, the number of bus stops selected

is set by the designer and randomly taken from the total number available. This may result in a spatial layout of stops that is not reflective of the real city. The distance between stops is based on actual travel time using the Google distance matrix API but averaged in both directions resulting in a symmetric distance matrix. An asymmetric travel times matrix could have been produced given the availability of the data. The roads traversed are not limited so may include low bridges that may prevent bus travel. Demand information is related to population density whereas demand for the Mumford instances is randomly generated between upper and lower bounds. While this means demand may more closely relate to real world travel demands it is still an estimation made on the assumptions of the designer.

2.6 Network Evaluation

As outlined in Section 2.3, our formulation of the network design problem evaluates the performance of a route set from both the passenger and operator perspective. The calculation of the passenger objective involves performing an all pairs shortest path (APSP) algorithm on the transit network. This determines the passengers' paths through the network and any transfers that may have taken place. Accurate evaluation of a route set is an important component of any algorithm to solve the UTNDP as it has a direct impact on the quality of solutions produced. The evaluation metric should provide a good approximation of the operational cost for the network operator, and a measure of service from the passenger's perspective. In this section we first detail our method for obtaining the passenger objective upon the transit network. We then introduce network evaluation using the route network and compare this to the evaluation upon the transit network in terms of runtime performance. An alternative method suggested by Mandl [105] is also presented and is compared to the transit network evaluation. Modifications to the transit network are then proposed to reduce the size of the graph to improve the efficiency. Finally we use a graphics processing unit (GPU) [91] implementation of Floyd's [60] all pairs shortest path algorithm and analyse the

speed-up that can be achieved over the serial implementation when using the transit network.

2.6.1 Evaluation using the Transit Network

In Section 2.3 we introduced the transport network which defines the public transport infrastructure i.e. the bus stops and roads used. A route network can be created by taking the vertices (i.e. bus stops) in the transport network and the edges used in a route set, as shown in Figure 2.7. Our method for determining the passenger cost (Equation (2.9)) requires the calculation of the cost of the shortest path between each pair of vertices in the transport network, given a set of available routes. As such, we must identify any transfers between routes that are necessary, and then penalise them to reflect the inconvenience caused to the passenger. To do this, the route network is expanded to include transfer points as shown in Figure 2.8(b).

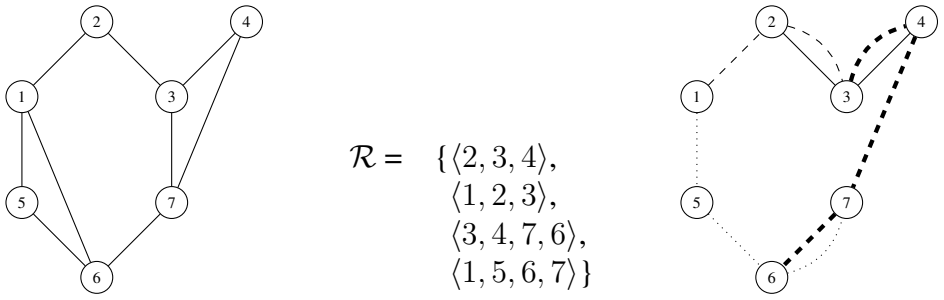
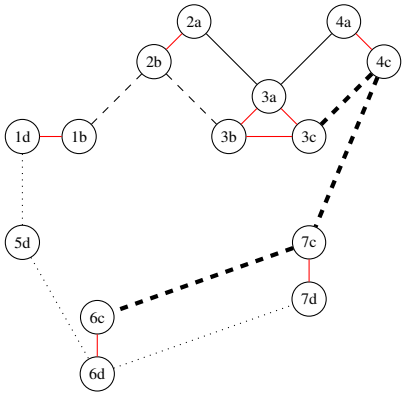


Figure 2.7: An example transport network (a), route set (b) and resultant route network (c).

Recall from earlier that in the expanded network, referred to as the *transit* network, duplicated vertices in a route set become transfer points. For example, the route set given in Figure 2.8(a) shows that vertex three is serviced by three routes labelled a, b, and c. This results in three vertices, 3a, 3b and 3c, being inserted into the transit network to represent the routes. A cycle is then formed between the transfer vertices with each edge having a cost equal to the transfer penalty.

$\mathcal{R} = \{\langle 2,3,4\rangle, \text{ a}$
$\langle 1,2,3\rangle, \text{ b}$
$\langle 3,4,7,6\rangle, \text{ c}$
$\langle 1,5,6,7\rangle\} \text{ d}$

(a)



(b)

Figure 2.8: An example route set with corresponding route label, (a), expanded into its transit network (b) with transfer edges given in red.

Calculation of the shortest path between all vertices is commonly referred to as the all pairs shortest path (APSP) problem. We use Floyd's [60] algorithm given in Algorithm 2.1 to calculate the APSP upon the transit network. As the APSP calculation is conducted on the transit network we must then 'collapse' the result i.e. removing the transfer vertices. This reflects the transport network, allowing passenger travel times to be extracted.

A vertex u in the transport network may map to one or many vertices, $\{u_1, \ldots, u_i\}$, in the transit network. This depends upon how many routes are incident to the vertex. To calculate the shortest path between any two vertices, u and v, in the transport network each occurrence u_j of u and v_k of v in the transit network are taken. The path with the lowest cost is selected as per our assumption given in Section 2.3. If more than one path shares the same cost then the path with the least number of transfers is taken, as we assume a passenger will always select the path that minimises their inconvenience.

The APSP is an expensive operation with computational complexity O(n^3) using Floyd's [60] algorithm. This is a significant drawback as the evaluation procedure is applied frequently during an optimisation. Although the complexity of Floyd's [60] algorithm is polynomial, the evaluation becomes expensive, even for relatively small problem instances following the creation of the larger transit network. This is shown in Table 2.2

where the average, minimum and maximum transit network sizes for a Pareto set of solutions is given for each instance. The duplication of vertices can lead to the production of a transit network considerably larger than the underlying transport network. Intuitively, reducing the number of duplicate vertices would reduce the size of the graph used for evaluation resulting in a reduction in computation time.

Alternative methods for calculating the APSP include applying Dijkstra's [48] algorithm to each vertex. Although Dijkstra's algorithm has a computational complexity of O($|E| + |V|log|V|$) when using Fibonacci heaps [62]. We found empirically that Floyd's algorithm outperformed Dijkstra's for calculating the APSP. As such Floyd's APSP algorithm was used for calculating the passenger cost on the transit network.

Instance	$\|\mathcal{P}\|$	$\|V\|$	μ	min	max
Mandl	97	15	29	20	41
Mumford0	99	30	86	41	179
Mumford1	93	70	231	150	450
Mumford2	101	110	715	560	1199
Mumford3	103	127	930	720	1500
Nottingham100	72	100	539	400	888
Edinburgh200	97	200	799	450	1744

Table 2.2: Mean, minimum and maximum number of vertices in a transit network for a Pareto set, $\mathcal{P}$, compared with the number of vertices in the original transport network.

1: **for** $\forall\ k \in V$ **do**
2: **for** $\forall\ u \in V$ **do**
3: **for** $\forall\ v \in V$ **do**
4: $c = m[u,k] + m[k,v]$
5: **if** $c < m[u,v]$ **then**
6: $m[u, v] = c$

Algorithm 2.1: Floyd's [60] algorithm for the all pairs shortest path.

2.6.2 Evaluation using the Route Network

As mentioned previously, reducing the size of the graph required for evaluation will offer a reduction in computation time. In this section we achieve this by using the smaller route network, as opposed to the larger transit network. The removal of transfer vertices, i.e. those vertices common to more than one route, from the transit network requires an alternative approach for calculating the shortest path through the network, in order to check for and penalise any transfers made. Our proposed approach is to check the vertices traversed along each path and map each to a given route. This mapping will enable any changes of routes (i.e. transfers) to be penalised and added to the cost of the path.

Using this approach the running time required to evaluate a set of route sets was compared to evaluation using the transit network. Table 2.3 shows the average time required for an evaluation is always greater using the route network. This can be attributed to the requirement of maintaining a list of vertices traversed by a path to allow for transfers to be identified and penalised. A further disadvantage to this method is that it does not always produce the correct shortest path. For example, the route network given in Figure 2.9 shows two routes and the associated cost of traversing each edge. We wish to find the shortest path from vertex five to eleven and will assume a transfer penalty of five minutes. There are two possible paths that can be taken, $\langle 5, 2, 1, 3, 11 \rangle$ with a cost of 18, or $\langle 5, 3, 11 \rangle$ with a cost of 19, with the second path incurring a transfer penalty. Using Dijkstra's [48] algorithm, the path $\langle 5, 3, 11 \rangle$ will be selected as the shortest path. However, this is the incorrect shortest path when transfer penalties are taken into account. The reason for this error is that Dijkstra's [48] algorithm and other shortest path algorithms, do not consider vertex costs i.e. vertex three from the path $\langle 5, 3, 11 \rangle$ has an additional cost of five, as a transfer is required.

This issue becomes clearer when we reflect upon how Dijkstra's [48] algorithm works, shown in Algorithm 2.2. At each iteration we explore the lowest cost unvisited vertex. If the algorithm is attempting to find a path between vertices five and eleven, it will

1: **for** $\forall\ v \in V$ **do**
2: dist[v] = ∞
3: dist[source] = 0
4: $Q = V$
5: **while** $Q \neq \varnothing$ **do**
6: u = min(Q)
7: $Q = Q - u$
8: **for** $\forall\ v \in N_D^+(u)$ **do**
9: t = dist[u] + dist(u, v)
10: **if** $t <$ dist[v] **then**
11: dist[v] = t

Algorithm 2.2: Dijkstra's [48] algorithm for the single source shortest path.

	Transit Network		Route Network	
Instance	Mean	Std. Deviation	Mean	Std. Deviation
Mandl	**0.09****	0.07	1.63	0.27
Mumford0	**2.13****	3.45	13.50	4.30
Mumford1	**30.00****	36.12	192.15	26.53
Mumford2	**651.93****	542.23	1395.25	93.74
Mumford3	**1466.87****	1274.47	2298.70	101.68
Nottingham100	**286.79****	242.80	839.00	68.89
Edinburgh200	**1340.92****	2041.10	9962.23	1529.57

Table 2.3: Evaluation time (milliseconds) using the route and transit networks. Asterisks indicate statistical significance according to a Related-Samples Wilcoxon Signed Rank test at the $p < 0.01$ level.

start from vertex five and visit vertex seven, two and then three – assuming that vertices are added to a priority queue that utilises a stable sorting algorithm. As the algorithm has now reached vertex three with a cost of four, no other path will be considered resulting in the 'true' shortest path never being found. A possible resolution to this problem would be to explore the adjacent vertices of the current vertex to determine if any transfers are required. In our example, this would mean examining vertices one, four and eleven when we reach vertex three. This approach introduces a further complication as a vertex could have multiple costs, depending upon whether or not a transfer is required. Alternatively every path between the origin and destination could be extracted and the cost of each path calculated. For large networks this approach

would incur a significant overhead. As such it is not given any more consideration here.

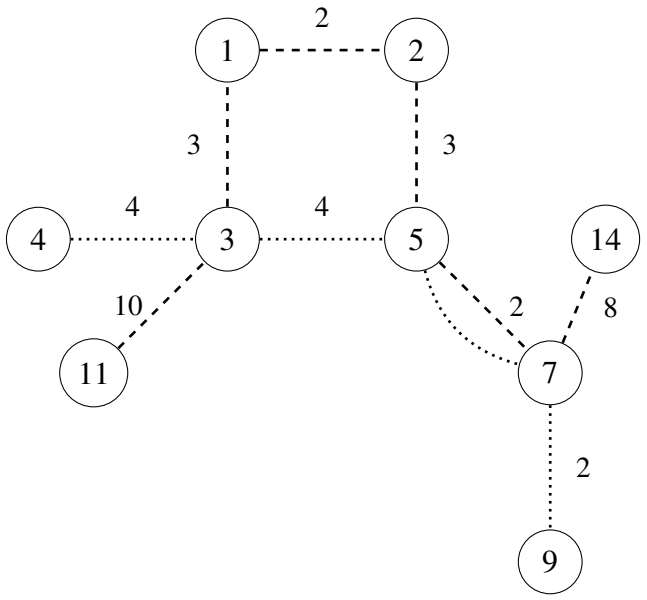


Figure 2.9: Route network extract with two routes.

2.6.3 Mandl's Evaluation Method

Mandl [105] proposed splitting the transit network into two different components: the transfer vertices and the remaining vertices. More formally, given an undirected graph $G = (V, E)$ the transfer vertices, $I \subset V$, are extracted from the network. The remaining vertices $Q = V - I$ are those that do not enable transfers i.e. they are contained in only one route. The closest transfer vertices $\forall\ v \in Q$ are then found using the following procedure: 1) if the vertex is a terminal of a route then there will only be one transfer vertex to find, 2) find a transfer vertex in each direction.

Floyd's [60] algorithm is suggested to find the shortest path costs between all pairs of vertices in I. All that remains is to calculate the distance between the vertices in Q and their closest transfer vertex. The distance between vertices on the same route must be computed independently as they do not require transfers. Once all the necessary pre-processing has been completed the shortest path cost between two vertices, u and

v, can be found by looking at the distance between u's closest transfer vertices and v's closest transfer vertices. For example, say u and v have the following transfer vertices:

$$u\text{: } a' \; c' \qquad v\text{: } b'.$$

The paths that must be considered to determine the shortest path, include those given below and the distance from u to v directly if there exists a route that contains both u and v.

$$u \text{ - } a' \text{ - } b' \text{ - } v$$
$$u \text{ - } c' \text{ - } b' \text{ - } v$$

The execution time of Mandl's proposed method was compared with that using the transit network. In our experiments a set of route sets were evaluated for each benchmark problem instance with the mean and standard deviation of the execution time computed for each set. This is given in Table 2.4.

	Transit Network		Mandl's Method	
Instance	Mean	Std. Deviation	Mean	Std. Deviation
Mandl	0.24	0.18	**0.19****	0.176
Mumford0	**2.39****	3.74	2.44	4.15
Mumford1	**30.75**	36.56	31.96	42.40
Mumford2	**659.12****	544.44	666.21	551.56
Mumford3	**1467.35****	1274.26	1480.69	1288.15
Nottingham100	**288.12**	241.84	289.47	250.42
Edinburgh200	1352.05	2049.63	**1320.62**	2064.99

Table 2.4: Comparison between evaluation time (milliseconds) using the transit network and Mandl's [105] proposed evaluation method. Asterisks indicate statistical significance according to a Related-Samples Wilcoxon Signed Rank test at the $p < 0.01$ level.

Table 2.4 shows that Mandl's evaluation method compares well with our method of evaluation using the transit network. Both methods are reasonably close in terms of mean runtime. However, evaluation using the transit network is able to achieve a lower

mean runtime in five out of the seven instances. Further investigation is required to determine if there are specific route set structures that provide more efficient evaluation using the transit network or Mandl's method. For example, if a route set contains a very small number of transfer vertices Mandl's method may have a far larger running time compared with evaluation on the transit network. This is due to the need to compute the APSP on the transfer vertices, then separate calculations on the non-transfer vertices to produce the travel times between all pairs of vertices i.e. shortest path calculations between vertices on the same route.

2.6.4 Evaluation Removing Overlapping Transfer Vertices

The previous sections have demonstrated the need to conduct the evaluation using the transit network, to allow for transfers to be identified accurately. In some situations it may occur that several routes share a set of adjacent transfer vertices. Figure 2.10 considers a city with a central boulevard that is served by many routes. The figure shows three routes, with each route containing the vertices four, five and six, leading to the duplication of these vertices in the transit network. In situations such as this, the routes share a large proportion of the stops on the boulevard. In terms of the shortest path calculation the passenger can choose to transfer to a different vehicle at any one of these vertices. In this case only the terminal transfer vertices of the overlapping section need be duplicated in the transit network, as shown in Figure 2.11.

Removal of the overlapping transfer vertices reduces the size of the transit network that is required for evaluation. However, there exists an issue with the representation. When considering a passenger whose path originates at vertex nine, but must terminate at vertex one, the representation in Figure 2.11 shows that the cost of the transfers will be disregarded when using current shortest path algorithms, even though one transfer penalty is incurred. This occurs because the passenger travels from vertex nine to six, where a transfer can be made, or can continue to vertex five remaining on the same route. At vertex five the passenger can transfer to any route with no associated transfer cost.

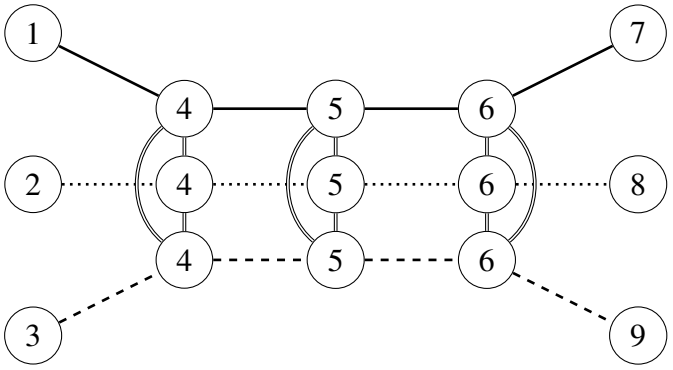


Figure 2.10: An example transit network with three routes, each with three identical transfer vertices.

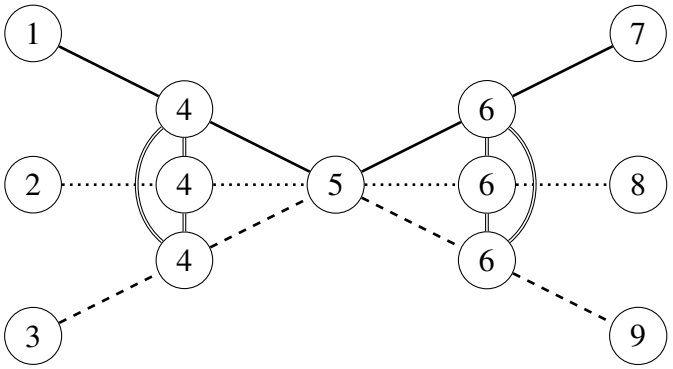


Figure 2.11: Removal of the intermediate overlapping transfer vertices in the transit network.

In this scenario the shortest path algorithm will select the least cost path, allowing the passenger to travel directly with no transfers between vertices nine and one. However, visually we can see that this cannot be achieved.

To resolve this problem, the passenger's path would need to be extracted each time a transfer vertex is encountered. The path can then be examined and a transfer penalty applied if necessary. In our example, we would extract the path $\langle 9, 6, 5, 4 \rangle$ revealing that vertices nine and four are not on the same route, allowing a transfer penalty to be imposed.

Similarities can be drawn between this representation and when using the route network. It was shown that the extraction of paths is expensive compared with the calculation of the shortest paths using Floyd's [60] algorithm on the transit network. As such, this

method is not given further consideration.

2.6.5 Evaluation Using the GPU

The previous sections have emphasised the need to conduct the APSP evaluation using the transit network. We have also shown that the use of alternative graph structures is not appropriate for the problem, given current shortest path algorithms. To reduce the running time required for evaluation when using the transit network, we have therefore implemented a GPU [91] version of Floyd's [60] algorithm. The algorithm was implemented using CUDA and the C standard library.

Katz and Kider's [91] method seeks to enable the processing of large adjacency matrices beyond the size of the GPU, as well as addressing the inherent dependency problems present in parallel implementation. The algorithm is comprised of a three phase process. However, before this can begin, the adjacency matrix must be decomposed into smaller blocks which fit on the GPU, and map optimally to the underlying block architecture of the hardware.

In phase one, *doubly dependent* blocks are identified. These run in a diagonal direction, from top left to bottom right, through the adjacency matrix, and each of these blocks has the APSP algorithm performed on them by the GPU. Moving into phase two, each of these blocks is passed back to the GPU in turn, but this time with its associated *singly dependent* blocks; the blocks in the same row and the blocks in the same column. Again, all of these have the APSP calculation performed on them. Finally, in phase three, all dependencies are now satisfied. This allows all blocks of the adjacency matrix to be passed back to the GPU, as space on the device allows, for a final third computation of the APSP algorithm.

Once this has been completed, the three phase process has the effect of calculating the APSP for the original adjacency matrix. Whilst it may appear that far more calculation is required than in a serial CPU implementation, clock cycles are cheap on the GPU.

Therefore, although additional computation needs to occur, due to the highly parallel nature of the GPU, it is still likely to execute considerably faster. This is more concisely described graphically and, as such, readers are referred to the original publication [91].

Table 2.5 provides a comparison between the time taken to evaluate a set of route sets using both the CPU and the GPU. It is shown that the GPU offers a seven-time speed up for the larger problem instances. We note that similar speed-ups are achieved for Mumford2, Mumford3 and Edinburgh, although they vary in the size of the transport network. This reflects that it is not the underlying transport network size that influences the running time, but the amount of route overlap in a route set.

	CPU		GPU		
Instance	Mean	Std. Deviation	Mean	Std. Deviation	Speed-up
Mandl	**0.26****	1.37	1.37	10.59	0.71
Mumford0	3.28	5.49	**1.77****	8.81	2.19
Mumford1	35.14	42.50	**7.62****	13.24	4.41
Mumford2	794.28	668.29	**107.83****	87.91	7.22
Mumford3	1802.77	1579.52	**230.85****	192.20	7.59
Nottingham100	348.13	303.05	**51.18****	45.29	6.60
Edinburgh200	1655.94	2.53	**211.16****	0.31	7.16

Table 2.5: Runtime comparison (milliseconds) using the CPU and GPU for evaluating the all pairs shortest path upon the transit network. Asterisks indicate statistical significance according to a Related-Samples Wilcoxon Signed Rank test at the $p < 0.01$ level.

For Mandl's instance, evaluation using the GPU causes an increase in running time. This is due to the small size of the instance and the extra overhead incurred when using a GPU i.e. copying memory to and from the device. Table 2.6 provides average running times for varying randomly generated graph sizes. It is shown that the GPU does not provide benefit until graph sizes reach approximately 200 vertices.

Vertices	CPU	GPU	Speed-up
100	**0.002**	0.005	0.40
200	0.014	**0.008**	1.75
300	0.049	**0.013**	3.77
400	0.115	**0.023**	5.00
500	0.226	**0.041**	5.51
1000	1.919	**0.243**	7.90
2000	19.741	**1.766**	11.18
3000	54.411	**5.935**	9.17
4000	125.663	**14.077**	8.93
5000	244.407	**26.902**	9.09

Table 2.6: Runtime comparison (seconds) for the all pairs shortest path on randomly generated graphs using the CPU and GPU.

2.7 Complexities of the Urban Transit Network Design Problem

We have shown that the UTNDP is a heavily constrained and difficult problem to solve. Several researchers have identified the complexities of the UTNDP with the majority of these being identified by Fan [53] in his PhD thesis:

1. The problem is $\mathcal{NP}$-Hard.

2. Accurate data for the design of route sets is difficult to obtain. In a real world situation the demand varies over an entire day and is based upon the current configuration of the route set, making the problem extremely complex [49]. Furthermore many services record only the entry point to the network, i.e. bus passes record where a person boarded a vehicle but not where they disembarked. This data is also commercially sensitive if the network is run for profit.

3. Modelling the arrival time of passengers at bus stops is challenging. Passengers arrival stochastically making accurate evaluation computationally expensive.

4. There is a high level of dependency between routes. Therefore routes cannot be

evaluated in isolation as the performance of one route is dependent upon the other routes in the set.

5. The UTNDP is inherently a multi-objective optimisation problem that must consider the minimisation of the operator costs, minimisation of the number of transfers and minimisation of the average passenger travel time. These objectives conflict as reducing the number of transfers for passengers will result in an increase in the operator cost.

2.8 $\mathcal{NP}$-Completeness of the Network Design Problem

Before introducing our proof that the network design problem is $\mathcal{NP}$-Complete we provide a summary of the theory of computational complexity defining the necessary terms. The foundations for computational complexity theory were laid down by Cook [35] and Karp [90] who put forward a framework for measuring the complexity of a problem. To define computational complexity theory in more detail we must first define some common terms.

A *decision problem* is a question or task that can have either a "yes" or "no" answer [36]. We shall define a *problem* more formally to be a question with a common structure that consists of a series of input variables requiring the output of an answer. An example of a question would be "Given an undirected graph G, does G have a Hamiltonian circuit?".

A *problem instance* or *instance* is a specific question of a given problem. Each instance consists of an exact specification of the input variables e.g. the number of vertices and a list of edges with the associated costs for finding a Hamiltonian circuit in a given graph.

An *algorithm* is a set of step by step instructions for solving a problem. An algorithm is said to solve a problem if it can be applied to any instance of the problem and prove or disprove the required question i.e. an algorithm does not solve the Hamiltonian circuit

problem unless it always proves the existence or otherwise of a Hamiltonian circuit. [68]

Measuring the computational complexity of an algorithm could be achieved using the running time on a given computer. The main issue with this approach is that computers are built upon varying architectures, processors and amounts of memory. Comparing the running time of an identical problem instance on several different platforms does not allow us to make a standardised comparison. Furthermore, some problem instances may be solved easily whilst others require significantly more steps to produce a solution.

Computational complexity theory overcomes these problems by utilising a framework that calculates complexity based upon how the time requirements for a problem increase as the size of the underlying problem instance increases. More accurately, an algorithm is expressed by a time complexity function that expresses its time requirements for each possible input length. It is generally the case that for each possible input length the worst case time required by the algorithm to solve the problem instance is given. This is referred to as Big O notation. [68]

To state the running time of an algorithm we use $O(.)$ notation that provides the number of elementary operations that must be performed for a given input length n. O notation provides an upper bound bounding function $f(n) = O(g(n))$, if a constant $c > 0$ exists, such that $f(n) \leq cg(n)$ for all $n > 0$. As the size of a problem increases we are mostly concerned with the increase in steps that must be performed to produce a solution. A polynomial time algorithm is defined to be one whose time complexity function is $O(p(n))$ where p is a polynomial function. An algorithm whose time complexity cannot be bounded by a polynomial function is called an exponential-time algorithm. A problem is termed hard if there does not exist any polynomial algorithm that can solve it. Conversely a problem is termed easy if there exists a polynomial algorithm that can solve it. A problem is referred to as intractable if it is so hard that no polynomial time algorithm can possibly solve it [68].

The class $\mathcal{P}$ contains the set of decision problems which can be solved in polyno-

mial time. $\mathcal{NP}$ (nondeterministic polynomial) denotes the class of problems that can be solved by polynomial time nondeterministic algorithms. A polynomial time nondeterministic algorithm is a theoretical construct capturing the notion of polynomial time verifiability. In essence it is an algorithm that can take all possible solutions and check them in parallel. Given a "yes" answer to a decision problem it can be easily verified in polynomial time. If the answer to a decision problem is "no" rather than "yes" and can be checked by a polynomial time nondeterministic algorithm then the problem is said to belong to the class $co\mathcal{NP}$. There are very few problems that have been proved to belong to both $\mathcal{NP}$ and $co\mathcal{NP}$ [36].

The class $\mathcal{NP}$ is much larger than the class $\mathcal{P}$, indeed $\mathcal{P} \subseteq \mathcal{NP}$. There might not be much difference between the problems in $\mathcal{NP}$ and $\mathcal{P}$ but as yet nobody has been able to prove that they are the same or different. The question of whether $\mathcal{P} = \mathcal{NP}$ remains one of the most infamous unanswered questions in computer science. If $\mathcal{P} = \mathcal{NP}$ can be shown the proof might unveil a new algorithm. On the other hand if it can be shown that $\mathcal{P} \neq \mathcal{NP}$, differences in the problems may highlight the reasons behind why problems in $\mathcal{NP}$ are harder to solve than others and provide an indication of how these problems can be solved more easily [36].

A problem is termed $\mathcal{NP}$-Hard if a polynomial time algorithm for that problem can be translated into a polynomial time algorithm for solving every problem in $\mathcal{NP}$. A problem that is both $\mathcal{NP}$-Hard and in the class $\mathcal{NP}$ is said to be $\mathcal{NP}$-Complete. Problems in the class $\mathcal{NP}$-Complete are among the hardest problems in the class $\mathcal{NP}$. Consequently if one could find a polynomial time algorithm for any $\mathcal{NP}$-Complete problem then there are polynomial time algorithms for all problems in $\mathcal{NP}$ proving that $\mathcal{P} = \mathcal{NP}$.

Magnanti and Wong [104] have shown that the network design problem is $\mathcal{NP}$-Hard. However, as far as we are aware no attempt has been made to determine if the problem is $\mathcal{NP}$-Complete. We believe that in the general case the network design problem is a generalised form of the travelling salesman problem (TSP). The TSP is one of the

$\mathcal{NP}$-Complete problems identified by Gary and Johnson [68] and is a transformation of the Hamiltonian circuit problem. Stated formally the TSP attempts to find the minimum cost Hamiltonian tour of a given set of cities that visits each city exactly once.

Theorem 1. *The network design problem is $\mathcal{NP}$-Complete.*

Proof. It is straightforward to prove that the network design problem (NDP) is $\mathcal{NP}$-Complete by showing that the TSP is a special case, giving TSP $\propto$ NDP.

To show this, consider an instance of the UTNDP where $r = 1$ and $d_{i,j} = 0 \; \forall \; i, j \in V$. In this case a valid route according to the constraints given in Section 2.3 is a Hamiltonian path. In addition, the operator cost is now defined $\sum_{\forall e_j \in E_{R_1}} w_j$ which is the sum of all the edge weights in the path, which is equivalent to the cost function of the TSP. □

If we relax the constraints on the number of routes and minimum and maximum vertices per route then the network design problem is a generalisation of the problem put forward by Johnson et al. [89]. Johnson et al. [89] provide a $\mathcal{NP}$-Complete proof for a slightly different network design problem where we are given a weighted directed graph and wish to find a subgraph which connects all vertices and minimises the sum of the shortest path weights between all vertex pairs. This can result in a spanning tree which violates the problem constraints we have defined. For example, a route that is a tree would have repeated vertices. If we relax our problem constraints and allow the tree to be created from M routes where $M \neq r$ then it is clear that the problem described in this thesis is a generalisation of that given by Johnson et al. [89].

2.9 Summary

This chapter has formulated both the network design and frequency setting problems. We have shown that both are heavily constrained multi-objective optimisation problems

that must balance the cost for the passenger and network operator. Problem instances used for the experimental work in this thesis were also introduced and a summary of their construction was presented.

An empirical analysis was then performed on four alternative graph structures for obtaining the passenger objective value for the network design problem. Alternate graph structures were explored and found to be unsuitable as they result in transfer penalties not being applied. As such the evaluation of the passenger objective requires a specialised graph to be created which we refer to as the transit network. A GPU algorithm for the APSP was then compared to the CPU algorithm on the transit network and found to offer a significant speed-up for larger problem instances.

Complexities of the UTNDP as a whole were then summarised followed by a proof that under our problem formulation the network design problem is $\mathcal{NP}$-Complete. In the next chapter we survey the relevant literature for the network design and frequency setting problems along with available commercial software to aid in the design and planning of public transport systems.

Chapter 3

Literature Review

This chapter first introduces the broad class of vehicle routing problems and puts the UTNDP into context. We then examine the manual approaches to the UTNDP, with a focus on the common guidelines that have been adopted by the public transport industry. Mathematical and heuristic approaches are then surveyed followed by a review of the metaheuristic techniques that have been applied to the UTNDP. Frequency setting approaches are then examined moving onto a discussion of the problems and limitations of the current research. Finally, commercial software packages used for public transport planning are discussed.

3.1 Vehicle Routing Problems

A Vehicle Routing Problem (VRP) requires the definition of an optimal set of routes for a fleet of vehicles in order to best serve a set of customers [142]. The VRP was first introduced by Dantzig and Ramser [41] who describe the real world problem regarding the delivery of gasoline between a terminal and a large number of service stations via a fleet of vehicles. The objective is to minimise the mileage of the vehicles whilst ensuring the demands of the service stations are met.

The VRP can be considered a generalisation of the Travelling Salesman Problem (TSP) [41]. The TSP, first posed by Whitney in 1934, obtained its name from the problem description whereby a salesman wishes to travel by the shortest distance between n

given cities before returning to his starting point [59]. Generalisations can then be made for the TSP by introducing additional constraints, such as a maximum capacity, adding to the complexity of an $\mathcal{NP}$-Hard problem.

Since the original formulation of the problem many variants of the VRP have emerged some of which are summarised below:

- Capacitated VRP (CVRP): deliveries must be made to a set of customers where the demands are deterministic and may not be split. Each of the vehicles is identical and based at a central depot with a capacity constraint defined for the vehicles. Each of the vehicles must visit the depot and each customer is only visited by one vehicle. The objective is to minimise the cost required to serve all the customers. If a distance constraint is added to the vehicles along with the capacity constraint the problem is then referred to as the Distance-Capacitated VRP.

- VRP with Time Windows (VRPTW): an extension to the CVRP in which each customer must be served in a given time window. Given a customer *i* a time window is defined as [a_i, b_i] inside of which the vehicle must arrive to service the customer. A time period, s_i, defines the time period for which the vehicle must service the customer for. Once the time period s_i has expired the vehicle can progress on to the next customer.

- VRP with Backhauls (VRPB): another extension to the CVRP. Given a partition of the set of customers, where, subset A contains those customers who require a delivery and subset B contains those customers who require a product to be picked up. If a vehicle route contains customers from both subsets then all of A must be served before B and the vehicle capacity constraint must not be exceeded.

- VRP with Pick-up and Delivery (VRPPD): each customer is associated with both a delivery and pick-up. Given a customer *i*, the customer whose pickup is the delivery for *i* when different from the depot must be served before *i*.

- Dial-a-Ride problem (DARP): consists of determining a set of routes and schedules for a given number of customers who each specify a pick-up and drop off location. The DARP can be seen to generalise several VRPs including the VRPTW and VRPPD. A significant difference between the DARP and other VRPs is the need to consider the human perspective. A service must be provided that balances the inconvenience to passengers whilst ensuring that the cost to the operator is minimised. [38]

VRPs can be divided into two further categories: 1) static, and, 2) dynamic. In the static case the transportation requirements are known beforehand whereas in the dynamic case the requests are revealed throughout the day and the routes of the vehicles are adjusted in real time or periodically to the meet the demand.

The VRP has been extensively studied resulting in a varied range of approaches being proposed for its solution. For example, exact algorithms which Laporte and Nobert [98] subdivided into three classifications; 1) direct tree search methods 2) dynamic programming 3) integer linear programming (examples of which can be seen in Gendreau et al. [69], Toth and Vigo [141], Baldacci et al. [10], Kok et al. [95] and Baldacci et al. [11]), heuristic approaches (e.g. Gillett and Miller [70], Clarke and Wright [32], and Renaud et al. [127]), meta-heuristic approaches such as genetic algorithms, simulated annealing and tabu search (examples of which can be seen in Bell and McMullen [14], Toth and Vigo [143], Baker and Ayechew [8], Tan et al. [139], and Czech and Czarnas [40]). Multi-objective algorithms have been acquiring an increasing amount of attention over recent years due to their ability to trade-off multiple objectives for a given problem. As such, a number of multi-objective formulations for the VRP have been proposed (these can be seen in Tan et al. [138], Ombuki et al. [121], and Baños et al. [12]).

Similarities can be drawn between the DARP and the UTNDP as both require the transportation of passengers from an origin to a destination. The DARP operates on a daily basis fulfilling the needs of the passengers on the given day, that is, the demand from day-to-day is not fixed. In comparison, the UTNDP assumes the demand per

day is invariant. Furthermore, each customer must supply their pick-up and drop-off location in the DARP. This leads to a smaller size of problem compared to the UTNDP due to the complexity of having to record possibly tens of thousands of individual trips for the latter problem. The DARP is a demand responsive application, aimed at short term planning whereas the UTNDP is very much a longer term planning tool given the investment needed from the network operator to change the route configuration and also the difficulties passengers would face having to remember constantly changing routes and schedules. It can be concluded that the UTNDP is a unique VRP that requires the design of bespoke algorithms to solve it. [53]

3.2 Practical Guidelines for the UTNDP

Despite recent advances in technology and research into the UTNDP the vast majority of transit planners still make use of past experience and practical guidelines to design or redesign bus routes and their accompanying schedules [114]. Suggested guidelines include service area, route coverage, route structure, route spacing, route length, route duplication and directness. Some of these guidelines are summarised below:

- The service area of a transit network is defined by the local operating authority and should cover major employment concentrations, schools and hospitals [116]. The service provided should enable potential passengers to reach their destination via public transport [102].

- Routes should make use of major street and land use patterns. For example, provide a grid system where the underlying street structure forms a grid [116].

- Only one route per arterial link should be utilised apart from approaches to the central business district or major transit stations – a desired maximum of two routes per street [116].

- Transit routes should be direct and try to avoid any circuitous paths. The route distance should not be more than the comparative distance by car. [116]

- Routes should be as short as possible whilst ensuring the required level of service, with overly long routes being avoided [116].

- Service frequency should not be below 30 minutes for peak periods and off-peak periods should not exceed 60 minutes. It is also common place to have clock-face headways where the headways on routes evenly divide 60 as these make it easier for passengers to predict bus arrival times and eliminate complex schedules. [111, 61]

- Load factor is the primary variable used to assess how effectively buses are allocated among different routes. The load factor is expressed as a percentage of the seating capacity of a vehicle at the maximum load (busiest) point of a particular route. The load factor is tied directly with the service frequency. A high load factor may mean an increase in service frequency is required or the use of larger vehicles. Loading factor guidelines vary however it seems that average load factors of 100 to 140 seem acceptable for off-peak and peak periods respectively. [111]

- For the service period different countries have different criteria. For example in the USA buses operate between the hours of 6am and 12am on weekdays and between 7am and 7pm at weekend. [116]

- In the European Union there are restrictions on driving hours for drivers of public transport vehicles. Daily driving hours should not exceed 9 hours with an exemption twice a week when it can be extended to 10 hours. Drivers must also take a break of at least 45 minutes after every $4\frac{1}{2}$ hours. [52]

Transit planners have been able to produce relatively good route sets over past years utilising the above guidelines and their local knowledge. However, with the increasing

world population and continued migration from urban to sub-urban areas a redesign of the public transport networks of most major cities may soon be warranted. Given the $\mathcal{NP}$-hard nature of the UTNDP [104] it is extremely unlikely, if not impossible, for a transit planner to produce a redesign for a transport network based upon guidelines and experience alone. [5]

3.3 Methods for tackling the UTNDP

3.3.1 Mathematical Approaches

Mathematical approaches for the UTNDP have tended to focus on specific aspects of the problem perhaps due to the $\mathcal{NP}$-hard nature of each sub-component, prohibiting exact solutions as the problem size grows.

In 1976, Byrne [21] modelled a radial transit system, due to its similarity with most cities transport systems, using polar coordinates in order to determine the routes and frequencies. The objective was the minimisation of both user and operating costs under both a constrained and unconstrained fleet size.

Schéele [130] proposed a non-linear model with the intention of determining the travel pattern and frequencies to be used on a given transit network. The objective of the problem was to minimise the total passenger travel time under a constrained fleet size. The frequencies and associated optimal transit trip pattern were solved simultaneously.

Another approach by Constantin and Florian [34] formulated a non-linear non-convex mixed integer programming problem, which was then reformulated as a bi-level Min-Min problem to find the optimum route frequencies. The objective was to find frequencies that minimise the total travel time including waiting time using a projected sub-gradient approach whilst adhering to fleet size constraints.

Bussieck [20] in his doctoral thesis concentrated mainly on rail transport; however,

his ideas can easily be adapted for use with other public transportation modes. The author proposed cost optimal planning, to determine routes, maximising the number of direct travellers whilst minimising the cost to the operator formulating this originally as a non-linear integer program. This problem was then solved using relaxation and branch-and-bound.

Wan and Lo [148] in 2003 presented a mixed integer formulation for solving the network design and frequency setting components of the UTNDP with the objective of minimising the sum of the operating costs, modifying an existing transit network. The mixed integer formulation was transformed into a mixed integer linear formulation to enable instances, of small size, to be solved on standard commercial solvers. The approach taken was only suitable for small problem instances (the authors illustrate an example with 10 vertices producing three routes that requires 363 binary variables, 30 integer variables and 303 continuous variables) with the authors stating "devising efficient solution heuristics and algorithms is crucial for applying the approach for practical size networks" [148, p. 308].

Lee and Vuchic [99] used an iterative procedure that minimised the passenger travel time under variable demand. The approach consisted of generating an initial network using the shortest path algorithm between all pairs of vertices then undesirable paths were eliminated such as those that are a subset of another. A transit assignment procedure was then used to assign the demand concentrating it on certain routes allowing less efficient routes to be eliminated. Finally an improvement procedure was applied aimed towards decreasing the passengers' travel time. The authors considered fixed total travel demand and variable demand whereby the choice of travel i.e. car or public transport was made using a logit formulation.

Guan et al. [75] attempted to design a set of routes and assign the passengers to routes. They combined the operator objective in terms of the sum of the length of all routes and the passenger objective consisting of the sum of in-vehicle travel time and total number of passenger transfers. The two subproblems, route configuration and passenger

assignment, were solved simultaneously using a convex combination of the operator and passenger objectives. The authors applied their approach to several generated minimum spanning tree networks and a very simplified model of the Hong Kong Mass Transit Railway where the number of stations is reduced from 49 to 9 to allow the problem to be solved. The authors acknowledged the need to reduce the size of the network to enable the problem to be solved on a normal desktop computer due to the enormous number of variables and constraints. Finally the authors noted the need to consider metaheuristic methods for solving large network problems.

Mathematical programming formulations can provide benefit only when an optimal solution can be obtained, given the combinatorial nature of the network design problem mathematical approaches do not scale well to practical sized instances [25]. This stance is echoed by Israeli and Ceder [81] who argue that conventional mathematical approaches that contain a significant simplification of the UTNDP are still unable to solve the problem. As such mathematical approaches have fallen out of favour for the UTNDP but work on similar problems still highlights their inability to solve practical sized instances. For example, Wu et al. [151] focus their attention on bus lane selection on existing transport networks for rapid transport. This problem clearly has parallels with the network design problem. The authors found that CPLEX can only solve instances with 110 nodes and 12 bus lines.

3.3.2 Heuristics

The adoption of heuristic methods for solving the UTNDP has been relatively wide spread due to their ability to approximately solve large problem instances. Heuristics are optimisation methods that exploit domain specific knowledge to produce a reasonable solution to the problem. However, compared with mathematical methods they do not guarantee to find the optimal solution [129]. Generally heuristic approaches to the UTNDP have concentrated on the individual subproblems with an emphasis on network design and frequency setting, few have solved the entire UTNDP.

In 1967 Lampkin and Saalmans [97] proposed a complete solution to the UTNDP through a four stage procedure. Firstly, a set of routes was generated using a heuristic procedure to maximise the number of journeys that could be satisfied directly whilst ensuring that the routes were reasonably direct, not excessively long and enabled transfers. In subsequent stages the frequencies of the routes were found, timetables were constructed and scheduling was undertaken respectively.

By far the most detailed description of the UTNDP is due to Mandl [105], 1979. Mandl highlighted some of the major difficulties of the problem along with a network instance that has become the defacto benchmark for researchers. Mandl paid particular attention to the network design problem and developed a two stage procedure, whereby an initial set of routes were generated and then improved upon using heuristics. A shortest path algorithm was utilised to first find a feasible set of routes using a set of terminals designated by the designer, attempting to find the path that included as many vertices as possible to avoid transfers. An analysis procedure then calculates the total transportation time which is to be minimised under a constrained fleet size. On completion of the analysis the feasible set of routes were improved using an iterative procedure that makes one of four changes to the routes. If the resultant route set is an improvement it is kept and the improvement operators are applied to this set until no more improvements can be made.

Mandl [106] also developed a route improvement heuristic such that, given a feasible set of routes, new feasible route sets are developed via a sophisticated technique of trial and error. Mandl proposed three route improvement heuristics which are summarised below:

- Creation of new routes by exchanging parts of a route at a point of intersection.
- Including a vertex in a route if there is a high travel demand between that vertex and the vertices already present in the route.

- Removing a vertex from a route if the travel demand between the vertex and the other vertices in the route is low, and the vertex is present within another route.

A search procedure was utilised to find the best possible improvement for each of the above improvement heuristics. The average transportation cost was computed for the improved route set and replaces the current best solution if the average transportation cost is lower than the current best. This procedure iterates until no more improvements can be made to the route set that result in a lower average transportation cost.

As mentioned previously, Ceder and Wilson [24] identified five different activities for the UTNDP consisting of network design, frequency setting, timetable development, bus scheduling and driver scheduling. A two-level approach was formulated: level I considered the passengers point of view and attempted to minimise the total travel time. In level II both passenger and operator viewpoints were then taken into account by balancing the passenger travel time and waiting time against the number of vehicles required. Frequency setting, timetable development and vehicle scheduling were also calculated at level II.

An alternative approach by Israeli and Ceder [81] proposed a seven step approach to the UTNDP, creating the routes, identifying transfers and then setting frequencies. In the final stage of the procedure two objectives were used in a multi-objective optimisation procedure allowing a human decision maker to choose a trade-off between a set of solutions. The first objective consists of the weighted sum of the passenger waiting hours, empty space hours, and the deviation of passenger hours from the minimum possible. The second objective considers the operator perspective only reflected in the fleet size required to operate the routes.

Baaj and Mahmassani [5, 6] constructed a three stage approach composed of a route generation algorithm, an analysis procedure and a route improvement algorithm. In the route generation algorithm the designer's knowledge and experience are utilised as well as the demand matrix to generate a set of routes. The generation procedure

is heavily dependent upon the demand matrix generating a user prescribed number of skeleton routes for the highest origin-destination demand pairs. These skeleton routes are generated using the shortest path or next-shortest path subject to constraints that can be imposed by the designer. Skeleton routes are then expanded using one of three heuristic procedures:

- Maximum demand insertion
- Maximum demand per minimum time insertion
- Maximum demand per minimum route length insertion

Once the number of skeletons specified by the user have been generated the route generation algorithm proceeds to check the demand satisfied directly. If the demand satisfied directly is below a user specified value then the highest as yet unsatisfied directly origin-demand pair is taken and expanded, this process continues until the required level is met. The demand satisfied directly and in one transfer is next analysed and if below a user prescribed level a similar approach to that described previously is taken.

On completion of the route generation algorithm the analysis procedure, TRUST (Transit Route Analyst), is used to determine a range of performance measures including the total travel time for passengers, number of passenger trips satisfied in zero, one or two transfers or unsatisfied, links flows, frequencies and number of buses required to keep the load factor on the network under a specified maximum (see [4] for a detailed discussion of TRUST). The computed measures are then utilised by the route improvement algorithm to improve and produce a feasible route set using a variety of approaches such as merging routes with a low rider-ship to form a route with a medium to high ridership.

In [133, 134], Baaj and Mahmassani's work was extended to include the concept of transit centres and coordinated operations. Transit centres operate as hubs with bus

routes 'feeding' these hubs. As such they are commonly referred to as feeder routes. Shih and Mahmassani [133] identify that the major disadvantage of the transit centre concept is the requirement for passengers to transfer in order to complete their journey. Therefore coordinated operations are proposed whereby the bus schedules are timed such that a very short waiting time at a transit centre is required before a passenger can continue their journey.

The approach proposed in [133, 134] progresses by firstly applying a route generation procedure. If the transit centre concept is utilised then n skeletons are generated with the remainder of the skeletons generated using high demand vertices pairs – assuming that n is less than the number of routes specified by the designer. An analysis procedure then computes an array of network, route and vertex level descriptors, determines the frequencies and finally computes performance measures. Finally, a route improvement procedure is applied in an attempt to improve the generated route set using a selection of methods.

Fusco et al. [64] combined the approaches put forward by Baaj and Mahmassani [6] and Pattnaik et al. [124] to put forward a three stage approach to network design. In the first stage a three phase approach to route generation was used. Firstly, the shortest paths between vertices with a demand higher than a given minimum value were generated, secondly a process based on link flow analysis was used to generate main routes and feeder routes using a set of constraints which are relaxed for the latter. Finally, the existing transit network routes are included into the pool. The second stage then applied a GA to select a configuration of routes that minimised the operator and passenger costs, setting the frequencies simultaneously. Lastly route improvement was applied to the set of routes selected by the GA.

3.3.3 Metaheuristics

Metaheuristics are a general framework that can be adapted to solve specific problems, utilising heuristics and search algorithms to find solutions in large and complex solution spaces. Originally metaheuristics were defined as a method able to escape a local optimum using a combination of local search improvements and a higher level strategy [72]. However, it is now generally accepted that metaheuristics are any technique, especially those exploiting neighbourhood operators to identify admissible solution moves, that enable the escape of local optimum in complex solution spaces [72].

Evolutionary Algorithms

Evolutionary algorithms (EAs) are a broad class of population based algorithms using mechanisms such as selection, crossover and mutation inspired by the concepts of natural selection and natural genetics [74]. A genetic algorithm (GA) is an example of an EA utilising a population of chromosomes (candidate solutions) each with an associated fitness (cost).

A GA is usually first initialised with a random or heuristically generated population of candidate solutions that are evolved over a series of iterations called generations. A generation proceeds by selecting parent solutions from the population, crossover can then be applied with a certain probability creating offspring solutions by exchanging parts of the parent solutions. The offspring can then be mutated again with a certain probability in the hope of producing fitter individuals (solutions that improve upon the objectives being optimised). A replacement procedure can then be used to determine whether the solution should enter the population for the next generation. The basic structure of a generic GA is given in Algorithm 3.1.

In 1998 Pattnaik et al. [124] proposed a system to attempt to minimise the overall cost composed of the operator and passenger cost as with Baaj and Mahmassani [5]. A route set generation phase was then utilised and guided by the demand matrix, the designers

```
1: P_0 = Generate N random candidate solutions
2: Evaluate(x) ∀x ∈ P_0
3: t = 0
4: repeat
5:    for i = 1 to N/2 do
6:       Select parents at random from P_t
7:       if random < p_c then
8:          Apply crossover to create offspring
9:       else
10:         Create a copy of each parent to become the offspring
11:      Mutate each variable in the offspring with probability p_m
12:      Evaluate offspring solutions
13:      P_{t+1} = P_{t+1} ∪ offspring
14:   t = t + 1
15: until stopping condition satisfied
```

Algorithm 3.1: A basic genetic algorithm.

knowledge and route constraints to generate a set of candidate solutions. In the first instance terminal vertices are identified by the designer and the shortest path between them found. In subsequent stages each link in the shortest path is clamped successively and a new shortest path between the terminal vertices found. All the routes that obey the problem constraints are then added to a candidate route set. The authors introduce fixed string length coding (FSLC) and variable string length coding (VSLC) whereby the number of routes is fixed and allowed to vary in the coding itself respectively. It was determined that FSLC provided the better performance measure however these improvements were marginal to VSLC given the extra computation time required.

In 2002 Chakroborty and Wivedi [26] proposed a three stage strategy to the network design problem, consisting of initial generation, evaluation and modification. During the initial generation heuristic methods were utilised to produce route sets that were generated from simple logic rather than an arbitrary process. The fitness function weighted three separate components namely the average in-vehicle travel time including transfer time of all the passengers on the network, the percentage of passengers who can reach their destination in zero, one or two transfers and finally the percentage of passengers who are unable to reach their destination in two transfers.

The GA used was provided with two crossover operators termed inter-string and intra-string crossover. Inter-string crossover exchanged routes from the parent route sets, however intra-string crossover exchanged parts of a route in a parent if two routes shared a common vertex. Mutation was then applied to the offspring by selecting a random vertex and changing it to any of its acceptable vicinity vertices [26].

In 2003 Tom and Mohan [140] created a two phase approach whereby a large set of potential solutions were first generated and a GA was used to select a solution route set along with their associated frequencies from this pool of solutions. The objective was to minimise the total system cost, the sum of the operating cost and passenger cost according to that of Baaj and Mahmassani [5].

Fan and Machemehl [57] in 2006 proposed an approach with the aim of minimising the sum of the user cost, operator cost and unsatisfied demand using a weighting factor for each. Initial routes were generated using shortest path and k-shortest path algorithms with slight modifications such that all feasible paths between a vertex pair were produced – any path with a route length between the minimum and maximum allowable number of vertices. A GA was then used to select a set of routes from the set of routes generated. The frequencies were also set for each route using an iterative process. Fan and Machemehl [57] stated that they found GAs outperformed local search with multiple start points and provided no worse solution quality when compared with simulated annealing and tabu search with similar running times.

In 2009 Fan et al. [55] proposed a simple multi-objective algorithm to the network design problem trading off the passenger cost, defined as the total travel time over all passengers, and operator cost defined as the sum of the route lengths. The Make-Small-Change procedure was applied as the mutation operator where a vertex can be either added to the end of a route or removed from the start of a route. The EA used was based upon the simple evolutionary algorithm for multi-objective optimisation (SEAMO) [146, 113] however lacked a crossover operator with only a mutation operator utilised.

Bagloee and Ceder [7], 2011, tackled real size road networks using a combination of

heuristics and a genetic algorithm equipped with an ant-system [7]. Their algorithm firstly determined the location of stops based upon the distance to high concentrations of travel demand and then used a system inspired by Newton gravity theory to produce a set of candidate routes. Finally a GA was used to search through the candidate routes to find a good solution. The authors proposed mutation operator randomly selected a route to remove and replaced it with a candidate route. The frequency of routes was computed simultaneously.

Szeto and Wu [137] solved a real world instance for a suburban residential area in Hong Kong, although their formulation differs to that found in the majority of the literature. Both network design and frequency setting were tackled simultaneously using a weighted sum approach composed of the number of transfers and total passenger travel time (in-vehicle travel time and waiting time). The underlying transport network had 23 bus stops and seven terminals and all the bus routes originate from these terminals and terminate at one of five destination vertices. The authors solution approach used a GA to solve the network design problem and incorporate a frequency setting heuristic based upon neighbourhood search into the GA to solve the frequency setting problem. Two crossover operators were proposed, the first exchanged entire routes between solutions and the second exchanged sequences of intermediate stops between routes that shared the same destination in both parent. Four mutation operators were also proposed that could either add or delete a vertex from a route, exchange a vertex between two routes or move a vertex from a route into another route. An improvement heuristic was also used to improve the solution quality of the GA by improving the sequence of stops for each route in terms of minimising the trip time. A method for maintaining the population diversity was also proposed where the probability of selecting an individual for the next generation is based on its hamming distance from the best individual.

Cipriani et al. [29] tackled redesigning Rome's public transport network with a focus on the bus network. The approach utilised a weighted sum of the operator's costs, defined as total bus travel distance and travel time, the passenger's cost, defined as the sum of

in-vehicle travel time, access time, waiting time and transfer penalties, and a penalty related to the level of unsatisfied demand. A heuristic generation procedure was used to generate a large set of candidate feasible routes then a GA applied to find the optimal set of routes and associated frequencies. The authors split routes into three separate classes: 1) A-type – these are direct routes connecting the highest demand node pairs not served by the rail system. Generated by applying a shortest path algorithm. 2) B-type – routes that connect main transit centers and links carrying the highest passengers' volumes levels. These are generated using the flow concentration procedure presented in [23]. 3) C-type – existing routes in the transit network. The heuristic generation procedure first generated all the types of routes and stored them in a set of feasible routes. A check was then made against the problem constraints and any routes that violate the constraints were removed, the routes that satisfy the constraints were then used as input to a parallel genetic algorithm (PGA). An initial population was formed for the PGA by randomly selecting a given number of routes from the set of feasible routes, forming a single individual. Crossover randomly selected half of the routes from one parent and the other half from the second. Mutation was then applied by replacing each route with a randomly selected route from the set of feasible routes with a given probability. The authors went on to show that their approach was able to offer a more efficient service using a smaller number of routes.

In 2012 Miandoabchi et al. [110] considered both road network design and transit network design consisting of constructing new streets, adding new lanes to existing streets, determining the direction and location of one way streets, lane allocations in two way streets and the topology of the bus networks. Important differences between the problem formulated in the paper and the remainder of the literature are that the effects of private car flows on bus flows or vice versa are taken into consideration and that bus routes may not pass the same sequence of stops on their forth and back routes. To solve the mathematical problem formulation the authors proposed a hybrid of a GA and simulated annealing and a clonal selection algorithm hybridised with simulated annealing.

Mumford [114] built upon the work of [57] and [55] proposing new genetic operators and an efficient heuristic method for seeding the population. In their work the network design problem was formulated as a multi-objective problem attempting to minimise both the average passenger travel time, composed of both the in-vehicle travel time and a transfer penalty, and the sum of all the transit route lengths. A new crossover operator that alternates the selection of routes between two parents favouring routes with as yet unused vertices whilst ensuring route connectivity has been designed as well as two new mutation operators that adds or deletes a bounded random number of vertices from a route set. A repair procedure was also given to ensure that route sets produced from initial generation and crossover are valid. Similar to Fan et al. [55] SEAMO [146, 113] was used as the evolutionary algorithm for the multi-objective optimisation. Four new benchmark instances were introduced and made available for other researchers.

Chew et al. [27] attempted to minimise the passenger and operator costs similar to Mumford and used the same problem formulation and objective functions. An initial population was generated by randomly selecting two nodes, these become the origin and destination of the route, Floyd's algorithm was then used to find the shortest path between the two nodes if they are not adjacent. It should be noted that the authors made the assumption that the minimum number of nodes is permissible in a route is fixed at 2 and the user will only set a maximum limit. A crossover operator inspired by Chakroborty and Wivedi [26] where a random route is selected from each parent. Substrings between the chosen routes swap their position between the two parents, producing two offspring. If the offspring violate the problem constraints then the crossover process is repeated until two valid offspring are produced. If none of the possibilities yield valid offspring then two new parents are selected and crossover is run again. Mutation is then applied using a modified version of the identical-point mutation operator proposed by Ngamchai and Lovell [117]. The operator selects a random node that is contained in at least two routes and then swaps the nodes before the selected node in two routes contained in the route set. Mandl's benchmark instance is used for comparison between Fan et al. [55] and Mumford [114] with the authors able to make

improvements over the majority of test cases.

Simulated Annealing

Simulated annealing (SA) was first posed by Kirkpatrick et al. [93] in 1983 and is an extension to random descent and utilises the Metropolis algorithm proposed by Metropolis et al. [109]. In random descent a move to a random neighbour is made at each iteration this can result in becoming trapped in a local optimum. SA attempts to escape local optima by making a move to a "worse" solution with a given probability in an attempt to explore more of the search space. SA is analogous to the process of physical annealing with solids, where a solid is heated and then allowed to cool very slowly until it reaches the most stable lattice configuration possible [72]. If the cooling schedule is slow the resulting configuration will usually be superior to that of the original [72].

In each iteration of SA a new solution is produced by making a small neighbourhood move to the current solution, this change is usually small to allow for a gradual change in solution such that the search does not jump to vastly different areas of the search space at each iteration. The two solutions are evaluated and compared, if the new solution improves upon the current solution then the current solution is replaced with the new solution else it is probabilistically decided whether to replace the current solution or not. At high temperatures (usually at the beginning of the SA algorithm) the probability of accepting worsening moves will be relatively high enabling the algorithm to escape local optima, as the temperature decreases the probability of acceptance also decreases allowing the algorithm to converge. A basic SA is given in Algorithm 3.2.

In 2006 Fan and Machemehl [56] used a SA algorithm to select the best set of routes from a pool of candidate routes. The process was similar to that of Pattnaik et al. [124], a set of initial solutions were created using Dijkstra's shortest path algorithm and Yen's k-shortest path algorithm. Route frequencies were determined simultaneously via a network analysis procedure to enable the computation of the required performance

1: $s_{current}$ = generate initial solution
2: Evaluate($s_{current}$)
3: $s_{best} = s_{current}$
4: t = initial temp
5: **repeat**
6: **for** i = 1 to MAX_ITERATIONS **do**
7: s_i = get random neighbour of $s_{current}$
8: Evaluate(s_i)
9: **if** $s_i < s_{current}$ **then**
10: $s_{current} = s_i$
11: **if** $s_{current} < s_{best}$ **then**
12: $s_{best} = s_{current}$
13: **else**
14: **if** $e^{\frac{s_{current}-s_i}{t}} >$ random **then**
15: $s_{current} = s_i$
16: decrease t
17: **until** stopping temperature reached

Algorithm 3.2: A simulated annealing algorithm.

measures. The objective was to minimise the sum of the user costs, operator cost and unsatisfied demand. The user cost is composed of walking cost, waiting cost, transfer cost and in-vehicle travel cost. The operator cost consisted of the cost of operating the buses on the routes. Headway, load factor, fleet size, route length and maximum number of routes constraints were imposed and required to be satisfied making the final solution quite realistic.

Fan and Mumford [54], 2010, proposed a SA algorithm using the Make-Small-Change procedure as the neighbourhood operator. A route set was generated at random according to a series of constraints then the Make-Small-Change procedure applied. The Make-Small-Change procedure could apply three moves to a randomly selected route: 1) Adding a vertex to the last position in a route 2) Deleting the first vertex in a route 3) Inverting the order of vertices in a route. The SA method was compared with a basic hill climbing algorithm for Mandl's benchmark instance and was found to obtain better results although did not improve over those of the GA proposed by Chakroborty and Wivedi [26].

Tabu Search

Tabu search (TS) was introduced by Glover [71] in 1986 however some elements were introduced by Glover previously in 1977 [72]. TS is an extension to steepest descent where a move to a neighbour solution is decided based upon the "best" neighbour, the solution that improves most upon the objectives. TS improves upon steepest descent by imposing restrictions on the search via a set of memory structures commonly referred to as tabu lists, these can be short term or long term memory. A neighbourhood move that is contained within one of these lists are termed tabu and forbidden [73]. A neighbourhood is a user defined operator that identifies solutions that are adjacent to the current solution in the solution space. A tabu move can be accepted under certain criteria, called aspiration criteria, if the evaluation of the solution provides an improvement of the best so far objective value. The basic TS algorithm is given in Algorithm 3.3.

1: $s =$ generate initial solution
2: $t = 0$
3: **repeat**
4: $N(s)$ generate the neighbours of s
5: $T(s,k)$ get the tabu set for s
6: $A(s,k)$ get the aspirant set for s
7: $s_t =$ choose best solution from $\{N(s) - T(s,k)\} + A(s,k)$
8: **if** $s_t < s$ **then**
9: $s = s_t$
10: update memory lists
11: $t = t + 1$
12: **until** stopping condition is satisfied

Algorithm 3.3: A basic tabu search algorithm.

Fan and Machemehl [58], 2008, made use of TS attempting to minimise a weighted sum of passenger cost, operator cost and unsatisfied demand. A route set generation procedure generated all candidate routes constrained by user input on minimum and maximum length. An analysis procedure and TS were then used iteratively to generate a good set of routes simultaneously producing the route frequencies.

In 2007 Lei and Yan [100] formulated an approach generating an initial route set then

applying three neighbourhood operators to gradually improve the solution by applying TS. The objective was the minimisation of the total cost composed of the in-vehicle travel time, waiting time and operator cost. The neighbourhood operators used were:

- Route-merge – attempt to combine two routes with common vertices.
- Route-break – split a route into two separate routes only if the number of routes in the current route set is lower than a predetermined minimum value.
- Add-link – attempts to add vertices to a route that is of a small length.

Ant Colony Optimisation

Dorigo [50] proposed the ant colony optimisation (ACO) metaheuristic in his PhD thesis for finding solutions to combinatorial optimisation problems that can be reduced to finding paths through a graph. The process is analogous to the real-world process ants use to find food from their nest. As an ant moves it lays down a pheromone trail, if another ant comes across this trail it will decide whether to follow the trail based upon the strength of the pheromone. The pheromone evaporates over time meaning that shorter paths through the graph to the food are more heavily utilised containing a strong pheromone trail.

ACO has received relatively little attention in the literature for solving the UTNDP. Yu et al. [152], 2005, proposed a parallel ACO algorithm using the passenger flow on each link as the pheromone. Their approach utilised parallel hardware due to their statement that the enumeration method used required a large amount of computation for large-scale practical instances.

A similar method is proposed by Jiang et al. [82] however solution stagnation is taken into account by reinitialising a pheromone trail after a certain threshold is reached. A penalty mechanism was also introduced to place constraints on a route such as a suitable length and the inability to contain cycles.

Other Metaheuristic Approaches

In 2009 Mauttone and Urquhart [108] based their approach on the GRASP [128] metaheuristic. The authors approach used a shortest path algorithm to generate a set of routes that conformed to a given number of constraints. An initial set of frequencies were determined for the routes and local search was then applied using a random vector of weights for the conflicting objectives. The objectives used were the passenger costs, in-vehicle travel time, waiting time and transfer time and the operator cost is given as the required fleet size needed to operate the routes with the required frequencies.

Blum and Mathew [16] proposed an agent based approach incorporating creation, deletion, modification and scheduling agents. A route generation approach using shortest path calculations and a link modification weight every time a link is used were utilised to produce all routes present in the transport network. Creation agents select routes from this pool to create route sets with several types of creation agents given. Modification agents were then used to improve the solutions using a variety of approaches ranging from route splitting to route deletion.

Blum and Mathew [17] tackled redesigning the bus network in Mumbai under the constraint that existing routes must remain in the network although the stops on the existing routes could be serviced less frequently to enable new routes to be created. Similar to their previous work ([16]) an agent based optimisation was used consisting of six modification agents. Their approach first created potential routes which were used to seed a solution pool with a set of initial solutions that were evaluated to ensure that they did not violate any of the problem constraints and models the passenger route choice. Multiple iterations of modification agents were then used to take solutions from the pool and create a new solution based upon the given solutions. The solution pool size was kept within predefined bounds via the use of a delete agent that removes poor solutions. The authors found that despite having to keep the existing routes significant improvements could still be found with a decrease in operator cost of 18.1% whilst maintaining the same level of performance for passengers. On the other hand an

improvement of 5.5% could be made from the passenger objective under the current operator cost.

Nikolić and Teodorović [119] put forward the use of bee colony optimisation (BCO) utilising a simple greedy algorithm to generate an initial solution then applying the BCO algorithm to improve the given solution. The authors concentrated on the passenger objective defined as the sum of the total travel time spent by passengers on a service, a weighted penalty for the number of transfers and a weighted penalty for the number of unsatisfied passengers. No thought was given to the operator point of view in the optimisation making the algorithm extremely biased in terms of the solutions produced. The greedy construction algorithm took the vertex pairs that have the highest demand values and generated the shortest path between them in turn generating a route. Heterogeneous bees were used where the first type takes a given route with a certain probability then destroys the route replacing it with the shortest path between one of the original terminal vertices and a new terminal selected with a given probability. The second type of bee selected a route in the same manner as the first type but this time decides to remove one of the terminal vertices with a certain probability. If the vertex is not removed then a new vertex is added to the route by randomly selecting from the adjacent vertices. The approach was compared using Mandl's instance and another instance although the majority of comparison was done using Mandl's instance.

3.4 Frequency Setting

In the previous section we looked at network design which is the first component on the UTNDP. We now look at the frequency setting problem, which is applied to a particular network design.

Schéele [130] formulated the frequency setting problem as a non-linear programming problem taking into account capacity constraints on buses attempting to minimise the total generalised travel time – walking time, in-vehicle travel time and waiting time.

The distribution of passengers was based upon the demand with the demand divided between the routes according to an entropy and the capacity constraints. Not all bus stops were included only the most relevant vertices were included reducing the size of the problem. The round trip time of a route was assumed to be independent of the loading of the bus. The proposed model was tested on the town of Linöping with 6 routes.

Han and Wilson [76] adopted the objective of minimising the occupancy level at the most heavily loaded point on any route in the network due to the complexity involved in specifying accurate waiting times and crowding level functions. A constraint was imposed upon the maximum number of buses; however fractional buses were allowed due to the possibility of interlining. The passenger assignment problem was tackled using the following approach: if a unique route for an origin-destination pair was available then this route was used, if a set of routes were available the demand was shared between them using frequency sharing [28]. The solution methodology first calculated a lower bound by iteratively assigning passenger flows and route frequencies until convergence was achieved. Secondly a much simpler surplus allocation problem was formulated defined only by linear constraints. The proposed model was tested on a small instance with 6 vertices and 3 routes.

Constantin and Florian [34] formulated a non-linear mixed integer formulation with the objective of minimising the total expected travel time and waiting time whilst satisfying both constraints on the fleet size and lower bounds. This problem was then reformulated as a Min-Min non-linear bi-level program. The upper level represents the planner's viewpoint and minimises the overall travel time subject to a fleet size constraint. The lower level represents the passenger viewpoint with the objective of minimising the overall travel time assigning passengers to routes based upon optimal strategies proposed by Spiess and Florian [135]. The model was tested using instances related to the cities of Stockholm (38 routes), Winnipeg (67 routes) and Portland (115 routes).

Gao et al. [65] proposed a bi-level model with the upper level attempting to minimise the total deterrence of the system, composed of the average in-vehicle travel time on the link and waiting time at the origin vertex of the link, and the lower level was a transit equilibrium model. The passenger assignment proposed in [42] was used which assigns the demand based upon the frequencies on the routes. The solution approach firstly selected an initial set of frequencies which were then iteratively improved upon using a sensitivity analysis until convergence. A numerical analysis was carried out on a very small example containing 4 vertices and 4 routes.

Yu et al. [153] formulated a bi-level problem with the upper level optimising the frequencies of the routes based upon the passengers on each route and the lower level assigning passengers to each route based upon the optimal strategy by [135]. A GA was used with an integer encoding of the frequencies and genetic operators that reassigned the vehicles among the different routes iterating on the passenger assignment until the frequencies converged. The proposed solution methodology was tested on two instances, the first with 6 vertices and 4 routes, the second with 3,004 vertices and 89 routes.

Huang et al. [79] proposed a bi-level formulation for frequency setting. In the lower level the proportional flow eigenvalues were calculated using the optimal strategy transit assignment model. The upper level was concerned with two factors: 1) the network cost composed of the passengers' expected travel time and operating costs, and 2) the network robustness indicated by the variance in passenger travel time. Constraints were placed upon the maximum fleet size available. A GA was proposed using a real-coded scheme to represent the frequencies.

Similar to Yu et al. [153], Martínez et al. [107] used the optimal strategy proposed by [135] to assign the demand to the network. Two approaches were put forward, the first a mathematical approach based upon the approach by [34] that was used to solve small real-world sized instances with 84 vertices. The authors identified the need for an approximate method to allow for larger instances to be solved and put forward a tabu search approach. Tabu search was applied under a constrained fleet size with infeasible

solutions penalised. To prevent a large number of neighbours being explored due to the high cost of assigning the demand to the network the number of neighbours that are explored are limited. The authors also discretise the set of allowed frequencies further reducing the size of the search space. It was further assumed that there is sufficient capacity on every route to carry the assigned passengers.

3.5 Limitations of Published Research

The current literature tends to concentrate on highly specific problem instances or use Mandl's problem instance to perform comparisons. Furthermore there is relatively little discussion regarding the heuristics or genetic operators applied. The lack of detail along with the vast number of objective functions means a comparison between methods can be challenging. A range of metrics are utilised in the literature due to the range of beliefs as to what constitutes a good solution in terms of both the passenger and operator perspective.

A further issue faced in the literature is the lack of benchmark instances available with the majority of researchers tailoring their methods to specific instances of the UTNDP for which they have acquired data. This makes comparison almost impossible if the instance is not placed in a publicly accessible location. Mumford [114] has made a step towards resolving the problem with the publication of a set of instances that are publicly available. The lack of benchmark instances previously available has led to researchers comparing methods using Mandl's benchmark instance which is extremely small compared with real-world scenarios, and as such, methods that may perform more favourably on large instances (heuristic methods for example) are penalised.

Benchmark instances created based upon real world cities such as those given in [114] have gone some way in addressing the issue with a lack of problem instances. However, there is still a need for a set or real world instances, of ranging sizes, to enable methods to be compared on practical networks. When real world instances are used in the

literature they tend to be highly specific and are not made available to other researchers.

The computation time involved for even relatively small instances (100 vertices) is a serious limitation with running times measured in days rather than hours. For real world instances the computation time is a limiting factor in terms of the problem size that can be tackled in a reasonable amount of time. Progress must therefore be made in terms of improving the efficiency of developed algorithms or the exploitation of high performance computing resources to enable these larger instances to be solved.

Computation time must however be balanced against the time frame for decision making for the UTNDP. A network redesign may only take place once every decade or more. A computation time of weeks or even months may therefore be tolerable given that the majority of network planning will be for the long term. Engagement from industry is key to develop approaches that provide insight in time frames that are synced to commercial processes.

Industry have adopted very few methods published in the literature, a notable exception is Hasselström's [78] method although this was adopted a significant number of years ago. If industry participation could be improved the research could benefit by producing standard performance metrics that industry are concerned with rather than the theoretical metrics used in the literature.

3.6 Commercial Software

A large number of commercial transportation software applications are available with integration of geographical information systems (GIS) enabling the overlay of routes on to maps. The majority of applications provide a graphical user interface allowing for easy use for transit planners who may not have significant information technology experience.

VISUM is the world's leading demand-driven and service-oriented public transport

planning tool [125]. VISUM supports both public transport, bus and rail for example, and private transport such as a private car. The system is designed to be very easy to use with the planner being able to click two terminals for a transit line and VISUM generating a route between these [63]. Modelling is inbuilt allowing for new transit networks to be evaluated using a series of performance metrics for both the passenger and operator perspective.

Emme is a complete transportation forecasting package that can model travel demand for rural, urban and national transportation networks [80]. Planning and modelling facilities are provided along with an API allowing the framework to be extended by the planner to provide their own user interfaces and access to the underlying demand data [80].

SATURN is a suite of tools developed at the University of Leeds, providing network analysis with six basic functions. These functions include traffic simulation and assignment, network editor, matrix manipulation package and simulation of individual junctions. [1]

Cube Voyager allows for the forecasting of personal travel via a selection of models providing an open and user friendly framework solution for modelling planning policies and improvements at the urban, regional and national level [31]. Modelling functions are provided for networks, highways, public transport and travel demand [30].

Trapeze are a commercial enterprise that specialise in software designed specifically for supporting, building, managing and measuring transportation services [144]. Services are provided for several types of transport ranging from public to healthcare with a particular focus on scheduling and the storage of complex data associated with large transport systems [145].

In general the commercial software mentioned above is well utilised in industry and mainly used for visualisation, simulation and decision support. None are able to automate the process of designing a new route network from scratch. However, given the

significant disruption to passengers that a total redesign would cause it is questionable whether such capability is needed. For example, the ability to add stops to a route currently in operation or add a single new route may provide more benefit to the industry. This reflects more closely processes that happen in the real world, where new housing developments for example are incorporated into existing routes.

3.7 Summary

This chapter has provided a brief introduction into the broad class of vehicle routing problems and shown how the UTNDP relates to the DARP. Guidelines used by transit network designers were presented before surveying the mathematical approaches to the network design problem. It is evident that exact approaches have fallen out of favour due to their inability to tackle problems of a practical size.

Heuristic approaches relevant to the network design problem were then covered before moving onto the metaheuristic approaches. The majority of approaches lack detail in terms of the local search or genetic operators used. There is wide ranging formulations for the network design problem with some authors only considering a single objective formulation. Furthermore, empirical comparisons are made using Mandl's Swiss road network which is very small in comparison to real world networks. Where real world instances are used the data is not made publicly available for other researchers to compare.

Methods for tackling the frequency setting problem were then described before moving onto a discussion regarding the limitations of current research for the UTNDP. Finally a brief overview of commercially available software to aid in the design of public transport networks was given.

Chapter 4

An Improved Approach to Network Design

In this chapter we present a new heuristic construction method for the network design problem that is used to seed a multi-objective evolutionary algorithm (MOEA). Several problem specific mutation operators are proposed and combined with an NSGAII framework, producing improved Pareto approximate sets compared to state of the art methods from the literature. We have opted for an evolutionary algorithm (EA) due its ability to cope with large structural changes in solutions compared with local search algorithms. Using an EA means that it is possible to make large changes, such as removing or adding an entire route to a solution, by applying a crossover and one or more mutation operators i.e. the offspring is substantially different from both parents before an evaluation takes place. An EA allows for these larger changes in structure, that may be required to remove an undesirable characteristic, that do not fit well with local search methods that utilise a small neighbourhood move. It has been shown by Chakroborty [25] that genetic algorithms (GAs) are "extremely well suited" for the network design problem.

Creation of feasible solutions for the network design problem is in itself complex with multiple constraints that need to be satisfied. Heuristics have been popular in the literature for the creation of solutions, although many opt for generating a large pool of routes. The pool of routes is then used in a pick and mix selection procedure to select a good combination of routes. However, for larger instances, the computation time

involved to generate the pool may be prohibitive.

In our methods we attempt to trade off the passenger and operators cost as defined in Section 2.3. We show that the use of a heuristically generated initial population can provide an improved Pareto set compared with a randomly generated population.

4.1 Heuristic Construction

Before the adoption of metaheuristic methods, heuristic construction was widely used in the literature for this problem. The majority of approaches used heuristic construction to generate one or more candidate solutions using shortest path and/or *k*-shortest path algorithms. Improvement heuristics were then applied in an effort to improve the quality of these. We use the same approach here to generate an initial population for the MOEA. As noted, an alternative approach [124, 64, 29] is to generate a pool of routes and then select a good combination from this pool; however this is less suitable for large instances where the time required to generate the pool can be infeasible.

Shih and Mahmassani [133] have proposed a heuristic procedure for route set generation where routes are continually added until (a) the number of passengers who can reach their destination directly, and (b) the number of vertices serviced by the route set, reach user defined levels. This violates our problem formulation as the number of routes is fixed. We base our approach upon this to produce route sets that obey Constraints (2.4-2.8) whilst balancing the cost to the operator and passenger.

Recall that we are given a graph $G = (V, E, W)$ where $V = \{v_1, \ldots, v_n\}$ is a set of vertices, $E = \{e_1, \ldots, e_m\}$ is a set of edges and $W = \{w_1, \ldots, w_m\}$ is a set of weights that define the cost to traverse edge e_i. We are also given a demand matrix $\mathbf{D}_{n \times n}$ where D_{v_i,v_j} gives the passenger demand between a pair of vertices v_i and v_j.

In our approach W and $\mathbf{D}$ are first normalised such that their values are in the range $[0, \ldots, 1]$ producing W' and $\mathbf{D}'$ respectively. A set of edge weighted graphs $\mathcal{G} =$

$\{G_1, \ldots, G_l\}$ is then created where $G_i = (V, E, W_i)$. Let $w_j^{(i)}$ be the weight of the j^{th} edge of the set W_i. Similarly, let $w_j^{'}$ be the j^{th} edge of the set W'. We set

$$w_j^{(i)} = w_j^{'}\lambda_1 + (1 - D_{u,v}^{'})\lambda_2$$

where the j^{th} edge joins vertices $u, v \in V$. λ_1 and λ_2 are weights specified in advance by the user. In our case we use weights varying from 0 to 1 with an interval of 0.05, giving 441 combinations in total.

For each $G_i \in \mathcal{G}$ we first create a spanning subgraph to ensure that Constraint (2.4) is satisfied. The spanning subgraph is created a route at a time with the intention of minimising the sum of the edge costs using an iterative procedure. In the first iteration the vertex pair (seed pair) with the lowest edge cost is selected. In subsequent iterations the vertex pair with the lowest edge cost is selected such that one vertex is already contained in the spanning subgraph ensuring Constraint (2.7) is satisfied. The remainder of the route is formed from this seed pair using an expansion process that adds the minimum weighted edge incident to one of the two terminal vertices. We prioritise the insertion of edges that result in vertices not present in the spanning subgraph being included. The expansion process continues until the inclusion of a further vertex would cause a constraint to be violated. If all vertices are not contained in the spanning subgraph the procedure is repeated providing $|\mathcal{R}| < r$. In some situations the creation of a spanning subgraph is unsuccessful, i.e. more than r routes are required, in these cases G_i is abandoned and we repeat the process for $G_{i+1} \in \mathcal{G}$. The process for creating the spanning subgraph is given in Algorithm 4.1.

Once a spanning subgraph has been created, if the number of routes in the solution is less than r then an additional procedure is applied. During this stage we utilise the approach of Shih and Mahmassani [133] given in Algorithm 4.2. Vertex pairs (v_i, v_j) that are not yet satisfied directly (i.e. it is not possible to travel from v_i to v_j without having to make a transfer) are extracted from the network and sorted in descending

order based upon the demand D_{v_i,v_j}. Each vertex pair is then taken in order and a *k*-shortest path algorithm is applied, using the weighted edge costs, to determine if a valid route, originating at v_i and terminating at v_j, can be constructed that obeys the problem constraints. We limit the number of 'shortest' paths that are to be explored to ten following the recommendations of Shih and Mahmassani [133]. If a valid route R can be constructed and the cost of the calculated route is less than $\alpha_{v_i,v_j}(\mathcal{R})$ before R's insertion, then R is added to $\mathcal{R}$. This process is applied iteratively until $|\mathcal{R}| = r$ as required by Constraint (2.8).

1: { Valid($\mathcal{R}$) = checks if $\mathcal{R}$ obeys problem constraints excluding number of routes, GenerateRoute($\mathcal{R}$, G) = returns a route generated using Algorithm 4.2 }
2: $\mathcal{R} = \varnothing$
3: **while** $\bigcup_{i=1}^{|\mathcal{R}|} V_{R_i} \neq V$ **do**
4: $R = \varnothing$
5: **if** $|\mathcal{R}|$ is 0 **then**
6: select lowest cost edge in G_i and insert into R
7: **else**
8: select lowest cost edge in G_i that shares a vertex with $\mathcal{R}$ and insert into R
9: **while** $|R| < m_2$ **do**
10: $EDGES\_AVAILABLE$ = edges incident to terminal vertices in R
11: sort $EDGES\_AVAILABLE$ based upon edge weight ascending
12: $EDGE\_INSERTED$ = false
13: **if** $EDGES\_AVAILABLE$ contains a vertex not in $\bigcup_{i=1}^{|\mathcal{R}|} V_{R_i}$ **then**
14: **while** $EDGE\_INSERTED$ is false and $EDGES\_AVAILABLE$ contains a vertex not in $\bigcup_{i=1}^{|\mathcal{R}|} V_{R_i}$ **do**
15: insert lowest cost edge that contains an unseen vertex into R
16: **if** Valid($\mathcal{R}$) **then**
17: $EDGE\_INSERTED$ = true
18: **else**
19: remove inserted edge from R
20: remove edge from $EDGES\_AVAILABLE$
21: **if** $EDGE\_INSERTED$ is false **then**
22: **while** $EDGES\_AVAILABLE \neq \varnothing$ and $EDGE\_INSERTED$ is false **do**
23: insert lowest cost edge into R
24: **if** Valid($\mathcal{R}$) **then**
25: $EDGE\_INSERTED$ = true
26: **else**
27: remove inserted edge from R
28: remove edge from $EDGES\_AVAILABLE$
29: **if** $EDGE\_INSERTED$ is false **then**
30: $break$
31: $\mathcal{R} = \mathcal{R} \cup R$
32: **if** not Valid($\mathcal{R}$) **then**
33: **return** no valid route set found
34: **while** $|\mathcal{R}| \neq R$ **do**
35: $\mathcal{R}$ = $\mathcal{R}$ $\cup$ GenerateRoute($\mathcal{R}$, G_i)
36: **return** $\mathcal{R}$

Algorithm 4.1: Heuristic construction procedure for a route set, $\mathcal{R}$, given a weighted graph G_i.

1: VP = extract vertex pairs not satisfied directly by $\mathcal{R}$
2: sort pairs (v_i, v_j) based upon demand descending
3: **for** (v_i, v_j) $\in VP$ **do**
4: $ROUTES$ = apply *k*-shortest path algorithm to vertex pair p using G
5: **for** $R_i \in ROUTES$ **do**
6: **if** R_i is valid and $\sum_{\forall e_j \in E_{R_i}} w_j < \alpha_{(v_i,v_j)}(\mathcal{R})$ **then**
7: **return** R_i
8: **return** no route found

Algorithm 4.2: Route generation approach of Shih and Mahmassani [133] given a route set, $\mathcal{R}$, and graph G.

4.2 NSGAII

Having used the method discussed in the previous section to generate an initial population, we use NSGAII to evolve the population. NSGAII is an elitist non-dominated sorting MOEA, widely used to solve multi-objective optimisation problems. It has been shown to often find an improved spread of solutions and while converging nearer to the true Pareto-optimal front compared with other Pareto based methods [46] . We have chosen to use NSGAII over NSGAIII as we are dealing with two objective functions in our problem. NSGAIII is designed for any objective optimisation problems which have four or more objectives [44].

The basic form of an NSGAII generation, shown in Algorithm 4.3, proceeds by creating an offspring population of size *N*. This is combined with the parent population of size *N* to produce a population, $P = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{2N}\}$. We define two attributes of a route set $\mathcal{R}_i$: 1) $\mathcal{R}_{i_{\text{rank}}}$ the non-dominated front that $\mathcal{R}_i$ belongs to, and 2) $\mathcal{R}_{i_{\text{dist}}}$ the crowding distance associated with $\mathcal{R}_i$ as defined by Deb et al. [46]. P is sorted such that $\forall\, \mathcal{R}_i, \mathcal{R}_j \in P\ \mathcal{R}_{i_{\text{rank}}} \leq \mathcal{R}_{j_{\text{rank}}}$ and $\mathcal{R}_{i_{\text{dist}}} \geq \mathcal{R}_{j_{\text{dist}}}$ for $i < j$. The successor population is formed by taking the first N solutions in P.

In our case, similarly to Deb et al. [46], a new population is generated using binary tournament selection with a crossover probability of 0.9. For each offspring the number

of mutations is distributed by a binomial random variable with r trials and a success probability of $\frac{1}{r}$.Our proposed mutation operators operate on a route set, as opposed to individual routes.

1: P_0 = generate initial population
2: Evaluate(p) $\forall\ p \in P_0$
3: $t = 0$
4: **repeat**
5: Q = generate new population
6: $P_t = P_t \cup Q$
7: F = NonDominatedSort(P_t)
8: $P_{t+1} = \varnothing$ and i = 0
9: **while** $\mid P_{t+1} \mid + \mid F_i \mid \leq N$ **do**
10: CalculateCrowdingDistance(F_i)
11: P_{t+1} = $P_{t+1} \cup F_i$
12: $i = i + 1$
13: Sort(F_i)
14: $P_{t+1} = P_{t+1} \cup F_i[1 : (N - \mid P_{t+1} \mid)]$
15: $t = t + 1$
16: **until** the stopping condition is satisfied
17: **print** all non-dominated solutions in the final population.

Algorithm 4.3: NSGAII.

4.3 Genetic Operators

NSGAII generates a new population by applying a series of crossover and mutation operators, collectively referred to as genetic operators. A crossover operator usually combines information from two parents to form an offspring, which is then mutated through one or more mutation operators. In the following two sections we introduce our crossover and mutation operators used for the network design problem.

4.3.1 Crossover

We use the crossover operator proposed by Mumford [114], to ensure that the produced offspring solutions obey Constraints (2.4-2.8). Given two parents, the operator con-

structs an offspring, $\mathcal{R}'$, from scratch by alternatively selecting a route from each parent such that the proportion of unseen vertices is maximised, until $|\mathcal{R}'| = r$. First a route is selected at random from one of the parents to seed the offspring. Next a route is selected from the second parent. However, to ensure connectivity the routes eligible for insertion into the offspring are extracted. Eligible routes are defined as those that share one or more vertices in common with those already in the offspring (i.e. a route R_i is only eligible for insertion into $\mathcal{R}'$ if and only if $R_i \cup \bigcup_{j=1}^{|\mathcal{R}'|} V_{R_j} \neq \varnothing$). For each eligible route we then calculate the proportion of unseen vertices defined as $V_{\text{unseen}} = V - V_{\mathcal{R}'}$. Consider a route $R_i = \langle 1, 7, 8, 9, 12, 14 \rangle$ that is contained in one of the parents and is being considered for insertion into $\mathcal{R}'$. If $\mathcal{R}' = \{R_1\}$ where $R_1 = \langle 12, 15, 0, 5, 3 \rangle$ then $V_{\text{unseen}} = R_i - R_1 = \{1, 7, 8, 9, 14\}$. Therefore the proportion of unseen vertices is $\frac{|V_{\text{unseen}}|}{|R_i|} = \frac{5}{6}$ in this case. The route that maximises the proportion of unseen vertices is selected for insertion into the offspring. In the case of one or more routes sharing the same proportion a uniform random choice is made as to which will be inserted into $\mathcal{R}'$. Focus then moves back to the first parent and the above process is repeated until $|\mathcal{R}'| = r$.

After crossover has been applied it is certain that the offspring will be connected although not all the vertices in V may be present, which is a violation of Constraint (2.4). In these cases a repair operator is applied which takes each route in the offspring at random and attempts to add the missing vertices to the terminal vertices of the route, providing valid adjacency relationships exist. If the repair process is unable to add the missing vertices, crossover is abandoned and the next two parents are selected.

4.3.2 Mutation

In our approach eight mutation operators are used. Some of these apply heuristics to mutate the route set in a way that encourages an improvement in quality in at least one of the objectives, while others are intended more as perturbation operators. For each newly created offspring the number of mutations is distributed by a binomial

random variable with r trials and a success probability of $\frac{1}{r}$. Each attempt at mutation selects an operator at random. Mutation must be carefully controlled for this problem to prevent violations of the problem constraints. The names of these mutation operators are *add-nodes*, *del-nodes*, *exchange*, *merge*, *replace*, *remove-overlapping*, *two-opt* and *invert-exchange*, which we now consider in turn.

add-nodes and *del-nodes* were both proposed in [114]. At the start an integer $0 < I \leq \frac{m_2}{2}$ is generated at random denoting the number of vertices to be added or removed from $\mathcal{R}$. Each route $R_i \in \mathcal{R}$ is considered in turn (in a random order) and, in the case of *add-nodes*, $i < I$ vertices are added to the terminal vertices of the route until the addition of a vertex would cause Constraint (2.5) to be violated or result in R_i no longer being a simple path. This process is repeated for each route until I vertices have been added to $\mathcal{R}$ or all routes have been exhausted. The case is the same for *del-nodes*, except I vertices are removed from the ends of routes in $\mathcal{R}$ whilst maintaining feasibility.

The *exchange* operator, as proposed by Mandl [105], selects a single route $R_i \in \mathcal{R}$ at random. Routes that intersect with R_i are then extracted from the route set (Constraint (2.7) ensures that there will be at least one route). One of the extracted routes is then taken at random, together with the initially selected route, and split at the common vertex to produce four paths. The paths are then exchanged to produce two new routes. The two new routes are checked to ensure that $\mathcal{R}$ is feasible and, if not, the routes are disregarded. In this case we take the next route that shares a common vertex and continue with the originally selected route until a valid mutation is produced or until all routes sharing a common vertex are exhausted.

Similar to *exchange*, the *merge* operator, given in Algorithm 4.4 selects a random route and searches the remaining routes to find a route that shares a common terminal vertex. The two routes are then merged to create one continuous route, disregarding one of the common terminal vertices – providing that Constraint (2.5) is not violated and the merged route is a simple path. If successful, the route generation procedure of Shih and Mahmassani [133] described in Section 4.1 is used to generate a new route for insertion.

```
1: { Valid(R) = checks if R obeys problem constraints, GenerateRoute(R) = returns a
   route generated using Algorithm 4.2 }
2: MUTATED = false
3: repeat
4:    R = Choose a route from R at random without replacement
5:    for R_i ∈ R do
6:       if R and R_i share a common terminal vertex then
7:          if |R ∩ R_i| = 1 then
8:             remove R and R_i from R
9:             R_merged = merge R and R_i at the common terminal vertex
10:            R = R ∪ R_merged
11:            R = R ∪ GenerateRoute(R)
12:            MUTATED = true
13:            break
14: until routes exhausted or MUTATED = true
15: if Valid(R) then
16:    return R
17: else
18:    return incumbent solution
```

Algorithm 4.4: Merge operator for mutating a routeset $\mathcal{R}$.

The *replace* operator, Algorithm 4.5 removes a route $R_i \in \mathcal{R}$ that satisfies the least passenger demand in $\mathcal{R}$. A replacement route is then generated using the route generation procedure of Shih and Mahmassani [133] as before. The purpose of the *replace* mutation is to replace routes that serve a relatively low demand with a hopefully high demand route. Note that *replace* can create infeasible solutions if the removed route acts as a transfer hub for other routes, i.e. the route set is only connected when the removed route is present. If this situation occurs the repair procedure used during crossover is applied and, if successful, the mutated solution is returned, else the incumbent is returned.

The *remove-overlapping* operator replaces a route R_i that is a subsequence of another route R_j. If such a route is discovered it is removed and the route generation procedure of Shih and Mahmassani [133] described previously is used to produce a replacement. Replacing the route provides the operator with the ability to remove duplicate services and use these resources to serve other passenger demands.

1: { Valid($\mathcal{R}$) = checks if $\mathcal{R}$ obeys problem constraints, GenerateRoute($\mathcal{R}$) = returns a route generated using Algorithm 4.2 }
2: Calculate passenger demand served directly by each route
3: Remove the route that serves the lowest demand from $\mathcal{R}$
4: $\mathcal{R}$ = $\mathcal{R}$ $\cup$ GenerateRoute($\mathcal{R}$)
5: **if** not Valid($\mathcal{R}$) **then**
6: Repair($\mathcal{R}$)
7: **if** not Valid($\mathcal{R}$) **then**
8: **return** incumbent solution
9: **return** $\mathcal{R}$

Algorithm 4.5: Replace operator for mutating a routeset $\mathcal{R}$.

1: { Valid($\mathcal{R}$) = checks if $\mathcal{R}$ obeys problem constraints, GenerateRoute($\mathcal{R}$) = returns a route generated using Algorithm 4.2 }
2: MUTATED = false
3: **repeat**
4: R = choose a route from $\mathcal{R}$ at random without replacement
5: **for** $R_i \in \mathcal{R}$ **do**
6: **if** R is a subsequence of R_i **then**
7: remove R from $\mathcal{R}$
8: $\mathcal{R}$ = $\mathcal{R}$ $\cup$ GenerateRoute($\mathcal{R}$)
9: MUTATED = true
10: $break$
11: **until** routes exhausted or MUTATED = true
12: **if** Valid($\mathcal{R}$) **then**
13: **return** $\mathcal{R}$
14: **else**
15: **return** incumbent solution

Algorithm 4.6: Remove-overlapping operator for mutating a routeset $\mathcal{R}$.

two-opt, proposed in 1958 by Croes [39] for use with the travelling salesman problem, selects two vertices at random in a route and inverts the order of the vertices between them. Here, a route is selected at random without replacement until a feasible mutation is produced or the routes are exhausted. As we do not have complete graphs the inversion of the vertices can lead to the production of infeasible solutions. By inverting a sequence of vertices we seek to reduce travel time between vertices on the same route.

Finally the *invert-exchange* mutation operator, given in Algorithm 4.7, selects two routes at random and two random index locations valid for both routes. The vertices between

the two random index locations are then inverted and exchanged between the two routes. For example, given two routes $R_1 = \langle 3, 5, 8, 10, 12, 15 \rangle$ and $R_2 = \langle 1, 6, 9, 8, 11, 7 \rangle$ with the selected indices of *3* and *5*. We invert everything in R_1 between the indices giving $\langle 3, 5, 12, 10, 8, 15 \rangle$ then replace the vertices in R_2 between the indices with the inverted section from R_1. In this case the resultant two routes would be $R_3 = \langle 3, 5, 11, 8, 9, 15 \rangle$ and $R_4 = \langle 1, 6, 12, 10, 8, 7 \rangle$. *Invert-exchange* attempts to decrease the travel time between vertices and prevent passengers having to make transfers. Similarly to *two-opt* there is a high possibility that the majority of routes created using this approach will be infeasible. As such, two routes are continually chosen at random until a feasible solution has been found or the routes have been exhausted.

1: { Valid($\mathcal{R}$) = checks if $\mathcal{R}$ obeys problem constraints }
2: MUTATED = false
3: **repeat**
4: R_1 = choose a route from $\mathcal{R}$ at random without replacement
5: R_2 = choose a route from $\mathcal{R}$ at random without replacement
6: INDEX_START = generate random index location
7: INDEX_END = generate random index location > INDEX_START
8: R_1^{invert} = invert vertices in R_1 between INDEX_START and INDEX_END
9: R_2^{invert} = invert vertices in R_2 between INDEX_START and INDEX_END
10: $R_1[INDEX\_START : INDEX\_END] = R_2^{invert}$
11: $R_2[INDEX\_START : INDEX\_END] = R_1^{invert}$
12: **if** Valid($\mathcal{R}$) **then**
13: MUTATED = true
14: **else**
15: revert changes made to $\mathcal{R}$
16: **until** routes exhausted or MUTATED = true
17: **return** $\mathcal{R}$

Algorithm 4.7: Invert-exchange operator for mutating a routeset $\mathcal{R}$.

4.4 Measuring Population Diversity

Population diversity defines how different the individuals in a population are. Many evolutionary algorithms lose diversity through premature convergence and thus get

trapped in a local optima. Measuring the similarity between solutions requires a different distance measurement based upon the solution representation used [66]. Several, alternative, methods exist for measuring the similarity between two solutions based upon their representation e.g. Hamming distance, Sorensen coefficient, Jaccard coefficient and Cosine coefficient. We propose a similarity measure between two route sets expressed as real-coded variables as follows.

Given two solutions, for example, $\mathcal{R}_1 = \{\langle 1,2,3,4\rangle, \langle 5,2,6\rangle\}$ and $\mathcal{R}_2 = \{\langle 1,2,3,4\rangle, \langle 5,2,3,6\rangle\}$ we define two multisets that contain the edges present in each solution. In our example this will produce the multisets $E_{\mathcal{R}_1} = \{\{1,2\},\{2,3\},\{3,4\},\{5,2\},\{2,6\}\}$ and $E_{\mathcal{R}_2} = \{\{1,2\},\{2,3\},\{3,4\},\{5,2\},\{2,3\},\{3,6\}\}$. $E_{\mathcal{R}_2}$ contains the edge $\{2,3\}$ twice. This is an important feature of the solution and should be present due to the use of multisets.

The distance between $E_{\mathcal{R}_1}$ and $E_{\mathcal{R}_2}$ is calculated as $1 - S(E_{\mathcal{R}_1}, E_{\mathcal{R}_2})$ where $S(E_{\mathcal{R}_1}, E_{\mathcal{R}_2})$ is the Sorensen/Dice measure, calculated:

$$S(E_{\mathcal{R}_1}, E_{\mathcal{R}_2}) = \frac{2|E_{\mathcal{R}_1} \cap E_{\mathcal{R}_2}|}{|E_{\mathcal{R}_1}| + |E_{\mathcal{R}_2}|} \tag{4.1}$$

In our example this gives us a similarity measure of $\frac{3}{11}$ as the solutions are very similar. To calculate the diversity of a population, P, we consider the average distance between each pair of solutions, calculated:

$$\frac{\sum_{\forall \mathcal{R}_1, \mathcal{R}_2 \in P} S(E_{\mathcal{R}_1}, E_{\mathcal{R}_2})}{\binom{|P|}{2}} \tag{4.2}$$

4.5 Experimental Method

Our first experiment will show how the algorithm of Mumford [114] (Algorithm A), based on the SEAMO2 [146] framework, can be improved, by seeding the MOEA with

our heuristically generated population (Algorithm B). Using our heuristic construction procedure, a subset of unique solutions are randomly selected for insertion into the initial population. Randomly generated solutions are then used to top-up the initial population if there are too few heuristic solutions. These were created using the same approach as Mumford [114] to seed the MOEA which we now summarise.

A route set, $\mathcal{R}$, is generated by selecting a vertex, $v \in V$, at random to seed the first route. For the remaining routes a seed vertex is selected at random from those already in $\mathcal{R}$. A desired route length is then generated between m_1 and m_2. The route terminals of the current route under construction are augmented by adding a random adjacent vertex until the desired route length is achieved. If the desired route length cannot be achieved, i.e. adding another vertex would cause a constraint to be violated, the attempt is abandoned and a new seed vertex selected.

In our second experiment we look at the effects of adding our proposed mutation operators (Algorithm C) to Mumford's original *add-nodes* and *del-nodes* operators together with heuristic seeding. A comparison is then made between SEAMO2 and NSGAII (Algorithm D).

The effect of population size is investigated in our third experiment by varying the initial population from 100 to 500 solutions while keeping the number of generations fixed. We look into population diversity and the effect this has on the final $\mathcal{S}$ metric, by preventing duplicate solutions from entering the population (Algorithm E).

Finally, we perform convergence testing using a population size of two hundred evolved for a thousand generations examining the population diversity, Pareto set size and $\mathcal{S}$ metric conversion as the population is evolved.

Unless stated otherwise, all experiments use a population of two hundred solutions evolved for two hundred generations. Final Pareto sets are achieved by combining the results from twenty independent runs. Each algorithm is now summarised below:

- Algorithm A – SEAMO2 framework using the crossover and mutation operators

put forward by Mumford [114]. The initial population is randomly generated.

- Algorithm B – SEAMO2 framework using the crossover and mutation operators put forward by Mumford [114]. The initial population is seeded using our heuristic construction procedure.
- Algorithm C – SEAMO2 framework using Mumford's [114] crossover operator and our proposed mutation operators. The initial population is seeded using our heuristic construction procedure.
- Algorithm D – NSGAII framework using Mumford's [114] crossover operator and our proposed mutation operators. The initial population is seeded using our heuristic construction procedure.
- Algorithm E – NSGAII framework using Mumford's [114] crossover operator and our proposed mutation operators. The initial population is seeded using our heuristic construction procedure. The population contains only unique solutions.

4.6 Experimental Results

Firstly, we augment Mumford's algorithm with our heuristic method for generating the initial population (Algorithm B). Table 4.1 presents the best solutions from the passenger and operator perspective compared to the findings of Mumford [114]. We see that our heuristic is clearly beneficial, producing an improvement over all the instances. A small improvement in passenger objective can result in a substantial number of passengers requiring less transfers. For example, an improvement of almost two minutes in the passenger objective for Mumford3 results in 96% of passengers requiring at most one transfer to reach their destination compared with only 78% using Mumford's method. We also note that where an improvement is made in the objective value, either from the passenger or operator perspective, we can see that an improvement is made to the other objective on all but the Mandl and Mumford0 instances from the operator perspective.

Note that an improvement from the operator perspective on Mandl's instance is not possible, as 63 is the lower bound calculated by Mumford [114] for this instance. However, a decrease in the passenger objective is still made compared with Mumford's findings.

		Mandl	Mumford0	Mumford1	Mumford2	Mumford3
Best for passenger	F_1	**10.25** (10.33)	**15.47** (16.05)	**23.65** (24.79)	**26.82** (28.65)	**29.23** (31.44)
	F_2	213 (224)	718 (759)	1928 (2038)	5379 (5632)	6554 (6665)
	d_0	95.83 (94.54)	69.94 (63.20)	43.37 (36.60)	36.47 (30.92)	37.21 (27.46)
	d_1	4.17 (5.14)	30.06 (35.82)	52.81 (52.42)	60.00 (51.29)	58.34 (50.97)
	d_2	0.00 (0.32)	0.00 (0.98)	3.81 (10.71)	5.53 (16.36)	4.45 (18.76)
	d_{un}	0.00 (0.00)	0.00 (0.00)	0.00 (0.26)	0.00 (1.44)	0.00 (2.81)
Best for operator	F_1	13.48 (15.13)	33.19 (32.40)	33.84 (34.69)	32.68 (36.54)	36.66 (36.92)
	F_2	**63** (63)	**97** (111)	**458** (568)	**1837** (2244)	**2276** (2830)
	d_0	70.91 (59.34)	20.82 (18.42)	19.04 (16.35)	18.58 (13.76)	18.17 (16.71)
	d_1	25.50 (30.57)	17.86 (23.40)	44.10 (29.06)	48.11 (27.69)	42.47 (33.69)
	d_2	2.95 (9.06)	26.93 (20.78)	30.53 (29.93)	27.91 (29.53)	32.55 (29.18)
	d_{un}	0.64 (1.03)	34.38 (37.40)	6.33 (24.66)	5.40 (29.02)	6.82 (20.42)

Table 4.1: Best objective values extracted using heuristic seeding for the initial population (Algorithm B). Mumford's [114] results are given in brackets. Passenger demand satisfied in zero transfers, one transfer, two transfer or unsatisfied (i.e. more than two transfers) is given by d_0, d_1, d_2 and d_{un} respectively.

Using Algorithm B, we now examine the effect of further augmenting the algorithm using our proposed mutation operators. Comparing $\mathcal{S}$ metric values for Algorithms B and C (Table 4.2), we can see that an improvement is achieved for the Mandl, Mumford0, Nottingham100 and Edinburgh200 instances, with small decreases in $\mathcal{S}$ metric values for the remaining instances. If the Pareto sets are plotted for the larger instances, (Figure 4.1), it can be seen that there is an improvement in the passenger objective for the majority of solutions. However, we struggle to make improvements in the extremes of the operator objective. By exploring the extremes of the operator objective it is unlikely that these solutions would be chosen by a human decision maker due to their poor quality in respect to the passenger objective. We note that when introducing our mutation operators there is an increased rate of failed mutations compared with using *add-nodes* and *del-nodes* alone – this will be covered in a greater depth in Section 4.7. If we modify SEAMO2 to apply a maximum of r attempts at mutation where each

attempt selects a mutation operator at random, we note an improvement in $\mathcal{S}$ metric over five of the seven instances. This is shown in Table 4.3 where our modified version of SEAMO2 is named Algorithm C'.

Given the popularity of NSGAII and its stated ability to produce a Pareto set closer to the true Pareto-optimal front compared with other Pareto based methods [46], it was used in place of SEAMO2 in our third set of experiments. We use our mutation operators with heuristic seeding and Mumford's crossover operator. As mentioned earlier, a probability of crossover and mutation of 0.9 and $\frac{1}{r}$ respectively are used. A comparison of $\mathcal{S}$ metric values, (Table 4.2), shows NSGAII gives an improvement over all the problem instances. This is displayed graphically in Figure 4.1. These improvements can be attributed to the following: 1) a higher selection pressure compared with SEAMO2 due to the use of Pareto ranking and crowding distance, and 2) increased rate of mutation, SEAMO2 attempts to apply a single mutation to an offspring whereas NSGAII will attempt to apply r mutations with a probability of $\frac{1}{r}$, leading to a greater exploration of the search space.

Instance	Algorithm A	Algorithm B	Algorithm C	Algorithm D
Mandl	16679$^{\triangle\triangle}$[1]	16676	16684	**16696****[2]
Mumford0	159688$^{\triangle\triangle}$	159589	159891	**160717****
Mumford1	689832	698028$^{\triangle\triangle}$	696732	**704270****
Mumford2	4878053	4958426$^{\triangle\triangle\dagger}$[3]	4943131	**5007652****
Mumford3	15571117	15820123$^{\triangle\triangle\dagger\dagger}$	15781607	**15977073****
Nottingham100	691080	693072$^{\triangle\triangle}$	695255†	**703263****
Edinburgh200	10910282	10916727$^{\triangle\triangle}$	10928707††	**10956064****

Table 4.2: $\mathcal{S}$ metric comparison over the five benchmark problems for our proposed modifications.

We now examine the effect of varying the population size on the $\mathcal{S}$ metric value. Figure 4.2 shows the change in the $\mathcal{S}$ metric as the population size is increased. For Mandl,

[1] $\triangle\triangle$ indicates statistical significance between Algorithm A and B according to a Related-Samples Wilcoxon Signed Rank test at the $p < 0.01$ level.

[2] ** indicates statistical significance between Algorithm C and D according to a Related-Samples Wilcoxon Signed Rank test at the $p < 0.01$ level.

[3] Indicates statistical significance between Algorithm B and C according to a Related-Samples Wilcoxon Signed Rank test at the $\dagger$ $p < 0.05$ and $\dagger\dagger$ $p < 0.01$ level.

Instance	Algorithm C	Algorithm C'
Mandl	16684	**16685**
Mumford0	159891	**159892**
Mumford1	**696732**	698557
Mumford2	**4943131**	4935725
Mumford3	15781607	**15782044**
Nottingham100	695255	**695944**
Edinburgh200	**10928707**	10928356

Table 4.3: $\mathcal{S}$ metric comparison between SEAMO2 when making a single attempt at mutation, Algorithm C, and r attempts at mutation, Algorithm C'.

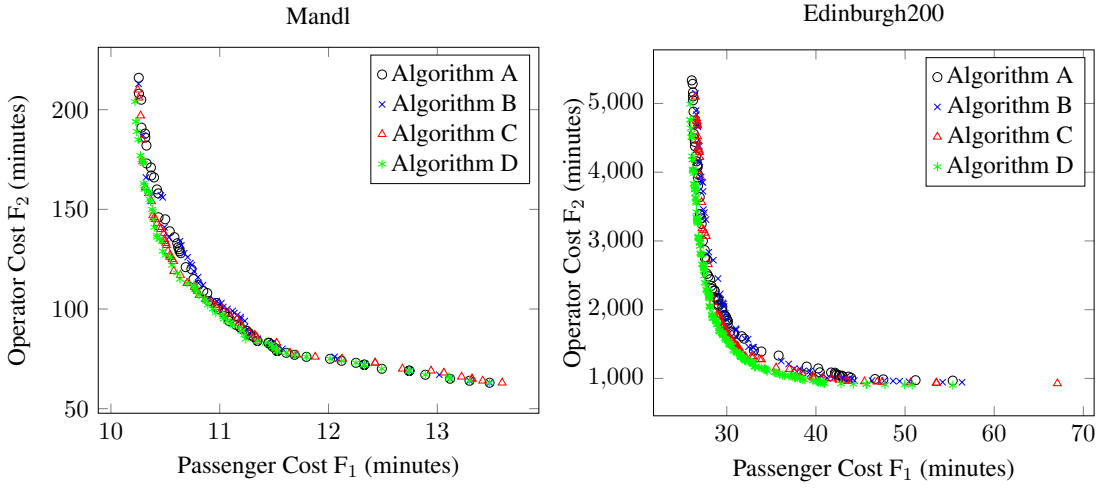


Figure 4.1: Combined Pareto fronts extracted from twenty runs for the smallest and largest benchmark instances using Algorithms A-D.

Mumford0, Mumford1, Nottingham100 and Edinburgh200, the plots demonstrate that an increase in metric can be achieved by increasing the population size whilst keeping the number of generations executed constant. Mumford2 and Mumford3 are more variable with increases in $\mathcal{S}$ metric for population sizes of 300 and 400, but a decrease once the population size reaches 500.

Population diversity is now investigated by preventing duplicate solutions from entering the population. We define a duplicate solution to be that which shares the same routes as that of another solution in the population i.e. the distance between the two solutions is zero. In this case we allow solutions that share the same objective values but have a different route set configuration to remain in the population. Table 4.4 shows

that preventing duplicates from entering the population produces an increase in the combined $\mathcal{S}$ metric for all the instances apart from Mumford3 and Edinburgh200. If we consider the twenty replicate runs there is no evidence to suggest that the differences between Algorithms D and E are statistically significant in terms of the $\mathcal{S}$ and Schott spacing metrics. However, when examining the diversity of the population, as shown in Figure 4.3, it is clear that preventing duplicate solutions helps to increase the diversity. For this reason, Algorithm E is selected over Algorithm D in the remainder of the experiments.

Instance	Algorithm D		Algorithm E	
	$\mathcal{S}$ Metric	Schott Metric	$\mathcal{S}$ Metric	Schott Metric
Mandl	16698	1.1500	**16701**	0.8011
Mumford0	160800	19.1716	**160913**	9.3842
Mumford1	703350	83.9315	**703880**	114.0468
Mumford2	5002556	340.9535	**5013202**	315.3071
Mumford3	**16011824**	524.8114	15999158	404.7256
Nottingham100	703263	75.4259	**705353**	50.5586
Edinburgh200	**10956064**	361.8055	10952858	337.8293

Table 4.4: $\mathcal{S}$ metric comparison over the five benchmark with duplicates allowed and duplicates prevented from entering the population.

Figure 4.4 shows the variation in $\mathcal{S}$ metric, Pareto set size and diversity over a thousand generations for the Mandl and Edinburgh instances. From the Mandl instance, we can see that the algorithm has almost converged after two hundred generations and the diversity of the population has dropped significantly. Although we do not achieve a continual increase in $\mathcal{S}$ metric over the generations, the size of the Pareto set continually increases, suggesting that we are still exploring the search space effectively. As the $\mathcal{S}$ metric does not significantly change over this period, it is suggestive that more solutions are being found along the Pareto front rather than making significant gains in terms of objective values. Similar trends are noted for the Mumford0 and Mumford1 instances.

Different conclusions are drawn from the Edinburgh instance, which is significantly larger than Mandl's instance. There are relatively large gains made in terms of $\mathcal{S}$ metric after two hundred generations. Although, the Pareto set size reaches its maximum size

after approximately four hundred generations. Reaching the ceiling of the population size indicates that more significant gains may still be able to be made if the population size was increased. Albeit small gains in the metric are still realised when the ceiling is reached. However, the diversity of the population plateaus upon reaching the population ceiling. When compared with the results from the population size experiment, it seems that, for larger instances, an increase in both population size and number of generations executed is warranted.

Given the increase in $\mathcal{S}$ metric that is achieved by extending the number of generations used, it seems beneficial to run the algorithm for longer. However, the running times for the algorithm are long and require the use of a high performance cluster to obtain acceptable running times. For example, the average CPU time needed for Edinburgh200 to complete a thousand generations was twenty four days. It therefore seems sensible in this work to limit the number of generations to the originally used two hundred and realising that gains can still be made on larger problem instances. Although, the largest increase in metric value is achieved in the first two hundred generations after which the rate of increase decelerates.

4.7 Genetic Operator Analysis

This section details the performance of our genetic operators. To analyse their effectiveness we performed 20 independent runs for each problem instance using a population size of 200 evolved for 200 generations. Two criteria for assessment are applied for an operator: 1) feasibility, and 2) improvement. We deem a 'successful' application of an operator to mean that the parents (crossover) or parent (mutation) produce an offspring that complies with all the constraints listed in Section 2.3, i.e., a feasible solution is produced. An 'improved' solution refers to a feasible offspring that dominates its parent (mutation) or parents (crossover).

Tables 4.5 and 4.6 show the percentage of successful applications of each operator and

the percentage of 'improved' or dominating solutions produced respectively. Table 4.6 also shows the percentage of offspring that are produced that possess a mutually non-dominating status with respect to their parent or parents. All results are averaged over the 20 runs.

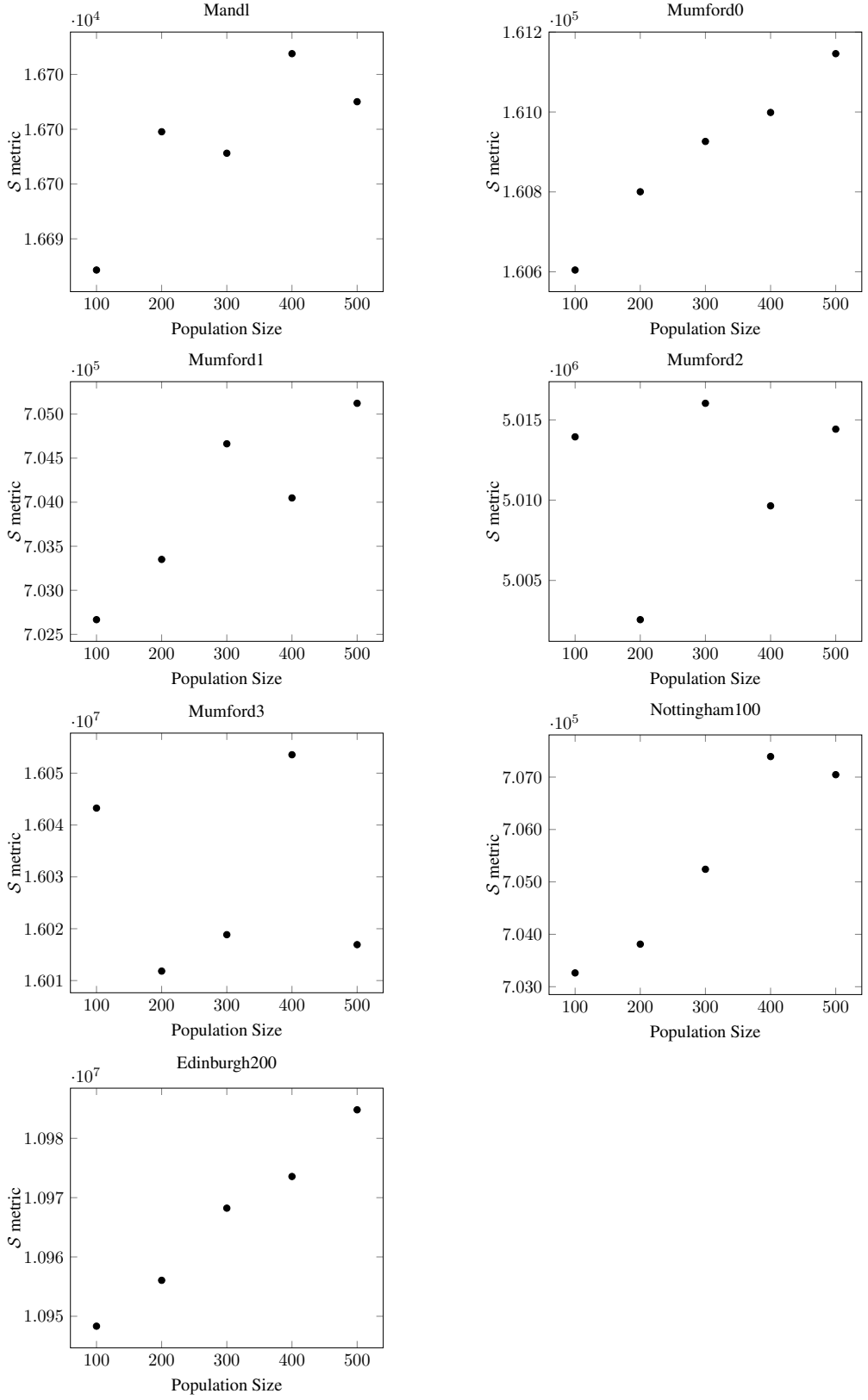


Figure 4.2: Plot of effect on $\mathcal{S}$ metric as the population size is increased.

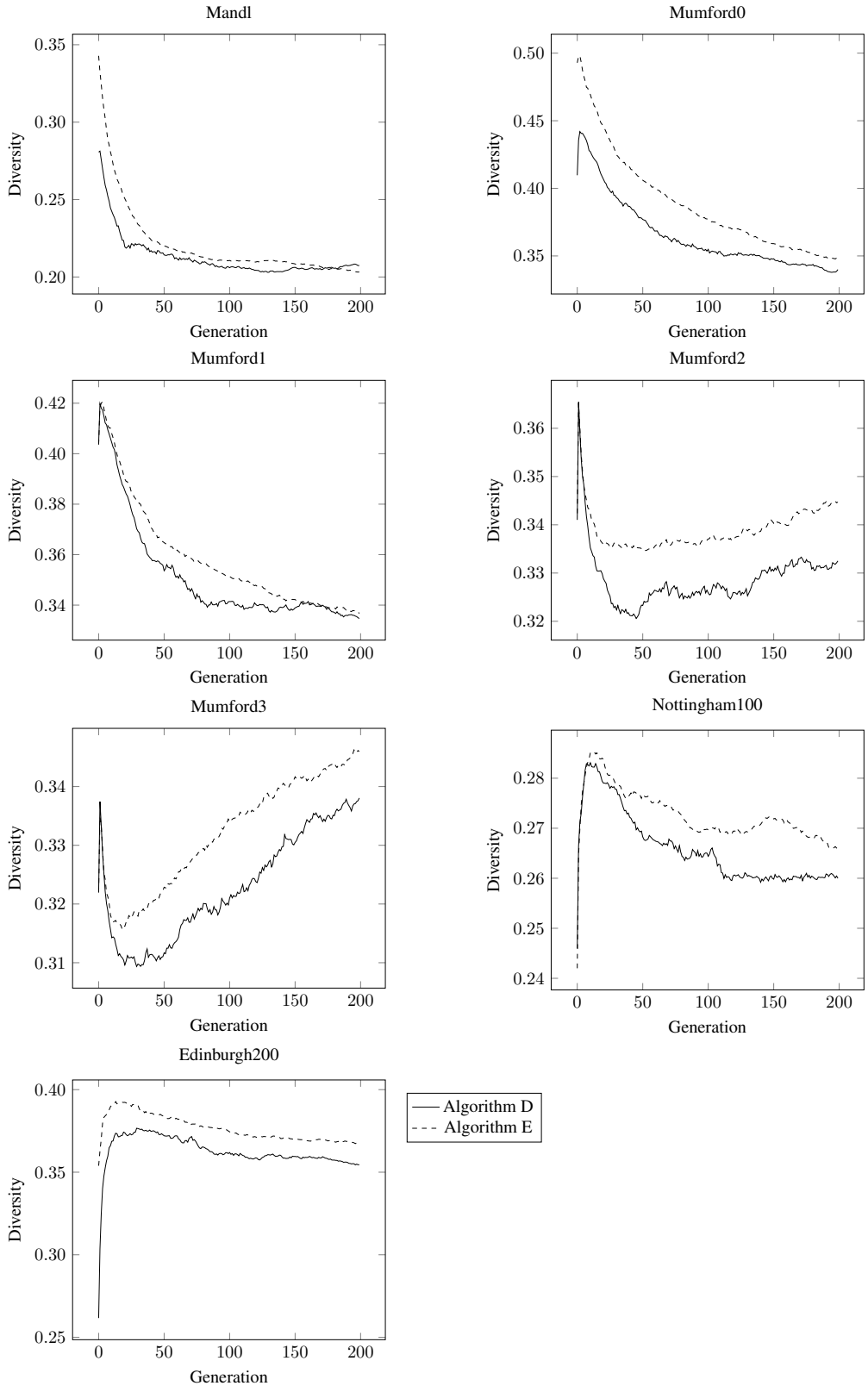


Figure 4.3: Comparison between population diversity using Algorithm D and Algorithm E.

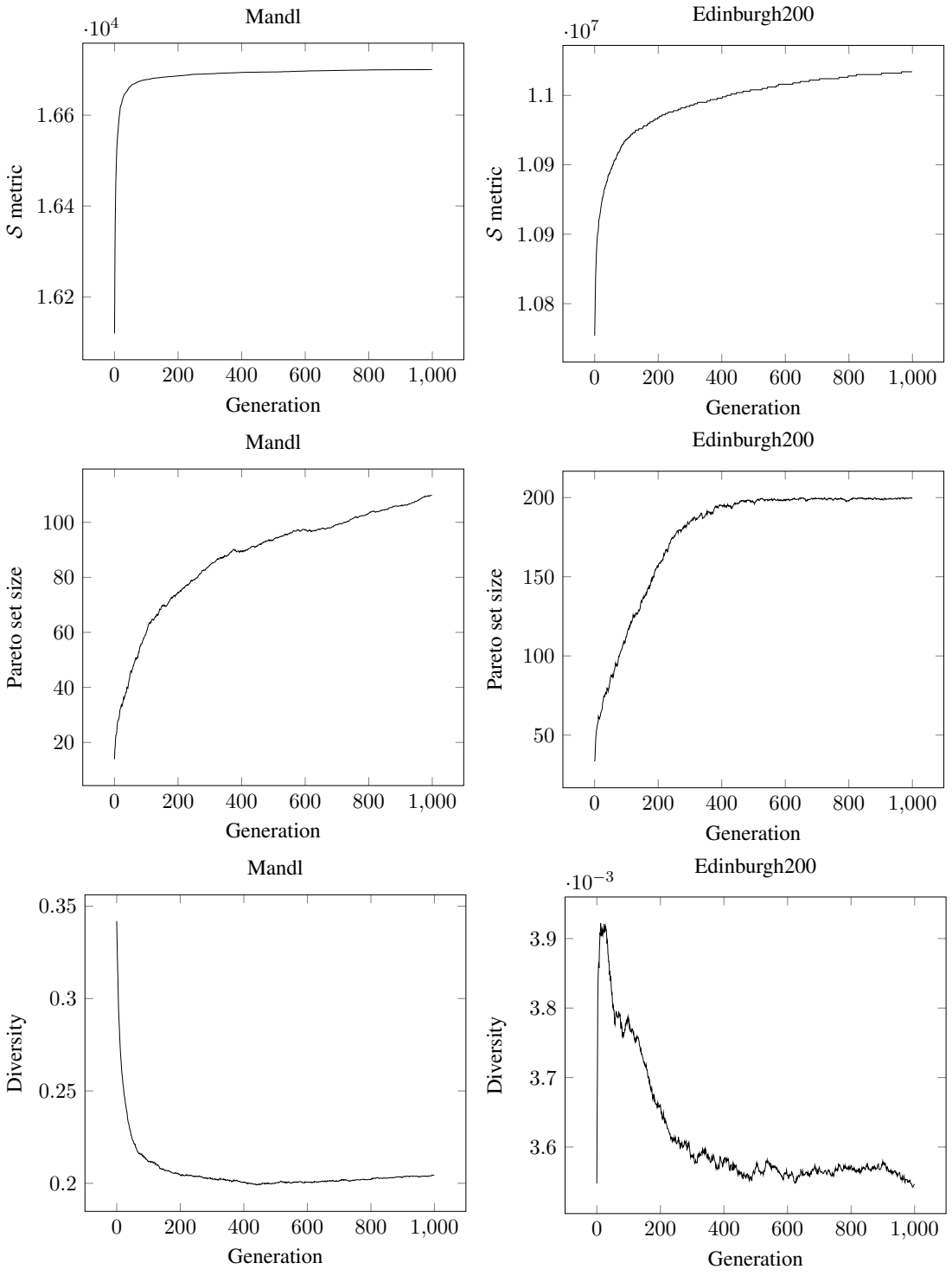


Figure 4.4: $\mathcal{S}$ metric, Pareto set size and diversity for Mandl and Edinburgh200.

Operator	Mandl	Mumford0	Mumford1	Mumford2	Mumford3	Nottingham100	Edinburgh200
crossover	89	92	**100**	**100**	**100**	**100**	87
add-nodes	**97**	**99**	99	98	98	98	98
del-nodes	94	96	86	84	82	91	96
replace	80	**99**	95	**100**	**100**	85	90
remove-overlapping	71	93	70	96	97	97	**100**
exchange	92	**99**	**100**	**100**	**100**	**100**	**100**
merge	40	78	50	91	91	79	94
two-opt	0	18	25	71	76	14	33
invert-exchange	4	8	10	76	82	48	69

Table 4.5: Percentage of applications of the genetic operators that were successful. Best performing operator per instance is highlighted in bold.

Operator	Mandl	Mumford0	Mumford1	Mumford2	Mumford3	Nottingham100	Edinburgh200
crossover	0 (48)	0 (70)	0 (65)	0 (65)	0 (68)	0 (66)	0 (77)
add-nodes	0 (96)	0 (100)	0 (100)	0 (100)	0 (100)	0 (100)	0 (100)
del-nodes	3 (97)	1 (99)	0 (100)	0 (100)	0 (100)	0 (100)	0 (100)
replace	16 (80)	2 (88)	21 (59)	19 (67)	25 (60)	31 (57)	30 (63)
remove-overlapping	**38** (61)	17 (80)	**48** (51)	38 (61)	34 (65)	**55** (43)	25 (73)
exchange	17 (44)	**33** (12)	32 (12)	**40** (15)	**41** (15)	39 (22)	**34** (28)
merge	0 (100)	0 (100)	0 (100)	0 (100)	0 (100)	0 (100)	0 (100)
two-opt	0 (0)	13 (27)	12 (34)	21 (43)	16 (51)	18 (32)	27 (26)
invert-exchange	2 (50)	7 (50)	7 (52)	15 (60)	14 (64)	9 (81)	6 (86)

Table 4.6: Percentage of dominating solutions produced from a successful application of the genetic operator. The percentage of mutually nondominating solutions is given in brackets. Best performing operator per instance, in terms of dominating solutions produced, is highlighted in bold.

As we saw earlier, the crossover operator is deemed to have failed if one or more of the vertices is not contained in a route following repair. However, this is rare with the larger problem instances, with success rates of 100% recorded. When repair *is* required (i.e. when crossover produces an infeasible offspring) we found the procedure was able to produce a feasible solution in 24% of the cases. With regards to the creation of dominating offspring, crossover performs poorly, producing no dominating offspring, though the majority of offspring produced are non-dominating with respect to both parents, as detailed in Table 4.6. Despite this, crossover is beneficial to the algorithm overall as improved $\mathcal{S}$ metrics are achieved for six out of the seven instances. This is shown in Table 4.7 where we compare the performance of our MOEA with and without the crossover operator.

Instance	Without	With
Mandl	16697	**16703** *[4]
Mumford0	160981	**161067** *
Mumford1	694780	**703141** **
Mumford2	4900654	**5006569** **
Mumford3	15583717	**16056110** **
Nottingham100	695126	**705353** **
Edinburgh200	10920957	**10952858** **

Table 4.7: $\mathcal{S}$ metric achieved with and without crossover using our proposed algorithm. Asterisks indicate statistical significance according to a Paired samples t-test at the ** $p < 0.01$ level.

With regards to mutation, *add-nodes* and *del-nodes* show similar behaviour to crossover with high success rates. *add-nodes* has a higher success rate compared to *del-nodes* as there is no risk of removing vertices that are required for route set connectivity. A success rate of 98% is seen across the majority of instances for *add-nodes*, while *del-nodes* is more variable with success rates in the range 82-96%. However we find that both *add-nodes* and *del-nodes* are unable to produce dominating solutions here. This is due to their design, with the operators tending to improve one objective while simultaneously degrading the other. For example, *del-nodes* will improve F_2 as we are

[4]Related-Samples Wilcoxon Signed Rank Test where * = $p < 0.05$

reducing the length of the routes; in turn this will degrade F_1 as passengers may now have to make vehicle transfers to reach their destination thereby incurring a penalty. As such, the percentage of non-dominating solutions produced for both operators is almost identical to the success rates. However, the success rates of *add-nodes* and *del-nodes* diverge as the evolutionary process progresses and the population as a whole improves. *add-nodes* has a relatively constant success rate whereas the success rate for *del-nodes* tends to decrease over time. This is reflective of the improvement in solution quality as the population is evolved.

Table 4.6 shows that the *exchange* operator is able to produce dominating solutions in 32-41% of the successful mutations applied for the Mumford problem instances, achieving more dominating solutions than mutually non-dominating solutions. The number of dominating solutions produced implies that exchange is able to reduce the passenger cost by reducing the number of transfers that are required whilst incurring a decrease in the operator cost. *Merge*, however, displays similar behaviour to *add-nodes* and *del-nodes* where a successful mutation is generally unable to create a dominating solution. Success rates of 40%, 70%, 50%, 91% and 91% are recorded over the Mumford problem instances and 79% and 94% for Nottingham100 and Edinburgh200 respectively. In the case of a successful merge, a non-dominating solution is expected because joining two routes will benefit passengers by reducing travel times. The generation of a replacement route satisfying the highest as yet unsatisfied demand vertex pair is also of further benefit to the passenger. This degrades the operator objective as the new route will incur extra distance that the operator must traverse.

remove-overlapping also shows interesting behaviour on the larger problem instances. In the first ten generations the operator is able to produce more dominating solutions than mutually non-dominating solutions. Success rates are also quite variable across the three smaller instances, although on the larger instances they are similar. We believe that this is due to the higher number of routes coupled with the increase in allowed route length that creates a suitable environment for duplication of vertices in multiple

routes. Creation of dominating solutions is relatively high compared with the other operators with averages of 38%, 17%, 48%, 38% and 34% over Mandl to Mumford3 with non-dominating solutions produced at rates of 61%, 80%, 51%, 61% and 65%.

Finally *two-opt* and *invert-exchange* both perform very poorly for the smaller instances although they do show some improvement for the larger instances. This is due to the constraints that need to be satisfied to allow either to be applied successfully. As the number of routes present in a solution increases so does the number of possible mutations that can be applied to the route set thus a greater likelihood of a feasible mutation. With respect to dominating solutions *invert-exchange* shows a better performance with 50%-86% of mutually non-dominating solutions produced compared with 0%-51% for *two-opt*.

Overall we can see from Tables 4.5 and 4.6 that as the problem size increases so does the performance of the proposed operators. The creation of dominating solutions is challenging given the multi-objective nature of the problem and design of the operators since each operator tends to focus on improving one particular objective while degrading the other. As such we believe that using all the operators together is the best approach due to the aforementioned design choice and the percentage of both dominating and non-dominating solutions produced. Further investigation into the effectiveness of each operator when combined would be an interesting topic for future work.

4.8 Comparative Results

In this section we now compare our proposed algorithm, Algorithm E, against the state of the art [114, 27, 115, 92]. It should be noted that we have only considered those methods that use a similar problem formulation to allow a direct comparison. We use an initial population of 200 solutions generated using our heuristic construction algorithm and evolve the population for 200 generations. 20 independent runs are combined to form an approximate Pareto set.

Tables 4.8 and 4.9 detail the best solutions found for each method for the passenger and operator viewpoints, respectively. As demonstrated a direct comparison is difficult as many authors only use Mandl's instance, or their problem formulations only consider one objective function. However, our proposed approach is able to achieve the lowest objective values observed from the passenger and operator perspective for all but Mandl's instance. Nayeem et al. [115] are able to achieve a lower passenger objective for Mandl's instance, though they have formulated their problem to focus solely on the passenger objective.

Tables 4.8 and 4.9 show that comparisons are generally made using the best performing solution from the passenger and operator perspectives. Given the inherent multi-objective nature of the network design problem, this method of comparison is inadequate. However, as far as we are aware, published Pareto sets from any of the publicly available instances used in this thesis do not exist. Clearly, the development of methods to tackle the network design problem will benefit greatly from the publication of reference Pareto sets allowing alternative methods to be compared on their multi-objective performance, rather than the current single-objective scenario.

Instance		Mumford [114]	Chew et al. [27]	Nayeem et al. [115]	Kılıç and Gök [92]	Our Method
Mandl	F_1	10.33	10.21	**10.10**	10.29	10.19
	F_2	224	224	-	216	217
	CPU Time	-	70	-	-	15
Mumford0	F_1	16.05	-	-	14.99	**14.96**
	F_2	759	-	-	707	668
	CPU Time	-	-	-	-	329
Mumford1	F_1	24.79	-	23.96	23.33	**23.04**
	F_2	2038	-	-	1944	1897
	CPU Time	-	-	-	-	6538
Mumford2	F_1	28.65	-	26.63	26.82	**26.20**
	F_2	5632	-	-	5027	5360
	CPU Time	-	-	-	-	174370
Mumford3	F_1	31.44	-	29.65	30.41	**28.79**
	F_2	6665	-	-	5834	6519
	CPU Time	-	-	-	-	315282
Nottingham100	F_1	-	-	-	-	**22.62**
	F_2	-	-	-	-	2761
	CPU Time	-	-	-	-	73847
Edinburgh200	F_1	-	-	-	-	**25.71**
	F_2	-	-	-	-	5727
	CPU Time	-	-	-	-	351676

Table 4.8: Best objective values obtained across all runs from the passenger perspective. CPU time given in seconds.

Instance		Mumford [114]	Chew et al. [27]	Our method
Mandl	F_1	15.13	13.48	13.48
	F_2	**63**	**63**	**63**
Mumford0	F_1	32.40	-	30.33
	F_2	111	-	**94**
Mumford1	F_1	34.69	-	47.53
	F_2	568	-	**445**
Mumford2	F_1	36.54	-	39.14
	F_2	2244	-	**1699**
Mumford3	F_1	36.92	-	47.38
	F_2	2830	-	**1879**
Nottingham100	F_1	-	-	35.79
	F_2	-	-	**856**
Edinburgh200	F_1	-	-	51.00
	F_2	-	-	**911**

Table 4.9: Best objective values obtained across all runs from the operator perspective. Note, results for Nayeem et al. [115] and Kılıç and Gök [92] are not available in this case. CPU time for each instance is the same at that given in Table 4.8.

4.9 Summary

This chapter has presented our approach to network design. We have shown that an improvement in solution quality can be achieved by seeding a population with solutions constructed using a novel heuristic construction algorithm. The approach combines information from the travel times and demand matrices to create route sets that balance the cost to the passenger and operator. Furthermore, the incorporation of several mutation operators from the literature combined with three mutation operators proposed in this thesis provide further benefit for the majority of problem instances. Using NSGAII can provide additional benefit when compared with SEAMO2. A method for comparing two route sets using the Sorensen/Dice measure was also presented followed by a population diversity metric.

An in-depth analysis of the performance of the genetic operators was performed and showed that as the problem size increases, so does the performance of our operators in

terms of the number of dominating and mutually nondominating solutions produced. However, it was evident that our operators usually fail to produce dominating solutions. This is due to their operation of improving performance from one perspective whilst simultaneously degrading the other.

Finally a comparison was made between our method and the state of the art from the literature where it was shown that our algorithm is able to achieve the best objective value from the passenger and operator perspectives for all but one instance. From this comparison, it was evident that a major obstacle to progress on the network design problem, is a lack of benchmark instances for researchers to compare their methods to. The variation in problem formulation also makes direct comparisons difficult. It was further identified that the current method used in the literature for comparing alternative approaches is unsuitable given the multi-objective nature of the network design problem. However, the publication of reference Pareto sets is needed to enable a comparison of the multi-objective aspect of algorithms to be compared[5].

[5]Pareto sets produced under Algorithm E are available at `http://users.cs.cf.ac.uk/M.P.John/pareto_sets/`

Chapter 5

Surrogate Models for Network Design

As discussed in Chapters 2 and 4, network evaluation for the passenger objective is an expensive computational task that we have found severely constrains the size of problem that can be tackled in a reasonable amount of time. On the other hand the operator objective is very quick to calculate and, as such, surrogate models are not needed for this. The use of high performance computing resources can, to some extent, increase the problem size that can be tackled, although, alternative evaluation strategies that can be quickly computed are desirable. Using an approximation to an objective value is referred to as surrogate-assisted optimisation, sometimes referred to as approximation evaluation, meta-modelling, response surface method or model emulation [149, 15, 120]. It was originally motivated from the need to reduce the computation time of expensive evolutionary optimisations [84].

In this chapter, we take Algorithm E, from the previous chapter which combined an NSGAII framework with our proposed mutation and crossover operators using a population of unique individuals. We augment the algorithm by replacing the procedure used for evaluating the passenger objective with our proposed surrogate models. For clarity, we shall refer to Algorithm E as the 'Original algorithm' throughout this chapter. In the following section, we provide an overview of surrogate-assisted optimisation in the field of evolutionary computation. We also discuss several different models for approximating the passenger objective in the network design problem. Finally we provide a comparison and discussion of the performance of the proposed models.

5.1 Overview of Surrogate-assisted Optimisation

Originally, surrogate-assisted evolutionary computation was motivated out of the need to reduce the computation time of expensive-to-calculate objective functions such as those encountered in drug [51] and aerodynamic design [85], where complex simulations are often required. A surrogate model can be described as a mathematical model that approximates the original objective function and that is also computationally less expensive to evaluate.

Early work on surrogate-assisted optimisation substituted the original objective function for the surrogate model assuming that the model could provide a sufficiently accurate evaluation [84]. This approach can lead to a number of issues, especially if the model introduces optima that do not exist in the original problem. Jin et al. [86] stressed the importance of model management in surrogate-assisted evaluation by making use of the approximate objective functions together with the original objective function.

Surrogate models can be divided into three general categories based upon their management technique: 1) individual based, 2) generation based, and 3) population based [83]. Individual-based model management utilises the original objective function to evaluate just some of the individuals per generation [18, 86, 87]. This is in contrast to generation-based methods where the surrogate model is used for evaluating some of the generations, with the original objective function being used in the remainder [19, 87, 101, 126]. Population-based strategies, on the other hand, co-evolve multiple populations each using its own surrogate model, with migration between the populations then taking place [84].

Moraglio and Kattan [112] define the traditional surrogate model based optimisation procedure as given in Algorithm 5.1. This approach first creates a surrogate model by evaluating a small sample set of randomly selected solutions using the original objective function to fit a mathematical model. A limit is placed on the number of original objective functions evaluations allowed and, until this limit is reached, an

iterative process of search and model creation is executed. At each iteration the search space defined by the surrogate model is explored using a search algorithm (such as an evolutionary algorithm) to find or approximate the optimum. The solution is then evaluated using the original objective function and the model is updated. The process is repeated until the limit on original objective function evaluations is reached.

1: Sample at random a small set of candidate solutions
2: Evaluate the solutions using the original objective function
3: **while** number of expensive evaluations not reached **do**
4: Generate new surrogate model from the current set of solutions
5: Determine the optimum of the surrogate model
6: Evaluate optimum solution using original objective function
7: Add solution to the set of solutions
8: **return** best solution found

Algorithm 5.1: Moraglio and Kattan [112] surrogate model based optimisation procedure.

5.2 Proposed Management Strategies

In our case we use two distinct management strategies for our MOEA which we will refer to as Strategies A and B. The proposed strategies do not fit into any of the categories defined previously. We have opted for this approach to remove the reliance on the original objective function. This is due to the time consuming nature of evaluation using the transit network as discussed previously. By removing the reliance of the original objective function it is hoped that the MOEA is able to execute significantly more generations, provide a greater exploration of the search space and in turn find improved solutions for both the passenger and operator perspectives.

Each strategy uses the surrogate model in all of the generations to evaluate each individual. Strategy A uses the surrogate model instead of the original objective function. Hence, in each generation the surrogate model is used to evaluate the passenger objective. After a set number of generations the model is then updated. At this stage all

the solutions currently in the population are re-evaluated with the original objective function. These original objective values are used together with the surrogate objective values to update the underlying mathematical model – introduced in the next section. It should be noted that the original objective function is never used to accept or reject a solution under this strategy.

For Strategy B, in each generation a solution is first evaluated using the surrogate model. If the solution shows promise, i.e. it would be inserted in to the population under the surrogate objective values, it is re-evaluated using the original objective function. The criteria for insertion are then rechecked using the original objective values and, if met, the solution is inserted into the population. Similarly to Strategy A, the underlying mathematical model is updated at a set number of generations using the process described below.

5.3 Proposed Surrogate Models

In this section we introduce the surrogate models that are used in conjunction with the above management strategies. Underpinning all of our models is the evaluation of the *route* network rather than the larger *transit* network as detailed in Section 2.3. Of course, this approach reduces the size of the graph used for evaluation but results in transfer penalties not being considered.

The relationship between the passenger objective value obtained when using the transit and route networks for evaluation is shown in Figures 5.1 and 5.2. These experiments used a population of 200 solutions generated using our heuristic construction procedure evolved for 200 generations. The figure demonstrates that there is a strong correlation between the objective values obtained even though the route network does not penalise transfers. In our first model a linear regression is carried out between these two variables every 10 generations (using the current population of two hundred individuals) to determine an intercept a and gradient b. For the next ten generations we then use

the surrogate model, Surr_1, where $F_1^{(1)}$ is the passenger objective value gained on the smaller route network.

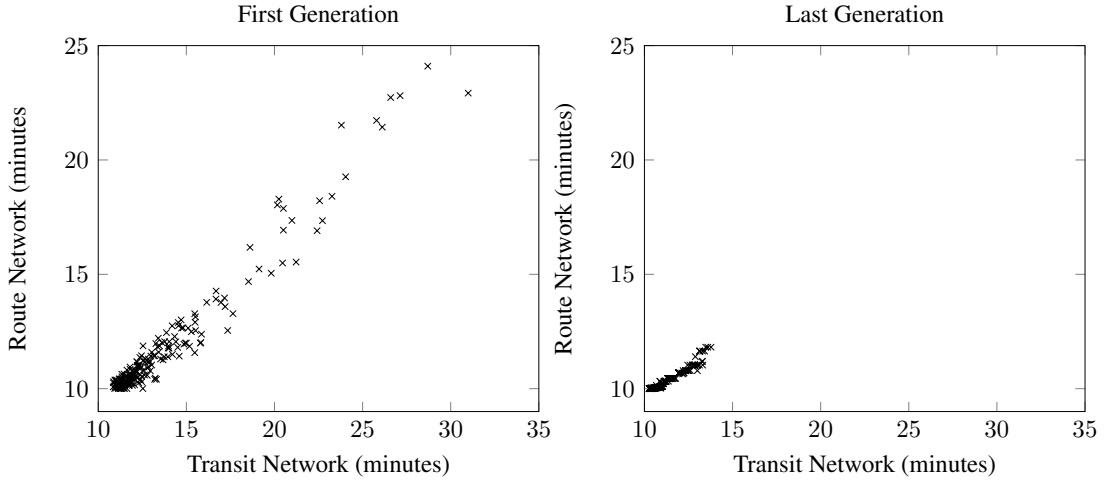


Figure 5.1: Relationship between the obtained objective value using the transit network and route network evaluation schemes for Mandl's benchmark instance using Algorithm E.

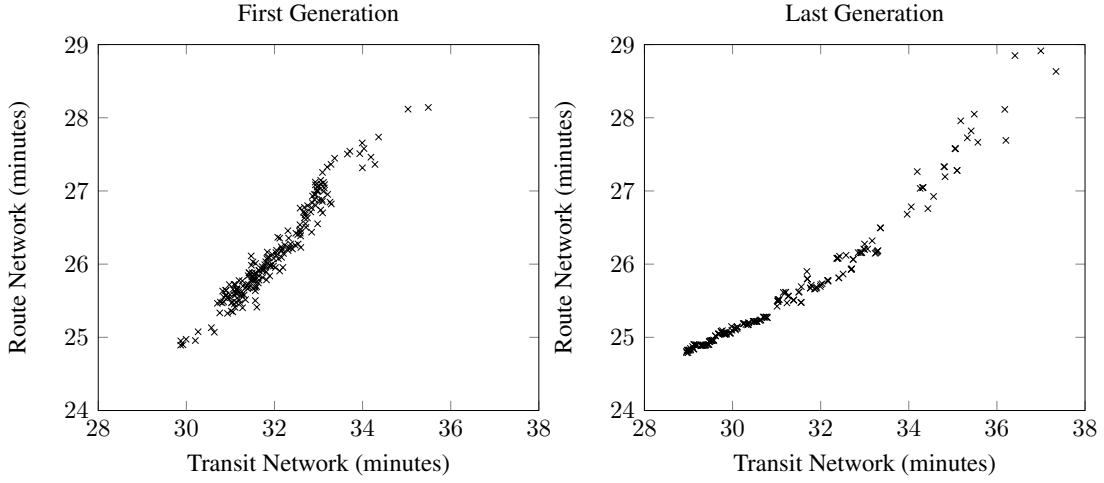


Figure 5.2: Relationship between the obtained objective value using the transit network and route network evaluation schemes for the Mumford3 benchmark instance using Algorithm E.

$$\text{Surr}_1(\mathcal{R}) = a + b(F_1^{(1)}(\mathcal{R})) \tag{5.1}$$

Our second model, Surr_2, uses a multi-variable regression upon the passenger objective

gained on the smaller route network and also the average route length to estimate the passenger objective on the transit network. The model is updated in the same fashion as the first model.

$$\text{Surr}_2(\mathcal{R}) = a + b_1(F_1^{(1)}(\mathcal{R})) + b_2(\frac{\sum_{\forall R_i \in \mathcal{R}} |R_i|}{r}) \tag{5.2}$$

In our surrogate models, because of the use of the route network, we neglect to consider the waiting time incurred by passengers making a transfer. Without the introduction of transfer vertices (as with the transit network) or an analysis of the paths passengers take, this cannot be accurately determined. However, to provide an approximation of the number of passengers requiring at least one transfer we propose the use of a binary transfer matrix $\mathbf{B_{n \times n}}$, where $B_{i,j}$ is set to 0 if v_i and v_j are on the same route and 1 otherwise. It should be noted that the shortest path between v_i and v_j may involve a transfer even if i and j are on the same route. However, in practise people may choose to stay on a vehicle resulting in a longer travel time to avoid the inconvenience of making a transfer. Indeed, the surrogate model might even be a better reflection of the real world in this particular case.

Using route network evaluation together with the binary transfer matrix we can now provide a more accurate estimation of the average passenger travel time using Equation (5.3), where $\beta_{v_i,v_j}(\mathcal{R})$ is the shortest path between vertices v_i and v_j using the route network on route set $\mathcal{R}$ and τ is the transfer penalty.

$$F_1^{(2)}(\mathcal{R}) = \frac{\sum_{i,j=1}^{n} D_{v_i,v_j} \beta_{v_i,v_j}(\mathcal{R}) + D_{v_i,v_j} B_{v_i,v_j} \tau}{\sum_{i,j=1}^{n} D_{v_i,v_j}} \tag{5.3}$$

Our previous models can now be amended to use $F_1^{(2)}$ producing the models Surr_3 and Surr_4.

$$\text{Surr}_3(\mathcal{R}) = a + b(F_1^{(2)}(\mathcal{R})) \tag{5.4}$$

$$\text{Surr}_4(\mathcal{R}) = a + b_1(F_1^{(2)}(\mathcal{R})) + b_2(\frac{\sum_{\forall R_i \in \mathcal{R}} |R_i|}{r}) \tag{5.5}$$

We can now define each of the four proposed surrogate models that are used with both management strategies giving eight different experimental scenarios. The models below are listed in increasing order of computational effort i.e. time taken to determine the passenger objective.

- Surr_1 – linear regression
- Surr_3 – linear regression with binary transfer matrix
- Surr_2 – multi-variable regression
- Surr_4 – multi-variable regression with binary transfer matrix

5.4 Experimental Method for Surrogate Models

To compare the proposed management strategies we choose to use both the NSGAII and SEAMO2 [146] methodologies as our evolutionary frameworks, as used in previous works [88, 114]. SEAMO2 allows a comparison between both strategies due to its population replacement rules [113]. SEAMO2 applies crossover and then mutation to the offspring, and the offspring is then directly compared to both parents to determine whether it should replace either. NSGAII only allows for Strategy A due to its use of Pareto ranking and crowding distance for determining the population for the next generation. During an NSGAII generation the size of the population is expanded from P to $2P$ and only at the end of generation is the population for the next generation produced.

For our experiments a population of 200 solutions generated using our heuristic generation procedure were evolved for a fixed running time, which was set to half the running

time of the original algorithm due to the large computation times involved. Final $\mathcal{S}$ metric values are produced by combining the Pareto sets from 20 independent runs. Statistical tests are performed over the 20 independent runs.

We first examine the effect that the management strategy has over the quality of solutions produced. For this we use SEAMO2 under both management strategies with each of the four surrogate models. Taking the best performing management strategy under SEAMO2 we then compare this to NSGAII using Strategy A and the four surrogate models. A comparison is then made between the best performing model and strategy to the original algorithm. Finally we combine the best performing surrogate model with GPU based evaluation introduced previously. We compare this against the original NSGAII algorithm using the original objective function with GPU based evaluation. It should be noted that all experiments were evolved for a fixed running time, as previously mentioned.

5.5 Experimental Results for Surrogate Models

Table 5.1 shows the $\mathcal{S}$ metric value for SEAMO2 under both Management strategies. In five out of the seven cases, Strategy A is able to achieve a higher $\mathcal{S}$ metric (reference points are given in Table 2.1). Comparing the number of generations executed for the surrogate models under fixed time limits, shown in Table 5.2 and graphically in Figure 5.3, we see that, as the problem size increases, substantially more generations are executed. This can be attributed to the difference in sizes of the route and transit networks. As the problem size increases so does the difference between the route and transit network sizes, as shown in Table 2.1, resulting in a vast reduction in the evaluation time and an increase in the number of generations executed. Furthermore, Strategy A executes significantly more generations than Strategy B per time period reflecting the fact that when using surrogate models under Strategy B we are still required to use the original objective function extensively. Table 5.1 also shows that as the problem size

increases the use of the original objective function becomes less crucial. Strategy A is able to evolve the population for more generations resulting in an improved Pareto set.

Instance	Original	Strategy	Surr_1	Surr_3	Surr_2	Surr_4
Mandl	16688	A	16436	16603	16572	16579
		B	16665**	16665**	16665**	**16668****
Mumford0	160019	**A**	154696	**159256****	158051**	154833
		B	158112**	157272	157795	157409**
Mumford1	696512	**A**	683028	**699525****	685546	683712
		B	686656**	687448	688053**	687025**
Mumford2	4935593	**A**	5052562**	**5096439****	4994117**	4953464**
		B	4897611	4902388	4905577	4899709
Mumford3	15727910	**A**	**16131847****	16122259**	15919207**	15725916**
		B	15659658	15645749	15667155	15668556
Nottingham100	6911423	A	676937	676151	675333	677677
		B	678132**	678808**	680873**	**681077****
Edinburgh200	10893021	**A**	**10968130****	10956145	10874199**	10931269**
		B	10658425	10702250	10645850	10665799

Table 5.1: $\mathcal{S}$ metric comparison between the proposed management strategies and surrogate models using SEAMO2 under fixed time limits. Asterisks indicate statistical significance according to a Related-Samples Wilcoxon Signed Rank test at the $p < 0.01$ level.

Instance	Original	Strategy	Surr_1	Surr_3	Surr_2	Surr_4
Mandl	162	A	373	256	283	300
		B	159	169	239	382
Mumford0	94	A	1736	586	609	588
		B	118	101	432	382
Mumford1	114	A	181	245	203	246
		B	138	130	201	216
Mumford2	101	A	5200	3293	5176	4370
		B	159	127	1858	1944
Mumford3	101	A	3714	1382	1509	1524
		B	152	116	760	849
Nottingham100	73	A	1550	1771	1290	1129
		B	105	104	427	486
Edinburgh200	56	A	8662	7692	1688	2215
		B	85	79	633	594

Table 5.2: Comparison between the number of generations executed under the different models and management strategies using SEAMO2 under fixed time limits.

Figure 5.4 shows the average elapsed running time versus the average $\mathcal{S}$ metric over the 20 independent runs for all the problem instances. Once problem sizes similar

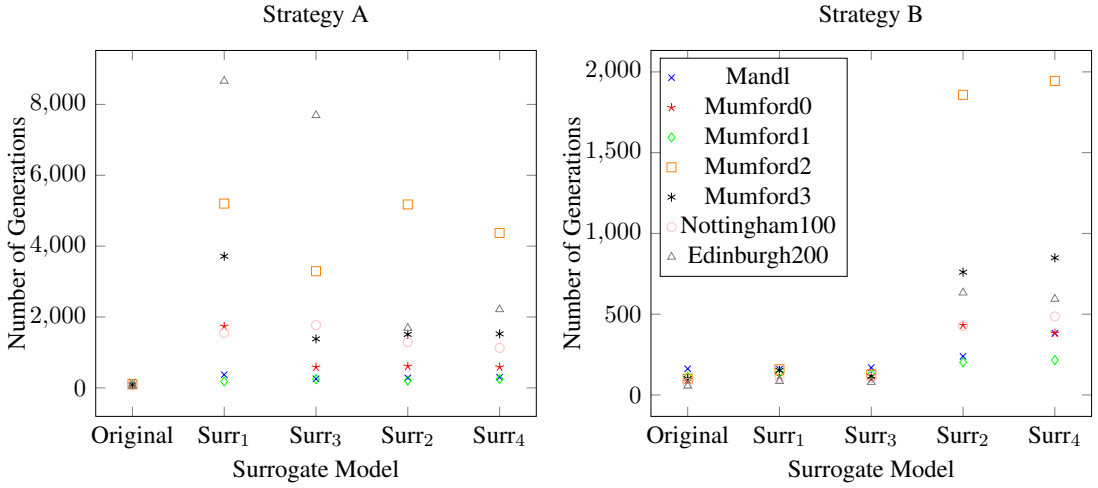


Figure 5.3: Comparison between the number of generations executed under the different models and management strategies using SEAMO2 under fixed time limits.

to Mumford3 are reached the surrogate models perform well in terms of $\mathcal{S}$ metric with Surr_3 and Surr_2 able to achieve higher average metric values in shorter amounts of time for Mumford3 and Surr_4 for Edinburgh200. We also note the decrease in metric value for Surr_1 on Mumford3 although this achieves the best combined metric value. This is due to Surr_1 exploring the extremes of the passenger objective. It can be seen from Figure 5.5 that Surr_3 is preferable to Surr_1 with regards to solution spread. Nottingham100 and Edinburgh200 show a degradation of $\mathcal{S}$ metric over time for Surr_1 and Surr_3; however when the final populations are combined from the 20 runs to form the combined final Pareto set Surr_1 achieves the best $\mathcal{S}$ metric. This can be attributed to a loss of diversity in the individual run populations but, when combined, they achieve a greater exploration of the extremes of the passenger objective resulting in a higher metric as shown in Figure 5.5. In terms of solution spread it is clear from the figure that the original algorithm is still preferred.

Taking Strategy A forward and using it for both SEAMO2 and NSGAII we found that NSGAII demonstrated improved performance over SEAMO2 for six out of the seven problem instances. Table 5.3 gives the $\mathcal{S}$ metrics for both algorithms. NSGAII using Surr_4 provides the best metric value in five out of the seven instances and, as shown

in Figure 5.6, achieves a similar solution spread. Surr_4 contains the most information about a route set in terms of the variables used with average route length and passenger objective value on the route network combined with the binary transfer matrix. This is indicative of the need for more variables that are quick to compute but reflective to help provide an indication of the route set's potential.

Instance	Algorithm	Original Alg.	Surr_1	Surr_3	Surr_2	Surr_4
Mandl	SEAMO2	16688	16436	16603	16572	16579
	NSGAII	16700	16492	16603	16514	**16661**
Mumford0	SEAMO2	160019	154696	159256	158051	154833
	NSGAII	160085	157670	159230	159257	**159421**
Mumford1	SEAMO2	696512	683028	699525	685546	683712
	NSGAII	701872	**700076**	700057	699952	696306
Mumford2	**SEAMO2**	4935593	5052562	**5096439**	4994117	4953464
	NSGAII	4954543	5047514	5057453	5055099	5070217
Mumford3	SEAMO2	15727910	16131847	16122259	15919207	15725916
	NSGAII	15800513	16133804	16121723	16117145	**16153183**
Nottingham100	SEAMO2	6911423	676936	676151	675333	677677
	NSGAII	696652	687042	700225	695289	**701663**
Edinburgh200	SEAMO2	10893021	10968130	10956145	10874199	10931269
	NSGAII	10909888	10930904	10964942	10930874	**10978840**

Table 5.3: $\mathcal{S}$ metric comparison between SEAMO2 and NSGAII using management Strategy A with the four proposed mathematical models.

If we compare the $\mathcal{S}$ metric over time for NSGAII under Strategy A, shown in Figure 5.7, to SEMAO2 under Strategy A, shown in Figure 5.4, we note that when using NSGAII, all the surrogate models are able to achieve higher metrics for the Mumford2, Mumford3 and Edinburgh200 instances compared with the original algorithm. This is also reflected in Table 5.3 where the combined $\mathcal{S}$ metric value is higher for all surrogate models for the mentioned instances. Figure 5.8 displays the Pareto sets for Edinburgh200 under Strategy A for the four surrogate models. From the figure it can be seen that Surr_1 and Surr_2 do not provide a greater exploration for the extremes of the passenger objective but do improve the operator objective associated with solutions whose passenger objective is above 30 minutes. On the other hand, Surr_3 and Surr_4 offer an improvement over the entire Pareto front, extending the extremes of the objective values also. This may be due to NSGAII being able to maintain diversity in the populations, through the use of

nondominated sorting and crowding distance, whereas SEAMO2 loses this, resulting in poor performance under the surrogate models.

If we take Surr_4 under Strategy A using NSGAII and compare it to the original algorithm, shown in Table 5.4, we see that for the smaller three instances the original algorithm is able to achieve the best combined $\mathcal{S}$ metric. For the larger four instances the surrogate model at worst performs just as well as the original, and in Mumford2, Mumford3 and Edinburgh200 performs significantly better. It would appear that for instances larger than 100 vertices it is advantageous to use surrogate models in place of the original algorithm due to the improvement in solution quality produced when operating under constrained running times.

Comparing the best achieved objective values for Surr_4 under Strategy A using NSGAII to the original algorithm, shown in Table 5.5, we can see that the surrogate model provides improvement from the operator viewpoint only. This is in-keeping with what we have seen throughout this chapter with the surrogate model exploring the extremes of the operator objective.

Instance	Algorithm	$\mathcal{S}$ Metric
Mandl	Original	**16700****
	Surr_4	16661
Mumford0	Original	**160085****
	Surr_4	159421
Mumford1	Original	**701872****
	Surr_4	696306
Mumford2	Original	4954543
	Surr_4	**5070217****
Mumford3	Original	15800513
	Surr_4	**16153183****
Nottingham100	Original	696652
	Surr_4	**701663**
Edinburgh200	Original	10909888
	Surr_4	**10978840****

Table 5.4: $\mathcal{S}$ metric comparison between the original NSGAII algorithm and NSGAII using Strategy A and Surr_4 using under fixed time limits. Asterisks indicate statistical significance according to a Related-Samples Wilcoxon Signed Rank test at the $p < 0.01$ level.

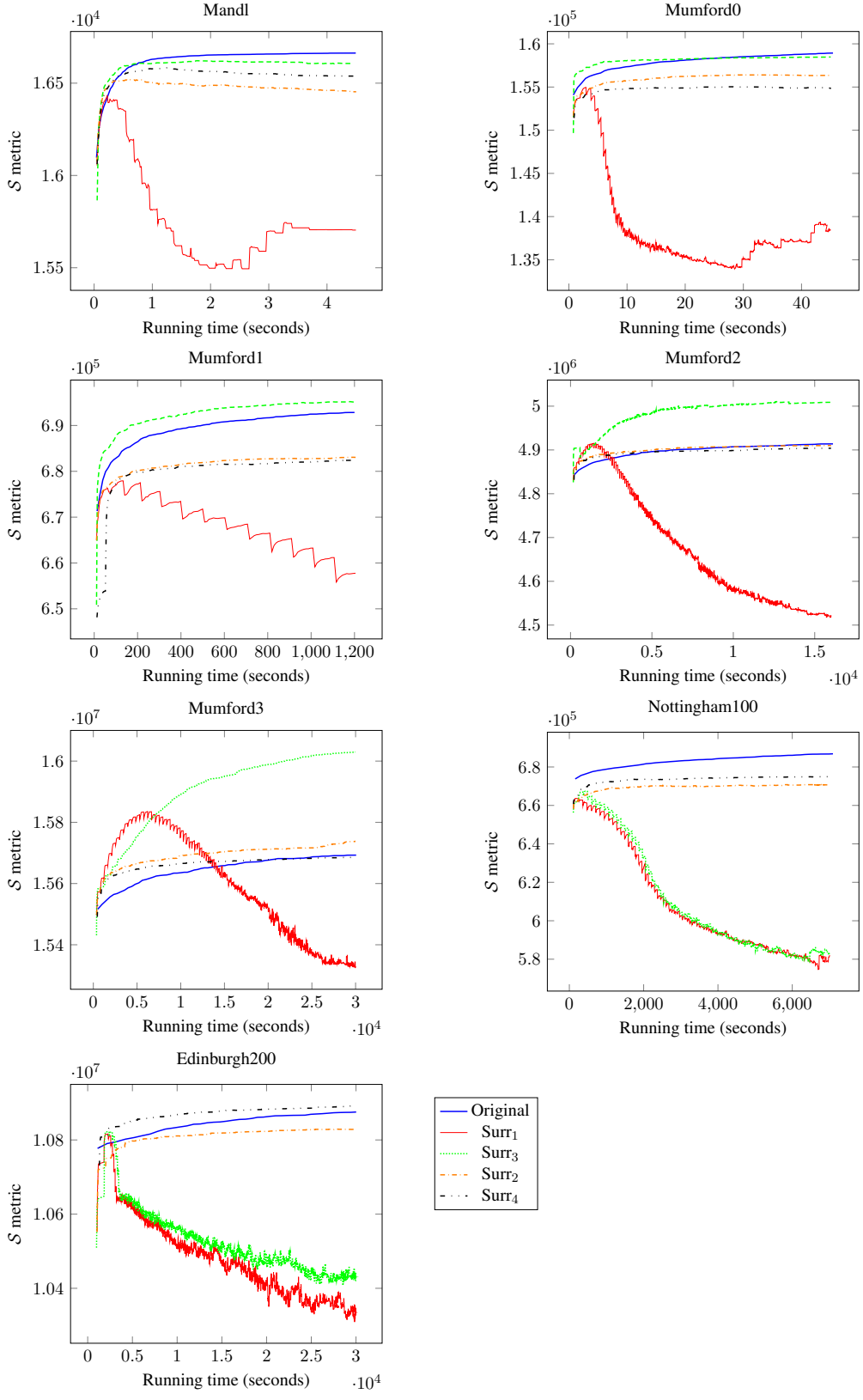


Figure 5.4: $\mathcal{S}$ metric comparison over time for our proposed mathematical models using Strategy A and SEAMO2. Averaged over 20 runs per generation.

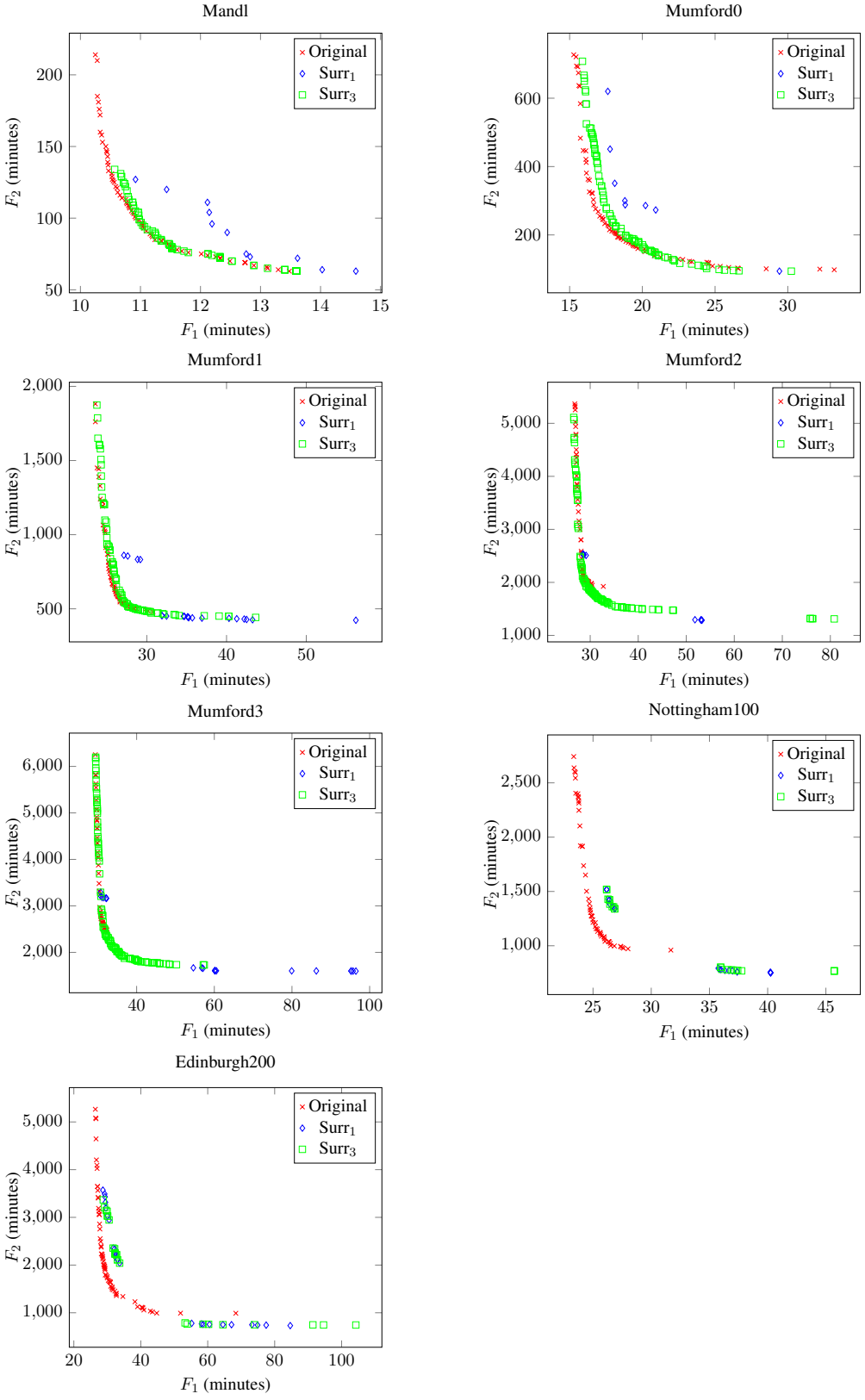


Figure 5.5: Pareto sets for Surr$_1$ and Surr3 using Strategy A with SEAMO2

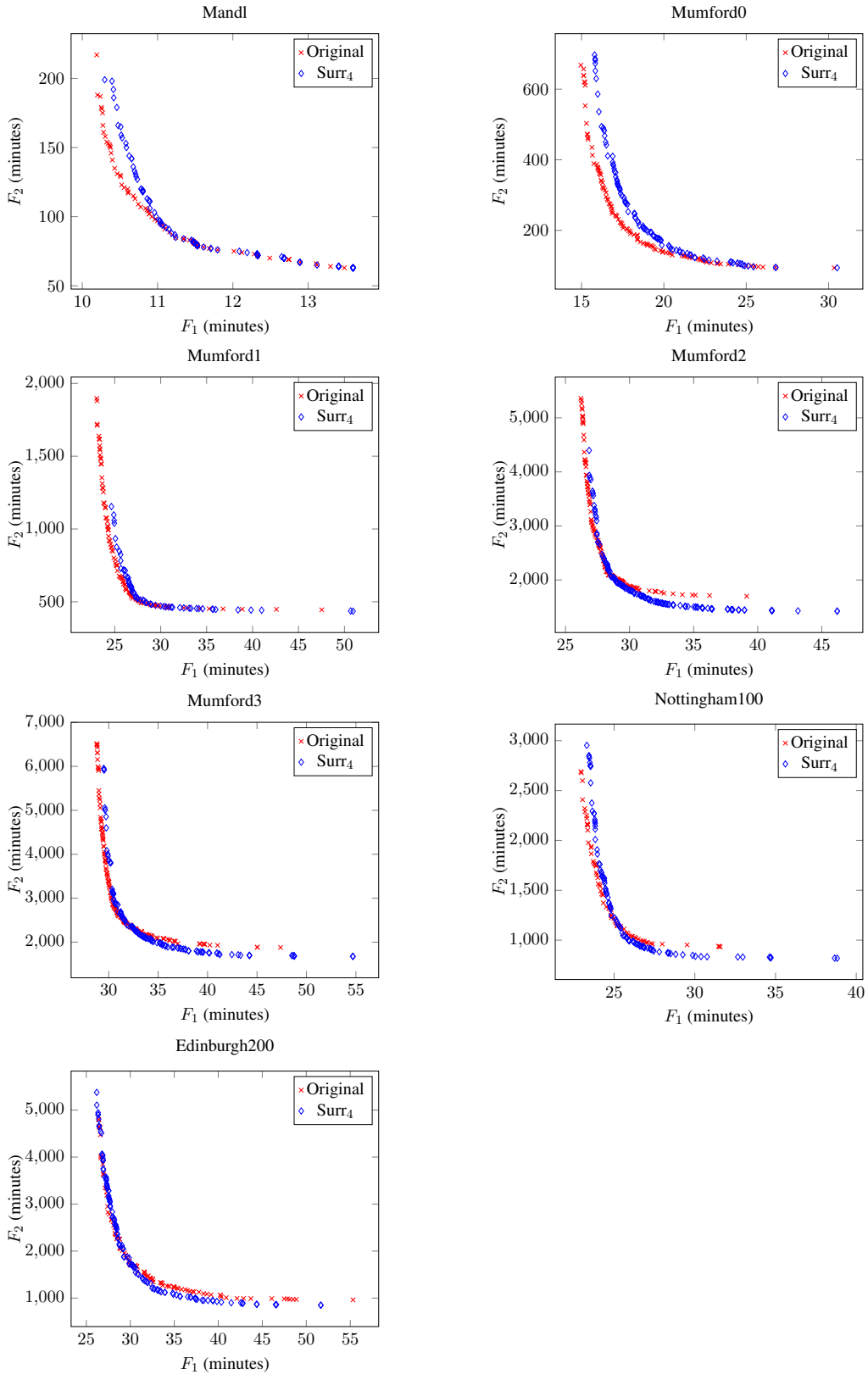


Figure 5.6: Comparison between Pareto sets for Surr_4 under Strategy A and the original algorithm using NSGAII.

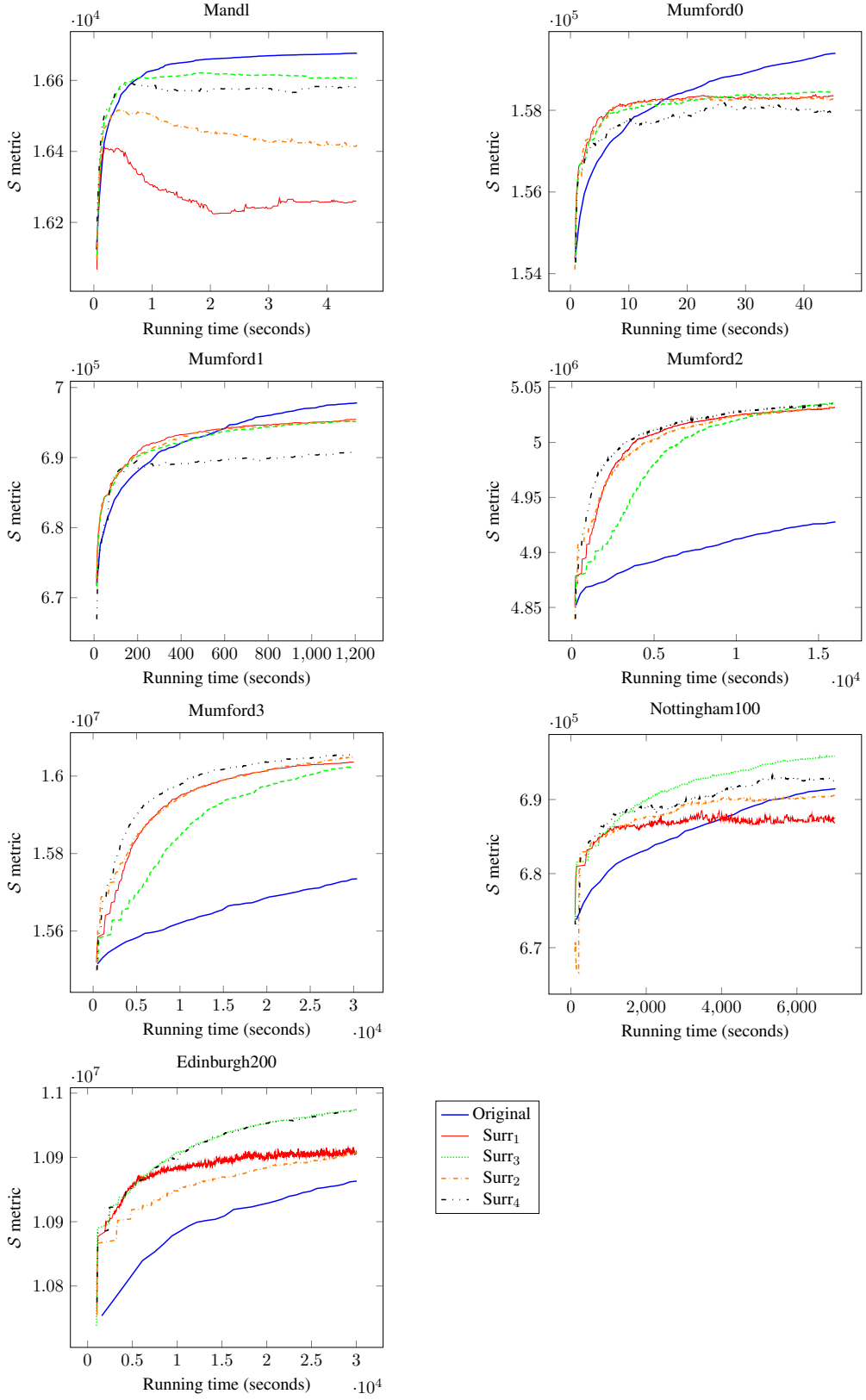


Figure 5.7: $\mathcal{S}$ metric comparison over time for our proposed mathematical models using Strategy A and NSGAII. Averaged over 20 runs per generation.

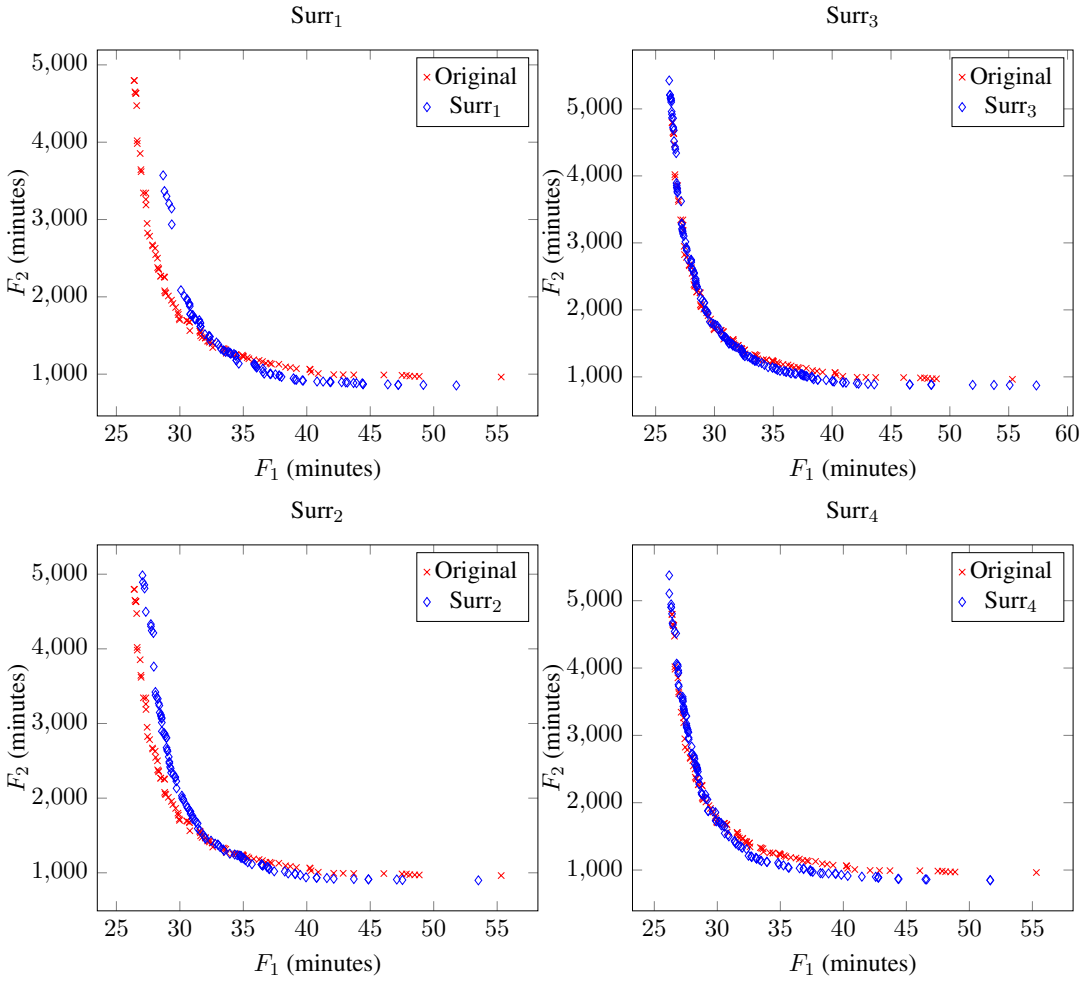


Figure 5.8: Comparison between Pareto sets for Edinburgh200 under Strategy A using NSGAII for the four surrogate models.

		Mandl	Mumford0	Mumford1	Mumford2	Mumford3	Nottingham100	Edinburgh200
Best for passenger	F_1	10.30(**10.19**)	15.8(**14.96**)	24.64(**23.04**)	26.85(**26.20**)	29.5(**28.79**)	23.31(**22.62**)	26.18(**25.71**)
	F_2	199(217)	697(668)	1154(1897)	4397(5360)	5957(6519)	2951(2761)	5376(5727)
Best for operator	F_1	13.60(13.48)	30.52(30.33)	50.93(47.53)	46.20(39.14)	54.71(47.38)	32.82(35.81)	51.66(51.00)
	F_2	**63**(**63**)	**94**(**94**)	**435**(445)	**1422**(1699)	**1674**(1879)	**818**(856)	**847**(911)

Table 5.5: Best objective values extracted using NSGAII under Strategy A with Surr_4 compared with those produced under the original algorithm given in brackets.

Finally, using NSGAII under Strategy A with Surr_4 and the GPU-based evaluation of the all pairs shortest path algorithm, we found that surrogate models were unable to provide benefit over NSGAII with GPU based evaluation when comparing the achieved $\mathcal{S}$ metric from Table 5.6 for our instances. This can be explained by the increase in generations executed by the original algorithm when using GPU evaluation with percentage increases of -44%, 231%, 152%, 1016%, 805%, 1428% and 2425% recorded across the problem instances respectively. If this is compared with the percentage increases for generations executed using NSGAII under Strategy A with Surr_4 of 87%, 93%, 46%, 100%, 67%, 94% and 165% we can see that GPU based evaluation does not provide a significant increase in the number of generations executed. Further investigation has shown that the all pairs shortest path algorithm using the GPU does not produce a speed-up over 100% until a graph size of approximately 200 vertices or greater is reached, as shown in Table 5.7. As our surrogate models use the smaller *route* network rather than the larger transit network evaluated for the original objective function the GPU does not provide a benefit on our instances (our largest instance has 200 vertices). However, as shown in Table 5.7, once the route network grows to realistic size – a few thousand vertices – combining surrogate models and GPU based evaluation will almost certainly provide additional benefit.

Instance	NSGAII	NSGAII Surr_4
Mandl	**16693****	16643
Mumford0	**161182****	159913
Mumford1	**703527****	699538
Mumford2	5054988	**5086349****
Mumford3	**16169859***	16158094
Nottingham100	**710696****	702273
Edinburgh200	**10986251****	10978313

Table 5.6: $\mathcal{S}$ metric comparison between NSGAII and NSGAII using Strategy A with Surr_4. Both algorithms used GPU based evaluation. Asterisks indicate statistical significance according to a Related-Samples Wilcoxon Signed Rank test at the * $p < 0.05$ and ** $p < 0.01$ level.

Vertices	CPU	GPU	Speed-up %
100	**0.002**	0.005	-60
200	0.014	**0.008**	75
300	0.049	**0.013**	277
400	0.115	**0.023**	400
500	0.226	**0.041**	451
1000	1.919	**0.243**	690
2000	19.741	**1.766**	1018
3000	54.411	**5.935**	817
4000	125.663	**14.077**	793
5000	244.407	**26.902**	809

Table 5.7: Runtime comparison (seconds) for the all pairs shortest path on randomly generated graphs using the CPU and GPU.

5.6 Summary

This chapter has introduced the general topic of surrogate-assisted optimisation. Two management strategies were proposed and empirically evaluated. The strategy that removed the reliance on the original evaluation function produced the best $\mathcal{S}$ metric executing significantly more evaluations. Four mathematical models were also discussed and empirically evaluated against the original algorithm (Algorithm E) described in Chapter 4. It was shown, that surrogate models are able to offer an improved Pareto sets compared with the original algorithm under a runtime constraint. We have demonstrated that incorporating more knowledge into the model provides an improvement in the resultant Pareto set with Surr_4 achieving the best metric values. As such the identification of variables that are quick to compute but also reflective of the route network, are key to the success of surrogate models for network design. This a topic for future work to aid in the development of techniques that are able to tackle real world instances in a reasonable time-frame.

Finally, we have shown that when combined with surrogate models, GPU based evaluation is unable to provide additional benefit on our current set of problem instances. This is due to the use of the smaller route network for evaluation under the surrogate

models. However, for real world instances, with a few thousand vertices, surrogate models combined with GPU based evaluation will almost certainly provide benefit.

In the next chapter, we empirically evaluate a number of algorithms for setting the frequency on routes designed using our approach to network design.

Chapter 6

Frequency Setting

In this chapter we now move on from network design and on to frequency setting. Frequency setting is the task of assigning a frequency to each route in a route set with the aim of minimising the travel time for passengers whilst balancing the operational costs for the network operator. The frequency determines the number of vehicles operating on a route, and thus the separation time between the arrival of vehicles at a given stop on the same route. It is obvious that routes that serve high demand areas should have a relatively high frequency. Similarly, routes that have a high number of passengers transferring on to them should also have a relatively high frequency, to avoid excessive waiting time for passengers when making a transfer. Compared with a redesign of the transit network, route frequencies are more easily configured due to the extensive disruption that an entire network redesign would cause to passengers. This chapter introduces work to first examine whether route sets that have been produced during network design can be improved by setting alternate frequencies on the routes under a constrained fleet size. We then compare two metaheuristics and a local search algorithm under an unconstrained optimisation scenario. Both vehicle capacity and fleet size are then constrained under a multi-objective optimisation algorithm to produce a set of approximate Pareto-optimal solutions, allowing a network operator to choose the desired configuration. Finally we consider alternative approaches to the UTNDP to overcome the issues identified.

6.1 Preliminary Investigation

Our first area of exploration is to determine whether a route set produced during network design can be improved, in terms of the average passenger travel time, by adjusting the frequencies of the individual routes. To start, we are given an existing route set with a default frequency of one vehicle every ten minutes and a corresponding fleet size needed to service these routes i.e. the fleet size is fixed and extra vehicles cannot be used. Is it possible to improve the average passenger travel time by adjusting these frequencies within a bounded range? A frequency of $\frac{1}{10}$ is used as it corresponds to the frequency used during the network design stage; that is, a frequency of $\frac{1}{10}$ corresponds to an average waiting time for passengers of five minutes. This frequency allows passengers to treat the service as "turn-up-and-go" [61]. Furthermore, for now we also assume that the capacity of each bus is infinite and fractional buses can operate on each route as per our assumptions during the network design stage. The allowed frequency is therefore a real number in the range of $\frac{1}{5}$ to $\frac{1}{30}$ minutes inclusive where $\frac{1}{5}^{-1}$ gives the headway between vehicles operating on the same route i.e. the separation time between vehicles.

In this section we propose a heuristic approach for the assignment of frequencies that utilises a ratio to assign the frequencies based upon the number of passengers that require a transfer on to the given route. It is assumed that a passenger will travel from their origin to destination using the shortest path as detailed in the evaluation scheme discussed in Section 2.6.1. Under the current simplified models, the capacity of vehicles is assumed to be infinite and we ignore the initial waiting time incurred. As such the average travel time for passengers can be decreased by increasing the frequency of vehicles on routes that have a high volume of passengers transferring on to them, compensated for by a decrease in service on less busy routes. It should be noted that if we consider the initial waiting time in the average travel time, then increasing the frequency on any route would result in a decrease in the average passenger travel time. This increase in frequency reduces the waiting time required for passengers when transferring to reach their destination therefore reducing the overall average passenger

travel time. We do not take account of the initial waiting time of passengers before they board the first vehicle, this is consistent with the calculation of the passenger objective used in the network design stage.

Using a ratio that splits the available fleet over all routes based upon the number of transfers made on to each route means that the assignment must be conducted in a controlled manner to prevent routes with very low transfer volumes from not receiving enough vehicles. As we are calculating the number of vehicles based on the number of transfers onto the route, a route with no transfers onto it would receive no vehicles. This obviously cannot be allowed to happen as the route must be operated at the minimum allowed frequency in this case. Hence if a route does not have enough vehicles to maintain the minimum allowed frequency of $\frac{1}{30}$ then the solution is deemed infeasible. Conversely if a route contains an excessive number of vehicles, that would result in a frequency exceeding $\frac{1}{5}$ the solution is also deemed infeasible. To prevent this from occurring, routes that have no passengers transferring onto them (i.e. passengers travel to from their origin to destination on a single route) or routes that have a proportion of transfer demand that means their required frequency is greater than $\frac{1}{5}$ are assigned to the lower and upper bound respectively. Vehicles are then assigned to routes based upon the proportion of the transfers they serve. An algorithmic description of our proposed algorithm can be seen in Algorithm 6.1. Algorithm 6.1 first assigns the lower or upper bound of the frequencies to those routes that have a transfer volume that would exceed a frequency of $\frac{1}{5}$ or be below $\frac{1}{30}$. The remaining routes that are yet to have a frequency assigned are then processed by taking a proportion of the remaining vehicles. We use this approach to ensure that we do not exceed the number of vehicles available, as our aim is to improve the passenger objective by augmenting the route frequencies, whilst keeping the fleet size constant. A frequency is assigned and checked to ensure that it does not exceed the upper bound, and if this value is exceeded the frequency is set to the upper bound. We use this process to ensure that the constraint on fleet size is not breached.

let $T[R_i]$ be the number of transfers made to R_i
for $R_i \in \mathcal{R}$ **do**
  if $T[R_i]$ = 0 OR required frequency for $R_i < \frac{1}{30}$ **then**
    set frequency of R_i to $\frac{1}{30}$
    decrease available_fleet
  else if required frequency for $R_i > \frac{1}{5}$ **then**
    set frequency of R_i to $\frac{1}{5}$
    decrease available_fleet
for $R_i \in \mathcal{R}$ **do**
  if frequency not set **then**
    vehiclesRequired = $\frac{available\_fleet \times T[R_i]}{\sum_{i=1}^{r} T[R_i]} \times \frac{1}{2}$
    frequency = $\lceil \frac{length\_route(R_i)}{vehiclesRequired} \rceil^{-1}$
    if frequency $> \frac{1}{5}$ **then**
      frequency = $\frac{1}{5}$

Algorithm 6.1: Frequency Setting Ratio Heuristic to distribute a fleet of vehicles (available_fleet) over a given route set.

We used the above heuristic under two different evaluation schemes. In the first the original passenger paths through the network were kept unchanged and the change in frequency was evaluated. Evaluating the frequency change in this manner significantly reduces the computation time because the need to re-perform the All Pairs Shortest Path (APSP) algorithm through the network is removed. We will call this Delta Evaluation. The second form of evaluation was made by re-calculating the APSP through the network. The transfer penalty for a given route is set to half the vehicle headway.

Using delta evaluation, the route set is initially evaluated with each route using a default frequency of $\frac{1}{10}$ as per the network design stage. This results in a transfer penalty of five minutes for each route. During the initial evaluation the path a passenger takes from their origin to destination is recorded. When a change in frequency is then applied to a route one must apply the new transfer penalty to any path that utilises the route, allowing the average passenger travel time to be calculated without recomputing the APSP.

Table 6.1 shows the average improvement and gap to the lower bound calculated by setting the frequency on each route to $\frac{1}{5}$. These results were taken by applying frequency

setting to a Pareto set produced from network design for each benchmark instance. We see that an improvement to the passenger objective has been achieved across all of the benchmark instances whilst ensuring that the cost to the operator, in terms of fleet size, is never increased. Although solutions with a smaller fleet size are accepted.

	Average Gap Lower Bound (%)		Average Improvement (%)	
Instance	APSP	Delta Evaluation	APSP	Delta Evaluation
Mandl	**2.1091**	3.5022	**1.4765**	0.1345
Mumford0	**6.8883**	8.2770	**2.3459**	1.0634
Mumford1	**7.4754**	9.0663	**2.5536**	1.1032
Mumford2	**4.2517**	7.8736	**4.9176**	1.6112
Mumford3	**3.2769**	6.8810	**4.7995**	1.4722

Table 6.1: Comparison between all pairs shortest path (APSP) and delta evaluation.

The comparison between using delta evaluation and the APSP, shown in Table 6.1, indicates that using the APSP to evaluate the frequency changes at each iteration is beneficial. This is justified because, when using delta evaluation, the paths passengers take through the network do not change. When re-evaluating using the APSP on the other hand the path a passenger may take from their origin to destination may change at each iteration because a change in frequency on a single route may result in a reduction or increase to the travel time the passenger encountered using the previous set of frequencies. A possible explanation for this is that the number of passengers journeys that can be satisfied directly is relatively low. As such frequency changes can result in shorter transfer times for passengers.

The APSP evaluation scheme has a number disadvantages associated with it. The initial waiting time is not taken into account, this means that if a high proportion of the demand is satisfied directly then the frequencies on routes can be set at the lower bound with no effect on the passenger objective. The computation time required to perform the APSP is also a significant factor. As shown in Chapter 2 the APSP on the transit network is an expensive operation. Using delta evaluation we remove the need to perform the APSP providing a reduction in computation time. However, given the improved results gained

by utilising the APSP it is preferred over delta evaluation.

6.2 Evaluation with Frequencies Considered

As previously mentioned, evaluating the frequencies of routes under the APSP makes simplifying assumptions such as the average waiting time being equal to half the headway. The initial waiting time is also not considered. In a more realistic evaluation scheme, a passenger may choose to travel on an alternative route given a certain probability if this is seen to go to their destination. That is, they may take the next vehicle that arrives at the current stop which services their destination. To provide a more accurate evaluation of average passenger travel time inclusive of initial waiting time and transfer time, the idea of Optimal Strategies, proposed by Spiess and Florian [135], is used.

Spiess and Florian's approach assumes that a passenger enters the network with no prior knowledge. The only information available to a passenger is that which he or she finds out whilst waiting at a vertex, i.e. bus stop, in terms of which route will be next served by a vehicle. Or in other words on which route will the next vehicle arrive on. Optimal Strategies takes into account the arrival time of buses at a bus stop and assumes that a passenger will choose to board the first vehicle that services a route currently part of the *optimal strategy*.

The Optimal Strategies algorithm can be broken down into two distinct stages: i) calculate the expected travel time from each vertex to a destination vertex along a set of edges (referred to as the *optimal strategy*), and ii) assign the demand to the edges contained in the *optimal strategy*. For consistency with Spiess and Florian's work we shall refer to the travel time between two vertices as the "expected travel time" denoted u_i, where i is the source vertex to a destination vertex d.

Similar to the evaluation methods discussed in Chapter 2.6, Spiess and Florian's approach requires that a specialised graph be constructed upon which evaluation is per-

formed. We have a graph $G = (I, A)$ where I is a set of vertices and A a set of arcs. The topology of G is similar to that given in Figure 6.1(b), we shall refer to this as the generalised transit network in keeping with Spiess and Florian. Figure 6.1(b) differentiates between two types of vertices. The square vertices are the entry points for passengers into the network and act as the physical bus stops. The arcs incident to these vertices are the boarding and alighting arcs allowing passengers to board a vehicle whose route services that particular stop or disembark the vehicle if they have reached their destination or require a transfer. The remaining vertices act as the routes with traversal costs between the edges indicated by the underlying transport graph.

For example, if we have a route set $\mathcal{R} = \{\langle 1, 3, 6\rangle, \langle 2, 3, 4, 5\rangle\}$ with the route network given in Figure 6.1(a) the generalised transit network used for evaluation will be that given in Figure 6.1(b). Associated with each arc in the generalised transit network is a tuple (a, b) where a is the time required to traverse the given arc and b is the frequency operating on the arc. It should be noted that infinite frequencies are assigned to every arc apart from boarding arcs (arcs from square to circular vertices). Boarding arcs are assigned the frequency of the route to which they are incident to. The remaining arcs are assigned infinite frequencies as there is assumed to be no waiting time required to alight from a vehicle and, when on the vehicle, there is no waiting time incurred. Taking Figure 6.1(b), the arc 1 to 1a would be assigned the tuple (0, $\frac{1}{10}$) assuming that the frequency on the route is $\frac{1}{10}$. The arc 1a to 1 would be assigned the tuple (0, ∞) and the arc 1a to 3a would be assigned the tuple (7, ∞) assuming that the travel time between vertices 1 and 3 in the underlying transport network is 7 minutes.

The calculation of expected travel time is completed for each destination vertex and is proportional to the number of arcs in the underlying graph with a computational complexity of O(log m) per destination vertex where m is the number of arcs in the graph. The algorithm for calculating the expected travel time, Algorithm 6.2, first initialises the cost to reach all vertices to infinity apart from the destination vertex, d, which has a cost of zero. Each vertex also has an associated sum of frequencies, f_i,

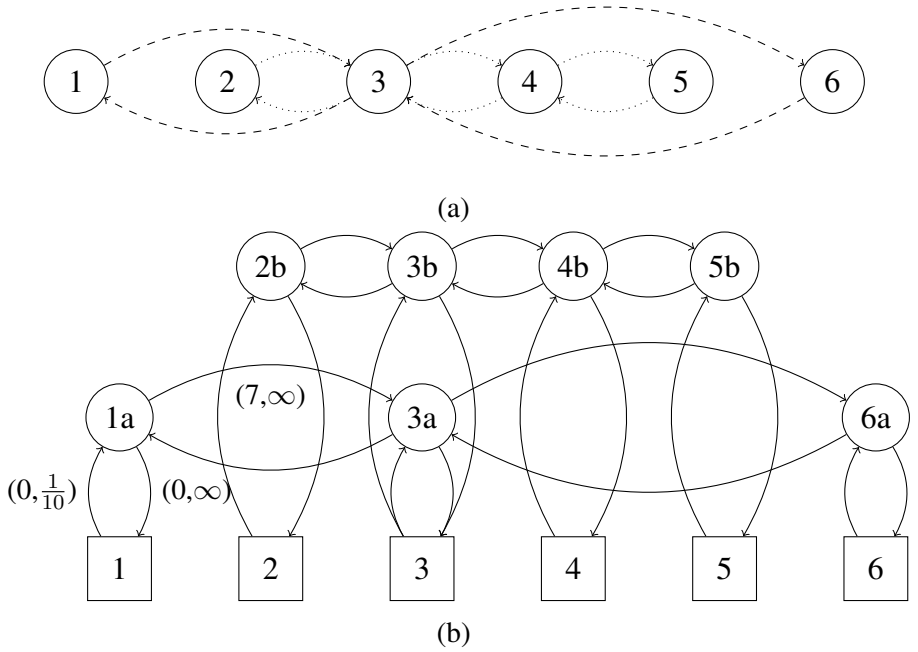


Figure 6.1: (a) Original route network (b) Generalised transit network required for Optimal Strategies evaluation.

which is the sum of all the frequencies of the arcs that intersect that vertex in the optimal strategy, initialised to zero. As the algorithm backtracks from the destination vertex we set the expected travel time to zero for u_d. Each arc in the generalised transit network is then added to the set S which details the arcs that still need to be explored. In each iteration the lowest cost unexplored arc, $a = (i, j) \in S$, with cost c_a and frequency f_a, (where j has already been visited) is selected. If the cost of traversing the edge, a, and the cost to reach j, u_j, is less than or equal to the current cost of reaching i, then a is added to the optimal strategy, $\bar{A}$. Upon adding an arc to the optimal strategy, u_i and f_i are updated using Equations (6.1) and (6.2) respectively. It should be noted that the first time a vertex is explored we allow a modification $f_i u_i = 0 \cdot \infty = \alpha$ following the convention used by Spiess and Florian. In our case we assume a constant inter-arrival rate of vehicles as such $\alpha = \frac{1}{2}$. Once the expected travel time has been calculated for each of the destination vertices it can be used in place of the shortest path in Equation (2.9).

1: $u_i = \infty \; \forall \; i \in I - d$
2: $f_i = 0 \; \forall \; i \in I$
3: $u_d = 0$
4: $S = A$
5: **while** S $\neq \varnothing$ **do**
6: find $a = (i, j) \in S$ which satisfies $u_j + c_a \leq u_{j'} + c_{a'}, a' = (i', j') \in S$
7: $S = S - \{a\}$
8: **if** $u_i \geq u_j + c_a$ **then**
9: $u_i = \frac{f_i u_i + f_a(u_j + c_a)}{f_i + f_a}$
10: $f_i = f_i + f_a$
11: $\bar{A} = \bar{A} \cup \{a\}$

Algorithm 6.2: Optimal Strategies calculation of expected travel time from a single destination vertex, d, to all other vertices.

$$u_i = \frac{f_i u_i + f_a(u_j + c_a)}{f_i + f_a} \tag{6.1}$$

$$f_i = f_i + f_a \tag{6.2}$$

On completion of Algorithm 6.2, for each vertex we will have an optimal strategy that details the arcs that will be used to reach the destination vertex from all other vertices. The second part of the Optimal Strategies algorithm makes use of this information to assign the passenger demand to the network. Each vertex has an associated demand volume V_i that gives the demand from the given vertex to the destination vertex. The demand from a vertex i to the destination vertex, g_i, is assigned according to the arcs contained in the optimal strategy, $\bar{A}$. Each arc $a \in \bar{A}$ is assigned a proportion of the volume at a vertex based upon the frequencies at that vertex. Each arc therefore has an associated volume v_a. Algorithm 6.3 details how the demand from all vertices to a single destination vertex is assigned.

Algorithm 6.3 proceeds by firstly assigning the demand from vertex i to the destination, g_i, to each of the vertices. Each edge $a \in \bar{A}$ is then taken in decreasing order of $u_j + c_a$ (i.e. in the reverse order as to which each edge was added to the optimal strategy) and

the proportion of the vertex volume V_i that corresponds to its frequency is assigned.

Spiess and Florian [135] propose a modification to the demand assignment portion of Algorithm 6.3 allowing for memory to be used more efficiently. In this implementation the variable $v_a, a \in A$ is used directly to store the accumulated arc volumes for all the destination vertices: i.e. $v_a = v_a + \frac{f_a}{f_i} V_i$. Their modification simply initialises all of the arc volumes to zero then accumulates the arc volumes, adding the proportion of demand directly to the arc volume.

1: $V_i = g_i \; \forall \; i \in \; I$
2: **for** $\forall \; a \; \in A$ **do**
3: **if** $a \in \bar{A}$ **then**
4: $v_a = \frac{f_a}{f_i} V_i$
5: $V_j = V_j + v_a$
6: **else**
7: $v_a = 0$

Algorithm 6.3: Optimal Strategies assignment of demand for a single destination vertex.

6.3 Problem Instance Demand Scaling

From the definition of the Optimal Strategies algorithm introduced in the preceding section it can be seen that demand plays a vital role. Problem instances created by Mumford [114] were designed specifically with the network design problem in mind and were thought to involve sensible numbers for the passenger demand between vertices. If the demand is analysed more closely, however, it is seen to be excessive and cannot be satisfied given physical bus sizes that operate in the real world. For the majority of the previous work the demand was not an important factor as the capacity of vehicles was assumed to be infinite. However, when imposing a capacity constraint the demand obviously becomes much more important.

Looking at the generated demand more closely, taking the Mumford1 problem instance, we can sum the demand to give 1,926,170 trips per day. As the Mumford instances are

symmetrical we can halve this to give 963,085 trips in each direction per day and further share this over the fifteen routes that are available, meaning a demand per route of approximately 64,206 in each direction. If we then take the demand per hour assuming a uniform demand pattern throughout the day we can calculate that 38 buses per hour in each direction will be required to satisfy all the demand assuming a bus capacity of seventy passengers[1]. Many simplifying assumptions have been made here such as the demand will be uniformly distributed between routes and that this will equal the peak volume on a link on the route. It does however demonstrate that the demand values need to be scaled down to become realistic.

Using an approach similar to that above we can reverse the process and determine more sensible demand values if we use the following assumptions: 1) the average bus capacity is thirty five persons, 2) average bus frequency is one bus every fifteen minutes, and 3) demand is spread evenly throughout the day. A capacity of 35 persons has been used as it equates to half the capacity of the smallest single deck bus specification released by the UK Government[1]. As bus ridership varies throughout the day, i.e. during morning and evening rush hours we assume that vehicles will be operating at or close to their capacity, an average ridership of 50% is used. Using these assumptions and the number of routes given for each of Mumford's problem instances we can use the following technique to rescale the demand values.

First we calculate the number of buses that will operate per day i.e. given one bus every fifteen minutes and fifteen routes this will equate to sixty buses per hour in each direction and 2880 buses per day. If each bus carries thirty five passengers, average ridership over the twenty four hour period, this will give an overall demand of 100,800. The demand value for each vertex pair can then be recalculated by multiplying the original demand value by the ratio of original demand to our calculated maximum demand per day. Table 6.2 gives the original and scaled demand values for each of the Mumford instances.

[1]Smallest single deck bus capacity is seventy persons available from https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/4260/buslength.xls

As the demand values for the instances have now changed we have produced four new benchmark instances with respect to frequency setting. As the instances have a rescaled demand they cannot be directly compared to their original unmodified instance for network design. Demand is an integer number and therefore during scaling rounding errors will have been introduced. This will result in different objective values being produced between the modified and unmodified instances. The modified problem instances are denoted Mumford0F, Mumford1F, Mumford2F and Mumford3F to indicate they are to be used for frequency setting optimisation.

It should be noted that the Mandl, Nottingham100 and Edinburgh200 instances did not require any scaling of demand. The demand figures were able to be adequately catered for given physical bus capacity. This is due to the way in which demand was created for the Nottingham100 and Edinburgh200 instances. Unlike the Mumford problem instances demand was not assigned at random between a bounded range set by the user. Demand was assigned based upon the population density surrounding each bus stop.

Instance	Original Demand	Perfectly Scaled Demand	Actual Demand
Mumford0	342160	80640	80640
Mumford1	1926170	100800	100846
Mumford2	4847900	376320	376406
Mumford3	6394950	403200	403292

Table 6.2: Comparison between the demand values for the Mumford problem instances when scaled.

6.4 Variable Fleet Size

We have shown in Section 6.1 that, given a constrained fleet size, the frequency of routes can be augmented in such a way as to offer an improved service to passengers whilst not increasing the operational cost to the operator in terms of vehicles required (for five out of the seven problem instances). Constraining the fleet size removes the possibility of exploring solutions, that for a slight increase in operator cost, could produce a significant

reduction in the passenger objective. Optimising using an unconstrained fleet size would also enable a network operator to plan for the future by providing the means to evaluate alternate frequencies with larger fleet sizes. Similar to the previous section it is assumed that vehicles operating on routes will have an infinite capacity.

The approach discussed previously allowed a frequency to be any real number in the range $[\frac{1}{30}, \frac{1}{5}]$. However, this is not a realistic real world scenario. As discussed in Section 3.2 the frequency for peak periods should not drop below 30 minutes and the vehicle headways should be a divisor of 60 to enable passengers to easily remember the timetable. The number of available frequencies between 5 and 30 that evenly divide 60 are few in number. We have chosen to discretise the frequencies into the following set $\{\frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}, \frac{1}{18}, \frac{1}{20}, \frac{1}{25}, \frac{1}{30}\}$. Although the majority of these frequencies do not divide 60 exactly they provide a good starting point. In reality the frequency discretisation would be conducted by the network operator and may be based upon the frequencies that are currently in use on the existing network or may be dictated by central and/or local government.

In the following three sections we introduce the multi-objective algorithms used for frequency setting: 1) NSGAII, 2) multi-objective first descent, and 3) multi-objective tabu search. The genetic operators used for NSGAII are introduced along with two neighbourhood operators that can be used with multi-objective first descent and multi-objective tabu search.

6.4.1 NSGAII

NSGAII, first described in Section 4.2, was selected for use as the evolutionary framework. NSGAII was used to give reference approximate Pareto sets for each of the benchmark instances using a naïve multi-objective evolutionary algorithm. As the benchmark instances have not yet been used for frequency setting there does not exist any reference Pareto sets to compare against. The production of reference approximate

Pareto sets will allow for the comparison with other algorithms. Similarly to the network design stage, probabilities of crossover and mutation of 0.9 and $\frac{1}{|\mathcal{R}|}$ are used respectively, as defined in Section 4.2.

Genetic Operators

For frequency setting we use a vector of real coded variables to define the headway on each route. Figure 6.2 shows the difference between the decision variables for network design and frequency setting and the representation used. It should be stressed that during frequency setting we do not modify the route structure, only the route frequency is changed. We propose the use of uniform crossover to produce a single offspring. The offspring is then mutated with a probability of $\frac{1}{r}$. Mutation randomly selects to increase or decrease the frequency of a given route in the offspring with equal probability. Consider a route with a frequency of $\frac{1}{10}$, if the mutation operator selects at random that the frequency should be increased the next highest discretised frequency value will be selected, in our case this would be $\frac{1}{9}$. Similarly, if a decrease in frequency is randomly selected the new frequency would be $\frac{1}{12}$.

6.4.2 Multi-objective First Descent

First descent is a type of local search algorithm. Local search is a method of finding a solution to a problem from a number of candidate solutions. A search progresses by moving from solution to solution in the search space by applying neighbourhood moves that modify the current solution until an optimal solution is found or the termination criteria is reached.

First descent is a method of optimisation that makes a move to the first observed neighbour of the current solution that offers an improvement. In the worst case when no neighbour improves upon the current solution all of the neighbours will have been evaluated and the search terminates at a local optimum. Here we use a modified version

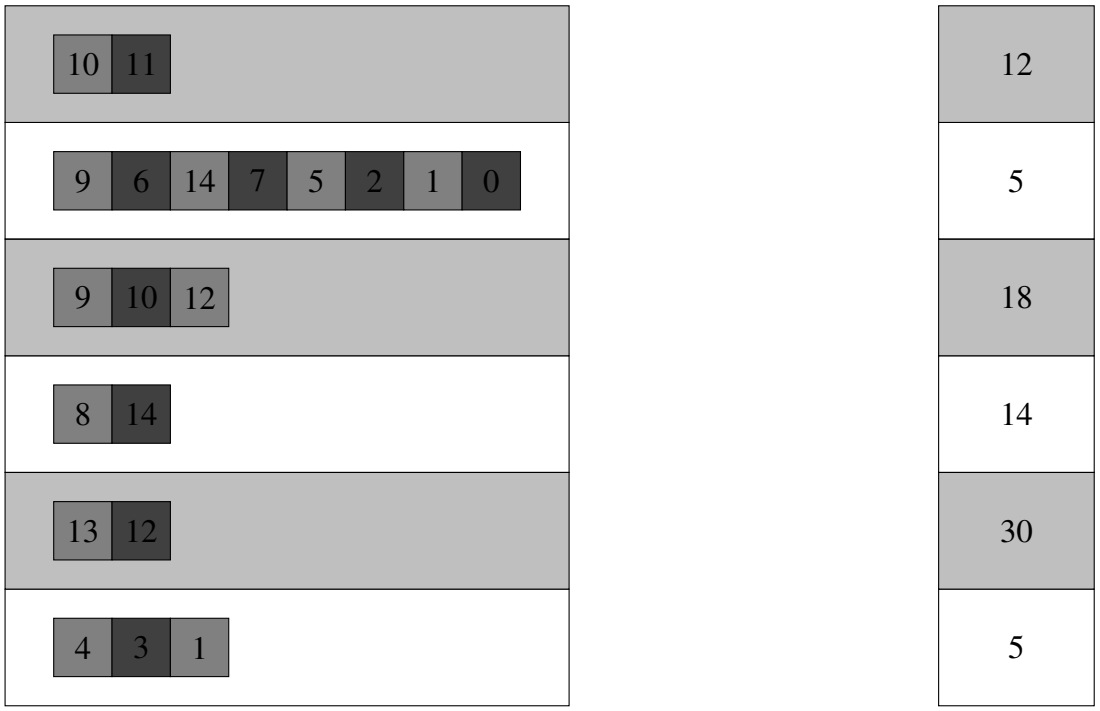


Figure 6.2: Comparison between the decision variables for network design (left) and frequency setting (right) together with the representation used.

of the first descent algorithm given in Algorithm 6.4 to allow for the multi-objective nature of the problem and to prevent the search from terminating when no improving neighbour solution is found. We shall refer to this approach as multi-objective first descent (MOFD).

As shown, Algorithm 6.4 maintains a nondominated set of solutions, which we call the archive or archive set. An initial solution is first generated and added to the archive. Until the stopping condition is reached (a given number of iterations or a maximum running time) the neighbours of the current solution are generated and evaluated in turn. If the solution currently being evaluated dominates the current solution then a move is made to that solution. The archive set is then updated to remove any solutions that are dominated by the incumbent solution. If the neighbour solution and the current solution are mutually nondominating then the neighbour is added to the archive set. When a dominating neighbour solution is not found, a random solution is chosen from the archive to become the current solution allowing the search to progress.

1: $s_{current}$ = generate initial solution
2: $archive = \{s_{current}\}$
3: **repeat**
4: N = get neighbours of $s_{current}$
5: **for** $s \in N$ **do**
6: evaluate(s)
7: **if** $s \prec s_{current}$ **then**
8: $s_{current} = s$
9: UPDATEARCHIVE(s)
10: $break$
11: **else if** s and $s_{current}$ are mutually nondominating **then**
12: UPDATEARCHIVE(s)
13: **if** no dominating solution found **then**
14: $s_{current}$ = $archive$[random()]
15: **until** stopping condition is satisfied
16: **return** $archive$

Algorithm 6.4: Multi-objective first descent with a nondominated archive.

MOFD is similar to Pareto local search (PLS) [122] which accepts a move to a neighbouring solution if it is not dominated by all nondominated solutions found so far in the search. The major differences between MOFD and PLS are: 1) MOFD moves to the first neighbour solution that dominates the incumbent, and 2) The archive set is only used to select a solution if the current solution has no dominating neighbours allowing the search to continue.

6.4.3 Multi-objective Tabu Search

As mentioned previously, tabu search imposes restrictions on the search via a set of memory structures referred to as tabu lists preventing the search from revisiting areas of the search space that have been recently explored. These restrictions can of course be overridden if for example moving to a tabu neighbour produces the best seen objective value so far (sometimes referred to as aspiration criteria). We could convert the first descent algorithm introduced previously into a tabu search, however we choose to adopt the well established multi-objective tabu search (MOTS) algorithm put forward by

Hansen [77]. The general algorithmic framework for MOTS is given in Algorithm 6.5 adapted for minimisation.

1: **for** $s_i \in X$ **do**
2: $s_i =$ random feasible solution
3: $TL_i = \varnothing$
4: $archive = \varnothing$, count = 1 and $\pi^k = 1/k$ for all k objectives
5: **repeat**
6: **for** $s_i \in X$ **do**
7: $\lambda = 0$
8: **for** $s_j \in X : s_j$ is nondominated by s_i and $f(s_i) \neq f(s_j)$ **do**
9: w = g(d($f(s_i), f(s_j), \pi$))
10: **for** all objectives k **do**
11: **if** $f^k(s_i) < f^k(s_j)$ **then**
12: $\lambda^k = \lambda^k + \pi^k w$
13: **if** $\lambda = 0$ **then**
14: $\lambda =$ randomly chosen vector
15: normalise(λ)
16: find solution y_i that minimises $\lambda \cdot f(y_i)\ \forall y_i \in N(x_i)$ and $y_i \notin TL_i$
17: **if** TL_i is full **then**
18: remove oldest element from TL_i
19: add move from s_i to y_i to TL_i
20: $s_i = y_i$
21: **if** y_i is nondominated by all points in ND **then**
22: UPDATEARCHIVE(y_i)
23: update π
24: **if** DRIFT-criterion reached **then**
25: randomly select one solution from X and set it equal to another randomly selected solution in X
26: count = count + 1
27: **until** stopping condition reached

Algorithm 6.5: Multi-objective tabu search [77] adapted for minimisation.

Hansen's method first initialises each of the solutions, $s_i \in X$, to a random feasible solution in the search space and associates with each an empty tabu list. A nondominated set, the archive set, is initialised and the range equalisation factors, π, are set to 1/k for each of the k objectives. The range equalisation factors are used to equalise the ranges

of the objectives and are calculated as:

$$\pi^k = \frac{1}{Range^k} \left[\sum_{i=1}^{n} \frac{1}{Range^i} \right]^{-1} \tag{6.3}$$

where $Range^k$ is the range width of objective k given a set of points. The last factor normalises the factors to ensure that $\pi \in \Delta$ and can be omitted. The λ-vector space Λ is used by Hansen defined as $\Lambda = \{\lambda \in \mathbb{R}^n | \lambda^k \in [0,1] \wedge \sum_{i=1}^{n} \lambda^i = 1\}$.

Each of the current solutions are then taken in turn and a move to a neighbouring solution is applied. The weight vector is determined such that the search moves away from other solutions. The closeness of a solution in terms of objective value determines the extent to which it influences the weight. We use the proximity function g(d) = $\frac{1}{d}$ where d is defined as the Manhattan distance norm used on the objectives scaled by the range equalisation factors i.e. d(x_i, x_j, π) = $\sum \pi^k |x_i^k - x_j^k|$ [77]. After determining the weights vector the standard tabu search procedure is applied to move to a non-tabu neighbour. In our case we define a neighbour to be tabu if a route has been assigned the same frequency in the last six iterations. If the neighbour is nondominated by all of the points in the current nondominated set it is added and the range equalisation factors recalculated. We permit a move to a tabu neighbour if it dominates any solution that is currently contained in the nondominated set. The DRIFT-criterion is used to insure a drift in movement over the Pareto front, helping to explore the entire length of the front and also helping to locate non-dominated solutions [77].

6.4.4 Neighbourhood Operators

First descent and tabu search both require the generation of neighbouring solutions for the search to progress. These are generated by a neighbourhood operator. We propose two neighbourhood operators. The first changes the frequency on a single route and the second operator simultaneously changes the frequency on two routes. It should be noted

that the change in frequency must be to an adjacent frequency in the set of discretised frequencies.

6.4.5 Candidate Solution Selection

As mentioned earlier, our problem formulation for frequency setting is conducted on an already established route network. In the case of optimising the frequencies on a network currently being used by the public, the initial route network is already decided. When redesigning a network the selection of route networks for use during frequency optimisation must be carefully considered. The structure of the network will have a direct impact on the quality of solutions produced from frequency setting. If a selected route network favours the operator then the passenger objective will be limited due to the underlying route construction. Conversely, if a route network favours the passenger then the resultant operator objective will be higher.

We saw in Chapter 4 that our approach to network design produces an approximate Pareto set of solutions. The best approach would be to take each individual solution produced and optimise the frequencies on it. The result from each individual solution could then be combined to produce a three dimensional Pareto set allowing a human decision maker to choose the solution that best fits their requirements. For small problem instances such as Mandl's, this approach works well and would produce the best approximation to the true Pareto set. For larger instances, however, the running time required is prohibitive. As all the solutions from network design cannot be optimised for frequency setting a subset of the solutions is therefore taken that provide a good representation of the Pareto front.

The selection of solutions from network design can have a direct impact on the quality of the solutions produced. If the solutions share similar objective values (i.e. clustered along the Pareto front) then it is likely that they will also share similar routes. An intelligent selection strategy is therefore required to select a "good" range of solutions

that does not show any bias towards the operator or passenger viewpoint. Under real world constraints it is expected that the network operator would be able to provide guidance with regards to selection criteria based upon their experience of the local area. We define a "good" selection of solutions to be one that results in solutions having a low similarity i.e. their route structure differs greatly. To achieve this we propose to use the objective values to determine which solutions should be selected for frequency setting.

In our case we propose a selection strategy, that is used with each of our three approaches to frequency setting, which selects five solutions from the approximate Pareto set produced during the network design stage. We concentrate on the passenger objective for selection as a high passenger cost equates to a low passenger cost and vice versa. Solution selection takes the best and worst solution i.e. the solution associated with the minimum and maximum passenger objective objectives. Three remaining solutions are then taken by selecting the solutions closest to the first, second and third quartile of the passenger objective values. For brevity we shall refer to the solutions as $\mathcal{F}_P, \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ and $\mathcal{F}_O$ to denote the minimum passenger objective, first, second and third quartile and maximum passenger objective respectively. This is shown in Figure 6.3 for the Edinburgh200 problem instance where the selected route network solutions are given in red.

6.4.6 Population Generation

The selection of network design solutions provides a basis upon which frequency setting can be applied. Given a network solution, a number of initial frequency setting solutions must be generated if using a population based approach, such as NSGAII or MOTS, whereas a single solution should be produced for MOFD. We use a vector of real coded variables equivalent in length to the number of routes in the underlying network where each element represents the frequency on a given route. Furthermore the variables contain the headway between vehicles on the same route to prevent issues with the representation of floating point numbers (recalling that frequency is the reciprocal of

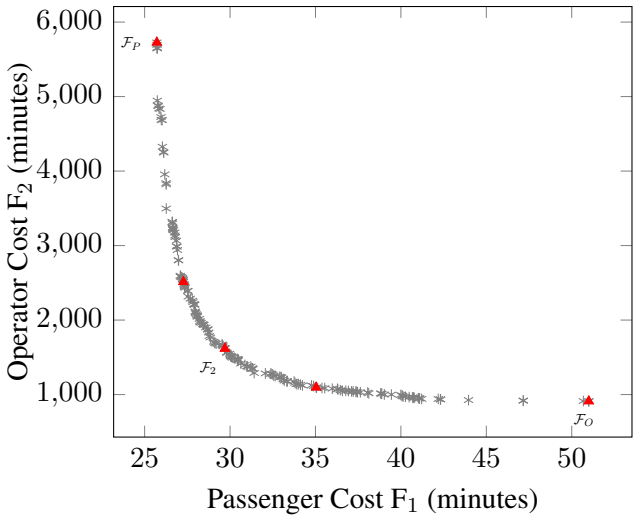


Figure 6.3: Pareto set for Edinburgh200 showing network solutions selected for frequency setting in red.

the headway).

Under an unconstrained scenario, the generation of initial solutions is trivial as there are no constraints that need to be met. In the latter sections of this chapter we will look at constrained scenarios where the choice of frequencies have a direct impact on the fleet size and therefore the capacity available on each route. For now we opt for a random generation scheme where the frequency for each route is selected at random from the set of discretised frequencies.

One of the major pitfalls in the creation of solutions is the need to assign a frequency to all the routes before it can be determined if the solution is feasible. To assign the demand to the network, and hence determine if there is sufficient capacity on each link, the Optimal Strategies algorithm must be run, resulting in the need for each route to have an assigned frequency. Similarities can be drawn here to the evaluation for the network design stage. A route set is a collection of interdependent routes that must be evaluated as a whole and not individually. As frequency setting involves assigning passenger flows through the network, (i.e. assigning demand to individual route segments), the frequency of each route must be known otherwise the frequency share rule cannot be applied correctly. In this situation the creation of an initial population is more time

consuming. There are no procedures that can be quickly applied to validate if the solution is feasible under real world conditions.

6.4.7 Experimental Method

In this section we first demonstrate the effect that the choice of solution from the network design stage has on the produced Pareto set using NSGAII. We then examine the use of MOFD using two different neighbourhood operators and compare the quality of solutions produced. Finally, MOTS is compared with NSGAII and MOFD. NSGAII was run for 200 generations using a population of 200 unique solutions with twenty replicate runs for each of the five solutions taken from network design. The twenty runs for each network solution are combined into an approximate Pareto set for the instance. MOFD and MOTS were each executed for the average running time of the twenty replicate runs for each solution using NSGAII – twenty replicate runs are used for each solution. It should be noted that the average running time over the five solutions is not consistent due to the difference in evaluation time required. In general $\mathcal{F}_P$ requires a greater running time due to a greater average route length compared with $\mathcal{F}_O$, leading to more vertices in the generalised transit network, and hence more arcs to be processed.

All experiments were run for each of the benchmark instances used for network design. Pareto sets from Algorithm E were used to extract the five solutions discussed above. Initial frequency solutions were generated by randomly selecting frequencies for each route from the set of allowed frequencies as discussed in the previous section.

6.4.8 Experimental Results

Figure 6.4 shows the Pareto sets achieved for each of the five solutions for the Mandl and Edinburgh200 benchmark instances. From the Mandl plot it can be seen that $\mathcal{F}_P$ is able to achieve the lowest average expected travel time and $\mathcal{F}_O$ achieves the lowest operator cost. This is more evident in the Edinburgh200 plot where there is a clear

differentiation between the five solutions with a clear increase in passenger cost and decrease in operator cost as we progress from $\mathcal{F}_P$ through to $\mathcal{F}_O$. This variation in approximate Pareto sets by network solution can be explained by the underlying route structure. Consider $\mathcal{F}_O$, the best solution from the operator perspective, the routes in this solution compared with $\mathcal{F}_P$ are significantly shorter, therefore a passenger will most likely have to transfer to reach their destination, resulting in a longer travel time. As the routes are shorter the number of duplicate vertices (i.e. transfer points) in the solution are reduced meaning that a passenger will generally have to travel further to reach a transfer point compared with $\mathcal{F}_P$. These factors result in the separation of the Pareto sets for the five selected solutions.

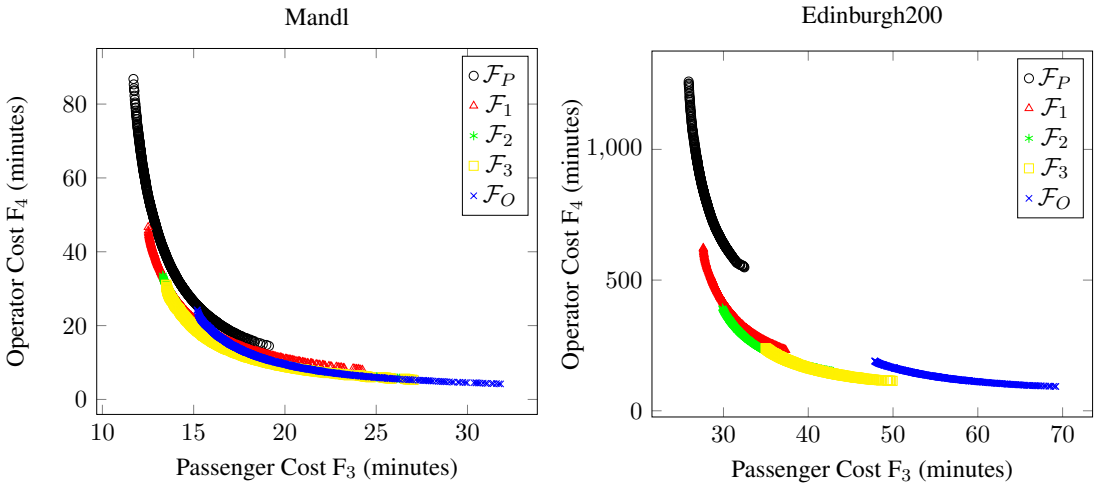


Figure 6.4: Pareto fronts for five selected solutions using the Mandl and Edinburgh200 problem instances.

MOFD was run using the parameters given above and compared with the combined approximate Pareto sets produced using NSGAII. Table 6.3 gives the $\mathcal{S}$ metric values for the combined Pareto sets. If we first examine the effect of the neighbourhood operator, we can see that changing the frequency on two routes causes a deterioration in $\mathcal{S}$ metric in six out of the seven instances, when compared to a frequency change on a single route. Changing the frequency on two routes means that larger neighbourhood moves are being explored which may be too disruptive to the search process. If we compare the $\mathcal{S}$ metric values for MOFD and NSGAII it is clear that NSGAII is able to provide

the best metric over all the instances.

Instance	MOEA	MOFD A	MOFD B
Mandl	**18938**	18932	18841
Mumford0F	**166295**	164783	165408
Mumford1F	**752960**	743756	739633
Mumford2F	**5414941**	5388060	5366840
Mumford3F	**17020405**	16962438	16933758
Nottingham100	**760768**	745637	744474
Edinburgh200	**11424005**	11378258	11366198

Table 6.3: $\mathcal{S}$ metric values for NSGAII and MOFD.

Given the nature of MOTS in that all neighbours must be evaluated in each iteration, the first neighbourhood operator was used generating a maximum of twelve neighbour solutions for Mandl's problem instance and one hundred and eighty for Edinburgh200. This is a reflection upon the computation time required to evaluate a single solution. Similarly to MOFD, MOTS was run for a running time not exceeding that of the average for the twenty replicate runs for each solution of the MOEA. A tabu list size of 6 was used and a population size of 10. Table 6.4 gives the $\mathcal{S}$ metric for MOTS using the given parameters compared with the MOEA.

Instance	MOEA	MOTS
Mandl	**18938**	**18938**
Mumford0F	166295	**166583**
Mumford1F	752960	**754580**
Mumford2F	**5414941**	5387978
Mumford3F	**17020405**	16946253
Nottingham100	**760768**	760107
Edinburgh200	**11424005**	11378031

Table 6.4: $\mathcal{S}$ metric values for Tabu search compared with NSGAII.

From Table 6.4 it can be seen that MOTS is able to achieve an improved $\mathcal{S}$ metric for the Mandl and Mumford1F instances, with NSGAII obtaining the best metric value for the remainder. If the number of iterations are compared for the two approaches there is a stark difference. For Mandl's instance MOTS executes 411 iterations; however,

for Edinburgh200 only 25 iterations are completed. This highlights the problem of examining neighbourhoods where, as the problem size increases, so does the number of neighbours that must be evaluated. If this is coupled with the increase in evaluation as the problem instance scales, then algorithms that rely on neighbourhood operators clearly become less suitable.

6.5 Constrained Capacity

In the previous section we assumed that the capacity of vehicles operating the routes was infinite or that the operator would ensure adequate capacity on routes. This is an unrealistic assumption as vehicles have restrictions imposed on the number of passengers that can be safely carried. Furthermore there are both physical and financial limitations to the number of vehicles that can operate a route. Restricting the capacity of vehicles available may mean that the number of vehicles needed to operate a route has to be increased to allow for the volume of passengers travelling. Many public transport systems allow for passengers to be seated or standing during their journey. We will ignore this detail and simply consider the maximum capacity i.e. the number of passengers allowed to be seated and standing. We also assume that the network is operated by a homogeneous fleet with each vehicle having identical capacity. As the demand information available from the benchmark instances is aggregated over a 24 hour period we assume that the load factor cannot exceed 100% of the available capacity as discussed in the Section 3.2 for off-peak periods.

To choose a sensible capacity for vehicles we use bus specifications published by the UK Government for single deck, double deck and bendy buses[2]. We have chosen to set our vehicle capacity as the smallest available single deck vehicle which has a total capacity of 70 passengers. This capacity relates to the demand scaling conducted previously where we assumed buses would maintain an average ridership of 35 persons over a

[2]https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/4260/buslength.xls

24-hour period.

To allow for the assignment of passengers to routes we now use the full Optimal Strategies algorithm defined in Section 6.2 to enable the passenger demand to be assigned to the network. Given that the demand information for the problem instance is aggregated over a 24-hour period, we assume that demand is uniform throughout the day. As such we can obtain an hour demand for each route by dividing the total passenger volumes, assigned using the Optimal Strategies algorithm, by 24.

Before progressing we first define the criteria for a feasible solution under this constraint. Given a solution we examine each route in turn and take the route segment with the highest hourly demand volume. If the demand on any route exceeds that which can be carried given the route's frequency and vehicle capacity then the solution is deemed infeasible. On the other hand, if the demand requirements are satisfied for each route in the solution, then it is feasible.

6.5.1 Experimental Method

We now examine the change in solution quality when introducing a capacity constraint to our proposed MOEA. NSGAII was executed for 200 generations using a population of 200 unique solutions generated at random using the approach described previously. Similarly to the unconstrained scenario, each network design solution has twenty independent runs with a final approximate Pareto set combined from the five solutions selected from the network design stage.

6.5.2 Experimental Results

From Table 6.5 we can see that, when constraining the capacity of the vehicles, a decrease in the $\mathcal{S}$ metric is recorded. For the Mumford1F, Mumford2F, Mumford3F and Edinburgh200 instances we were unable to generate any initial solutions for $\mathcal{F}_O$.

For $\mathcal{F}_O$ (the solution with the lowest operator cost) it was found that certain routes had extremely high passenger volumes. These routes played a central role in allowing passengers to transfer between vehicles, i.e. they became transfer hubs, resulting in demand volumes that could not be satisfied.

Instance	Unconstrained	Constrained
Mandl	**18938**	**18938**
Mumford0F	**166295**	165127
Mumford1F	**752960**	751344
Mumford2F	**5414941**	5387985
Mumford3F	**17020405**	16923982
Nottingham100	**760768**	760213
Edinburgh200	**11424005**	11394427

Table 6.5: $\mathcal{S}$ metric comparison using NSGAII under an unconstrained and constrained capacity.

If we look at the approximate Pareto sets produced for each of the five network solutions there is a clear trend as the size of the problem instance increases. From Figure 6.5 it can be seen that for Mumford0F, when applying the capacity constraint, the exploration of the higher ends of the passenger objective is reduced. This is shown to a greater extent in Figure 6.6 for the Edinburgh200 instance. No feasible solutions could be found for $\mathcal{F}_O$ and the spread of solutions for $\mathcal{F}_3$ is also reduced.

Reflecting upon the reduction in the higher ends of the passenger objective we note that this mainly affects the solutions that tend toward favouring the operator i.e. those with shorter route lengths. As the route lengths decrease more passengers will be forced to transfer to reach their destination. A number of routes will act as "transfer routes" allowing passengers to transfer, but their popularity results in demand that cannot be satisfied. The route networks that favour the network operator are clearly unsuitable and reflect the need for careful consideration when selecting the networks upon which frequency setting is applied.

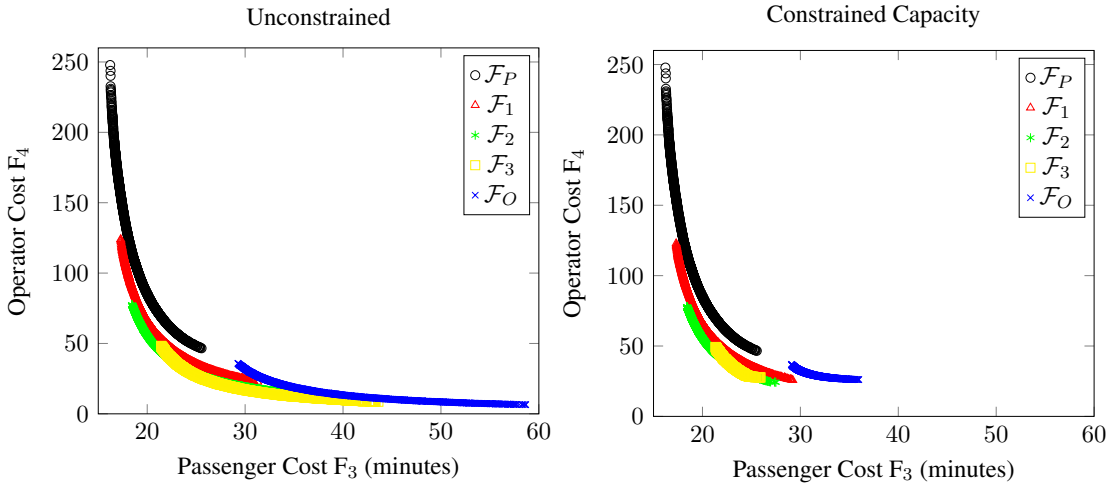


Figure 6.5: Pareto fronts obtained for the network design solutions when applying frequency setting for the Mumford0F instance with and without a capcity constraint.

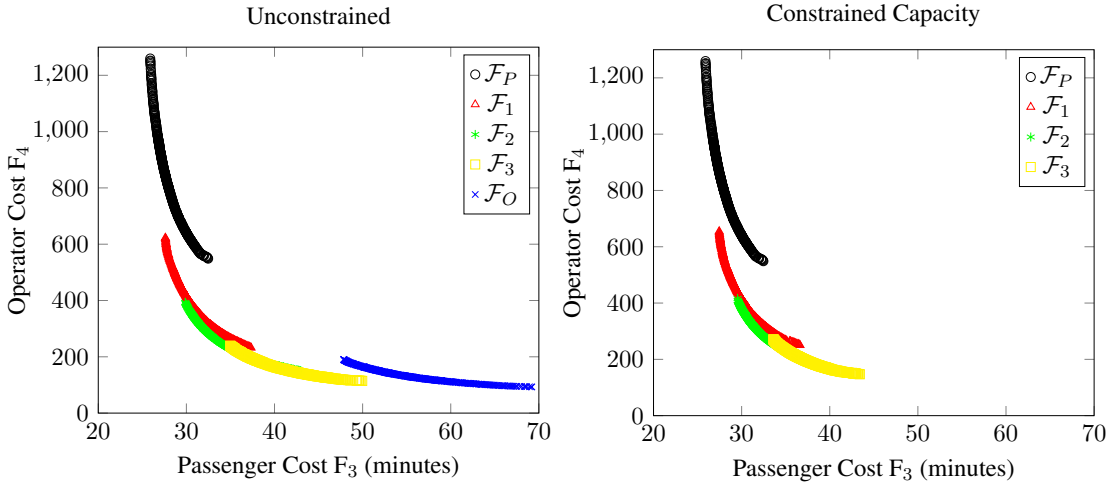


Figure 6.6: Pareto fronts obtained for the network design solutions when applying frequency setting for the Edinburgh200 instance with and without a capcity constraint.

6.6 Constrained Capacity & Fleet Size

Constraining the capacity of vehicles enables the network operator to obtain sensible solutions that are able to be implemented providing there is an adequate fleet available. However, even for long term planning there may be fleet sizes that cannot be achieved given the operators available capital. Introducing a constraint on the maximum fleet

size therefore seems a necessary step to ensure the optimisation algorithm does not spend time exploring areas of the search space that are not feasible from the operator viewpoint.

If a maximum fleet size constraint is introduced given our current route network solution selection strategy, the result is intuitive. From Figures 6.5 and 6.6 we can see that if a fleet size constraint is imposed, the effect will be to reduce the exploration of the solutions that favour the passenger objective i.e. $\mathcal{F}_P$. Combining this with the observations made from the constrained capacity experiments, it is evident that our strategy for selecting network design solution is therefore unsuitable.

The selection of network design solutions at the extremes of the passenger objective results in solutions that are infeasible for frequency setting. If capacity and fleet size were to be severely constrained then it is possible that $\mathcal{F}_1$ and $\mathcal{F}_3$ may also become unsuitable. This reflects the fact that network design was conducted without considering the available fleet size or capacity available. It therefore seems that an alternative approach to both frequency setting and network design are required to ensure feasible solutions are produced. In the next section we provide a discussion of these alternative approaches.

6.7 Discussion: Alternative Approaches to Frequency Setting

As we have seen, the selection of a route network has a direct impact on not only the quality of solutions produced from the operator and passenger viewpoints but also whether frequency setting can even be applied – given constraints on vehicle capacity and available fleet size. Designing a route network without taking into account the available capacity or fleet size may, therefore, not always be suitable, and alternative methods should be considered. However, evaluation of solutions for both the network

design and frequency setting stages simultaneously is time consuming and must be conducted on a network as a whole rather than on a single route. As such we now discuss a number of alternative approaches that may provide an improved approach to both network design and frequency setting.

One of the assumptions our problem formulation has made is that we are designing a public transport network to be implemented. However the UTNDP can take two differing approaches:

1. To design an operational public transport system that will be implemented by the network operator.

2. To provide long term planning support by producing a number of alternative networks that will allow an operator to plan for future investment.

Depending upon the aim of the network operator it may be prudent to combine or split the network design and frequency setting stages. It is therefore questionable whether a single approach will be able to tackle the UTNDP from both viewpoints. A network operator, for example, may wish for a network to be designed that causes minimal disruption. This may mean producing a network that does not make large route changes to that currently in use, allowing the operator to quickly implement the change.

The separation of network design and frequency setting is a significant reason for the production of infeasible solutions during the frequency setting stage. When considering both capacity and fleet size constraints, feasible solutions from the network design stage cannot be used (i.e. solutions at the extremes of the passenger and operator objectives). This impacts our initial problem formulation and indicates that constraints need to be added during the network design stage to ensure the solutions that are produced will be able to undergo frequency setting. Our network design stage produces infeasible solutions (from the frequency setting point of view) as they require an unrealistic frequency and/or capacity to ensure sufficient capacity on a route. An option

is to modify our initial network design formulation to assign passengers to routes, i.e. attribute to each route edge a passenger volume similar to Optimal Strategies, but then to continue to use the APSP for calculating the passenger paths. Solutions could now be checked to ensure that given the default frequency of ten minutes whether a sensible capacity is required on each route. The setting of a sensible capacity is difficult to judge and could become a parameter to the algorithm based upon the network operators current fleet size.

Alternatively one might set both the route network and frequency upon each route simultaneously. As previously mentioned, both the network design stage and frequency setting stage are $\mathcal{NP}$-hard. Combining both stages together will remove the issue of generating infeasible solutions as they could be prevented from entering the population. But this would result in a significant increase in running time. Evaluation would need to be conducted using the Optimal Strategies algorithm or similar method that assigns passenger demand to the network. Using this approach it may be difficult to maintain population diversity in terms of route networks when setting frequencies simultaneously. We have seen that a single route network can produce a range of frequency solutions and this could affect the route network diversity. To tackle this issue it may be possible to add a third objective that compares the structure of designed route networks to that currently in operation using a diversity measure similar to that introduced in Chapter 4. Adding the objective would aid in maintaining diversity in the population providing that a network redesign is being conducted rather than the design of a completely new public transport network.

If the running time of the algorithm is not of critical importance, then a more realistic evaluation of the route network along with the associated frequencies could be modelled via a public transport simulation package. Using a modelling package rather than the APSP or Optimal Strategies algorithm would allow for the consideration of a number of factors such as the stochastic nature at which passengers arrive at bus stops, other traffic on the roads that may cause congestion, and the walking time of passengers between bus

stops. Obviously this would be a very expensive optimisation procedure, but surrogate models may help to reduce the computational burden.

An alternative approach would be to run network design and frequency setting in an iterative procedure. This approach would allow network design to run for a set number of generations and then feed all or a subset of the solutions into frequency setting. Once frequency setting has completed, the average frequency could then be computed and fed back into network design as the transfer penalty. It may also be possible to use the route frequencies rather than computing an updated transfer penalty. The effects of applying crossover and mutation would have to be carefully examined to investigate if using the frequencies on routes from differing route sets would cause a degradation in solution quality.

If the operator does not consider longer term planning to be a priority and requires a public transport system that can be implemented in a short time frame, then it may be more appropriate to use a single objective optimisation for frequency setting. Sensible constraints regrading fleet size could be imposed during the network design stage. The network operator could then select a solution or solutions from the produced Pareto set. Using both capacity and maximum allowed fleet size constraints the passenger objective could be optimised in an attempt to improve the travel time whilst not incurring extra cost to the operator. This approach would reduce the size of the search space allowing for the optimisation to be performed in a shorter time frame.

Finally, one of the major issues with research in to the UTNDP is the lack of real world data available. Current research, including our own, focuses on artificially created instances that make it almost impossible to trial the produced networks in the real world i.e. using some form of simulation to accurately model the local population. With active industry collaboration to provide details on the route networks currently used and accurate demand figures it should be possible to progress the research and provide optimisation algorithms that are able to produce sensible public transport networks.

6.8 Summary

This chapter has described three multi-objective algorithms for the frequency setting stage of the UTNDP. NSGAII, multi-objective first descent and multi-objective tabu search were compared empirically. NSGAII was shown to consistently outperform the other two methods producing improved $\mathcal{S}$ metrics for all the available problem instances. The effects of constraining the capacity upon vehicles was then investigated and found to reduce the number of solutions with high passenger objective values. A discussion was then made regarding the introduction of a maximum fleet size constraint and the effect this would have.

Our study on the introduction of capacity and fleet size constraints highlighted that the selection of solutions from network design must be carefully considered. The effect of introducing a capacity constraint was to remove the network that favoured the operators. Introducing a fleet size constraint removed the network that favoured the passenger. Alternative approaches to network design and frequency setting were then discussed that would ensure solutions produced during network design are feasible with respect to frequency setting.

Chapter 7

Conclusions & Future Work

7.1 Conclusions

This thesis has examined the first two stages of the UTNDP, namely the network design and frequency setting problems. Problem definitions for both have been presented along with seven problem instances that are publicly available for other researchers. We then presented a proof that the network design problem is $\mathcal{NP}$-Complete. A discussion around the use of the transit network for the evaluation of the passenger objective was then made. The need to identify and penalise transfers is a key aspect of any public transport system. Passengers wish to avoid having to make a transfer and will ultimately not use a service if excessive transfers are necessary. To identify transfers we use the transit network, which is significantly larger than the route network, leading to large running times. Alternative approaches were presented that used the smaller route network but were found to be unsuitable in either being unable to identify transfers correctly or resulting in larger running times. The bottleneck when using the transit network is the application of an all pairs shortest path algorithm. To reduce the computation time we parallelised the Floyd-Washall algorithm on the GPU and found that it offered significant time savings.

Given the $\mathcal{NP}$-Hard nature of the network design and frequency setting problems we chose to approach each separately, despite them being highly coupled. We described a multi-objective evolutionary algorithm (MOEA) that is able to outperform the state of

the art from the literature. A heuristic construction algorithm was proposed to seed the MOEA with a high quality initial population. By seeding the initial population using the heuristic construction algorithm improved approximate Pareto sets were produced over five of the seven problem instances. Improvements were also achieved across all instances when comparing the best objectives values from the passenger and operator perspectives.

The approach was further improved by adding several mutation operators and replacing SEAMO2 with the NSGAII evolutionary framework. An extensive analysis of the performance of the proposed genetic operators was also conducted. It was found that as the underlying problem size increases, then so does the performance of the operators. Given the multi-objective nature of the network design problem the creation of dominating solutions is challenging. Furthermore, our operators tend to focus on improving one objective whilst simultaneously degrading the other. We therefore concluded that using all the operators together produces the best quality approximate Pareto sets for the majority of instances.

A comparison was then made between our proposed algorithm and the state of the art from the literature. Over the available problem instances our method was able to achieve the best result from the passenger perspective in six out of seven cases and all of the instances from the operator perspective. This comparison, however, highlighted a major issue with the UTNDP currently: a lack of problem instances that are publicly available along with multiple problem formulations make direct comparisons difficult. Although the lack of problem instances has been improved recently by Mumford [114] with the publication of four benchmark instances, there is still a long way to go before a library of instances are available ranging from small to real-world sizes.

A further complication is the range of problem formulations in the literature with some authors only considering a single objective optimisation. However, the appropriateness of tackling the network design problem using single or multi-objective optimisation depends upon the context. If the optimisation is for long term planning a multi-objective

optimisation may be more appropriate. It would allow for the operator to compare the improvement in the level of service provided to passengers versus the additional operating cost required i.e. extra vehicles. A single objective optimisation, focusing on the passenger objective, may therefore be used for short term planning where the operator wishes to maintain their current infrastructure and operating costs.

The comparison also highlighted issues with the publication of results for the network design problem. Given it's multi-objective nature, comparing the best route sets from the passenger and operator perspectives is unsuitable. Publication of the achieved approximate Pareto sets would allow for a multi-objective comparison reflecting the true performance of alternative approaches.

While deploying our approach to the network design problem, we saw that the vast increase in running time as the problem size grew was a significant drawback. High performance computing resources were used but this requires a network operator to have access to similar resources. An alternative approach was explored that provided an approximation to the actual objective values for the passenger objective in a significantly reduced time frame. This is know as a surrogate model or surrogate assisted optimisation.

Four surrogate models were presented and compared empirically. We found that surrogate models perform well compared with the original algorithm using evaluation on the transit network. Under constrained running times the surrogate models were able to outperform the original algorithm producing improved approximate Pareto sets. To provide a greater speedup, surrogate models were then combined with GPU based evaluation. It was found that given the small size of the route networks the GPU was unable to provide a speedup for our problem instances. However, for larger instances with a few thousand vertices GPU based evaluation coupled with surrogate models will almost certainly provide benefit.

Surrogate models provide a speedup by reducing the size of the graph needed for evaluation i.e. using the route network rather than transit network. It may be more beneficial

to provide a highly detailed evaluation that models all aspects of urban transport systems (i.e congestion, rail, subway, walking time). A more detailed evaluation will of course result in longer running times but also produce a route set that is better suited to the intended environment. A balance must therefore be made between the complexity of the model and what is considered as a reasonable running time. Without the engagement of industrial partners this will be difficult to achieve.

We then moved onto the frequency setting problem highlighting some of the issues involved when separating the network design and frequency setting stages. During frequency setting we assign passengers to routes using the Optimal Strategies algorithm of Spiess and Florian [135]. Here we must select route sets to have frequency setting applied. Also, all route sets produced during the network design stage cannot be evaluated due to the extensive running time involved as the problem size scales. A route network selection strategy was proposed that selected five solutions from the approximate Pareto sets output from the network design stage. Each of these route sets then independently undergoes frequency setting, producing five approximate Pareto sets that are combined to give a final approximate Pareto set. Three multi-objective algorithms were described and empirically evaluated under an unconstrained scenario. It was found that NSGAII performed well in comparison to multi-objective tabu search (MOTS) and multi-objective first descent (MOFD). The poor performance of MOTS and MOFD can be attributed to the expensive evaluation function that must be performed on each neighbour. It was noted that MOTS and MOFD completed far fewer iterations compared with NSGAII under the same running time.

A capacity constraint was then introduced limiting the number of passengers that can be carried on each vehicle. For the smaller problem instances the capacity constraint had little effect due to the small route lengths and low demand across the networks. However, as the problem size increases capacity plays a more pivotal role. Some solutions from the network design stage, that heavily favoured the network operator, were found to be infeasible when applying frequency setting with a capacity constraint.

The inability to create frequency solutions based upon a given route network configuration can be linked directly to the separation of the network design and frequency setting problems. Creation of route sets ignoring capacity is unsuitable and needs to be reconsidered to prevent time being wasted exploring infeasible areas of the search space. By assuming an infinite capacity during network design route sets are constructed that place the highest demand node pairs on the same route with minimal route overlap (when the route set favours the network operator). This has a two-fold effect: 1) the high demand cannot be served as it is now less likely to be distributed over multiple routes, and 2) higher transfer volumes on to already highly utilised routes. To overcome this issue several alternative approaches were discussed as a topic for future work.

7.2 Future Work

As previously mentioned there, is a lack of problem instances that can be used to compare methods. As such there is a strong need for a set of problem instances that allow researchers to compare their approaches over a range of scenarios, both artificial and real world. The production of instances is, however, challenging. For artificial instances the spatial layout and connectivity of the network should reflect that seen in the real world, otherwise methods may become over-fitted to these artificial instances and perform poorly on real world examples. Demand over the network must also be calculated carefully to ensure it is sensible and reflective of a town/city of the same spatial layout. Nevertheless the creation of a library of instances would greatly benefit future research.

Artificial instances can provide a means for comparing approaches, but real world instances must also play a far greater role. Industrial collaboration to obtain the necessary demand data and network requirements will be invaluable for future research. The acquisition of real world data will allow for route networks produced to be compared with those currently in use. This may be undertaken through simulation or limited trials

on the transit network itself. The more industrial collaboration that can be sought the greater the benefit for researchers and the UTNDP as a whole.

The commercial sensitivity of passenger demand for current services is a major obstacle to the creation of real world benchmark instances. Development of methods that are able to accurately predict passenger demand, for example based upon population density, will provide a far easier and cost effective means compared with direct passenger travel surveys. This will also overcome the issue of researchers developing approaches on a specific problem instance that has industry collaboration with a condition that the underlying data cannot be released.

While industrial collaboration may provide the means for creating more problem instances it can also enable more accurate problem formulations to be produced. Network operators can share their perspectives which can be pooled to look for similarities. Given the large range of formulations for the UTNDP this may have the added benefit of producing an agreed formulation resulting in an easier method for comparing alternative approaches for the UTNDP.

An in-depth analysis of transport networks currently in use may also be able to shed light on similarities in terms of route structure. The identification of similarities will help to constrain the problem reducing the size of the search space. Given the large number of public transit networks, it seems unlikely that there is not a level of similarity between them. For example degree of route overlap, distance covered by a single route and stops per route. If similarities can be found they can be fed into the creation of artificial instances to ensure that they are reflective of the real world.

Although the UTNDP is inherently multi-objective, compromises must be made to achieve a method that has a reasonable running time. If effective constraints can be introduced from either the passenger or operator perspective – whether that be through industry collaboration or analysis of existing networks – a single objective formulation may be achievable. This would have the advantage of being less computationally expensive. By utilising industry collaboration it may be found that operators would

rather improve upon a given solution rather than producing a Pareto set of solutions.

To provide solutions that can be deployed on existing networks both the network design and frequency setting stages need to be tackled simultaneously. Creating route sets without considering capacity and/or frequency is acceptable if the problem is academic with the focus on creating improved algorithms. However, for applications to real world networks it is questionable whether a multi-objective algorithm approach is truly warranted. Furthermore the extensive disruption a total redesign of a transit network would cause may lead to network operators favouring small improvements on their current network configuration. It is therefore prudent to consider two alternative approaches. One could take the current route network and make small improvements, whether this be through the use of heuristics or the application of an optimisation algorithm. Tied to this could be a function that measures the disruption caused to passengers, which could then be traded off against improvements from the passenger viewpoint. In this scenario we could assume that the resources available to the network operator are fixed.

A theme throughout this thesis has been the computational expense of evaluating solutions. From the network design perspective, this is due to the all pairs shortest path. Development of a shortest path algorithm that is able to efficiently deal with vertices that have conditional costs of traversal (i.e. a transfer penalty) without the need to introduce extra vertices could provide a significant time saving. However, more research is needed into the objective formulation and whether the approach used in this thesis is truly reflective of passenger travel patterns.

The UTNDP is complex with conflicting viewpoints and a stochastic nature that must be accounted for in an accurate formulation of the problem. It is therefore questionable whether a completely accurate formulation will ever be possible for the UTNDP without the need for expensive modelling to evaluate route sets that, in turn, may prove too costly. It is therefore our belief that the identification of similarities and discussion with network operators is key for the UTNDP to progress.

Bibliography

[1] *SATURN Manual*, 11.2 edition. URL http://www.saturnsoftware.co.uk/saturnmanual/.

[2] Google distance matrix API. URL https://developers.google.com/maps/documentation/distancematrix/intro.

[3] National public transport data repository (nptdr). URL http://data.gov.uk/dataset/nptdr.

[4] M Hadi Baaj and Hani S Mahmassani. Trust: A lisp program for the analysis of transit route configurations. *Transportation Research Record*, (1283), 1990.

[5] M Hadi Baaj and Hani S Mahmassani. An ai-based approach for transit route system planning and design. *Journal of advanced transportation*, 25(2):187–209, 1991.

[6] M Hadi Baaj and Hani S Mahmassani. Hybrid route generation heuristic algorithm for the design of transit networks. *Transportation Research Part C: Emerging Technologies*, 3(1):31–50, 1995.

[7] Saeed Asadi Bagloee and Avishai Avi Ceder. Transit-network design methodology for actual-size road networks. *Transportation Research Part B: Methodological*, 45(10):1787–1804, 2011.

[8] Barrie M Baker and MA Ayechew. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800, 2003.

[9] Richard Balcombe, Roger Mackett, Neil Paulley, John Preston, Jeremy Shires, Helena Titheridge, Mark Wardman, and Peter White. The demand for public transport: a practical guide. 2004.

[10] Roberto Baldacci, Eleni Hadjiconstantinou, and Aristide Mingozzi. An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations research*, 52(5):723–738, 2004.

[11] Roberto Baldacci, Nicos Christofides, and Aristide Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

[12] Raúl Baños, Julio Ortega, Consolación Gil, Antonio Fernández, and Francisco De Toro. A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows. *Expert Systems with Applications*, 40(5):1696–1707, 2013.

[13] Gabriela Beirão and JA Sarsfield Cabral. Understanding attitudes towards public transport and private car: A qualitative study. *Transport policy*, 14(6):478–489, 2007.

[14] John E Bell and Patrick R McMullen. Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18(1):41–48, 2004.

[15] Robert W Blanning. The construction and implementation of metamodels. *simulation*, 24(6):177–184, 1975.

[16] Jeremy J Blum and Tom V Mathew. Intelligent agent optimization of urban bus transit system design. *Journal of Computing in Civil Engineering*, 25(5): 357–369, 2010.

[17] JJ Blum and TV Mathew. Implications of the computational complexity of transit route network redesign for metaheuristic optimisation systems. *IET Intelligent Transport Systems*, 6(2):124–131, 2012.

[18] Jürgen Branke and Christian Schmidt. Faster convergence by means of fitness estimation. *Soft Computing*, 9(1):13–20, 2005.

[19] Larry Bull. On model-based evolutionary computation. *Soft Computing*, 3(2): 76–82, 1999.

[20] Michael Bussieck. *Optimal lines in public rail transport*. Citeseer, 1998.

[21] Bernard F Byrne. Public transportation line positions and headways for minimum user and system cost in a radial case. *Transportation Research*, 9(2-3):97–102, 1975.

[22] Massimiliano Caramia and Paolo Dell'Olmo. *Multi-objective management in freight logistics: Increasing capacity, service level and safety with optimization algorithms*. Springer Science & Business Media, 2008.

[23] S Carrese and S Gori. An urban bus network design procedure. In *Transportation planning*, pages 177–195. Springer, 2002.

[24] Avishai Ceder and Nigel HM Wilson. Bus network design. *Transportation Research Part B: Methodological*, 20(4):331–344, 1986.

[25] Partha Chakroborty. Genetic algorithms for optimal urban transit network design. *Computer-Aided Civil and Infrastructure Engineering*, 18(3):184–200, 2003.

[26] Partha Chakroborty and Tathagat Wivedi. Optimal route network design for transit systems using genetic algorithms. *Engineering optimization*, 34(1):83–100, 2002.

[27] Joanne Suk Chun Chew, Lai Soon Lee, and Hsin Vonn Seow. Genetic algorithm for biobjective urban transit routing problem. *Journal of Applied Mathematics*, 2013, 2013.

[28] Claude Chriqui and Pierre Robillard. Common bus lines. *Transportation science*, 9(2):115–121, 1975.

[29] Ernesto Cipriani, Stefano Gori, and Marco Petrelli. Transit network design: A procedure and an application to a large urban area. *Transportation Research Part C: Emerging Technologies*, 20(1):3–14, 2012.

[30] Citilabs. Cube Voyager - Modelling Functions, . URL `http://www.citilabs.com/products/cube/cube-voyager/cube-voyager-modeling-functions`.

[31] Citilabs. Cube Voyager, . URL `http://www.citilabs.com/products/cube/cube-voyager`.

[32] GU Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

[33] Carlos Coello Coello, Gary B Lamont, and David A Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media, 2007.

[34] Isabelle Constantin and Michael Florian. Optimizing frequencies in a transit network: a nonlinear bi-level programming approach. *International Transactions in Operational Research*, 2(2):149–164, 1995.

[35] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[36] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. Wiley-Blackwell, 1997.

[37] Ian M Cooper, Matthew P John, Rhydian Lewis, Christine L Mumford, and Andrew Olden. Optimising large scale public transport network design problems using mixed-mode parallel multi-objective evolutionary algorithms. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 2841–2848. IEEE, 2014.

[38] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem (darp): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(2):89–101, 2003.

[39] Georges A Croes. A method for solving traveling-salesman problems. *Operations research*, 6(6):791–812, 1958.

[40] Z.J. Czech and P. Czarnas. Parallel simulated annealing for the vehicle routing problem with time windows. In *Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*. Institute of Electrical & Electronics Engineers (IEEE), 2002.

[41] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[42] Joaquin De Cea and Enrique Fernández. Transit assignment for congested public transport systems: an equilibrium model. *Transportation science*, 27(2):133–147, 1993.

[43] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[44] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[45] Kalyanmoy Deb and Sachin Jain. Running performance metrics for Evolutionary multiobjective optmization. Technical Report 2002004, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology, Kanpur, India, 2002.

[46] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[47] Department for Transport. Action for Roads: A network for the 21st century. Technical report, 2013.

[48] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[49] Thomas A Domencich and Daniel McFadden. Urban travel demand-a behavioral analysis. Technical report, 1975.

[50] Marco Dorigo. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992.

[51] Dominique Douguet. e-lea3d: a computational-aided drug design web server. *Nucleic acids research*, page gkq322, 2010.

[52] European Commission. Driving time and rest periods. URL `http://ec.europa.eu/transport/modes/road/social_provisions/driving_time/index_en.htm`.

[53] Lang Fan. *Metaheuristic Methods for the Urban Transit Routing Problem*. PhD thesis, Cardiff University, January 2009.

[54] Lang Fan and Christine L Mumford. A metaheuristic approach to the urban transit routing problem. *Journal of Heuristics*, 16(3):353–372, 2010.

[55] Lang Fan, Christine L Mumford, and Dafydd Evans. A simple multi-objective optimization algorithm for the urban transit routing problem. In *2009 IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE, 2009.

[56] Wei Fan and Randy B Machemehl. Using a simulated annealing algorithm to solve the transit route network design problem. *Journal of transportation engineering*, 132(2):122–132, 2006.

[57] Wei Fan and Randy B Machemehl. Optimal transit route network design problem with variable transit demand: genetic algorithm approach. *Journal of transportation engineering*, 132(1):40–51, 2006.

[58] Wei Fan and Randy B Machemehl. A tabu search based heuristic method for the transit route network design problem. In *Computer-aided Systems in Public Transport*, pages 387–408. Springer, 2008.

[59] Merrill M Flood. The traveling-salesman problem. *Operations Research*, 4(1): 61–75, 1956.

[60] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5 (6):345, 1962.

[61] Transport for London. Guidelines for Planning Bus Services. 2012. URL `http://content.tfl.gov.uk/bus-service-planning-guidelines.pdf`.

[62] Michael L Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3): 596–615, 1987.

[63] Markus Friedrich, Ptv Ag, Thomas Haupt, and Klaus Nökel. Planning and analyzing transit networks-an integrated approach regarding requirements of passengers and operators. In *Journal of Public Transportation*. Citeseer, 1999.

[64] Gaetano Fusco, Stefano Gori, and Marco Petrelli. A heuristic transit network design algorithm for medium size towns. In *Proceedings of the 13th Mini-EURO Conference, Bari*, 2002.

[65] Ziyou Gao, Huijun Sun, and Lian Long Shan. A continuous equilibrium network design model and algorithm for transit systems. *Transportation Research Part B: Methodological*, 38(3):235–250, 2004.

[66] Abel Garcia-Najera and John A Bullinaria. Bi-objective optimization for the vehicle routing problem with time windows: Using route similarity to enhance performance. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 275–289. Springer, 2009.

[67] Benjamin Gardner and Charles Abraham. What drives car use? a grounded theory analysis of commuters' reasons for driving. *Transportation Research Part F: Traffic Psychology and Behaviour*, 10(3):187–200, 2007.

[68] Michael R Gary and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freeman and Company, New York, 1979.

[69] Michel Gendreau, Gilbert Laporte, and René Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation science*, 29(2):143–155, 1995.

[70] Billy E Gillett and Leland R Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations research*, 22(2):340–349, 1974.

[71] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[72] Fred Glover and Gary A. Kochenberger, editors. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2003.

[73] Fred Glover and Manuel Laguna. *Tabu Search*. Springer Science Business Media, 1997.

[74] David E Golberg. *Genetic algorithms in search, optimization, and machine learning*. 1989.

[75] JF Guan, Hai Yang, and SC Wirasinghe. Simultaneous optimization of transit line configuration and passenger line assignment. *Transportation Research Part B: Methodological*, 40(10):885–902, 2006.

[76] Anthony F Han and Nigel HM Wilson. The allocation of buses in heavily utilized networks with overlapping routes. *Transportation Research Part B: Methodological*, 16(3):221–232, 1982.

[77] Michael Pilegaard Hansen. Tabu search for multiobjective optimization: Mots. In *Proceedings of the 13th International Conference on Multiple Criteria Decision Making*, pages 574–586, 1997.

[78] Dick Hasselström. Public Transportation Planning, 1981.

[79] Zhengfeng Huang, Gang Ren, and Haixu Liu. Optimizing bus frequencies under uncertain demand: Case study of the transit network in a developing city. *Mathematical problems in Engineering*, 2013, 2013.

[80] INRO. Emme. URL `http://www.inrosoftware.com/en/products/emme/index.php`.

[81] Yechezkel Israeli and Avishai Ceder. Designing transit routes at the network level. In *Vehicle Navigation and Information Systems Conference, 1989. Conference Record*, pages 310–316. IEEE, 1989.

[82] Hong Jiang, Qingsong Yu, and Yong Huang. An improved ant colony algorithm for urban transit network optimization. In *2010 Sixth International Conference on Natural Computation*, volume 5, pages 2739–2743. IEEE, 2010.

[83] Yaochu Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing*, 9(1):3–12, 2005.

[84] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[85] Yaochu Jin and Bernhard Sendhoff. A systems approach to evolutionary multiobjective structural optimization and beyond. *IEEE Computational Intelligence Magazine*, 4(3):62–76, 2009.

[86] Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, pages 786–793. Morgan Kaufmann Publishers Inc., 2000.

[87] Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on evolutionary computation*, 6(5):481–494, 2002.

[88] Matthew P John, Christine L Mumford, and Rhyd Lewis. An improved multiobjective algorithm for the urban transit routing problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 49–60. Springer, 2014.

[89] David S Johnson, Jan Karel Lenstra, and AHG Kan. The complexity of the network design problem. *Networks*, 8(4):279–285, 1978.

[90] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[91] Gary J Katz and Joseph T Kider Jr. All-pairs shortest-paths for large graphs on the gpu. In *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, pages 47–55. Eurographics Association, 2008.

[92] Fatih Kılıç and Mustafa Gök. A demand based route generation algorithm for public transit network design. *Computers & Operations Research*, 51:21–29, 2014.

[93] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[94] Joshua Knowles and David Corne. On metrics for comparing nondominated sets. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, volume 1, pages 711–716. IEEE, 2002.

[95] A. L. Kok, C. M. Meyer, H. Kopfer, and J. M. J. Schutten. A dynamic programming heuristic for the vehicle routing problem with time windows and european community social legislation. *Transportation Science*, 44(4):442–454, 2010.

[96] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1): 48–48, 1956.

[97] W Lampkin and PD Saalmans. The design of routes, service frequencies, and schedules for a municipal bus undertaking: A case study. *Journal of the Operational Research Society*, 18(4):375–397, 1967.

[98] Gilbert Laporte and Yves Nobert. Exact algorithms for the vehicle routing problem. *North-Holland Mathematics Studies*, 132:147–184, 1987.

[99] Young-Jae Lee and Vukan R Vuchic. Transit network design with variable demand. *Journal of Transportation Engineering*, 131(1):1–10, 2005.

[100] Deming Lei and Xinping Yan. Urban transit route network design problem using tabu search algorithm. In *International Conference on Transportation Engineering 2007*. American Society of Civil Engineers (ASCE), 2007.

[101] Dudy Lim, Yew-Soon Ong, Yaochu Jin, and Bernhard Sendhoff. Trusted evolutionary algorithm. In *2006 IEEE International Conference on Evolutionary Computation*, pages 149–156. IEEE, 2006.

[102] Roger L Mackett and Marion Edwards. Guidelines for planning a new urban public transport system. In *Proceedings of the Institution of Civil Engineers. Transport*, volume 117, pages 193–201. Institution of Civil Engineers, 1996.

[103] Peter Mackie, James Laird, and Daniel Johnson. Buses and Economic Growth. Technical report, University of Leeds, Institute for Transport Studies, 2012.

[104] Thomas L Magnanti and Richard T Wong. Network design and transportation planning: Models and algorithms. *Transportation science*, 18(1):1–55, 1984.

[105] Christoph E Mandl. *Applied network optimization*. Academic Press, 1979.

[106] Christoph E Mandl. Evaluation and optimization of urban public transportation networks. *European Journal of Operational Research*, 5(6):396–404, 1980.

[107] Héctor Martínez, Antonio Mauttone, and María E Urquhart. Frequency optimization in public transportation systems: Formulation and metaheuristic approach. *European Journal of Operational Research*, 236(1):27–36, 2014.

[108] Antonio Mauttone and María E Urquhart. A multi-objective metaheuristic approach for the transit network design problem. *Public Transport*, 1(4):253–273, 2009.

[109] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[110] Elnaz Miandoabchi, Reza Zanjirani Farahani, Wout Dullaert, and WY Szeto. Hybrid evolutionary metaheuristics for concurrent multi-objective design of urban road and public transit networks. *Networks and Spatial Economics*, 12(3): 441–480, 2012.

[111] Mark Mistretta, Jay A Goodwill, Rob Gregg, and Chirstopher DeAnnuntis. Best practices in transit service planning. 2009.

[112] Alberto Moraglio and Ahmed Kattan. Geometric generalisation of surrogate model based optimisation to combinatorial spaces. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 142–154. Springer, 2011.

[113] Christine L Mumford. Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm. In *Genetic and Evolutionary Computation Conference*, pages 1389–1400. Springer, 2004.

[114] Christine L Mumford. New heuristic and evolutionary operators for the multi-objective urban transit routing problem. In *2013 IEEE Congress on Evolutionary Computation*, pages 939–946. IEEE, 2013.

[115] Muhammad Ali Nayeem, Md Khaledur Rahman, and M Sohel Rahman. Transit network design by genetic algorithm with elitism. *Transportation Research Part C: Emerging Technologies*, 46:30–45, 2014.

[116] NCHRP. *Bus Route and Schedule Planning Guidelines*. Transportation Research Board National Research, 1980.

[117] Somnuk Ngamchai and David J Lovell. Optimal time transfer in bus transit route network design using a genetic algorithm. *Journal of Transportation Engineering*, 129(5):510–521, 2003.

[118] G Nielsen, J D Nelson, C Mulley, G Tegner, G Lind, and T Lange. Public transport–planning the networks. HiTrans Best Practice Guide. 2005.

[119] Miloš Nikolić and Dušan Teodorović. Transit network design by bee colony optimization. *Expert Systems with Applications*, 40(15):5945–5955, 2013.

[120] Anthony O'Hagan. Bayesian analysis of computer code outputs: a tutorial. *Reliability Engineering & System Safety*, 91(10):1290–1300, 2006.

[121] Beatrice Ombuki, Brian J Ross, and Franklin Hanshar. Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24(1):17–30, 2006.

[122] Luis Paquete, Marco Chiarandini, and Thomas Stützle. *Pareto Local Optimum Sets in the Biobjective Traveling Salesman Problem: An Experimental Study*, pages 177–199. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-642-17144-4. doi: 10.1007/978-3-642-17144-4_7. URL http://dx.doi.org/10.1007/978-3-642-17144-4_7.

[123] Vilfredo Pareto and D Cours. Economic politique. *Lausanne, Switzerland, Rouge*, 1896.

[124] SB Pattnaik, S Mohan, and VM Tom. Urban bus transit route network design using genetic algorithm. *Journal of transportation engineering*, 124(4):368–375, 1998.

[125] PTV Group. How Can You Create Perfect Services All Along the Line?

[126] Alain Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In *International Conference on Parallel Problem Solving from Nature*, pages 87–96. Springer, 1998.

[127] Jacques Renaud, Fayez F Boctor, and Gilbert Laporte. A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on Computing*, 8(2):134–143, 1996.

[128] Mauricio GC Resende and Celso C Ribeiro. Greedy randomized adaptive search procedures. Technical Report TD-53RSJY, AT&T Labs, 2002.

[129] Franz Rothlauf. *Design of modern heuristics: principles and application*. Springer Science & Business Media, 2011.

[130] Siv Schéele. A supply model for public transit services. *Transportation Research Part B: Methodological*, 14(1):133–146, 1980.

[131] Jason R Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. Technical report, DTIC Document, 1995.

[132] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002.

[133] M C Shih and H S Mahmassani. A design methodology for bus transit networks with coordinated operations. Technical Report SWUTC/94/60016-1, 1994.

[134] Mao-Chang Shih, Hani Mahmassani, and M Baaj. Planning and design model for transit route networks with coordinated operations. *Transportation Research Record: Journal of the Transportation Research Board*, (1623):16–23, 1998.

[135] Heinz Spiess and Michael Florian. Optimal strategies: a new assignment model for transit networks. *Transportation Research Part B: Methodological*, 23(2): 83–102, 1989.

[136] Stephen Stradling, Michael Carreno, Tom Rye, and Allyson Noble. Passenger perceptions and the ideal urban bus journey experience. *Transport Policy*, 14(4): 283–292, 2007.

[137] WY Szeto and Yongzhong Wu. A simultaneous bus route design and frequency setting problem for tin shui wai, hong kong. *European Journal of Operational Research*, 209(2):141–155, 2011.

[138] Kay Chen Tan, YH Chew, and Loo Hay Lee. A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems. *European Journal of Operational Research*, 172(3):855–885, 2006.

[139] KC Tan, TH Lee, K Ou, and LH Lee. A messy genetic algorithm for the vehicle routing problem with time window constraints. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 679–686. IEEE, 2001.

[140] VM Tom and S Mohan. Transit route network design using frequency coded genetic algorithm. *Journal of Transportation Engineering*, 129(2):186–195, 2003.

[141] Paolo Toth and Daniele Vigo. An exact algorithm for the vehicle routing problem with backhauls. *Transportation science*, 31(4):372–385, 1997.

[142] Paolo Toth and Daniele Vigo, editors. *The Vehicle Routing Problem*. Society for Industrial & Applied Mathematics (SIAM), 2002.

[143] Paolo Toth and Daniele Vigo. The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing*, 15(4):333–346, 2003.

[144] Trapeze Group. About Trapeze Group, . URL `http://www.trapezegroup.co.uk/about`.

[145] Trapeze Group. Public Transport Scheduling and Planning, . URL `http://www.trapezegroup.co.uk/solutions/public-transport-scheduling-and-planning`.

[146] Christine L Valenzuela. A simple evolutionary algorithm for multi-objective optimization (seamo). In *Congress on Evolutionary Computation (CEC)*, pages 717–722, 2002.

[147] Cushman & Wakefield. UK cities monitor 2008. URL `http://investinmanchester.com/wp-content/themes/midas/docs/indicies-02.pdf?f43d0a`.

[148] Quentin K Wan and Hong K Lo. A mixed integer formulation for multiple-route transit network design. *Journal of Mathematical Modelling and Algorithms*, 2 (4):299–308, 2003.

[149] Chen Wang, Qingyun Duan, Wei Gong, Aizhong Ye, Zhenhua Di, and Chiyuan Miao. An evaluation of adaptive surrogate modeling based optimization with two benchmark problems. *Environmental Modelling & Software*, 60:167–179, 2014.

[150] Peter R White. *Public transport: its planning, management and operation*. Routledge, 2008.

[151] Peng Wu, Ada Che, Feng Chu, and Yunfei Fang. Exact and heuristic algorithms for rapid and station arrival-time guaranteed bus transportation via lane reservation. *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[152] Bin Yu, Zhongzhen Yang, Chuntian Cheng, and Chong Liu. Optimizing bus transit network with parallel ant colony algorithm. In *Proceedings of the Eastern Asia Society for Transportation Studies*, volume 5, pages 374–389, 2005.

[153] Bin Yu, Zhongzhen Yang, and Jinbao Yao. Genetic algorithm for bus frequency optimization. *Journal of Transportation Engineering*, 136(6):576–583, 2009.

[154] Fang Zhao and Albert Gan. Optimization of transit network to minimize transfers. Technical report, 2003.

[155] E Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*. PhD thesis, Swiss Federal Institute of Technology Zurich, 1999.

[156] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms – a comparative case study. In *International Conference on Parallel Problem Solving from Nature*, pages 292–301. Springer, 1998.

[157] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.