Cardiff University

Hyper-Heuristics for Two Complex Vehicle Routing Problems: The Urban Transit Routing Problem, and a Delivery and Installation Problem

Leena Ahmed, B.Sc. (Hons), M.Sc.

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

School of Computer Science and Informatics

December 2020

Declaration of Authorship

Statement 1
This Thesis is being submitted in partial fulfilment of the requirements for the degree of PhD.
Statement 2
I, Leena Ahmed, hereby declare that this thesis entitled, **“Hyper-Heuristics for Two Complex Vehicle Routing Problems: The Urban Transit Routing Problem, and a Delivery and Installation Problem"**, is all my own work, except as indicated in the text. The report has not been accepted for any degree and it is not being submitted currently in candidature for any degree or other reward.
Statement 3
I hereby give consent for my thesis, if accepted to be available online in the University's Open Access Repository.

Signed: ______________________________

Date: ______________________________

Abstract

Hyper-heuristics have emerged as general purpose search techniques that explore the space of low-level heuristics to improve a given solution under an iterative framework. They were introduced to raise the level of generality of search techniques representing self-configuring and automated reusable heuristic approaches for solving combinatorial problems. There are two classes of hyper-heuristics identified in the literatire: *generation* and *selection* hyper-heuristics. In this thesis, we focus on the class of selection hyper-heuristics and their efficient design and application on complex routing problems. We specifically focus on two routing problems: the Urban Transit Network design Problem (UTRP), and a rich vehicle routing problem for the delivery and installation of equipment which was the subject of the VeRoLog solver challenge 2019.

The urban transit routing problem (UTRP) aims to find efficient travelling routes for vehicles in public transportation systems. It is one of the most significant problems faced by transit planners and city authorities throughout the world. This problem belongs to the class of combinatorial problems whose optimal solution is hard to find with the complexity that arises from the large search space, and the multiple constraints imposed in constructing the solution. Furthermore, realistic benchmark data sets are lacking, making it difficult for researchers to compare their problem solving techniques with those of other researchers. We evaluate and compare the performance of a set of selection hyper-heuristics on the UTRP, with the goal of minimising the passengers' travel time and the operators' costs. Each selection hyper-heuristic is empirically tested on a set of known benchmark instances and statistically compared against all the other hyper-heuristics to determine the best approach. A sequence-based selection method utilising a hidden markov model achieved the best performance between the tested selection methods, and better solutions than the current known best solutions are achieved on benchmark instances. Then, we propose a hyper-heuristic algorithm specifically designed to solve the UTRP with defined terminal nodes that determine the start and end points of bus journeys. The algorithm is applied to a novel set of benchmark instances with real world size and characteristics representing the extended urban area of Nottingham city. We compare the hyper-heuristic performance on the data set with the NSGAII algorithm and real world bus routes, and prove that better solutions are found by hyper-heuristics. Due to the clear gap in research between the application of optimisation algorithms in public routes network optimisation and the real world planning processes, we implemented a hyper-heuristic algorithm that interactively work with interface procedures to optimise the public transport lines in Visum transportation modelling software. We adopt Selection Hyper-heuristics for two optimisation problems and the optimisation objectives include the passengers' average travel time and operators' costs. The results demonstrate the successful implementation of the applied optimisation methods for multi-modal public transport networks. Finally we introduce a population based hyper-heuristic algorithm and apply it on a complex vehicle routing problem consisting of two stages: a Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) for the delivery of equipment, and the Service Technician Routing and Scheduling Problem (STRSP) for the installation of the delivered equipment. This problem was the subject of the VeRoLog solver challenge 2019. We apply the hyper-heuristic population-based algorithm on a small and large size data sets, and show that our approach performed better in terms of results and run time on small instances compared to the results of a mathematical model implemented for this problem. We perform analysis of the new proposed algorithm and show that it finds better quality solutions compared to its constituent selection hyper-heuristics when applied individually. Finally we conclude the thesis with a summary of the work and future plans.

Acknowledgements

I would like first to pay my gratitude to God almighty who gave me the strength physically and mentally to complete my research tasks and to prepare this thesis.

I would like to thank the Computer Science and Informatics department in Cardiff University for offering me this great opportunity and funding my PhD studies, which allowed me to persue the research field I adore the most. Without this opportunity, this thesis would not have seen the light, I will be forever in debt for this incredible support and opportunity.

I would like to express my sincere gratitude to my supervisor Dr.Christine Mumford for her guidance, support, patience, understanding, and encouragement during the past four years of my PhD. Her hard work with me throughout paper writing and conducting experiments helped me a lot to overcome many challenges and shape my experience especially in writing scientific papers. I owe her the success of this dissertation and genuinely thank her for being the great person that she is.

I thank our colleagues from Nottingham University, Dr.Yong Mao, and Philipp Heyken who I worked with closely and has been a great contributor in many of the projects proposed in this thesis. I also thank the other authors from Lancaster University, and Prof.Gromicio who gave incredible support for our VeRoLog project.

I would also like to thank my husband Dr.Ahmed Kheiri, first for playing a major role in completing the projects on this thesis with his advice, guidance and hard work, and second for being a great supporter to complete this thesis by encouraging me, pushing me to achieve all this, and believing in me.

I also thank my family for their continuous support during my studies, and my two kids who made this journey way more difficult but joyful at the same time.

Contents

List of Figures

List of Tables

Abbreviations

ACO	Ant Colony Optimisation
ALNS	Adaptive Large Neighbourhood Search
BCO	Bee Colony Optimisation
CO	Combinatorial Optimisation
COPs	Combinatorial Optimisation Problems
CVRP	Capacitated Vehicle Routing Problem
CVRPTW	Capacitated Vehicle Routing Problem with Time-windows
DARP	Dial-a-Ride-Problem
EA	Evolutionary Algorithm
GA	Genetic Algorithm
GD	Great Deluge
GR	Greedy
HC	Hill Climber
HMM	Hidden Markov Model
IE	Improve or Equal
ILP	Integer Linear Programming
LA	Late Acceptance
MCVRP	Multi Compartment Vehicle Routing Problem
MILP	Mixed Integer Linear Programming
MOOP	Multi Objective Optimisation Problem
MPVRP	Multi Period Vehicle Routing Problem
NSGAII	Nondominated Sorting Genetic Algorithm II
OI	Only Improve

POHH	Population-based Hyper-heuristic
PSO	Particle Swarm Optimisation
PuT	Public Transport
RD	Random Descent
RP	Random Permutation
RPD	Random Permutation Descent
RR	Record-to-Record
SA	Simulated Annealing
SDVRP	Split Delivery Vehicle Routing Problem
SR	Simple Random
SSHH	Sequence based Selection Hyper-heuristic
SS-GD	Sequence-based Selection combined with Great Deluge
STRSP	Service Technician Routing and Scheduling Problem
TSP	Travelling salesman Problem
TS	Tabu Search
UTNDP	Urban Transit Network Design Problem
UTRP	Urban Transit Routing Problem
VeRoLog	The Euro working group on Vehicle Routing and Logistics optimisation
VRP	Vehicle Routing Problem

Chapter 1

Introduction

Vehicle Routing Problems (VRPs) are one of the most important classes of NP-hard combinatorial optimisation problems that have been subject to research for more than fifty years [1]. Due the complex nature of the VRPs and their real-world practicality, various versions have been implemented with different structures and operational constraints, challenging researchers to develop a rich suite of methodologies and efficient solution methods for solving such a complex problem. There is a growing interest nowdays in the application of optimisation techniques in real-world routing problems due to the increasing demand by industrial and commercial partners who are competing to deliver efficient services for their customers while effectively reducing their expenditures. As a result, several well-known companies have been approaching academic institutions and organising challenges in cooperation with research groups to encourage researchers into developing efficient solution methods and to compete in delivering high quality results for real-world problems encountered in their businesses. The research field also benefits from such connections, which can result in the availability of real-world benchmarks, and enrich the research with novel versions of routing problems from real life applications.

One important application of VRPs in the current urban societies is the development of efficient public transit services. The ever-increasing use of private transportation throughout the cities of the world is resulting in unacceptable levels of congestion, pollution, and environmental, social and economic cost. This has led to move towards improving public transportation services and encouraging citizens to use them more. To fulfil the current needs of modern cities in delivering efficient, economical, and environmentally friendly transportation systems, careful planning is required in the design phase to avoid excessive waiting and travelling times and reducing the operational costs. The design of public transit systems is a complicated task that needs to satisfy the requirements of many stakeholders with conflicting needs, including passengers and transportation companies. One key stage is the design of routes over a given network to provide an efficient service for passengers and network operators. This problem is referred to as the Urban Transit Routing Problem (UTRP). The UTRP is considered an enormous challenge for optimisation algorithms, because of the huge complexity imposed by the multiple constraints which define the criteria for accepting feasible solutions, and the many conflicting objectives that the designed network should satisfy. This makes finding near optimal solutions extremely difficult.

Years of research in the optimisation of combinatorial problems led to the development of a variety of methods which participated in finding increasingly competitive results in many intractable computational problems such as VRPs. However, most of these methods have been finely tuned to work well on one problem or an instance of a problem. This has lead to the lack of general problem solving methodologies and the creation of a range of methods that work well on specific problem structures, while not being able to perform as well in other problem versions without significant human input. This issue has urged researchers to develop methods that are more general and can adapt to changes in the problem domain. Recently, research has focused on a class of algorithms known as hyper-heuristics [2] that have the potential to adapt to changes in the problem domain, making their application to different problems and instances easier than other search methodologies. Hyper-heuristics, which are defined as heuristics to choose heuristics, is separated from any specific domain knowledge by what is called "domain barrier", and therefore it can focus on providing sufficiently good solutions without the need for lengthy run-times or significant input. Since the development of the hyper-heuristic framework, it has been utilised in solving several combinatorial problems, and routing problems is a domain in which hyper-heuristics has had an outstanding record of success.

In this thesis, we apply a selection hyper-heuristic framework to two routing applications, one is the aforementioned UTRP, and the other problem is a rich VRP problem comprising of two routing stages for delivery and installation. Despite the differences between the two problems in their description and operational constraints, it was worth investigating the effectiveness of the hyper-heuristic framework and its generality on different versions of real world routing problems, and showing that it can adapt to different domains while delivering high quality performance.

We explore the capabilities of hyper-heuristics in overcoming challenges that other meta-heuristic algorithms have struggled with, and show the impact of online learning in improving the performance of hyper-heuristics. We test different combinations of selection hyper-heuristics and perform extensive experiments on data sets with different sizes and characteristics, and show the scalability of our methods and its adaptability when more complexity is added to the problem domain. We provide a solid contribution to the Urban Transit Routing Problem (UTRP) by developing a novel hyper-heuristic algorithms for solving three different versions of the problem, and in one of them we integrate our algorithms with a professional transportation modelling software. We further explore our selection hyper-heuristic framework as a population-based approach in a routing problem of two interconnected stages and prove that our algorithm is able to find optimum solutions in a short time.

1.1 Research Motivation and Contributions

Meta-heuristics have enjoyed some success on versions of the UTRP, with genetic algorithms (GAs) as a particularly popular choice [3–6]. However, one of the main shortcomings of applying population-based algorithms to solve the UTRP are the significantly long run times when solving large instances, which can extend to days rather than hours as has been reported in [7]. Running such algorithms on large instances may often require the use of a high performance cluster, and yet the execution time remains unreasonably long especially when the number of generations increases. This has led to limiting the implementation of population-based algorithms to relatively small instances. Cooper et al. [8] used parallelism to solve the run time problems of the UTRP, but this cannot be a definitive solution as it requires the use of a cluster of high performance computers.

In this thesis we propose hyper-heuristics as a possible way forward. Hyper-heuristics have a clear advantage in terms of run time over population-based methods such as GAs because their focus is on a single point in the search space, rather than a population of points. Furthermore, hyper-heuristics have built-in mechanisms that carry out the tuning and parameter setting without the need for human intervention, and use only simple low level heuristics that are fast and easy to design. Although hyper-heuristics are designed as problem independent methods, many researchers have shown that the choice of selection hyper-heuristics components highly influence their performance [9, 10]. Thus, we focus on examining and comparing the performance of several selection hyper-heuristics, combining different known selection and move acceptance methods on the route design problem (UTRP) with the goal of minimising the average passengers travel time, and the costs to the operators. Moreover, there is a lack in the UTRP research for simplified models that are also applicable to real world size instances. Most of the currently available methods applied to real world size instances of cities and towns are specifically designed to work on those instances, and they are not available publicly in the majority of these studies. This has motivated us to contribute with our methodologies to find state-of-the-art results to a new published set of instances with real world size and characteristics, and also to prove that hyper-heuristics continue to perform well in terms of run time and solution quality compared to multi-objective evolutionary frameworks such as NSGAII.

We have also observed the gap in the research between the purely academic studies in the automatic public transport route optimisation and real world planning processes. Therefore, there is an urge to develop algorithms that can bridge the gap between the theoretical research of the UTRP and the real world transportation planning. A hyper-heuristic is a good candidate for such application, being a single-point based framework, and therefore able to facilitate the interaction with a transport modelling software package. Finally, we wanted to explore the generality of hyper-heuristics to solve different VRP versions that is as complex as the UTRP, with a real world impact. The VeRoLog solver challenge 2019 problem was an ideal platform to showcase our results and to test the hyper-heuristic capabilities in terms of run times and solution quality against the performance of exact mathematical methods, and the results of the challenge finalists.

The key research questions we are addressing in this thesis are:

1. How can a selection hyper-heuristic being a single-point based framework succeed in overcoming the run time issues in population-based methods while delivering high quality solutions in small as well as large size instances ? (chapter 4 publication 1)
2. How can we extend our implementation of the hyper-heuristic framework to be applied on more complex versions of the UTRP and on instances with real-world size and characteristics ?(chapter 5, publication 2)
3. How can we bridge the gap between academic versions of the UTRP and real-world transportation systems planning by integrating the algorithms used to solve the UTRP theoretically with a commercial software package used by transportation systems planners ? (chapter 5, publication 3)
4. How can we generalise the application of hyper-heuristics in different domains of complex routing problems and prove its effectiveness and computational efficiency? (chapter 6, publication 4)

The main contributions of this thesis are:

- A novel implementation of a selection hyper-heuristic algorithm for solving the UTRP by implementing and testing several components of selection and move acceptance methods.
- Testing an online learning selection method based on the Hidden Markov Model (HMM) and show that it is more effective compared to other non-learning random selection methods.
- Comparison with the state-of-the-art methods from the literature and finding new best results for Mandl instance with 6, 7, and 8 routes and the instances of Mumford data set.
- Using the weighted sum approach to mitigate the effect of maintaining a population of solutions which causes serious run time limitations while solving the UTRP.
- A hyper-heuristic algorithm for solving the UTRP on real world scale instances with fixed terminal nodes. A set of specialised operators are implemented to handle the presence of fixed terminals.
- A comparison with the NSGAII evolutionary multi-objective framework approximate Pareto front, and real-world bus routes to show the excellence of our results.
- The integration between hyper-heuristics as an optimisation method and the evaluation tools offered by the commercial transport planning software Visum, thus allowing to solve versions of the UTRP that are more applicable to transport planners in real world.
- A novel population-based selection hyper-heuristic for solving the VeRoLog solver challenge 2019.
- The comparison of the population-based selection hyper-heuristic framework with a mathematical model results and the results of the competition finalists on large scale instances.

1.2 Structure of Thesis

The thesis is structured as follows:

- Chapter 1: Introduces the thesis topic and describes the motivation of this study.
- Chapter 2: Introduces the main concepts applied throughout the thesis, and defines the VRP variants solved.
- Chapter 3: Summarises the methods applied previously in the literature to solve Vehicle Routing Problems (VRPs). The chapter also presents a comprehensive survey of the literature in the UTRP, and some well-known VRP variants related to the problems tackled in the thesis. An introduction to the hyper-heuristic framework and the online algorithm applied in the thesis is described.
- Chapter 4: This chapter describes the applied hyper-heuristic methodology for solving the UTRP based on our work in [11]. We present our results with an extensive analysis, and compare these results with the state-of-the-art methods from the literature.
- Chapter 5: Explains the application of the developed hyper-heuristic algorithms in two problems: a version of the UTRP with defined terminal nodes on a realistic size instances , and the optimisation of the public transport lines in Visum transport modelling software.
- Chapter 6: Describes our methodology and the developed population based hyper-heuristic framework for solving a VRP version for the delivery and installation of equipment which was the subject of the VeRoLog solver challenge 2019.
- Chapter 7: Summarises the research findings of this thesis and the future work.

1.3 Academic Publications Produced

The following academic articles, conference papers and abstracts have been produced as a result of this research, and on which the thesis chapters are based on:

1. Leena Ahmed, Christine Mumford and Ahmed Kheiri (2019) Solving urban transit route design problem using selection hyper-heuristics. European Journal of Operational Research, 274(2):545-559. [journal] (Chapter 4).
2. Leena Ahmed, Heyken Soares, Christine Mumford, Yong Mao. 2019. Optimising bus routes with fixed terminal nodes. Presented at: The Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 13-17 July 2019GECCO '19

Proceedings of the Genetic and Evolutionary Computation Conference (Chapter 5).

3. Heyken Soares, Leena Ahmed, Yong mao, Chrisitne Mumford (in press) Public Transport Network Optimisation in PTV Visum using Selection Hyper-Heuristics.Public Transport (Chapter 5).

4. Ahmed Kheiri, Leena Ahmed, Burak Boyaci, Joaquim Gromicho, Christine Mumford, Ender Ozcan and Ali Selim Dirikoc (2020) Exact and hyper-heuristic solutions for the distribution-installation problem from the VeRoLog 2019 challenge. Networks, 76(2):294-319 (Chapter 6).

There are also a number of abstracts produced out of this thesis that were presented in well known conferences including: IMA and OR Society Conference 2017, OR Society Annual Conference 2018 (OR60), New to OR (2019) OR Society Biennial Conference, 28th PTV Traffic User Seminar(2018).

Chapter 2

Vehicle Routing Problems and the Variants Solved

2.1 The Concept of Optimisation

In mathematics and computer science, optimisation refers to the selection of the best element from a set of available alternatives using a mathematical function, or a criterion on which to base the selection decision. In the simplest form, optimisation can refer to the minimisation or maximisation of a function named the "objective function", by choosing an input or a set of inputs, and calculating the value of the objective function. The optimisation is usually subject to a set of constraints defined according to the problem domain, and the optimised problem is either a maximisation or a minimisation problem, where in the former the calculated objective value is maximised, and minimised in the latter.

An optimisation problem can be formulated in the following way: $f : A \to \mathbb{R}$, a function from a set A to the real numbers $\mathbb{R}$, the goal is to find an element $x_o \in A$, such that $f(x_o) \leq f(x) \quad \forall x \in A$ (minimisation), or $f(x_o) \geq f(x) \quad \forall x \in A$ (maximisation). The set A is known as the "solution space", or the space of candidate solutions, and f is the objective function that calculates the value of the candidates in A. The solution x_o is the best global optimum and which the optimisation procedure seeks to find.

A variety of optimisation algorithms have been developed over the years motivated by the practical importance of optimisation in taking critical decisions in many large scale applications, and its contribution to cost saving and service improvements. However, due to the enormous search spaces involved in many real-world problems, manual application of these algorithms is not realistic. For this reason, the automation of the optimisation process is the only valid option. In the next sections and chapters we will further explore the time and computational complexity associated with some optimisation problems, and describe the algorithms suitable for solving them.

2.1.1 Optimisation of Combinatorial Problems

As mentioned above, an optimisation algorithm aims to find the best configuration from a set of variables defined on the solution domain to achieve defined goals. There are two important paradigms of optimisation that are used to categorise optimisation problems: discrete optimisation and continuous optimisation. In the discrete optimisation, some or all of the decision variables belong to a discrete set of values, in contrast to the continuous optimisation, in which the variables are allowed to take a value from within a range of values. Within the discrete optimisation problems, there is a category of problems known as the *Combinatorial Optimisation problems* (COPs).

Combinatorial optimisation is a special case of of discrete optimisation, where the search for an optimal solution is conducted on a finite set of solutions which can be represented by a structure, such as a graph or a permutation. According to Papadimitriou and Steiglitz [12], in COPs we are looking for an object from a finite set or possibly countable infinite set, and this object can be a subset, a permutation, or a graph structure. As in the general optimisation problems, the goal in a COP is to find a set of globally optimal solutions as defined by an objective function. However, in cases of COPs when the space of finite solutions is very large and increases exponentially with the increase in the problem size, exhaustive search methods become intractable and impractical to apply. Typical example of problems involving combinatorial optimisation are: the Travelling Salesman Problem (TSP), the Bin Packing Problem, Boolean Satisfiability (SAT), Quadratic Assignment (QAT), and scheduling and timetabling problems.

Beside the theoretical relevance of COPs, they are also practically important due to their applicability in many real-world scenarios. Such domains in which we can see COPs include: routing, scheduling, decision making, production planning, energy, transportation and telecommunication. Many COPs can be represented as graphs. In this class of problems, the solution domain is represented by a graph structure, and the goal is to find an optimal solution in the form of a sub-graph containing a subset of the graph edges and nodes. Typically, route design optimisation problems are classified as graph-based COPs.

In principal, if the feasible solution space is finite, any COP can be solved exactly by an algorithm that can identify all the feasible solutions and find the best of them according to the objective function evaluation. However, the feasible solution space grows exponentially with the size of the instance to be solved, and therefore such a simple approach is not applicable for practical problems. According to Blum and Roli [13], solution methods for COPs can be broadly classified as complete, or approximate algorithms. In the complete (exact) solution methods, an optimal solution is guaranteed to finite size instances in bounded time, while for NP-hard COPs that cannot be solved in polynomial time (section 2.1.2), these methods will require an exponential time in the worst case to find an optimal solution which is a significantly high computation time for practical purposes. For this reason, approximate methods have been more popular and received increasing attention. Amongst these methods are heuristics (constructive and local search methods), and meta-heuristics methods (discussed further in chapter 3).

2.1.2 $\mathcal{NP}$-Hard and $\mathcal{NP}$-Complete Problems

The foundations of the computational complexity theory were put down by Cook [14] and Karp [15], who introduced a framework for measuring the computational complexity of a problem. The computational complexity theory provides a basis for calculating the complexity of a problem based on how the required time for solving the problem increases as the problem size gets larger. In other words, the time complexity of a problem is expressed in terms of a complexity function that calculates the time requirements for each possible input length. This function is referred to as the big $\mathcal{O}$ notation. For a given input of length n, the notation $\mathcal{O}()$ provides a function proportional to the maximum number of operations that should be performed, given that input.

Algorithms with time complexity $\mathcal{O}(n^k)$ for some constant k, are called polynomial time algorithms. These algorithms are described as *tractable*, as it is quite feasible to run algorithms of this complexity with large inputs using the kinds of computers we have today. In contrast, exponential time algorithms of time complexity $\mathcal{O}(k^n)$ are intractable and grow much faster than any polynomial function. An intractable problem is a problem that cannot be solved by any polynomial time algorithm, such as the above example of exponential time algorithms, and the factorial run time algorithms $\mathcal{O}(n!)$ that grow even faster.

An important class of computational problems are the non-deterministically polynomial problems ($\mathcal{NP}$), which are described as the decision problems that are whether or not it is tractable to find their solution, the verification of this solution is polynomial or tractable. The class of polynomial problems which can be solved by means of a polynomial-time algorithm, is called P. Trivially, P is a subset of NP ($P \subseteq NP$).

If M and M' are NP problems such that M' is significantly harder than M (i.e., any reduction, or translation of M' to M takes more than polynomial time), then M cannot be amongst the hardest of NP problems. For M to be one of the hardest NP problems, it is necessary that any NP problem M' is reducible to it in polynomial time. This is the motivation for the following definitions: A decision problem M is NP-hard if any NP problem can be reduced to M in polynomial time (so no NP problem is more than polynomially harder than M), and M is not necessarily an NP problem (not a decision problem that has a "yes" or "no" answer). A problem is NP-complete if it is both NP and NP-hard, i.e., it is an NP problem for which no NP problem is more than polynomially harder. Therefore, the NP-complete problems are the hardest amongst the NP problems. Figure 2.1 illustrates the relation between P, NP-Hard, and NP-Complete problems.

We should note that the existence of $\mathcal{NP}$-Hard, and $\mathcal{NP}$-Complete problems are based on the assumption that $P \neq NP$. Whether or not this assumption is actually true remains one of the most famous unsolved problems in computer science. Currently, no one has yet discovered an algorithm to solve an NP-complete problem in a polynomial time, and it is also unproved that such algorithm does not exist. If a polynomial time algorithm that solves one of the NP-complete problems is found, this will consequently mean that the entire class NP is contained in P, so we would have P=NP. If this has ever became true, it will have a profound scientific consequences, because it will lead to the availability of polynomial algorithms to solve a large class of important practical problems which are thought to be intractable and to which a great effort has been spent to develop algorithms that solve it approximately.

Almost all vehicle routing and scheduling problems are NP-hard and cannot be solved in polynomial time. Exact approaches are only successful in solving small versions of such problems and are not feasible for larger instances in terms of the computational time requirements.

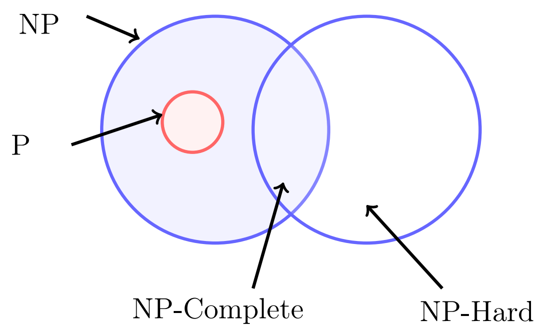


Figure 2.1: Relationship between P, NP, NP-Hard, and NP-Complete problems

2.1.3 Single and Multi-Objective Optimisation

A single objective optimisation problem can be defined as: the minimisation or maximisation of a single function $f(x)$ subject to a set of constraints, where $x \in \Omega$. The function $f(x)$ is named the objective function, and $x = \{x_1, x_2 \ldots x_n\}$ is an n-dimensional decision variable vector from some universe Ω. Ω is a domain that contains all the possible x that satisfies the evaluation of $f(x)$ and all its constraints. The vector x and the scalar function $f(x)$ can be discrete or continuous. In the single objective optimisation, the optimisation focuses solely on minimising or maximising $f(x)$ (i.e., single decision optimisation).

Another category of optimisation problems named *"multi-objective optimisation problems"* (MOOPs) involve multiple objective functions that are to be minimised or maximised simultaneously, and similar to the single-objective problems, the multi-objective problems contain a set of constraints that must all be satisfied by a feasible solution. In a MOOP, the concept of a globally optimal solution does not apply, but rather the goal is to find a set of solutions that provide a good balance between several contradicting objectives. Most of the real world problems are multi-objective in nature and involve several stakeholders and decision making criteria. Examples are Vehicle Routing Problems (VRPs), where the efficiency of the service delivered to the customers must be balanced with the total encountered costs.

MOOPs can be formally stated as follows: $F(x) = \{f_1(), f_2(x) \ldots f_k(x)\}$, subject to: $g_i(x) \leq 0; \quad i = 1 \ldots, n$, s.t: $x \in U$, where x is a solution, $k \geq 2$ is the number of objective functions, n is the number of constraints, and U is a feasible set. The solution x is given as a decision vector $x = \{x_1, x_2, \ldots x_m\}$, where m is the number of decision variables. A single solution in MOOP cannot improve all objective functions simultaneously, but rather Pareto optimality is used to describe the set of solutions that

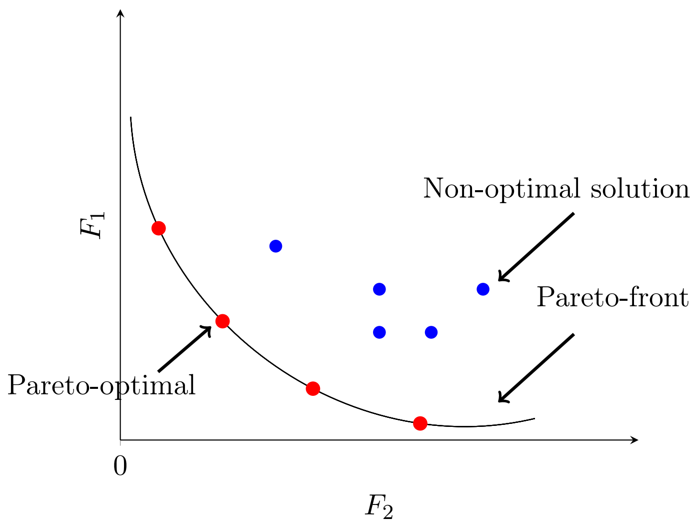


FIGURE 2.2: Pareto-front and Pareto optimal solutions

provide a trade-off between the multiple objectives. A solution is said to be Pareto optimal, if it is not possible to move from this solution to a solution that is better for one objective without worsening the other objectives. The set of all Pareto optimal solutions is known as the *Pareto front* or the non-dominated front (figure 2.2), and these solutions are not dominated by any other solution in the search space.

Multi-objective optimisation techniques can be classified based on how to combine the decision making and search into the following approaches:

- Priori approach: in this method, the weights and the preferences of the objectives are set prior to the search process by the decision maker. An example of this is the weighted sum approach, where the weight of each objective is set before the start of the search.
- Posteriori approach: the search is conducted to find a set of solutions, and a decision process is then applied to select the most appropriate solutions (trade-off solutions). Examples of this approach are evolutionary algorithms.
- Interactive (progressive approach): in this class, the decision maker can adjust the preferences while the search is ongoing, or alternatively it can be done automatically by the algorithm (e.g. SAWing).

Several methodologies have been developed for solving MOOPs, including the weighted-sum approach, the ϵ-constraint method, and Evolutionary Algorithms (EAs). Further detail on some of these methods will be discussed in the following chapter.

2.2 Graph Structure

As mentioned in section 2.1.1, an important class of CO problems such as route design optimisation problems are graph-based, where the solution forms part of a sub-component of the graph, and all the operations are preformed on a graph structure. In this section we represent the fundamentals of the graph theory, which is essential to the understanding of the problems definitions proposed in this thesis.

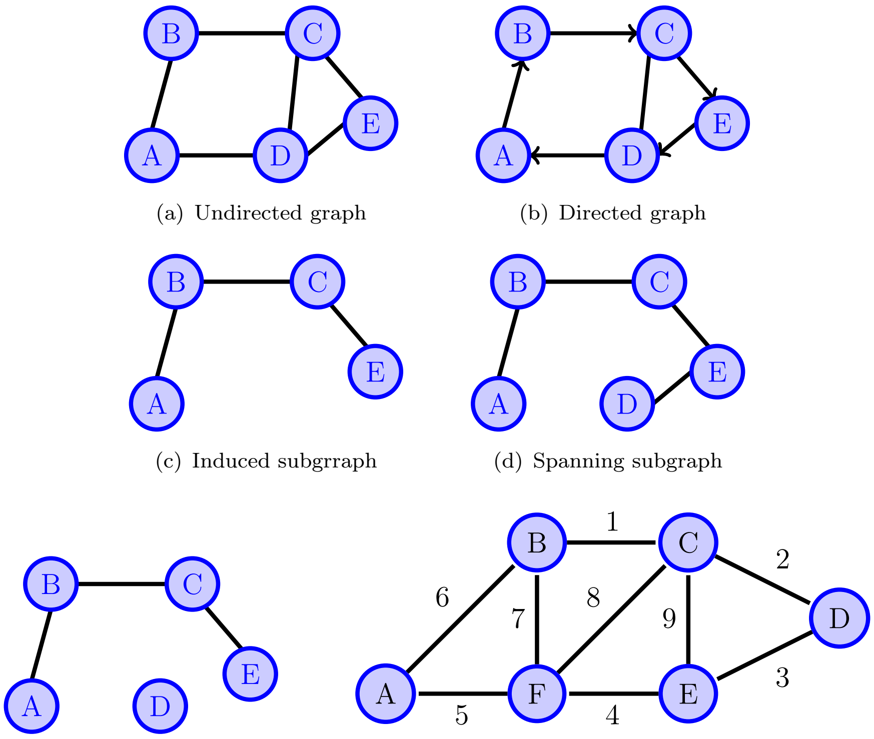
Formally, a graph $G = (V, E)$ is defined as a set of vertices $V = \{v_1, v_2, ...v_n\}$, and a set of edges $E = \{e_1, e_2, ...e_m\}$, where each edge $e = \{u, v\}$ is associated with a set of two unordered vertices belonging to V. In this example, the edge e, is said to be connecting the two vertices u, v. An edge is defined as *incident* to a vertex, if this vertex is contained within the set that defines this edge.

A *simple graph* is defined as a graph that is undirected, has at most one edge between any two pair of vertices, and has no self loops. A loop exists in a graph when there is at least one edge in the edge set that does not connect two distinct pair of vertices (i.e., it connects an edge to itself, $e = (u, u)$). An edge that connects an edge to itself is named a “loop”, and if it connects two distinct pair of vertices, it is named a “link”.

A multi-edge exists when the edge set contains more than one edge with the same incident vertices, and a graph is named a *multi-graph*, if it contains a loop, or several edges between the same pair of vertices. A graph is also defined as a *directed graph* when an edge connects a pair of two ordered vertices, such that: $\{u, v\} \neq \{v, u\}$. An example of a directed and an undirected graph is shown in figure 2.3.

A graph G' is defined as the subgraph of the graph G, if it contains a subset of its vertices $V' \subseteq V$, and a subset of its edges $E' \subseteq E$. Of a particular interest is the *induced subgraph*, in which a specific subset of the vertices is selected, along with the set of edges that connected the pairs of vertices in this subset in the original graph. It can be defined as the graph G'_s that has the subset of vertices S, and has all the edges in E with endpoints that belong to S. Figure 2.3(c), is an induced subgraph from the undirected graph (a) that includes the vertices A, B, C, and E. Another important class of subgraphs is the *spanning subgraph* which can be defined as $G' = \{V, E'\}$, in other words, it is the subgraph that contains all the existing vertices of the original graph and a subset of its edges.

A walk in a graph is defined as a sequence of edges, that are connected by a sequence of vertices. We can refer to it by $W = v_1, e_1, v_2, e2, \ldots e_n, v_n$, such that each edge e_i



(a) Undirected graph (b) Directed graph (c) Induced subgrraph (d) Spanning subgraph (e) Unconnected graph (f) Undirected weighted graph

FIGURE 2.3: Demonstration of different types og graphs

has the the vertices v_{i-1}, v_i as its endpoints. A trail is defined as a walk with distinct edges, and the path is a trail with distinct vertices. A connected graph is said to have a walk between every possible pair of vertices in the vertices set, otherwise the graph is unconnected leaving some of its vertices isolated (i.e., has no adjacent vertices).

A weighted graph, is the graph where each edge is associated with a weight value (figure 2.3 (f)). This graph is particularly useful in applications where the shortest path or the distance between pairs of vertices should be calculated.

Finally, we define the concept of adjacency between the graph vertices. Two vertices are adjacent to each other if and only if there exists an edge connecting these two vertices. In other words, adjacent vertices are incident on an edge. A single vertex can be adjacent to several vertices, and the degree of a vertex is defined as the number of its adjacent vertices. An isolated vertex has a degree zero, and the presence of a such vertex in the graph means that this graph is unconnected (i.e., figure 2.3 (e), vertex D).

A graph can be simply represented by a structure called an *adjacency matrix*. For a graph G, where $|V| = n$, the adjacency matrix A_{nxn} is a matrix with entries of zero and one values, where each entry $A_{i,j}$ equals one if the two vertices v_i, v_j are adjacent, and zero otherwise. This matrix is symmetrical in the case of simple undirected graphs (i.e., directed graphs are generally not symmetric).

2.3 The Vehicle Routing Problem

The well-know class of combinatorial optimisation problems, known as *vehicle routing problems*, emerged in the fields of transportation, and distribution management motivated by the needs of these industries to save expenditures, where major savings can occur by optimising some measures of the transportation system. In fact, a large portion of the logistics costs are related to distribution, leading to more attention to apply optimisation techniques for costs saving.

Routing and scheduling problems encountered in industry or travel have a high degree of complexity, involving multiple variables and constraints. Modelling these problems require some types of simplifications and adjustments to make them tractable by optimisation algorithms. Nevertheless, even with these simplifications, obtaining optimal solutions is still a challenge.

The VRP is one of the most well-known and extensively studied problems in combinatorial optimisation, and the foundation of this research dates back to 1959 in the study by Dantzig and Ramser [16] named "A Truck Dispatching Problem". The study tackled a real-life application of distributing gasoline among a number of service stations, with the goal of finding optimal travel routes with the minimum travelling distance. Since the introduction of this problem sixty years ago, it has attracted a great number of researchers given its complexity and challenging nature, and most importantly its practical applicability in the real-world. Due to this, many other VRP variants were proposed to model real life situations.

The main goal in VRPs is to determine a set of routes for a fleet of *vehicles* located at one or multiple *depots* to serve the demand at a number of geographically dispersed locations known as *customers*. These vehicles are operated by a crew of drivers and travel through an appropriate *road network*. In the most basic version of a VRP (figure 2.4), a homogeneous fleet of vehicles serves customers' demand by visiting each customer exactly once, and the journey of each vehicle starts and terminates at the depot. The objective is to find a set of routes, each performed by a single vehicle, such that all customers' requests

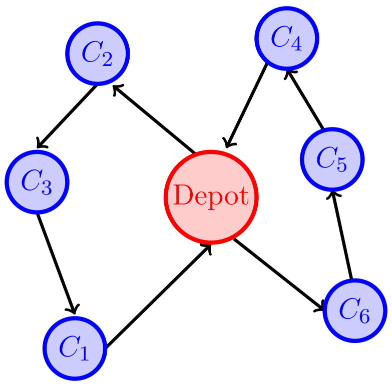


FIGURE 2.4: Solution to the basic VRP

are delivered, the operational constraints are satisfied, and the overall transportation costs (i.e., travel time, distance) are minimised.

Further, there are many other constraints that can be added to to the basic version, depending on the nature of the delivery/distribution problem. One of the most commonly applied constraints is a capacity constraint on the size of the vehicles. This version of the VRP is named the *Capacitated VRP*, in which each customer demand must not exceed the vehicle capacity. The CVRP is one of the most well studied versions of the VRP and forms the basis of other variants (see section 2.3.2 for more on VRP variants). The CVRP can be considered the simplest VRP version, in which it is assumed that the vehicles are identical and belong to the same depot. Using the graph notations, the CVRP can be described as: a set of customers $C = \{c_1, c_2, \dots c_n\}$ scattered on different geographical locations $(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$, and a depot node $\{0\}$ located at position (x_0, y_0). A graph $G = (V, A)$ exists, where the graph vertices $V = \{0\} \bigcup C$. There are a number of vehicles M located in the depot to serve customers' requests, and each has a limited capacity Q, where customer demand d_i cannot be divided, and hence cannot exceed the vehicle capacity $(d_i \leq Q \quad \forall c_i \in C)$.

2.3.1 The Travelling Salesman Problem

VRPs are $\mathcal{NP}$-hard combinatorial problems originating from the classical Travelling Salesman Problem (TSP). The Travelling Salesman Problem (TSP) is a well-known $\mathcal{NP}$-hard problem that has been studied by many researchers due to its various applications in real-world problems. Some of these applications include: computer wiring, dashboard design, job sequencing, vehicle routing, and warehouse automation systems. It was firstly defined by the two mathematicians William R Hamilton and Thomas Kirkman in the

19th century. The basic definition of the problem involves a salesman who wishes to travel between a number of cities returning home at the end, and the goal is to find the sequence in which he can visit all the cities while minimising the total travelled distance. Although the problem definition appears to be simple, it is until now considered one of the most challenging problems in the field of operational research [17]. We will briefly describe the mathematical definition of the TSP, as it is considered the basis for other important routing applications.

Following the graph theory described in section 2.2, the TSP involves a graph $G = \{V, E\}$ containing a set of vertices V, and a set of edges E. The goal is to find a minimum distance circuit that passes each vertex only once and returns back to the origin vertex. This cycle is known as a *Hamiltonian cycle*. For example, if the graph G has the set of vertices $V = \{a, b, c, d\}$, a Hamiltonian cycle can be $\{(a, b), (b, c), (c, d), (d, a)\}$. Therefore, the TSP aims to find the minimum cost Hamiltonian cycle through the given cities. Typically, the TSP is formulated as a weighted graph, and the edge weights can be calculated in several ways such as the euclidean distance between two points using the formula: If it is required to calculate the distance between two cities at locations $i = (x_1, y_1)$, and $j = (x_2, y_2)$ then: $D_{i,j} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Several problems have originated from the TSP, such as the multiple TSP and other Vehicle Routing Problems (VRPs). In the multiple TSP, more than one salesman can be used in the solution and accordingly the solution consists of multiple routes. This draws a similarity to the VRP, where Dantzig and Ramser [16] proved that this problem is a actually a generalisation of the TSP. However, VRP variants involve various operational constraints, in terms of the vehicles capacity, deliveries time windows, and the distribution and scheduling of deliveries making them even more challenging to address.

2.3.2 Overview of VRP Variants

In real world applications, and because of the complexity and diversity of the real-world systems, the CVRP only represents a narrow class of cases in a simplified way. However, the CVRP is one of the elementary variants of VRP from which other variants originated. Recalling from section 2.3, in the basic CVRP version, the goal is to find the minimum cost routes for serving a set of geographically dispersed customers with known demands. A fleet of homogeneous vehicles with fixed capacity located at a central depot serves the customers' requests, and each customer is visited exactly once to satisfy his/her demand. Here we present the main categories of other VRP variants in more detail.

- The VRP with Time Windows (VRPTW): imposes a time interval ("time window") on the delivery of each customer's request. Two further categories can be identified: the VRP with soft time windows (VRPSTW) [18] in which violating the time windows is allowed but associated with a penalty, and the VRP with hard time windows (VRPHTW) [19] in which the time windows must be respected.

- The Multi-Period VRP Problem (MPVRP) [20]: in this variant deliveries are scheduled within a planning period, with each customer requiring one or more visits during this period. The service days are known and the frequency of customer visits is predetermined.

- The Multi-Compartment VRP, and the Multi-Commodity VRP (MCVRP): concentrate on delivering different types of commodities to the customer, by either using a single vehicle, or by splitting them to several vehicles, thus requiring multiple visits to the same customer.

- The Pickup and Delivery Problem (PDP) [21, 22]: This problem involves picking up and delivering customers' requests from certain points of pick up and delivery, and this must be achieved by the same vehicle. Other variants of VRP have originated from the PDP such as: the VRP with backhauls, the VRP with simultaneous pick up and delivery, the VRP with mixed pick up and delivery, and dial a ride.

- The Split Deliveries VRP (SDVRP) [23]: in the SDVRP, the constraint that each customer is visited by only one vehicle is relaxed, and thus customers' demand can be split between several vehicles for delivery.

- The Multi Depot VRP (MDVRP) [24]: In this variant, it is assumed that there are multiple depots from which customers can be served.

- The Multiple Trips Vehicle Routing Problem (MTVRP) [25]: This problem assumes that trucks can visit the depot more than once in the time horizon for stock replenishment.

- The Open VRP (OVRP): in this variant, it is not necessarily the case that the vehicles end their journey at the depot location.

- The Stochastic VRP (SVRP): some elements of the problem are stochastic and unpredictable such as the number of customers, their requests, or their serving time.

- The Workforce Routing and Scheduling Problem (WRSP): this problem can be categorised as a general VRP and is concerned with the routing of staff from their home location to their working sites. A similar problem to the WRSP, is the Service Technician Routing and Scheduling Problem (STRSP), which involves designing the least cost routes for vehicles carrying a number of service technicians.

2.4 The Urban Transit Network Design Problem (UTNDP)

The problem of designing urban transit routes and schedules for a public transport infrastructure with known demand following practical constraints is referred to as the Urban Transit Network Design Problem (UTNDP). The UTNDP is a combinatorial optimisation problem that is NP-hard and is characterised by its computational intractability. For this reason, research has attempted over the years to develop numerous algorithms for solving the problem efficiently in a short computational times.

The UTNDP can be considered a very special variant of VRP problems, where there is no central depot from which vehicles start and terminate their journeys. Rather, passengers are picked up and dropped off at several locations along the routes, for example at bus stops. The main focus as in the general VRP is to reduce the total travelled distance encountered by the passengers as well as the expenditure of the operators.

The UTNDP is an important practical problem, that has a high impact on the development of the current urban societies. Having a robust infrastructure for public transport is a reflection of urbanisation, as well as being an essential service for individuals. Moreover, it contributes considerably on reducing the dependability on private cars, which recently resulted in many social, and environmental problems, causing high rates of accidents, traffic, and pollution. Due to the challenging nature, and the important social and practical impact of the problem, researchers tackled it as early as 1925 and several solution approaches, and algorithms have been applied and new studies continue to compete to find state of the art results and efficient algorithms for the optimal design of public transport routes.

The two main components of the UTNDP problem are: the Urban Transit Routing Problem (UTRP) and the Urban Transit Scheduling Problem (UTSP). Generally, the UTRP deals with the design of efficient transit routes on a given transportation network with known pick-up/drop-off locations, while the UTSP deals with the development of schedules and timetables for the vehicles travelling along the designed routes to serve passengers between their origin and destination locations. These two problems are usually solved sequentially, as the routes must be designed first before setting the schedules and timetables. The UTNDP has been classified by Ceder and Wilson [26] into five main stages that together contribute in the design of a public transit system: (1) network design (2) frequency setting (3) timetable development (4) vehicle scheduling (5) driver scheduling. The first stage, the transit routes design, is the most important, and on which the other stages are based.

The UTNDP is a very difficult and heavily constrained optimisation problem, due to its composition of several sub-problems and design stages as mentioned in the above paragraph, that are all NP-hard in nature and require to search for optimal solutions for an extremely large solution space. The UTNDP also deals with a complete set of decision-making processes in transportation systems including strategic, tactical, and operational decision making [27]. Moreover, the transit systems of transportation modes are characterised by their stochastic nature and complexity, making the UTNDP extremely difficult, requiring simplifications to be tractable for modelling and solving by optimisation algorithms.

In fact, most studies focus on tackling either one of the design stages, or commonly the first two design stages are combined together in a problem named "route design and frequency setting". Moreover, this problem is inherently multi-objective consisting of several criteria that should be optimised simultaneously. Mainly, the objectives that most studies have focused on are: the passenger costs represented by the total travel time, the percentage of transfers, and the operator costs represented by the total covered distance, or the fleet size. Transportation companies try to reduce their costs which can affect the service provided to the passengers, making the objectives of the problem conflicting in nature. Some of the factors that affect the operator costs are the transit vehicle size, distance travelled, vehicle operation hours for specific routes configuration, and the fleet size. On the other hand, the passengers require a transportation service with rapid travel times, less transfers, and frequent service. Other stakeholders who are involved in the development of a transit system are national and local government, local businesses, and taxpayers. All these stakeholders have their own benefit from the designed system and evaluate its efficiency with respect to their own perception and view. In a following section, we focus on the description of the UTRP, defining the problem mathematically and showing its deep-rooted complexity.

2.4.1 Difficulties of the UTNDP

Over years of research, the UTNDP has been identified as an extremely difficult and challenging problem, even for the most efficient optimisation algorithms. The majority of the reasons for the UTNDP complexity have been identified in John [7], Fan [28] doctoral theses:

- The problem is NP-hard, which means that the difficulty in finding a solution increases exponentially with the size of the problem.
- Although many models have been presented in the literature for solving the UTNDP, these models differ hugely in their description of the problem, constraints, and the objective functions considered. Therefore, there is no standardised accepted model that can be adopted as a reference.
- The constraints of the problem can be difficult to model and to satisfy, making the search and check of feasible solutions a complex process that involves considerable computation. .
- Different parts of the solution heavily depend on each other, and therefore it is difficult to evaluate a single route in isolation from the other routes in the route set. The quality of the route set is determined by all the routes belonging to it, and should be evaluated as a whole.
- The problem involves multiple conflicting targets making the problem inherently multi-objective. For example, reducing the service costs, and maximising the passengers benefit and welfare are targets that compete with each other.
- The collection of the input data in order to efficiently design a route set that reflects the real nature of the transport system is extremely difficult. Demand figures change throughout the day, and the passengers' behaviours are stochastic, and therefore finding an accurate measures is challenging. The consequence of this is that the design of routes can be totally wrong if the input data is poor.

2.4.2 UTNDP and VRP

The UTNDP falls into the broad category of VRP problems, although there are many key differences between the UTNDP and the various delivery problems discussed earlier. Generally, most VRPs involve multiple trips that originate and terminate at a depot location, and each route (vehicle) services a number of customers with pre-determined demands (requests), and a time window for the delivery/pick-up time. In the route design problem of the UTNDP, a set of routes is designed to pick-up and drop-off passengers from their origin to destination points, following a fixed schedule. The passenger cannot determine a preferred time window, but rather waits at the bus stop for the next vehicle to arrive. The VRP problems are solved on a daily basis to serve the varying demand of the customers, while the UTRP aims to design routes for the long term strategic planning with estimated demand figures. The general objectives in both problems are common, aiming to reduce the total costs associated with the total travelled distance and raising the customers' (passengers) satisfaction.

The closest variant to the UTRP amongst the other variants of VRP is the dial a ride problem (DARP). The DARP deals with the transportation of customers from their selected origin point to multiple destination points, using vehicles that are shared by a number of customers with different requests. However, there are several differences that distinguish the two problems. In the DARP, the vehicles are assigned on the basis of customers' requests and accordingly the routes are determined, while in the UTRP the routes are pre-determined and fixed, and the passengers cannot request their pick-up vehicle or preferred time for pick-up and arrival, instead the timetables are followed. In other words, in the DARP, customers' preferences and choices are taken into account, in contrast to the UTRP where passengers make selections from available routes and schedules according to personal preferences. The DARP is also planned on day to day bases, accommodating to the new received requests, and therefore the demand is variable. In the UTRP, It is assumed that the demand levels remain the same with insignificant variance during the day. The DARP is on a smaller scale and is a more flexible application compared to the UTRP, which is a long term planning application that is performed on a larger scale. This makes the UTRP a unique VRP variant that requires specialised algorithms to solve it.

2.4.3 The Urban Transit Routing Problem (UTRP): Problem Description

The urban transit routing problem (UTRP) involves developing a set of routes for an existing transport network following a set of constraints, and can be defined as the physical design of the of the UTNDP [29]. The transport network defines the layout of the street network and the connections between bus stops. The layout of the transport network can be given as a graph structure, where the bus stops are the graph nodes, and their connections are the graph edges (arcs). The goal is to develop a set of routes that allow each passenger to travel from any source to any destination point in the network, with the minimum travel time and transfers, while keeping the expenditures of the operating companies at an acceptable levels.

The route network is a part of the transport network, where several paths (i.e., routes) are superimposed to form a network. It should form a spanning subgraph of the transport network, containing all the nodes of the transport network and a subset of its edges. Most importantly, the route network must be connected, such that, there is a path connecting every pair of nodes in the network in order to satisfy the network demand. Here, we will define these terms mathematically and the objective functions. The formulation we describe for the UTRP is a simplified formulation based on an undirected graph.

The transport network can be represented using an undirected graph: $G = \{V, E\}$. The vertices in the graph $V = \{v_1, v_2, \ldots, v_n\}$ represent access points (i.e., bus stops), and the edges connecting the vertices $E = \{e_1, e_2, \ldots, e_m\} \subseteq V \times V$ represent direct transport links (i.e. roads). We also present two symmetrical matrices:

1. A travel time matrix T, which associates each edge with a specific weight value representing time required to traverse the edge. t_{ij} is the travel time between v_i and v_j. Note that $t_{ij} = +\infty$ if v_i and v_j are not directly connected, and $t_{ii} = 0$

$$T = \begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,n} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ t_{n,1} & t_{n,2} & \cdots & t_{n,n} \end{pmatrix}$$

2. A demand matrix D, representing the number of passengers travelling between two points in the network (which may consist of several edges in the transport network), where d_{ij}, represents the number of passengers travelling from vertex v_i and v_j, and $d_{ii} = 0$.

$$D = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n,1} & d_{n,2} & \cdots & d_{n,n} \end{pmatrix}$$

In order to construct an efficient route network, the demand estimations should be realistic and accurate. There are many possible approaches to estimate the demand between given origin and destination points in the network, some of these approaches are: conducting a public and private vehicle analysis, carrying out surveys on the local population, and examining the current ticket sales. However, it is still difficult to acquire such estimations, as the demand changes dynamically and is very sensitive to factors like service quality, and pricing policies [28]. The demand is a point to point demand, and only gives the number of travellers between two points in the network and does not give the total passengers flow. The passengers flow in the network (i.e., determining the total passengers volume on a path or a link) is calculated using a demand assignment procedure, which is performed concurrently with the route design to adjust the route frequencies. In this case the vehicle capacity is an important factor, and the frequency of the route with the assignment procedure results can determine if the passengers volume can be serviced by the vehicles operating on the route. The travel times between two points in the network are calculated depending on the problem modelling procedure. Usually shortest path algorithms are employed for this purpose, and the average of the travel time in both directions is considered.

A route $r_a = \{v_{i_1}, v_{i_2}, \ldots, v_{i_q}\}$, where $v_{i_k} \in \{v_1, v_2, \ldots, v_n\}$, is defined as a simple path in the graph connecting a set of edges. A route network $R = \{r_a : 1 \leq a \leq |R|\}$, is a connected set of routes, and the spanning subgraph of the transport network, in the sense that it should contain all the vertices present in the transport network, and a subset of its edges. The route network is what actually represents a solution to the UTRP. We will also define the *transit network*, which is the network constructed during the evaluation of the route network. During the evaluation, transfer nodes are identified and each node that corresponds to a transfer point between two routes is duplicated, and the two duplicates are connected through a transfer edge (Figure 2.5). As a result, the size of the route network can grow as much as an order of magnitude during evaluation, depending on the number of routes, their lengths and connectivity.

Mathematically, let E_a be the set of edges of route r_a. $G' = \{V', E'\}$ is the transit graph in which $V' \subseteq R \times V$. We define the node $y_{r_a i}$ as a pair consisting of a route r_a, and a vertex, $v_i \in V$. Let $(y_{r_a i}, y_{r_b j}) \in E'$ be the edge from node $y_{r_a i}$ to node $y_{r_b j}$ in the transit graph. Consequently, we define two types of edges: (i) *transport edges* (E'_1) correspond to in-vehicle travel links between two nodes within the same route, and (ii) *transfer edges* (E'_2) correspond to transfers from one route to another.

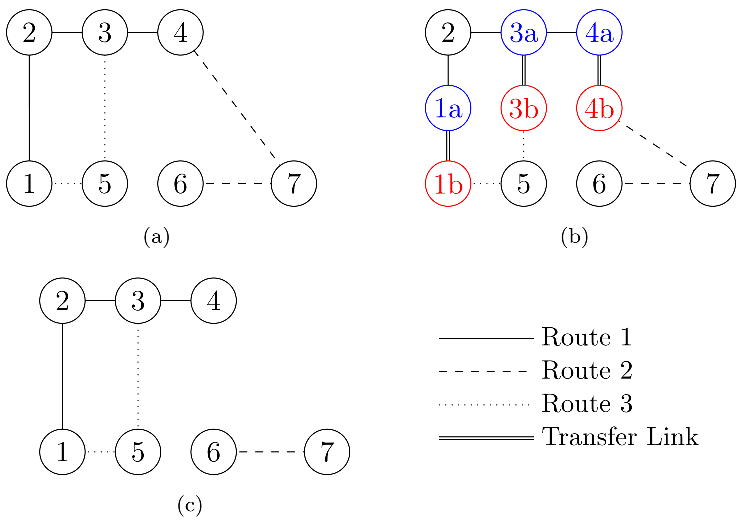


FIGURE 2.5: (a): Feasible route network containing three routes. (b): Transit network showing the duplication of nodes and transfer edges connecting the duplicates. (c): Infeasible disconnected network

Two objectives are considered in our model, the passenger cost, and the operator cost. An efficient public transportation system from the perspective of passengers, is a system with the lowest travel time, and the least number of transfers, or no transfers at all. Whereas the network operators are looking to reduce their cost and increase their profits. These objectives are contradictory, since reducing the overall expenditures may result in a poor service for passengers and vice versa.

To evaluate the passenger cost for a route network $R = \{r_1, r_2, \ldots, r_{|R|}\}$, the relevant transit network is built during the evaluation to incorporate both in-vehicle travel time, and transfer penalty. The passenger cost represents the average travel time of a single passenger when travelling in the network between source and destination. The minimum journey to travel between the two vertices v_i, v_j in the route network (R) is given by the shortest path $\alpha_{ij}(R)$ from node $\{y_{r_a i} : v_i \in r_a\}$ to node $\{y_{r_b j} : v_j \in r_b\}$ in the transit network, including both transport edges (Equation 2.1) and transfer edges (Equation 2.2).

$$E_1' = \bigcup_{r_a \in R} \{(y_{r_a i}, y_{r_a j}) : (v_i, v_j) \in E_a\} \tag{2.1}$$

$$E_2' = \bigcup_{v_i \in V} \{(y_{r_a i}, y_{r_b i}) : v_i \in r_a \cap r_b\} \tag{2.2}$$

The passenger cost is the total travel time made by all passengers who travel from their source to destination using the shortest path, over the entire demand served by the network [6]:

$$C_p(R) = \frac{\sum_{i,j=1} d_{ij}\alpha_{ij}(R)}{\sum_{i,j=1} d_{ij}} \tag{2.3}$$

The transport network operators aim to reduce the total cost of the system, while satisfying at least a minimum level of service quality. The operator costs include the fleet size required to satisfy the demand, the total distance travelled by the vehicles, the costs of maintenance, and the drivers employment costs. To simplify the operator costs, we use the sum of the cost (in time) for traversing all the routes in one direction [6]:

$$C_o(R) = \sum_{r_a \in R} \sum_{(v_i,v_j)\in E_a} t_{ij} \tag{2.4}$$

We use the following properties, which were also adopted by other studies [3, 30] to describe an efficient transit route network:

- The route network should satisfy most (if not all) of the transit network demand.
- The network provides directness for the passengers as much as possible, without the need to make a transfer during their journey.
- The network should reduce the overall travel times of the passengers.
- Prioritise the layout of those routes with the highest network demand.

We also use other performance indicators commonly quoted in the literature [3, 5, 6, 31] to evaluate the route sets more comprehensively.

- d_0: The percentage of demand satisfied with zero transfers.
- d_1: The percentage of demand satisfied with one transfer.
- d_2: The percentage of demand satisfied with two transfer.
- d_{un}: The percentage of demand unsatisfied (assuming that making three transfers or more is unacceptable).

We have a set of constraints that define the feasibility criteria of a route network. We list these constraints as follows:

- **C1:** A route network must contain all the vertices (bus stops) present in the road network.
- **C2:** All routes present in the route network must be free of cycles. This means a route with duplicate vertices is not accepted.
- **C3:** The route network must be connected allowing a passenger to reach any destination in the network from any source.
- **C4:** The number of vertices in any route must not be less than a minimum number, and not exceed a maximum number. These numbers are present as problem parameters set by the user.
- **C5:** The total number of routes in the route network is set to a specified number determined as a parameter by the user.

We have previously mentioned the complexity of the UTRP, and the difficulty of incorporating real world assumptions and constraints. For this reason, we have applied a set of constraints to simplify our model, in order to allow us to focus on a general methodology for network design that can be compared to previous work. Our assumptions are listed below [6]:

- Vehicles travel back and forth on the same route, and reverse their direction every time they reach the route terminal.
- The choice of a route from the passenger perspective is based on the shortest path (in terms of travel time + transfer time) between their origin and destination.
- The time the passenger needs to make a transfer is set to 5 minutes.
- Only symmetrical networks are considered. The values of the demand and travel time are the same regardless of the travel direction between any two points.
- The demand and the travel time between any two points in the network is fixed.
- One way streets are not considered. We assume that all road segments are traversed in two directions.

2.5 The VeRoLog 2019 Solver Challenge

Gromicho et al. [32] introduced the VeRoLog solver challenge (2018-2019) of the Euro working group in logistics and optimisation in cooperation with the Dutch based company ORTEC. Similar challenges were introduced in the years 2014 [33], 2015 [34], and 2017 [35][36]. The organisation of the challenges is a collaborative effort with companies, such as PTV, the leading German company in developing software solutions and consultations in traffic, transportation, and logistics, who organised the first and second editions in 2014 and 2015. The third edition in 2017 was organised by ORTEC, one of the largest providers of advanced planning and optimisation solutions and services. The aim of these challenges is to motivate the efficient design of algorithms to solve complex vehicle routing problems with multiple constraints, inspired by a real world situation, and to challenge participants to provide efficient solution methodologies under certain time restrictions.

The VeRoLog 2019 challenge tackled a multi-period VRP problem for a number of vehicles that travel from a central depot to deliver machines to customers based at several locations on their request. Each vehicle has a fixed capacity, and therefore can carry machines up to their limit and can travel back to the depot to transport additional machines. Additionally, each customer has a preferred time window for the delivery of their request. The subsequent installation of these machines requires a number of technicians, where the routing and the scheduling of the technicians is required such as to reduce each technician working days, and to minimise the number of days between the delivery and the installation of a certain request (see section 6.1 in chapter 6 for further details on the problem description).

More formally, the overall problem is an integrated version of the capacitated vehicle routing problem with time windows (CVRPTW) [1], and the service technician routing and scheduling problem (STRSP) [37]. The problem is highly complex, combining two interacting and co-dependent NP-hard routing problems into a single model, each problem having its own set of constraints, making it a unique and challenging topic. Additionally, the problem of the challenge shares characteristics with other VRP variants, such as: 1) The Split Delivery VRP (SDVRP): a single customer can order more than one machine type and these orders can be splitted between several vehicles 2) the Multi-period VRP (MPVRP): each customer can be visited more than once during the planning period 3) Multiple Trips VRP (MTVRP): the vehicles can travel back to the depot to carry more machines if they have not exceeded their maximum travelling distance per day.

The VeRoLog 2019 challenge is a unique VRP version considering its set of constraints and the integration with the staff rostering and routing problem. To the best of our knowledge there is no similar version investigated in the literature, and the best matching study to our problem description is by Bae and Moon [38] where they extended the Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW) to a problem of service vehicles used for delivery and installation of electronics. They developed a mixed integer programming model, a heuristic and a genetic algorithm, and compared their performances. There are differences between this study and the VeRoLog problem, for example we consider a longer planning horizon, and deal with multiple types of machines. We also allow multiple visits to the customer by different vehicles, while their model only allows a single visit for both delivery and installation. They also assign a maximum period of time between delivery and installation that must not be exceeded, while in our model we restrict this time by imposing a penalty.

The full description of the challenge and the competition rules can be found in the challenge official web page [1]. To summarise, the challenge was conducted in a number of phases each with specified submission dates, where each phase requires the submission of the results for a set of instances published publicly to the competitors. The final phase which names the restricted resources challenge, required the submission of the competitors' solvers to run it by the competition organisers on a hidden set of instance. Based on the results of this challenge, the competitors were ranked and the finalists were announced. Eight teams were announced finalists, and the top three teams received the first, second, and third prizes. Here, we will summarise the approaches of some of the finalists teams who published their methods and results.

The third ranked team, COKA coders published their results in [39] and they adopted a matheuristics approach hybridising heuristic methods with mathematical programming tools, more specifically a mixed integer programming method. First they decomposed the problem into sub-problems: trucks and people. The people (i.e., technicians) problem was further decomposed into routing and scheduling sub-problems, while such decomposition was not required for trucks. Further they described their novel matheuristic approach: "columnwise neighborhood search" to solve the decomposed sub-problems optimally or near-optimally as MIPs. They also show how the solutions to the sub-problems were fused to find a solution to the overall problem. The approach used by this team was different from the other approaches by using integer programming. The winner team published their results in [40] describing their solution method which combines an Adaptive Large Neighbourhood Search (ALNS) meta-heuristic with a Variable Neighbourhood Descent (VND) procedure and integrates these components through an adaptive layer that guides the search. The adaptive layer adjusts the solution method to the characteristics of the instance to be solved and balances intensification and diversification during the search. It also controls the search by allocating more time to solve the sub-problem that has the highest weight in the objective function. A decomposition approach was applied resulting in decomposing the problem into two main sub-problems for the routing of trucks and technicians respectively, such that the improvement on each sub-problem is handled

[1]https://verolog2019.ortec.com/

separately using the specialised heuristics in turn. The comparison of their methods with the second ranked team showed the efficiency of their heuristic-based approach in finding very competitive results in short time duration specifically for smaller instances.

Our solution method for solving the VeRoLog solver challenge is based on a population-based hyper-heuristic approach and a specialised set of low-level heuristics that take into account the inter-connectivity of the two routing phases for the trucks and technicians. We also develop a feasibility test to check that all the problem constraints are satisfied at each iteration. A mathematical model for the problem was developed, where we compare its results with the results of our hyper-heuristic framework and show that it is possible to find optimal solutions through hyper-heuristics in a faster computational time compared to the mathematical approach. In chapter 6 we will discuss the challenge further, and propose our solution methodology and results.

2.6 Summary

In this chapter, we introduced the basic definitions and concepts related to this research. We begin the chapter by defining the concept of mathematical optimisation, focusing on a special class of discrete optimisation problems named Combinatorial Optimisation Problems (COPs). We further demonstrate that optimisation problems differ in their complexities and some cannot be solved exactly in a polynomial time requiring the use of solutions methods such as heuristic methods to find sub optimal solutions in a feasible run time. We move forward to define the main problems tackled in the thesis starting with the VRP problem, demonstrating its origin, structure, and its several variants. We also define the Urban Transit Network Design Problem (UTNDP) and describe the reasons for its complexity, relating and comparing it to the VRP to show its differences and similarities. We formulate the UTRP model which will be adopted for solving the problem in chapters 4 and 5 describing the problem constraints, objectives, and the route set assessment. We concluded the chapter by discussing the problem central to the VeRoLog solver challenge 2019 which we tackled in chapter 6. We give an introduction to the VeRoLog challenges and the purpose for organising them with emphasis on the challenge of 2019. We also survey some of the approaches of the competition winners.

Chapter 3

Methods for Solving VRP Problems

In this chapter we overview the methods widely used to solve COPs, starting with exact mathematical approaches, and shifting towards heuristic and meta-heuristic methods, demonstrating the evolution of the research due to the application of meta-heuristic algorithms. We further explain the application of these various methods in solving different variants of the VRP for delivery applications, and the UTRP focusing mainly on the important contributions made by meta-heuristics. Further, we introduce the hyper-heuristic framework and describe the hyper-heuristic methodologies that were applied on the several problems addressed in this thesis with specific emphasis on online learning in selection methods and population-based hyper-heuristic algorithms. We finalise the chapter by summarising methods commonly used in solving MOOPs and describe the approach applied in this thesis for solving multi-criteria problems such as the UTRP using a single-point based framework such as selection hyper-heuristics.

3.1 Methods for Solving Combinatorial Optimisation Problems

Methodologies for solving COPs fall into one of the following categories: mathematical modelling, heuristics, and meta-heuristics (see figure 3.1). In the following sections we will be discussing each of these categories in further detail.

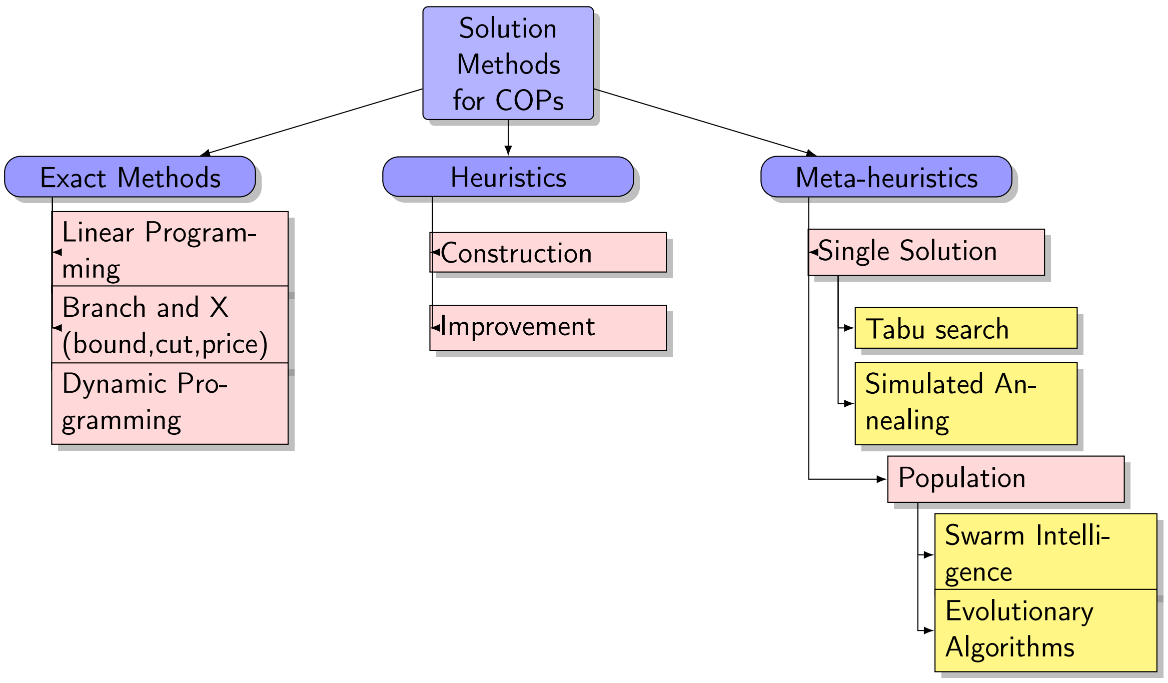


FIGURE 3.1: Solution methodologies for CO problems

3.1.1 Exact Mathematical Approaches

Mathematical approaches rely on mathematical formulations for the design of the objective function and its constraints. One of the best known mathematical programming methods is *linear programming*, where the objective function and the constraints are given as linear functions. When some or all of the decision variables are restricted to integers, the mathematical optimisation is called Integer Linear Programming (ILP), and if the decision variables are a mix of discrete and continuous, it is called a Mixed Integer Linear Programming (MILP). Other classical mathematical algorithmic frameworks include Dynamic Programming [41], and Branch and Bound (B&B) [42]. Mathematical programming approaches are useful when optimal solutions are to be found, and they can tackle COPs that are solvable in polynomial time as well as large size COPS in some cases.

Nevertheless, one of their main drawbacks is that for many COPs, they fail to scale to large size instances and cannot solve them in a finite amount of time. Chakroborty [43] stated that transportation engineering contains a multitude of optimisation problems that pose an extreme difficulty on traditional mathematical approaches, and one of such problems is the UTNDP. For these problems, simplifications should be introduced to the size and complexity of the model in order to apply the mathematical approaches, and for this reason their application became less favourable. In such cases, approximation methods can handle the practical size and the complexity of the problem instances.

3.1.2 Heuristic Methods

As a result of the failure of mathematical approaches in solving large scale versions of CO problems, heuristic methods have grown in popularity with the advance in computing technology, and have been applied since then in many studies.

Heuristic methods are used to approach an intractable optimisation problem by finding sub-optimal solutions in a polynomial time. Pearl [44] defined heuristics as an intelligent search strategy for computer problem solving. In optimisation problems, a heuristic is defined as a "rule of thumb" to guide the computational search for finding a solution. Although heuristics are designed to speed up the search process, they cannot guarantee to find an optimal solution unlike mathematical methods. However, in some NP-hard problems such as VRPs, it is a better choice to give up the search for optimal solutions in favour for improvements in run-time, or to find solutions for larger instances.

Heuristic methods are categorised according to how they explore the search space into construction and improvement heuristics. A construction heuristic attempts to build an optimal solution from scratch, while an improvement heuristic starts from a candidate solution and iteratively moves from one solution to another in its neighbourhood (i.e., a neighboured solution is generated by making small changes in the candidate solution). Heuristic methods are characterised as problem-dependent, which means they are specific to the problem they are trying to solve. This has motivated researchers to develop domain-independent and more generally applicable methods such as meta-heuristics and hyper-heuristics.

3.1.3 Meta-heuristics

The last decades have witnessed a great growth in computing power, and as a result meta-heuristic approaches have emerged as popular techniques to solve hard combinatorial problems. Meta-heuristics were defined by Sörensen and Glover [45]: "A meta-heuristic is a high level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimisation algorithms". Sörensen and Glover [45] also used the term meta-heuristic to define " a problem-specific implementation of a heuristic optimisation algorithm according to the guidelines expressed in a meta-heuristic framework". Due to the adaptability of meta-heuristics to different problems with complex structures, and their efficient computational performance and speed, research shifted towards using them for solving NP-hard problems. Meta-heuristic methods such as genetic algorithms, tabu search, simulated annealing, and particle swarm optimisation have played an important role in developing the research of CO and have outperformed previously applied heuristic approaches.

Meta-heuristics can be classified into two main broad categories: population based and single solution based. The single solution based meta-heuristics employ a single solution during the search, while the population based maintain a pool of candidate solutions (population). Some hybrid algorithms combine the two approaches into a single method. Here we demonstrate some of the well-known classes of meta-heuristic algorithms that are commonly used in solving VRPs and the UTNDP.

3.1.3.1 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are a class of population based search methodologies, which are inspired by the Darwinian theory of evolution. Each iteration of EA corresponds to a generation, through which a number of solutions in the population named individuals can reproduce. New solutions are created (i.e., offspring) by the recombination of two individuals (i.e, parent solutions) which are selected from the population using a selection strategy, and mutation is then performed on the new individuals to encourage diversity. The fitness of the new solutions are evaluated, and a selection strategy is applied to determine which individuals to remain in the next generation. The main operations of a generic EA are demonstrated by algorithm 1. Some of the well-known algorithms grouped under the category of EAs are: Genetic Algorithms (GAs), Evolution Strategies (ES), Evolutionary Programming (EP), and Genetic Programming (GP) [46].

Genetic Algorithms (GAs) are one of the most well known and mostly used algorithms amongst EAs that use the concepts of natural evolution and genetics for problem solving. Generally, a GA consists of a similar set of functions as the generic EA, but their implementation can different substantially according to the problem. The main components of a general GA are: solution representation (chromosomes), selection strategy, type of crossover, and mutation operators.

Until now, GAs are the dominant approach for solving the UTNDP, and several variants and approaches were developed and proved to find competitive results. For more in depth overview on GA methods and their application reader can refer to [47]

Algorithm 2: Evolutionary Algorithm

```
1 CreateInitialSolutions() // Initialise the population ;
2 Evaluate()// Evaluate each individual in the population;
3 repeat
4     SelectParents()// Select individuals from the population for mating;
5     Crossover() // apply cross over operator with a probability ;
6     Mutate() // apply mutation operator with a given probability ;
7     Evaluate()// Evaluate new individuals;
8     ReplaceSolutions() // Generate new population of solutions ;
9 until noGenerations;
```

3.1.3.2 Swarm Intelligence

Swarm Intelligence (SI) is an artificial intelligence discipline concerned with the design of intelligent multi agent systems by taking inspiration from the collective behaviour of social insects such as ants and bees [48]. The idea is that multiple smart agents can interact locally to achieve a common complex goal without the need for a centralised control system. Every individual uses simple rules to govern their actions, and the swarm reaches a desirable goal by the interaction of the entire group. Currently, the most well known swarm intelligent algorithms widely used in optimisation problems are Ant Colony Optimisation (ACO) [49] which mimics the way ants search food in nature, Bee Colony Optimisation (BCO) [50] inspired by the movement of bees during nectar collection process, and Particle Swarm Optimisation (PSO).

3.1.3.3 Single Solution Based Meta-heuristics

Single solution based meta-heuristics share the advantage of their ability to intensify the search on local regions in the search space, focusing on a single solution that is iteratively improved. Moreover, they are generally computationally faster by eliminating the need to maintain and evaluate individuals in the population. Some of the commonly applied single-point based meta-heuristics include: Local search algorithms (Hill Climbing (HC), Iterated Local Search (ILS)), Simulated Annealing (SA), Tabu Search (TS), and Greedy Randomised Adaptive Search Procedure (GRASP).

Local search algorithms [51] is a widely used set of algorithms that are based on the idea of examining the search space by initialising a solution and iteratively move to other neighbouring solutions. The neighbourhood of a solution are the set of solutions that can be generated by making changes (usually small) on the candidate solution, and the decision at each step of the search is based only on information about the local neighbourhood. The Hill Climbing algorithm (HC) is one of the well known local search methods which gradually improves a candidate solution by selecting the best neighbour based on an evaluation function. However this method can easily get stuck in a local optimum, a state where no more better neighbouring solutions can be found. Iterated Local Search (ILS) [52] improves the hill climbing local search by avoiding the easy entrapment in a local region in the search. It performs repeated iterations of perturbation and local search on a local minimum solution generated by the local search until satisfying a termination condition. This way, ILS maintains the balance between the exploration an exploitation processes by using perturbation and local search operators respectively.

Simulated Annealing (SA) is a probabilistic meta-heuristic framework that imitates the process of annealing in solids. At each decision point a new solution is generated, and it is accepted if its better than the previous solution. Worsening solutions are occasionally accepted to prevent entrapment in local optima. A worsening solution is accepted with a probability equals $P = e^{\frac{\Delta}{T}}$, where Δ is the solution quality change, and T is the method parameter, called temperature which regulates the probability to accept solutions with higher objective value (cost). Generally speaking, the search starts with a high temperature, and then according to the cooling schedule, the temperature decreases gradually throughout the search process.

Tabu search is a meta-heuristic introduced by Glover [53] in 1986. The idea of the algorithm is that it prohibits the recent moves in the search by maintaining a memory structure named a *tabu list* that prevents the repetition of the recently visited solutions. This can help the search in escaping the local optima.

GRASP [54] is an iterative meta-heuristic framework in which each iteration is made up of two phases: construction phase and local search phase. In the construction phase a solution is built, and repaired if it is not feasible, and a local search procedure is then applied to improve this solution until a local optima is reached, while keeping the best solution. Repeating the two phases with a new solution constructed at each iteration enhances the local search diversification.

3.2 Solving the Urban Transit Routing Problem

There is a considerable amount of research published on transit planning, due to its practical importance. Some researchers focused on a single aspect of transit planning, while others tried to solve multiple aspects simultaneously. Solution methodologies, problem models, and objectives differ substantially between different studies, making it difficult for any researcher to effectively compare their algorithm's performance with others. We try in this section to survey the different studies and methodologies that attempted to design algorithms for the problem of the optimal design of routes in urban transit systems on benchmark and on larger scale instances. We show the advance in research over the years, and how the emergence of meta-heuristics has helped significantly in the design of powerful algorithms that can handle such a computationally complex problem.

3.2.1 Analytical and Exact Mathematical Approaches

Analytical and mathematical methods were the first approaches for solving the UTRP, although they tend to focus on specific aspects of the problem using simplified network structures. Analytical methods attempt to find route attributes for a given network such as routes length and spacing, and develop relationships between the transportation network components rather than designing the actual routes. The disadvantages of analytical models have been pointed out by researchers, where Ceder [55] mentioned that these methods are suitable only when approximate values for the design parameters are needed to assess policies and not for a complete design. Tom and Mohan [56] stated that these methods are of a theoretical interest only. One of the early studies that applied analytical methods in the design of bus transit services is Holroyd [57], where they attempted the problem of finding the optimum positions of bus routes and the optimum frequency on these routes in an urban area. Their objectives are to minimise the sum of time costs associated with bus journeys, and the cost of providing a bus service. The problem is studied theoretically in a large uniform area with the bus routes forming a square grid. Byrne [58] built a model of a transit system in polar coordinates and radial transit lines with the purpose of finding line positions and headways that minimise the user and the transit agency costs. This is achieved in relation to a population density function, where the author stated that determining transit line locations must be done in relation to the population density in the region. In Byrne [59] different line speeds are introduced to a similar model under similar objectives. The study concluded that low speed lines should be terminated in some cases if they are adjacent to a high speed line. Examples of other studies that extensively analysed the UTRP under such methods are [60–62].

Mathematical programming approaches were attempted in the UTRP as early as 1970. Van Oudheusden et al. [63] used two well known location-allocation mathematical programming problems: the set Covering Problem (SCP), and the Simple Plant Location Problem (SPLP) for the design of bus network routes. It was found that the SCP is more effective when fixed demand is assumed, while SPLP is more powerful under the more realistic assumption of variable demand. van Nes et al. [64] combined heuristic and mathematical methods to design a model suitable for redesigning parts of a transport network, or the design of a complete network and the assignment of the frequencies. The designed model uses a single optimisation process that can be used for several design methods. It provided good results with test networks and actual data. Bussieck [65] presented in his doctoral thesis a mathematical programming approach for the problem of line planning in a public rail transport system. The study describes an approach based on Integer Linear Programming (ILP) which provides an effective tool for modeling line optimisation problems focusing on a specific approach for line planning that aims to maximise the number of direct travellers. A cost optimal line planning problem is also introduced and solved as an integer nonlinear program. The ideas in his research can be generalised to other modes of transportation networks.

Wan and Lo [66] implemented a mixed integer model for solving the route design problem and the frequency determination of the lines simultaneously. The problem is solved as a mixed integer formulation with the objective of minimising the sum of operating costs for all transit lines. The model was tested on a small example network of 10 nodes, and the final solution consisted of three routes. The authors stated that their work provides a good starting point for extending the model to consider both user and operator perspectives. They also mentioned that devising heuristic methods is crucial for application in practical size networks.

Guan et al. [67] dealt with the problem of simultaneous transit line configuration and passenger line assignment. The study focused on large city railways and the tests are carried out in Hong Kong city. Using a linear binary integer program that can be solved in any integer system such as a standard branch and bound method, the work attempts to model the transit line planning and the passenger transfer process. The objective is to minimise the total length of transit lines and the total length travelled by passengers, under constraints of routes length, maximum number of transfers and capacity. Several simplifications were introduced to reduce the computational burden on the binary integer program. The authors concluded the study by pointing out the need for more efficient algorithms for solving large size networks, and suggested meta-heuristics such as Simulated Annealing, Tabu Search, and Genetic Algorithms for this purpose.

Barra et al. [68] proposed a constraint satisfaction model to the solve the transit network design problem which was tested on a Constraint Programming (CP) system. They considered several service parameters in their model including minimum frequency, maximum transfers, and routes limits. The method is designed to be interactive, allowing an expert to use their experience by changing the design parameters to enhance the results. Their objectives included passenger satisfaction, and budget constraints represented by the total travelled distance, or the necessary fleet to operate the designed network. The model was tested on small instances derived from Mandl's network, and they stated that the CP package is unable to process large instances.

3.2.2 Heuristic Methods

Probably, Patz [69] was the first to tackle the route design problem using heuristics. He developed an iterative procedure to generate network lines (routes) based on penalties. Initially the network contains lines between each origin- destination pair with associated penalties calculated from the number of passengers who need transfers to complete their journey. The network lines are iteratively deleted based on these penalties. His approach was applied on a small ten node instance, but was not extensible.

Another early attempt to apply heuristic approaches was the study by Lampkin and Saalmans [70], who solved a case of a municipal bus problem by tackling it in four stages: reorganising the routes structure, determining frequencies on the routes, designing bus schedules and setting the timetables. A heuristic procedure is developed to design the network by building an initial route skeleton, and nodes are added one by one to this skeleton in later steps. Frequencies are then allocated to the routes so as to maximise the passenger service. A linear programming model is then used to assign buses to journeys.

Dubois et al. [71] studied the problem of modifying a transportation network to fit with the existing demand. A set of heuristic procedures were applied to the re-planning of bus routes in some medium size towns networks. They mentioned that heuristics are the only viable methods when the network size is no longer small, and proposed three greedy heuristic procedures for minimising the travel time and maximising accessibility. His methods has been tested in the design of transport networks in ten towns in France.

Sonntag [72] solved the transit network design on a rail system using a heuristic approach. In his work, an initial route set is created between every pair of stops and the initial routes are iteratively improved and infeasible routes are removed to find the best shortest paths with reasonable travel times and least number of transfers for passengers.

Following that, Mandl [73, 74] published his work, which is considered one of the most fundamental studies in the UTNDP, that assisted future research in understanding the principles of applying heuristics to the route design problem. His pioneering work produced the Swiss 15 node instance, which has become a defacto benchmark used by most researchers. The process consists of two phases: route generation, and route improvement. In the route generation phase, a shortest path algorithm is applied to compute the shortest paths for every vertex pair. These routes are then added in the route set by selecting the routes that has the most number of nodes, while the unseen nodes are added iteratively to the routes in the most convenient way. In the second phase, a heuristic algorithm is proposed that improves a given transportation network with the average transportation cost as an objective. This heuristic algorithm idea is to search for new routes, so that the entire route set remains feasible, and the average transportation cost is improved. If a new set of routes is found that is better than the old one, it is accepted and the search procedure continues until no further improvement is achieved. The improvements on the route set that were considered are:

- Create new routes by exchanging parts of routes at an intersection point.
- Add a node to a route if this node is close the route and the transportation demand between this node and the other nodes in the route is high.
- Reduce the length of a route by excluding nodes that are served by other routes, and the transportation demand between this node and the other nodes in the route is low.

Ceder and Wilson [26] identified the sequence of operations involved in the bus system design and planning: network design, frequencies setting, timetables development, bus scheduling, and driver scheduling. They also proposed a two level approach, stating that it is desirable that the design process incorporates alternative levels of complexity, due to the overall complexity of the bus system design and the vast number of involved factors. The first level of their model tackled the design of routes considering only the passenger perspective, while the second level determines the bus schedules and timetables taking both passenger and operator objectives into account. They also presented a route construction algorithm which produces an output that can be fed into both levels and also produces a set of feasible routes and their directness measures. Despite the sophisticated ideas presented in their work, its application was demonstrated on a very simple example of a five nodes network which does not sufficiently test the effectiveness of the proposed model.

Baaj and Mahmassani [75, 76] also developed a three stages heuristic approach based on artificial intelligence tools composed of a route generation algorithm guided by the passengers demand, followed by an analysis procedure to compute a number of performance measures for the initially generated route set. Finally a route improvement algorithm utilises the computed measures to produce feasible, improved route sets. Lee and Vuchic [77] developed an iterative heuristic procedure to solve the network design and frequency setting problem with variable transit demand under a given fixed total demand. The objective was to decrease the total travel time through an improvement procedure on an initially generated network utilising the shortest paths. A transit assignment procedure was also applied to concentrate demand on certain routes eliminating less efficient routes.

Some heuristic methods are based on heuristic construction procedures, which attempt to build optimal public transport route sets from scratch. An example of such method is the heuristic algorithm developed by Simonis in 1981 [78]. This method starts by generating a route using the shortest path between the highest density demand points and then deletes the demand satisfied by this route. The process iterates to the next highest demand points until a maximal number of routes is reached.

3.2.3 Meta-heuristic Approaches

3.2.3.1 Genetic Algorithms

Genetic algorithms (GAs) have been a very popular choice for solving the UTNDP and its components, despite their requirement for high computational power an their long run times compared with other methods. Nevertheless, recent research still focuses on their implementation and competitive results are acquired by this method.

Pattnaik et al. [79] was one of the first attempts to apply GAs to the transit route network design problem. They attempted to find transit routes and their associated frequencies with the objective of reducing the overall system cost (i.e. user and operator). They designed a two phase model: the first phase generates candidate solutions guided by the demand matrix and the route set constraints, and the second phase applies a GA to improve the quality of the route sets. They experimented with both fixed-length and variable-length string encoding schemes. Similar work was developed later by Tom and Mohan [56] using a simultaneous route and frequency coded model.

Chakroborty and Wivedi [3] proposed a three stage approach: an initial generation procedure using heuristic methods, a modification procedure based on a GA, and finally an evaluation procedure where they used a fitness function weighting three components: passengers in-vehicle and transfer times, percentage of demand satisfied with zero, one, and two transfers, and the percentage of unsatisfied demand. They applied their proposed method to Mandl's benchmark and could find better results compared with other methods at that time. In [43], Chakroborty addressed both transit route design and scheduling problems sequentially by applying the same GA approach they used in their previous work on Mandl's benchmark. The work also focused on showing the effectiveness of GA approaches on this problem compared to the previous traditional approaches.

Fan and Machemehl [4], used a genetic algorithm approach to examine the underlying characteristics of an optimal bus transit network with variable transit demand. The framework of the proposed solution is constructed of three main components; an initial candidate route set generation procedure, a network analysis procedure that decides transit demand matrix, assigns transit trips and determines service frequencies, and finally a genetic algorithm procedure that selects a route set from the huge solution space.

In Szeto and Wu [80] a bus network design problem for the town of Tin shui wai in Hongkong was solved, with the aim of improving bus services by reducing transfers, and passengers travel time. They proposed an integrated solution method to solve the route design problem and frequency setting problem simultaneously, and used the real world instance of the town. The authors used a GA to solve the network design problem and incorporated it with a frequency setting heuristic based on neighbourhood search to add frequencies to the routes.

Cipriani et al. [81] addressed the transit network design with elastic demand to define lines, frequencies, and vehicle sizes with aim of reducing operator costs, waiting time, and unsatisfied demand. The authors propose a solution approach consisting of two stages: (i) implementing a heuristic algorithm to generate potential lines and their frequencies and (ii) a GA that recombines lines to generate a new population of individuals while the fitness function evaluates them using a probabilistic modal split model which determines the mode choice behaviour of users, and a hyper-path transit assignment model that determines the route choice behaviour of users.

Mumford [6] developed several intelligent genetic operators within an evolutionary bi-objective framework, with the joint goals of minimising passengers average travel time, and operator's cost. A heuristic-based method was implemented to seed the population with feasible route sets, in addition to a new crossover operator, and mutation operators to add and delete groups of nodes to the routes. This work proposed four new benchmark instances which were made public for researchers.

Chew et al. [82] also approached the UTRP as a bi-objective problem. In their proposed algorithm the initial population is created with the aid of Floyd's algorithm for all pairs shortest paths. Their experiments were tested on Mandl's instance and compared with the work of Mumford [6] and Fan and Mumford [5], where they reported improved results.

The work in John et al. [83] is built upon the work of Mumford [6] using an NSGA-II bi-objective framework. They developed a new powerful heuristic construction method for candidate route sets generation and implemented eight new operators to perform replace, exchange, and merge operations. Their approach found improved results from both the passenger and the operator perspectives. The method was later implemented in a parallel model by Cooper et al. [8] to improve its efficiency in terms of run times. In Heyken Soares et al. [84] the algorithm of John et al. [83] was adjusted to solve a version of the UTRP with terminal nodes at the routes ends, and additional mutation operators were introduced. The algorithm was used to provide preliminary results for a new data set presenting the extended urban area of Nottingham city, which was generated from real-world data available in public sources.

Nayeem et al. [85] presented a genetic algorithm with an elitism approach. They generated the initial population using a greedy algorithm, and created the route set through choosing the pair of nodes with the highest demand and finding the shortest path between them. They also improved the genetic algorithm proposed in their previous work by allowing the population size to increase through copying high quality individuals from current generation to the next. The approach outperformed the known state-of-the art. However their objective evaluation focused only on the passenger perspective.

Arbex and da Cunha [86] addressed the network design problem and the frequency setting of the routes simultaneously and proposed an Alternative Objective Genetic Algorithm (AOGA) to efficiently solve the problem. Their approach consisted of alternating the focus of the optimisation to one of the objectives at each generation, and they incorporated both passenger objective as the sum of in-vehicle travel times, transfer and waiting times and the operator objective as the total operating fleet in the network. They applied their method on Mandl's instance with routes sizes 4, 6, and 8 and proved that their proposed GA is competitive with the current published results.

Most recently, Yang and Jiang [87] implemented a novel initial route set generation algorithm, and a route set size alternating heuristic that changes the number of routes in a solution, and embeds them into a NSGAII framework to provide an approximate Pareto front in terms of the passenger and operator costs. Their initial generation procedure assumes that maximising the demand satisfied directly is a key component to generate an efficient initial route set. They tested their algorithms on Mandl and Mumford benchmark instances, and their results on Mumford data set outperformed all the current available results including the recently developed hyper-heuristic approach in [88].

Other studies along with those listed above that applied GA to the route design problem/route design and frequency determination are given in table 3.1.

3.2.3.2 Swarm Intelligence

Various Swarm Intelligence algorithms have been applied to solve the UTRP and proved to be competitive and efficient compared to GA implementations. Yu et al. [89] developed a method that aims to maximise demand density on routes by dividing the transit network design process into three stages: First an empty network is built, and skeleton routes are added such as to maximise direct traveller density until constraints such as route length, demand, and directness constraints are exceeded. After this, main routes are laid into the transit network according to the maximum traveller density. Last, branch routes are laid on the transit network which includes skeleton and main routes. Their model aims to maximise the demand density of each route which is the transit demand divided by the length of the route, and the transit demand includes both direct trips and transfers. The ACO algorithm is applied to determine the design of the skeleton, main and branch routes while adhering to the problem constraints. Two test cases were used in this work, a simple network which has six nodes and nine links, and data from Dalian city in China, and the results showed that the optimised transit network can be improved with respect to transfers. Poorzahedy and Rouhani [90] tackled the route design problem for bus networks to minimise the travel time of the users, while maintaining the fleet size requirements. The ACO algorithm was applied to solve their model on the network of Sioux Falls, as well as a real scale network representing the city of Mashhad in Iran. Their developed Ant System works on a decision graph instead of the bus network itself, and their method has been calibrated to both network examples where they demonstrate its efficiency in each.

Nikolić and Teodorović [91] developed a model for solving the network design problem based on Bee Colony Optimisation (BCO), with the objectives of maximising the number of satisfied passengers, minimising transfers and minimising the total travel times of all served passengers. They proposed a simple greedy algorithm to generate the initial route set which aims to increase the number of direct trips by finding the nodes with the highest direct service demand, and calculating the shortest path between these nodes. These shortest paths are then added to the initial route set until the desirable number of routes in the route set is reached. They proposed two types of artificial bees to perform modifications on the solution. The algorithm was applied to Mandl's network and compared to the current best known solutions, where new best solutions were achieved for Mandl's problem versions with 4, 6, 7, and 8 routes. In [92] the authors extended their work to simultaneously determine the frequencies on the designed routes.

Kechagiopoulos and Beligiannis [93] proposed a PSO algorithm as a first attempt to apply it for solving the UTRP. Their model focused on the solution representation in terms of the route network and the evaluation procedure which evaluates two objectives: the passenger and the operator costs. They compared the performance of their method with other seven known methods in the literature and found that their approach is competitive on Mandl's instance with 4, 6, 7, and 8 routes. Recently, Jha et al. [94] implemented a Multi-objective particle swarm optimisation with multiple search strategies (MMOPSO) to solve the bus route design problem and frequency setting. Their approach consisted of two stages: a route set generation phase for the route design based on a GA implementation, and the frequency setting phase which is solved as a multi-objective problem by applying the MMOPSO framework to generate an approximate Pareto set of solutions between passenger and operator costs. They applied their methods to Mandl's benchmark and compared it with the state-of-the-art in the route design phase. They also justified the results of the second phase by comparing them with NSGAII results. Buba and Lee [95] proposed a hybrid differential evolution algorithm with particle swarm optimisation (DE-PSO) to simultaneously optimise the routes' configuration and their associated frequencies with the objective of minimising the passenger and operator costs. They conducted their experiments on Mandl's benchmark and a larger instance representing the Rivera city, Northern Uruguay. They demonstrated that their algorithm is competitive with other approaches on Mandl's benchmark and their multi-objective framework finds a diverse set of non-dominated solutions.

3.2.3.3 Single Solution based Meta-heuristics

Fan and Machemehl [96] used a SA algorithm to select the best set of routes from a pool of candidate routes. A set of initial solutions was created using Dijkstra's shortest path algorithm and Yen's k-shortest path algorithm. Route frequencies were determined simultaneously using a network analysis procedure to enable the computation of the required performance measures. The objective was to minimise the sum of the user costs, operator cost and unsatisfied demand. Three experimental networks were tested and a GA was implemented for comparison with the results. Their results proved the success of SA over GA in most cases of the tested example networks. Fan and Mumford [5] applied SA with a make-small-change procedure as a neighbourhood operator. At first, a random route set is generated and the make-small-change procedure applies one of three moves: add a vertex to a randomly selected route, delete a vertex from the route, or invert the route vertices order. The SA algorithm was applied and compared to a simple hill climbing algorithm on Mandl's benchmark, and was able to find better results.

Fan and Machemehl [97] used tabu search to solve the optimal bus transit route design problem at the distribution node level. Their approach consists of three stages: an initial candidate route set generation procedure that generates all feasible sets of routes following the practical transit guidelines, a network analysis procedure that computes performance measures, assigns transit trips and calculates frequencies, and a Tabu search procedure that guides the candidate solution generation process. The objective is a weighted sum of passenger and operator costs and unsatisfied demand. Three different variants of Tabu algorithm were implemented and compared to genetic algorithms to measure the performance quality of TS, and the results showed clearly that it outperforms GAs. Mauttone and Urquhart [98] solved the UTRP as a multi-objective problem by applying a heuristic based on the GRASP meta-heuristic. They proved that their method produced better non-dominated solutions than the weighted sum method with the same computational efforts for Mandl's instance and another real test case. Kılıç and Gök [31] reported the importance of good quality initial solutions. They proposed a new initial route generation method that employs the level of demand as guidance for their construction. They used hill climbing and tabu search algorithms to test their method, and implemented simple operators to modify route sets including add, delete and swap.

TABLE 3.1: Some selected studies applying various meta-heuristic algorithms for the optimisation of the route or the routes and frequencies in the UTNDP. Studies applying GA methods are highlighted to show the prominence of GA in previous literature.

No.	Reference	Year	Objectives	Decision Variables	Metaheuristic
1	Pattnaik et al. [99]	1998	Total travel time	Routes, Frequencies	GA
2	Chakroborty and Wivedi [3]	2002	Passenger and Operator costs	Routes	GA
3	Bielli et al. [100]	2002	Multiobjective	Routes	GA
4	Tom and Mohan [56]	2003	Passenger and operator costs	Routes, Frequencies	GA
5	Ngamchai and Lovell [101]	2003	Passenger and operator costs	Routes, Frequencies	GA
6	Agrawal and Mathew [102]	2003	Generalised travel cost and operator cost	Routes and frequencies	Parallel GA
7	Fan and Machemehl [4]	2006	Passenger and operator costs	Routes, Frequencies	GA
8	Fan and Machemehl [96]	2006	Passenger and operator costs	Routes, Frequencies	SA
9	Yang et al. [103]	2007	maximum direct travellers per unit length	Routes and stops	Parallel ACO
10	Fan and Machemehl [97]	2008	Passenger and operator costs and unsatisifed demand	Routes and frequencies	Tabu Search
11	Fan [28]	2009	Total travel distance and transfers	Routes	SA
12	Yu et al. [104]	2010	Total distance and transfers	Routes and frequencies	GA
13	Bagloee and Ceder [105]	2011	Minimum passenger discomfort	Routes and frequencies	GA and decomposition
14	Szeto and Wu [80]	2011	Total travel time	Routes and frequencies	GA
15	Cipriani et al. [81]	2012	Passenger and operator costs	Routes and frequencies	Parallel GA
16	Chew and Lee [106]	2012	Passenger cost	Routes	GA
17	Mumford [6]	2013	Passenger and operator costs	Routes	EA
18	Chew et al. [82]	2013	Passenger and operator costs	Routes	GA
19	Nikolić and Teodorović [91]	2013	total travel time	Routes	BCO
20	Afandizadeh et al. [107]	2013	Passenger and Operator costs	Routes	GA
21	John et al. [83]	2014	Passenger and operator costs	Routes	NSGAII
22	Cooper et al. [8]	2014	Passenger and operator costs	Routes	Parallel GA
23	Nayeem et al. [85]	2014	Passenger costs	Routes	GA
24	Nikolić and Teodorović [92]	2014	Passenger and operator cost, unsatisfied demand	Routes and frequencies	BCO
25	Kılıç and Gök [31]	2014	Passenger and operator costs	Routes	Tabu Search
26	Amiripour et al. [108]	2014	total waiting time	Routes	GA
27	Kechagiopoulos and Beligiannis [93]	2014	Passenger and operator costs	Routes	PSO
28	Arbex and da Cunha [86]	2015	Passenger and operator costs	Routes and frequencies	GA
29	Zhao et al. [109]	2015	Passenger cost and unsatisfied demand	Routes and frequencies	Memetic Algorithm
30	Owais et al. [110]	2015	Passenger and operator costs	Routes and frequencies	GA
31	Nayeem et al. [111]	2018	Multi-objective	Routes	EA
32	Buba and Lee [112]	2018	Passenger cost and unsatisfied demand	Routes and frequencies	Differential Evolution (DE)
33	Buba and Lee [95]	2019	Passenger and operator costs	Routes and frequencies	DE-PSO
34	Heyken Soares et al. [84]	2019	Passenger and operator costs	Routes	NSGAII
35	Islam et al. [113]	2019	Minimise travel time and maximise satisified demand	Routes	Stochastic Beam Search (SBS)
36	Fan et al. [114]	2019	Average passenger travel time and number of transfers	Routes	Flower Pollination Algorithm (FPA)
37	Jha et al. [94]	2019	Travel time and operator cost	Routes and frequencies	MMOPSO/GA
38	Duran et al. [115]	2019	Total travel time and CO_2 emmisions	Routes and frequencies	GA
39	Mahdavi Moghaddam et al. [116]	2019	Passenger and operator costs	Routes and frequencies	GA
40	Duran-Micco et al. [117]	2020	Total travel time and CO_2 emissions	Routes and frequencies	Memetic Algorithm (MA)
41	Yang and Jiang [87]	2020	Passenger and operator costs	Routes	NSGAII
42	Chai and Liang [118]	2020	Passenger travel time and number of vehicles	Routes and frequencies	NSGAII
43	Liang et al. [119]	2020	Passenger and operator costs	Routes and frequencies	Cooperative coevolutinary MOEA

3.2.4 UTRP Algorithms in Real-world Planning

Despite the huge amount of research on computer-based solutions for solving the UTRP, there are few studies that have been actually used in real-world planning processes [120]. One example is the work by Pacheco et al. [121], who proposed a solution algorithm to a bus design problem posed by the city council of Burgos city in Spain. The problem consisted of designing bus routes and assigning buses to these routes to optimise the service level such as to reduce the waiting times at bus stops and the duration of trips. Two algorithms were implemented, a local search and a tabu search and the two decision levels are optimised in alternating steps. Their algorithm was able to perform better than the tools used in planning by the city authorities. Another example is the study from 2012 by Cipriani et al. [81] in cooperation with the mobility agency of Rome, Italy. It included the application of a genetic algorithm on undirected graph representing the street network of Rome. The results show improvements over the existing bus route network in terms of waiting times, operator costs and unsatisfied demand. According to the authors, the mobility agency of Rome started implementing their results in 2012. Many other studies compare their results against the real existing routes, showing that their methods can lead to improvements over an existing service (e.g. [84, 100, 105, 122]). However, the results of these studies have not been verified in real-world planning processes.

3.2.5 Limitations of Previous Research in the UTRP

There are some clear limitations with most previous approaches applied to solve the UTRP. The lack of benchmark data for the problem is a serious issue, and many researchers implemented methods that are highly specific to given towns or cities. Furthermore, these instances are not publicly available so we cannot judge the generality of the applied methods. Other researchers who implemented and tested their methods on benchmark instances used Mandl's 15 node benchmark, which is a very small instance. We cannot judge the performance of a method in terms of scalability based on such a small network. If we consider GA approaches in particular, with a population of perhaps several hundred solutions to maintain, the run-time for a road network of 100 vertices or above can be measured in days rather than hours.

Moreover, we notice there is a wide range of objective functions and route set evaluation methods used by different studies. The lack of standardised methods for the evaluation and assessment of efficient route networks makes the direct comparison among different studies not possible. Moreover, we identified that GAs for solving the UTRP are the dominant algorithms in the literature so far with the most competitive results despite their computational burden. The proposed single point based algorithmic solutions so far do not compete with the results of the studies applying GAs. There is a lack of computationally efficient algorithms with ability to scale and provide high quality results at the same time. This has given the inspiration to test the hyper-heuristic approach as a possible way forward and to address these limitations.

3.3 Solution Methods for VRP Delivery Problems

In this section, we turn our attention to a different class of VRPS, concerned mainly with the delivery of goods, rather than the transport of passengers such as in the UTRP. The term VRP was first introduced in [16] as a truck dispatching problem, where they modelled the problem of how a homogeneous fleet of vehicles can serve oil demand of a number of gas stations from a central hub, with a minimum travelling distance. This method was the first proposed heuristic approach for solving a VRP, and was then generalised in [123] to a linear optimisation problem: how to serve a number of customers located around a central depot, using a fleet of vehicles with varying capacities. These two studies have put the fundamentals and the first formal definition of the currently known Capaciated Vehicle Routing Problem (CVRP).

Due to the wide available literature on VRPs, we will limit the survey here to the VRP variants related to the problem addressed in VeroLog solver challenge 2019, and the reader can refer to published books and survey papers for more literature and taxonomy of VRP studies [1, 17, 20, 124].

The CVRP problem has been tackled by several approaches and algorithms including exact mathematical approaches, with branch-and-cut (BC) algorithms [125, 126], and algorithms based on the set partitioning formulation [127, 128] being the most successful. One of the first attempts to describe an exact BC algorithm for solving the CVRP is in [125], where they successfully solved for the first time a CVRP instance with 135 customers. Many studies also applied heuristic methods in the CVRP, an example study is Fisher and Jaikumar [129] who proposed a heuristic that performs a generalised assignment procedure to assign customers to vehicles with an objective function that approximates delivery costs. Also, one of the earliest proposed heuristic algorithms for solving the CVRP is the study of Clarke and Wright [123], which is considered the first study that proposed the CVRP as its current formulation. In terms of meta-heuristics, Genetic Algorithms (GAs) have been widely applied to the CVRP. In the work of Morgan and Mumford [130] a hybrid GA was developed to solve the CVRP by slightly perturbing the customers coordinates to fool the the Clark and Wright simple heuristic and produce better solutions than if the heuristic is applied to the original customer locations. It is also common in many studies to combine local search techniques with GAs to improve offspring [131–134].

For the version of CVRP with Time Windows (CVRPTW), exact methods have been successful for cases with up to 100 customers [135], and as a result heuristic and meta-heuristic methods have been preferred for solving instances of large scale. Examples of heuristic methods applied to the VRPTW that use route construction and local search algorithms can be found in [136–138], and other studies that applied meta-heuristics such as genetic algorithms, ant colony, tabu search, and simulated annealing are in [139–142].

In Beltrami and Bodin [143], the Periodic VRP (PVRP) problem was addressed for the first time to solve a problem related to garbage collection at industrial sites, and the nature of the demand required several visits to each site in a duration of a week. The authors introduced two key heuristics to solve the problem. A study by Rahimi-Vahed et al. [144] applied a path relinking algorithm to the multi-depot periodic vehicle routing problem by generating a reference set of elite solutions, and combining characteristics from those solutions to find better solutions. The computational results show that this method produces good results in both run-time and solution quality. Archetti et al. [145] presents three ways to formulate the multi-period vehicle routing problem with time windows and solve the problem using a branch-and-cut algorithm. The algorithm was able to find good solutions for small problems of ten orders, but was unsuccessful in larger problems. Alonso et al. [146] proposed a tabu search algorithm for the periodic vehicle routing problem with multiple trips and accessibility restrictions such that not every vehicle can visit every customer. When tested on randomly generated test problems, it performed reasonably well with regards to solution quality. Furthermore the computation time was manageable for instances as large as 1000 orders. We refer the reader to [20] for more literature on the PVRP.

Mirzaei and Wøhlk [147] conducted research on two variants of the multi-compartment VRP (MCVRP), one concentrates on split deliveries for different commodities, and the second focuses on delivering all commodities by a single vehicle. They proposed a branch-and-price method and compared the optimal costs of the two variants. The computational results were presented for instances with up to 100 customers, and the algorithm optimally solved instances with up to 50 customers and four commodities. Heuristic examples such as [25] proposed an iterated local search algorithm for solving the multi-commodity multi-trip VRP with the objective of minimising the number of used vehicles. In [148] the authors addressed the commodity constrained split delivery VRP, where multiple commodities can be mixed in a single vehicle while satisfying the capacity constraint and each customer can be visited more than once, but a single commodity type should be delivered in one delivery. They proposed a heuristic based on the adaptive large neighbourhood search (ALNS) and tested their approach on benchmark instances. Among the meta-heuristic methods applied to the MCVRP, genetic algorithms are the most common so far. [149] worked on a VRP encountered in the cold supply chain logistics of the frozen food delivery. In their model, they associated a penalty cost for late delivery based on the types of products and proposed a GA for solving the model with real data.

In the Split Delivery VRP (SDVRP), the first heuristic approaches were introduced by [23, 150]. After these studies, most of the subsequent work focused on meta-heuristics or hybrid schemes. One example is the work of [151] who applied a tabu search algorithm, and [152] used a genetic algorithm combined with a local search procedure. Hybrid algorithms have then grown in popularity, examples are found in [140, 153]. Many exact models have also been proposed on this problem, and one example is the study in [154]. For further literature on SDVRP we refer to [151].

Finally, we mention the Service Technician Routing and Scheduling Problem (STRSP), which is the focus of the second part of the VeRoLog solver challenge. Cordeau et al. [37] solved a real life technician scheduling problem for a large telecommunication company set as a competition by the French Operational Research Society in 2007. In this paper an adaptive large neighbourhood search algorithm is implemented. In [155], the authors concentrated on a field technician scheduling problem in the telecommunications industry, and their purpose was to maximise the number of served requests as well as considering the request's priority and the technician's skill level. A local search algorithm, a Greedy Randomised Adaptive Search Procedure (GRASP) and a greedy heuristic algorithm were proposed to solve the problem. [156] studied the service technician routing and scheduling problem with the objective of minimising the total routing and outsourcing costs. The authors used an adaptive large neighbourhood search algorithm for solving the problem on artificial and real-world instances. [157] proposed a parallel matheuristic approach for solving a variant of the TRSP in which a number of technicians with a set of accompanying skills, tools and spare parts need to be scheduled and routed within given time windows. The study dealt with the availability of tools and spare parts for the technicians and routing them to the depot for the replenishment of tools. [158] used an iterated local search algorithm to solve the TRSP. They studied a variant where it was given which technicians can serve which orders. The algorithm was benchmarked on instances ranging from 25 to 100 orders and compared to an ALNS algorithm, where it was found that it performs significantly better on large instances with fast computational times.

3.4 Optimisation with Selection Hyper-heuristics

Research in computational intelligence and optimisation fields has a significant influence on the design of bespoke heuristic algorithms for solving real-world complex optimisation problems. However, most of these methods require significant modification when applied to different problem domains, making them highly specific to a single problem domain or a specific class of problems or instances. This was the main inspiration for the development of a general, problem-independent heuristic search methods, known as *hyper-heuristics*.

Hyper-heuristics have emerged as solution methodologies that raise the level of generality of search techniques for computational search problems. They can be defined as a high level automated search methodology, that explores the space of low level heuristics, or heuristic components while solving computationally difficult problems. Since their development, they have received significant attention in the research community, and competed against other known heuristics and meta-heuristics in solving different variants of complex optimisation problems.

A study by Fisher [159] concluded that mixing and combining different low level heuristics leads to better quality solutions than applying them separately, where the single heuristic application can be effective at some stages of the search, and perform poorly at others. This study was the earliest motivation for the design of the general purpose framework of selection hyper-heuristics. Following this, Cowling et al. [160] was the first to use the term *hyper-heuristic* and defined it as "*a heuristic to choose a heuristic*", and mentioned that hyper-heuristics work at a higher level of abstraction than other meta-heuristics. In their study, they solved a personnel scheduling problem and introduced most of the known basic selection and move acceptance components.

Hyper-heuristics have higher abstraction capabilities than other meta-heuristics, where the search is conducted on the space of heuristics, controlling and perturbing a set of low level heuristics which work directly on the solution space. Therefore hyper-heuristics are isolated from any specific problem domain information and only control the low level heuristics as a set of black boxes. In this way, the search can be utilised to focus on other qualities, such as changes in the objective and the execution time of the search process. This concept is known as the domain barrier, which explicitly prevents any problem specific information from passing to the higher level of hyper-heuristics. Burke et al. [161] classified hyper-heuristics based on the nature of the heuristic search space into *selection* hyper-heuristics that select from an existing set of heuristics, and *generation* hyper-heuristics that generate new heuristics from the components of existing ones. The former class, selection hyper-heuristics, is the focus of the work in this thesis.

Selection hyper-heuristics are based on an single-point based iterative framework which repetitively applies a heuristic to a single solution at each step of the search. A low level heuristic [1] is selected and applied to an initial created solution at each iteration, and a decision is made if the new solution is accepted or not. A good selection hyper-heuristic selects a suitable low level heuristic to diversify the search, if the search process stagnates locally. This reflects the importance of the efficient design of a selection method that can automatically lead the search.

The iterative framework of selection hyper-heuristics consists of two successive stages, as has been identified by [10]: a *heuristic selection* to choose a low level heuristic and generate a new solution, and *move acceptance* to decide the acceptability of the new solution based on the fitness evaluation. The two processes iterate to improve an initially generated solution until meeting a termination condition as illustrated in figure 3.2. The general framework of selection hyper-heuristics is also demonstrated by algorithm 3. Most of the selection hyper-heuristics components are reusable, and can be applied in several problem domains, or instances of the same domain without requiring any modifications. Another crucial feature of selection hyper-heuristics as described in [9], is that the different components of selection and move acceptance methods deliver different performances on the same instance or problem domain. This observation means that different combinations of selection and move acceptance components can be applied, and yet get varying performances depending on the problem nature.

[1]low level heuristics are operators that perform simple neighbourhood moves in the solution when applied. They are designed according to the specifications of the problem domain and can be either perturbative, or constructive heuristics.

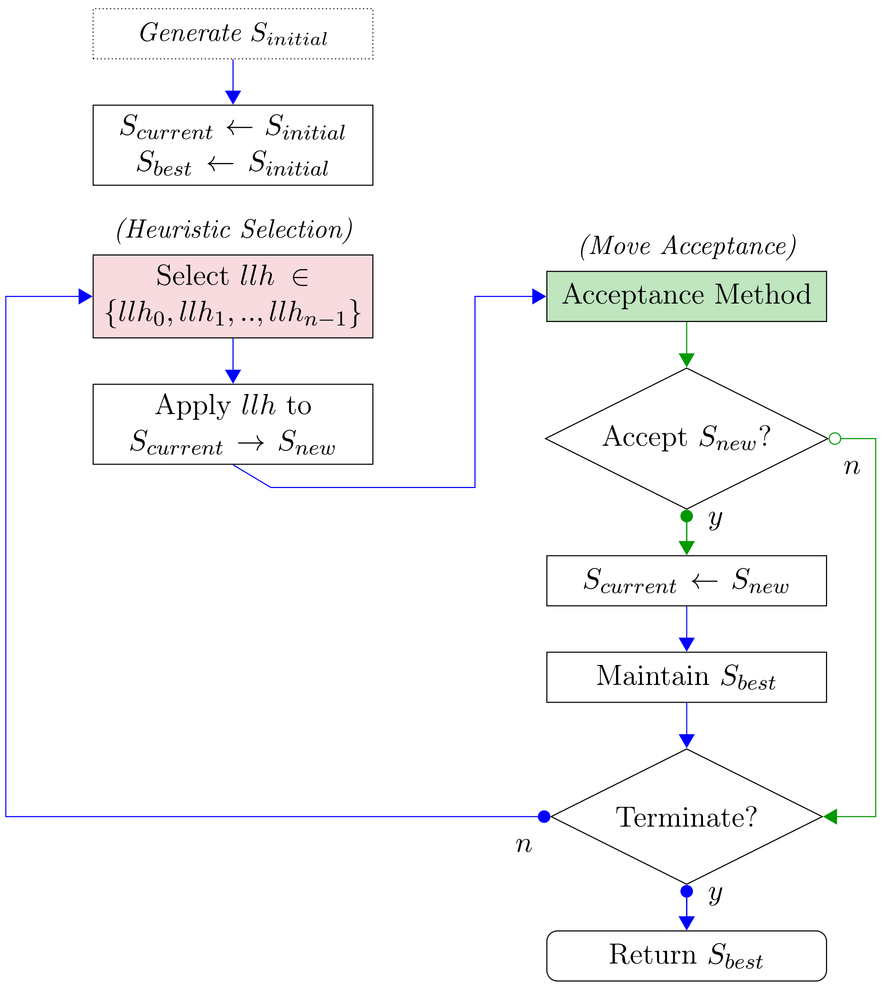

Generate $S_{initial}$
$S_{current} \leftarrow S_{initial}$
$S_{best} \leftarrow S_{initial}$
(Heuristic Selection)
Select $llh \in \{llh_0, llh_1, .., llh_{n-1}\}$
Apply llh to $S_{current} \rightarrow S_{new}$
(Move Acceptance)
Acceptance Method
Accept S_{new}?
n
y
$S_{current} \leftarrow S_{new}$
Maintain S_{best}
Terminate?
n
y
Return S_{best}

FIGURE 3.2: A generic selection hyper-heuristic framework. The green arrows represent the acceptance component

3.4.1 Classification of Selection Hyper-heuristics

Hyper-heuristic algorithms have been classified into sub-categories based on a number of criteria. Burke et al. [161] reviewed previous classifications and provided a general classification based on two considerations: 1) the nature of the heuristic search space, 2) source of feedback during learning. With the first consideration, hyper-heuristics are classified into selection and generation hyper-heuristics, where the former selects a heuristic from pre-existing perturbation or constructive heuristics, and the latter generates new heuristic methods from components of pre-existing perturbation or constructive heuristics. According to the feedback mechanism, selection hyper-heuristics can be classified into online learning, offline learning, or no learning mechanisms. The online learning selection hyper-heuristic learns from feedback during the search to improve the process of selecting low level heuristics, and the offline learning learns before the search starts

Algorithm 3: Algorithm for the general single-point based selection hyper-heuristics framework

```
Let $S, S', S_b$ be current, new, best solutions respectively;
Let $LLH = [llh_1, llh_2, \ldots, llh_{|LLH|}]$ be the set of low level heuristics;
$S_{initial} \leftarrow$ InitialGeneration();
$S \leftarrow S_{initial}$;
$S_b \leftarrow S$;
repeat
    $llh \leftarrow Select(llh_i, LLH)$;
    $S' \leftarrow$ ApplyLLH($llh_i, S$) ;
    if $Accept(S', S)$ then
        $S = S'$;
    end
    if $S$ isBetterThan $S_b$ then
        $S_b \leftarrow S$;
    end
until timeLimit;
return $S_b$;
```

on a set of test instances. Selection hyper-heuristics with no learning select a heuristic randomly, or from a fixed permutation without keeping a record of their previous performance. Selection hyper-heuristics are also classified based on the nature of the low level heuristics into two categories: selection perturbative hyper-heuristics, and selection constructive hyper-heuristics [2]. Perturbative hyper-heuristics work on complete solutions, while the constructive process partially built solutions.

Drake et al. [162], in their most recent survey on the advances of selection hyper-heuristics research have identified other classes of selection hyper-heuristics additional to the aforementioned. They have also extended the above classifications to include the following classes:

- Nature of the low level heuristics set: the selection hyper-heuristic can control the whole, reduced, or increased set of low level heuristics.
- Parameter setting: static, dynamic, or adaptive parameter setting. In the static setting, the parameters are set statically prior to the search process, and in the adaptive and dynamic setting, the parameters are allowed to change reactively or dynamically during the search.
- Nature of the move acceptance: either stochastic or non-stochastic based on whether a probabilistic framework is used on the acceptance decision.

3.4.2 Online Learning Selection Hyper-heuristics

In the above section, we have discussed how selection hyper-heuristics are classified based on the feedback mechanism to online, offline, or no learning hyper-heuristics. Research in hyper-heuristics has identified the importance of incorporating learning mechanisms in order to raise the level of generality of the framework and to make it more adaptive to the requirements of the search. The offline learning approach requires test instances for training before the search starts, and this is usually a time consuming process, and is not adaptive to the performance changes of the low level heuristics. Therefore, online learning is a more practical option and has more potential than offline learning.

Several studies have identified the impact of the choice of the hyper-heuristic components and the parameter settings in the overall performance across different problem domains or even instances of the same domain [9]. Online learning methods therefore, contribute to solving this issue by giving hyper-heuristics the ability to adapt, learn, and configure themselves and accordingly optimise better [163].

Most of the existing online learning hyper-heuristics use *reinforcement learning.* Here, the hyper-heuristic maintains a utility value for each low level heuristic which is updated through a reward and penalty scheme. Some examples of selected studies that used reinforcement learning are [164–166]. We will describe our online learning algorithm which uses a reward and penalty model based on transitioning between several states of the low level heuristics that simulates the states of the hidden markov model. Further details are in section 3.5.1.

3.4.3 Population-based Selection hyper-heuristics

There are various criteria used to classify selection hyper-heuristics, and one of them is the solution nature, where selection hyper-heuristics are classified based on this measure into single point or multiple point. Most of the previous research focused on the selection hyper-heuristic as a single-point framework, where a single solution is iteratively improved until a termination condition is met (figure 3.2). However, there are many studies that have applied the framework as a population by utilising multiple current solutions during the search, usually each of them is improved individually and a global best solution is found after a specific amount of running time. One of the advantages of using a population-based hyper-heuristic framework is to improve the diversity of the search and the exploration of the search space, which can help to discover new regions that can improve the solution. However, one of the downsides is that the total run time will be divided between the multiple solutions in the population, and therefore each solution will be improved only during that specified amount of time.

The majority of the previous studies on selection hyper-heuristics present approaches based on single-point-based search, and only a few used a population of solutions or a mixed approach alternating between using single and multiple solutions for the search. Moreover, those previously proposed population based approaches are mostly a hybrid between a selection hyper-heuristic and an evolutionary algorithm framework.

Cowling et al. [167] investigated a genetic algorithm based on hyper-heuristics for the personnel scheduling problem. A GA is implemented and applied as a high level selector, and a set of low level heuristics are used at each generation to locally improve the quality of each individual, where the low level heuristics are applied in any sequence. [168] proposed a Monte-Carlo tree search hyper-heuristic framework that tries to identify good sequences of heuristics using a Monte-Carlo search tree. A memory mechanism containing a population of solutions is utilised, and at each iteration a solution from the population is selected, and the population is subsequently updated using several updating rules. [169] proposed a memetic algorithm based on hyper-heuristics to solve an examination timetabling problem. Their approach constructs several heuristic lists based on graph colouring heuristics and applies evolutionary operators to generate new lists. A local search method is used to further optimise the solutions. [170] implemented a hyper-heuristic based on variable neighbourhood search (VNS) iterating in two stages, first using a population of solutions, and the second stage uses only a single solution. Their approach consists of two main steps, *shaking* and *local search.* The shaking phase improves the exploration of the search space, and the local search step looks for the local optima. A population of solutions is used in the shaking stage, where the authors argued that the diversity of solutions is important in the first stages of the search to explore the right search path, and after a period of time the best solution is picked from the population. Tournament selection is used to filter unfit solutions from the population. [171] introduced a hyper-heuristic that alternates between working on a single solution and a population of solutions. Their algorithm starts by scoring the available local search heuristics, and a serial phase working with single solutions starts by applying the heuristics sequentially according to their quality scores. A parallel phase uses a population of solutions, and a heuristic is applied to each individual in the population. The algorithm switches back to the serial phase whenever a global improvement is found (i.e., better than the best found solution so far).

3.5 Selection and Move Acceptance Methods

We will outline here the set of selection and move acceptance methods used in our study.

Most of the simple selection methods applied in this thesis were identified in [160]. These simple selection methods do not incorporate a learning mechanism, but rather choose a low level heuristic randomly from the set of low level heuristics or from a pre-determined permutation of the low level heuristics. Simple Random (SR) uses a uniform probability distribution to randomly select a low level heuristic at each step. Random Descent (RD) selects a low level heuristic randomly, and repeatedly applies it as long as it is making an improvement. Random Permutation (RP) forms an initial permutation of the low level heuristics and selects one at a time at each step. Random Permutation Descent (RPD) organises the low level heuristics in a similar way to RP but applies the selected heuristic repeatedly similar to RD if an improvement is made. The Greedy selection method (GR) applies all low level heuristics to a candidate solution, and chooses the heuristic that generates the most improved solution.

Move acceptance methods can be categorised as either *deterministic* or *non-deterministic*. Deterministic methods always return the same decision for any given set of input parameters, whilst non-deterministic methods (inspired by meta-heuristic methods) depend on the current time or step in making their decision. The deterministic methods applied throughout the thesis are: Only Improve (OI) which only accepts the improved solutions, and Improve or Equal (IE) which accepts non-worsening solutions. The non-deterministic move acceptance methods included are: Simulated Annealing (SA), Great Deluge (GD), Late Acceptance (LA), Record to Record (RR), and Naïve acceptance (Naïve).

Simulated annealing (see section 3.1.3.3) has been applied as a move acceptance component in selection hyper-heuristics by several studies [9, 172] and has proved to be successful. In [9], simulated annealing was used with the probability of accepting worsening moves given by the formula:

$$p_t = e^{-\frac{\Delta f}{\Delta F(1-\frac{t}{T})}} \tag{3.1}$$

where Δf is the change in the evaluation function at time t, T is the maximum time and ΔF is the range for the maximum change in the evaluation function.

The Great Deluge (GD) algorithm was first introduced by Dueck [173]. It is based on a stochastic framework that accepts all improved solutions by default. Non-worsening solutions are accepted if their objective value is equal to or better than a specific cost value called the 'level'. Initially the level is equal to the cost of the initial solution, and is updated afterwards at each step with the following formula:

$$\tau_t = f_0 + \Delta F \times (1 - \frac{t}{T}) \tag{3.2}$$

where ΔF is the maximum change in the objective value, f_0 is the final expected objective value, T is the time limit, and t is the time at the current step.

Late acceptance was first introduced in [174]. It compares the quality of the current solution with the solution generated L steps earlier during the search. This method requires the implementation of a circular queue of size L to save the objective values of L previously generated solutions. The performance of this method heavily depends on the queue size as stated in [174]. Similar to SA, in Naïve acceptance worsening solutions are accepted with a certain probability. The difference is that this probability is fixed for Naïve and is predefined by the user, while in SA, the probability varies in time and is calculated by the formula 3.2.

Record-to-Record (RR) is a variant of GD which accepts worsening solutions that are not much worse than the best solution in hand to an extent based on the following formula:

$$obj(S_{new}) \leq obj(s_{best}) + fr \times obj(S_{best}) \tag{3.3}$$

Where fr is a factor that is updated during the search, starting with a large value and gradually decreasing.

3.5.1 Sequence-based Selection Hyper-Heuristic

Generally, a selection method will choose a single heuristic and apply it to the current solution to generate a new solution. In this thesis, as one of our selection methods, we have utilised a scheme inspired by the hidden Markov model (HMM) [175] that applies sequences of heuristics to a given solution, where the low level heuristics represent the hidden states of the model. In this selection method each low level heuristic is associated with two probabilities: a probability to move to another low level heuristic including itself, and a probability to determine whether to terminate the sequence at this point. An outline of Sequence-based Selection Hyper-Heuristic (SSHH) is given in Algorithm 4.

Algorithm 4: Sequence-based selection hyper-heuristic

```
1  Let $S, S', S_b$ be candidate, new and best solutions, respectively;
2  Let $Tran$ be the transition matrix;
3  Let $Seq$ be the sequence construction matrix;
4  Let $[lh_0, llh_1, llh_2, \ldots, llh_{n-1}]$ be the low level heuristics set;
5  Let $HeuristicsSequence$ be the application sequence of low level heuristics;
6  $HeuristicsSequence \leftarrow [\ ]$;
7  $curr \leftarrow$ SelectRandomly$[0, 1, 2, \ldots, n-1]$;
8  $HeuristicsSequence$.add$(llh_{curr})$;
9  repeat
10     $next \leftarrow$ SelectNext$(Tran, curr)$;
11     $HeuristicsSequence$.add$(llh_{next})$;
12     $Status \leftarrow$ ComputeStatus$(Seq, next)$;
13     if $Status = end$ then
14         $S' \leftarrow$ Apply$(HeuristicsSequence, S)$;
15         if $S'$ isBetterThan $S_b$ then
16             $S_b \leftarrow S'$;
17             Update$(Tran, Seq)$;
18         end
19         $S \leftarrow$ Accept$(S, S')$;
20         $HeuristicsSequence$.clear$()$;
21     end
22     $curr \leftarrow next$;
23 until $TimeLimit$;
```

If the low level heuristics set is $[llh_0, llh_1, \ldots, llh_{n-1}]$, we define a transition matrix ($Tran = n \times n$) which specifies scores for each low level heuristic, from which we derive the probabilities of moving from one heuristic to another. We also define a sequence construction matrix ($Seq = n \times 2$) which stores scores for each of the n low level heuristics in two columns: *continue* and *end*. Following the addition of each low level heuristic to the sequence, the matrix Seq is used to compute the status of that sequence: either the sequence will end at this point and the low level heuristics within it will be applied to the current solution in the order in which they appear, or the sequence will continue, and the next low level heuristic will be selected. Initially every element in the two matrices is assigned the value '1', but these values are incremented to reward sequences of low level heuristics that are successful in improving the quality of the best solution so far.

At first, a random low level heuristic is selected (llh_{curr}) and added to the sequence (line 8). The next low level heuristic (llh_{next}) is chosen by selection procedure $SelectNext$ (line 10) based on the roulette wheel selection strategy with a probability equal to: $Tran[curr][next]/\sum_{\forall j} Tran[curr][j]$. The selected heuristic (llh_{next}) is then added to

	llh_0	llh_1	llh_2	llh_3
llh_0	1	1	1	1
llh_1	1	1	1	1
llh_2	1	1	1	1
llh_3	1	1	1	1

	con	end
llh_0	1	1
llh_1	1	1
llh_2	1	1
llh_3	1	1

(a) Matrices initial values

	llh_0	llh_1	llh_2	llh_3
llh_0	1	2	1	1
llh_1	1	1	1	2
llh_2	1	1	1	1
llh_3	1	1	1	1

	con	end
llh_0	2	1
llh_1	2	1
llh_2	1	1
llh_3	1	2

(b) Matrices updated values

FIGURE 3.3: Example of updating the values in the transition and sequence construction matrices: We assume the application of the sequence [llh_0, llh_1, llh_3] improved the best solution. The scores of these low-level heuristics in the "Transition Matrix" and the "Sequence Construction matrix" are updated. This update increases the probability of selecting this sequence in later steps.

the growing sequence of low level heuristics (line 11) and the status for this low level heuristic is computed with the procedure ($ComputeStatus$) (line 12) which determines whether or not the sequence will terminate at this point. The choice made here is also based on roulette wheel selection with the probability of continuing the sequence given by: $Seq[next][continue]/(Seq[next][continue] + Seq[next][end])$. If $Status = end$, the sequence is complete and will be applied to the current solution to generate a new solution (line 14). If the new solution is accepted and improved over the best solution, the scores in the matrices for the relevant low level heuristics are increased by one as a reward (line 17), increasing the chance of selecting the sequence that generates improved solutions. In addition, Seq is also updated by incrementing the *end* column for the final low level heuristic in the active sequence of low level heuristics and incrementing the *continue* columns for the non-terminal low level heuristics. For a more in depth description of the method with examples the reader can refer to [175, 176].

3.6 Hyper-heuristics in Routing Problems

Since the development of the hyper-heuristics framework, it has been utilised for solving various important combinatorial problems such as timetabling [9, 177], personnel scheduling [178] routing [179–181], Bin Packing [182], and Constraint Satisfaction [183]. VRPs are area in which hyper-heuristics proved to be particularly successful, and have been applied to solve different variants of VRPs (table 3.2). Pisinger and Ropke [179] used adaptive large neighborhood search (ALNS) hyper-heuristic, and achieved the state of the art results for multiple variants of the VRP. Garrido and Castro [184] presented

Table 3.2: Some selected routing problems in which hyper-heuristics were used as solution methodologies

Problem Domain	Reference
Ready-mix concrete delivery	[185]
Dynamic capacitated vehicle routing	[180]
Capacitated vehicle routing	[186]
Dial-a-ride with time window	[187]
Periodic vehicle routing	[188]
Vehicle routing with cross-docking	[189]
Inventory routing problem	[163]

a hill-climbing based hyper-heuristic to solve instances of the capacitated VRP, managing a set of perturbative-constructive pairs of low level heuristics and applying them sequentially. Their approach provided quality solutions compared to other methods in the literature. In their follow up work [180] the authors presented a self-adaptive hyper-heuristic capable of solving static and dynamic instances of the CVRP. They controlled a generic set of perturbative and constructive low level heuristics and designed a simple strategy based on reinforcement learning ideas to assign reward and penalty values and guide the operator's selection. They tested their approach on several benchmark instances and compared them to results obtained with previous hyper-heuristics and other well-known methods in the literature, and found that their approach provides high quality results with more adaptability to dynamic scenarios than other methods. Walker et al. [181] presented the CVRP with time windows as one of the problem domains in the HyFlex framework (Hyper-heuristic Flexible framework). They implemented data structures, and objectives for the evaluation of the problem, as well as a set of state of the art low level heuristics, and tested it using adaptive iterated local search hyper-heuristic. Their results showed the success of the adaption mechanism in improving the performance of hyper-heuristics.

The previous VeRoLog challenge 2016-2017 tackled a rich VRP problem related to a cattle improvement company that regularly measures the milk quality at a number of farms using specialised tools. These tools have to be delivered to a number of farms (customers) on request and picked up again a few days after delivery. The key challenge is how to schedule the deliveries to satisfy the requests, whilst at the same time design efficient routes for the pick-ups and deliveries. The second place winner on this challenge used a hyper-heuristic approach based on an online selection method [36].

In the UTRP, to the best of the author's knowledge, the use of hyper-heuristics is unexplored in the literature. Our reason for choosing it was driven by several motivations: (i) Hyper-heuristics are reasonably generic and are easy to implement and maintain. Thus we expect that our implementation can be applicable to other variants of the UTRP with minimal adaptation. (ii) The success of hyper-heuristics in solving several NP-hard optimisation problems generally and complex routing problems specifically (Table 3.2). (iii) The use of a single solution based framework can help solving the run-time issues of the problem. (iv) With the aid of appropriate low level heuristics, hyper-heuristics can handle complex solution spaces and therefore help in solving complex versions of the UTRP.

3.7 Methods for Solving Multi-objective Optimisation Problems

3.7.1 Evolutionary Algorithms

Evolutionary algorithms are a broad category of population-based meta-heuristics that have been widely accepted as a solution method for solving Multi Objective Optimisation Problems (MOOPs) because of their ability to produce multiple elements of the Pareto front in a single run. One of the most well-known and vastly applied evolutionary optimisation algorithms is Genetic Algorithms (GA) (section 3.1.3.1) which evolves through a number of iterations called generations, a population of initial candidate solutions called chromosomes each with a defined fitness value. As the search evolves, the population becomes fitter and eventually converges.

A GA is well suited for solving MOOPs because of its population-based nature. The generic single objective GA can be modified to produce multiple non-dominated solutions in a single run. One of the most important components of a multi-objective genetic algorithm is the ranking method. The ranking method uses the concepts of Pareto dominance to rank the solutions, and according to the dominance rules, the population is ranked and each solution is assigned a fitness value based on its rank in the population [190]. Also, maintaining diversity is an important consideration to ensure solutions are uniformly distributed over the Pareto front, and prevent the clustering of the solutions in specific regions which limits the exploration of the Pareto front.

There are various known multi-objective genetic algorithms implemented previously which are currently used in many applications. They differ in their fitness evaluation procedure, elitism, and diversification approaches. Some of these algorithms are: Multi Objective Genetic Algorithm (MOGA) [191], Strength Pareto Evolutionary Algorithm (SPEA) [192], and Fast Non-dominated Sorting Genetic Algorithm (NSGAI and NSGAII) [193].

3.7.2 The Weighted Sum Method

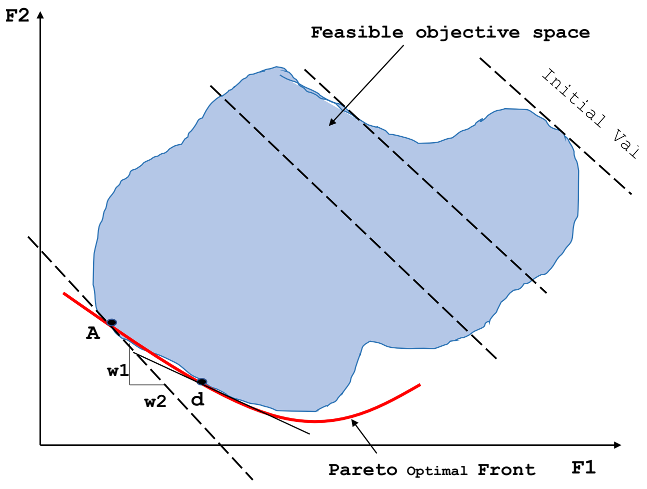
The weighted sum method casts the MOOP problem as a single objective optimisation problem. This is achieved by summing all the objective functions f_i, and weighting them using weighting coefficients w_i. Generally, the weighted sum approach can be described with the formula:

$$\sum_{i=1}^{k} w_i f_i(x) \tag{3.4}$$

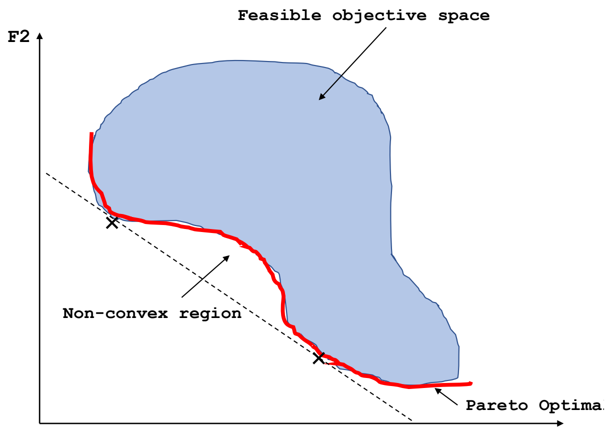
Where $w_i \geq 0$, and $\sum_{i=1}^{k} w_i = 1$. Ideally, the weight values in equation 3.4 are set by the decision maker based on their deep knowledge of the problem. However, as different objectives can have different magnitudes, the normalisation of the weights becomes essential in order to get a consistent Pareto optimal solution to the assigned weights [194]. In this case a single weight can be computed as $W_i = w_i \theta_i$ where w_i are the assigned weights and θ_i are the normalisation factors. Possible ways for normalising the weights can be:

- Normalise the weights using the initial value of the objective function such that: $W_i = \frac{w_i}{f(x_0)}$.
- Normalising using the minimum of the objective function: $W_i = \frac{w_i}{f(x_{min})}$ where x_{min} gives the minimum solution to the objective function f_i.

The weighted sum method is simple and straightforward to implement, with a key advantage of transforming an MOOP to a single solution optimisation problem, allowing the application of single point based optimisation methods to multi-objective problems. This approach is also computationally efficient. However, the application of this method is very sensitive to the weight adjustments and requires precise tuning by the decision maker and an intrinsic knowledge of the problem and its objectives in order to provide a good balance between the objectives through the weight coefficients. Another drawback



(a) Weighted sum approach in a minimisation problem



(b) Weighted sum in a non-convex Pareto front

FIGURE 3.4: Illustration of the weighted sum approach in a minimisation problem and in non-convex Pareto front

is that it requires a number applications for the single point-based optimiser in order to get a number of solutions, in contrast to evolutionary algorithms based methods which produce a full set of Pareto optimal solutions in a single optimisation run. Additionally, one of the identified problems in the weighted sum approach, is their inability to find any solutions laying in a non-convex region within the feasible solutions space (figure 3.4). Therefore, multi-objective frameworks based on the weighted sum approach have difficulties in finding solutions over a non-convex trade-off surface [190].

3.7.3 The Applied Weighted Sum Method

Throughout this thesis, we have adopted the simple weighted sum approach in handling the multi-objective nature of the problems tackled, and to create trade-off solutions as a part of the Pareto front. In the UTRP, our method is based on normalising the two objectives of the passenger and operator to ensure fairness and balance, as each objective represents a different measure. Different weight values are then used to find a spread of compromise solutions. These weight values were obtained by exhaustively trying several weights combinations and choosing the most successful. This approach suited the purpose of applying selection hyper-heuristics to solve the UTRP, as our objective was to find an efficient single point based computational method that can provide high quality route sets for a variety of instance sizes in a short computation time, and thus overcome the run time issues of GA methods. In the VeRoLog solver challenge problem, and according to the competition description and rules, a set of weights is provided with each instance to determine which objective is more important in that instance. In this case, the application of the approach was straightforward by using the supplied weights and no further tuning was required. Detailed description of our applied approach and its application will come in the following chapters.

3.8 Summary

This chapter covered the methods used for solving NP-hard COPs, classifying solution approaches and explaining why mathematical approaches sometimes fail to solve such computationally complex problems. It provides a comprehensive literature survey focused on meta-heuristic algorithms application to the UTRP and the success achieved so far. The second part of the survey focused on the general VRP problem with emphasis on the variants related to the VeRoLog 2019 solver challenge. Later, we outlined a full description of the hyper-heuristic framework describing its components, processes, and classification and showed its advantages over traditional heuristic methods and other meta-heuristic algorithms by being general and separated from any domain specific knowledge. We introduced the selection and move acceptance methods applied in our hyper-heuristic framework throughout the thesis with a detailed description of the SSHH online selection method. We also summarised the previous studies that applied selection hyper-heuristics in different complex routing problems. Finally, we briefly outlined two approaches for solving Multi-objective Optimisation Problems (MOOPs):

a mutli-objective approach based on evolutionary algorithms, and the weighted sum approach, and described our approach for handling the multi-objective problems addressed in this thesis.

Chapter 4

Hyper-heuristics for Urban Transit Route Design Problem

In this chapter, we demonstrate the application of the selection hyper-heuristics framework to solve the UTRP problem. Our aim is to develop a computationally efficient algorithm that works well on instances with different characteristics and sizes while providing high quality solutions from the perspective of passenger and operator in reasonable run times. We demonstrate here that selection hyper-heuristics are the potential way forward for solving the run time problems associated with the GA methods, and that the iterative improvement of a single solution can yield to high quality results. We will describe the application of thirty different selection hyper-heuristics on Mandl's benchmark instance and Mumford's dataset and compare between their performances. As stated in Bilgin et al. [9], different combinations of selection and move acceptance components can yield to different performances. We conduct these comparisons using statistical analysis methods to find the best combination of selection and move acceptance components on the UTRP. Moreover, we compare our results with the current state-of-the-art solutions on the applied set of instances and show the superiority of our results.

4.1 Problem Model

In chapter 2 (section 2.4.3), we demonstrated a simplified model for the UTRP based on a weighted undirected graph. The same model was adopted by other previous studies

[6, 82, 83]. We apply this model to test our selection hyper-heuristics using the objective functions described by equation 2.3, and equation 2.4 respectively to calculate the passenger and operator objectives. The two objective functions are subject to a set of constraints that defines the feasibility criteria of the route network. These constraints were also listed in section 2.4.3. The calculation of the passenger cost objective is based on expanding the route network into the transit network, and finding the shortest paths between all origin-destination points in the transit network. For the operator cost, we considered the costs of travelling all the routes in the route network in one direction. The fleet size is also an important consideration for the operator. We will not include it as an objective for this current study, but will describe a simple way to calculate the total fleet size required to cover the entire network demand. This could be simply done by dividing the total length of the routes (multiplied by 2 to represent travelling in the opposite direction) by the route headway (equals 10 considering an average waiting time of five minutes).

The calculation of the passenger objective is the most time consuming part of the algorithm requiring the expansion of the route network into the transit network in which the number of nodes increases by an order of magnitude to represent passengers transfers, and then applying an all pairs shortest paths algorithm to each candidate route set. We used an implementation of Dijkstra algorithm based on a priority queue [195] with an associated run time complexity of $O(n^2)$, and compared its performance with the Floyd Warshall algorithm [196] that has a run time complexity of $O(n^3)$. Through this comparison, we found that the priority queue used in Dijkstra implementation reduces the computational time required by the passenger cost. Floyd Warshall's algorithm was applied to calculate the passenger cost in [6, 83], and we believe it was a cause of the long run times in the large instances as has been reported in these studies.

For the scope of this work, we will assume that there are sufficient vehicles traversing each route, and with enough capacity to cover the demand between all the bus stops. The frequency of the routes though can be calculated using a demand assignment procedure [86] to determine high demand routes and transfer points and a simple heuristic procedure [80] to redistribute the total available fleet (calculated as mentioned in this section) between the routes, allowing larger number of buses to be assigned to busier routes. We will also assume that the time cost of waiting at transfer points and the penalty of making a transfer are combined in a single cost that equals 5 minutes (assumption in line with previous studies [3, 6, 31, 75, 82]).

For assessing the generated route sets, we use the average passenger and operator costs as performance indicators. We also use transfer statistics for a more comprehensive assessment of the route set (as described in section 2.4.3). These statistics calculate the percentage of direct travellers, travellers that requires one, two transfers, and unsatisfied demand (i.e. travellers requiring more than three transfers).

4.2 Hyper-heuristics Design and Solution Initialisation

4.2.1 Evaluation Method

The solution to our model is the route network, which consists of a number of fully connected routes, that allows a passenger to travel between any source-destination pair. We represent our solution S as a two dimensional vector. A candidate solution is evaluated using the following equation:

$$f(S) = \alpha F(S) + \beta C_p(S) + \gamma C_o(S) \tag{4.1}$$

α, β and γ are constants used to weight the three components of the objective function, where β and γ are positive values between $(0 - 1)$, and α is a positive value between $(0 - \infty)$. $C_p(S)$ and $C_o(S)$ are the passenger and operator objectives of solution S calculated using Equations 2.3 and 2.4 respectively; and $F(S)$ represents the feasibility of the solution S and it computes to what extent this solution meets the problem constraints. If any of these constraints is violated at any position in the route set, it is penalised by increasing its value by one. For example for each missing node in the route set, the constraint concerned with including all the vertices is increased by one. Similarly, for the other constraints with each individual violation. F is then calculated as the sum of these constraint violations. A route set is only accepted if this sum is zero (i.e. the solution is feasible). Note that F, C_p and C_o can be treated as separate objectives in a multi-objective formulation or combined into a single objective as in Equation 4.1. In the present work the objectives are considered separately in the main, simply to facilitate comparisons with published results produced by other state-of-the-art methods. However it is clear that compromise solutions between objectives will be required in practice, and for this reason we include a brief illustration of how our hyper-heuristic approach can be extended to produce such a set of solutions in Section 4.3.4.

4.2.2 Initial Solutions

Our implemented initial route set generation procedure focuses on the feasibility of the solution. The aim is to construct a single initial solution that obeys all feasibility constraints. The process consists of two stages: construction and repair. In the construction stage, a pool of candidate routes is generated by applying Dijkstra algorithm and finding the shortest travel time path between every pair of nodes in the network. Only the shortest paths that obey the user constraints for maximum and minimum route length are included in the pool. The route set is then built by selecting one route at a time from the pool, until the route set reaches the required size as pre-determined by the user. The first route is selected by trying all of the routes from the pool in turn, and choosing "the best" according to Equation 4.1, with parameters β, and γ set to zero, and α is set to one. The initial generation procedure is focusing solely on building a feasible route set, and not on its quality. Thus the candidate route that produces the minimum number of feasibility violations is chosen as the first route. The second route is then selected from the pool in a similar way, this time applying Equation 4.1 to the growing route set consisting of the first route, and the candidates for the second route, and once again the procedure makes "the best choice" for the route set according to the equation. The selection procedure continues in a similar way for every route in the route set, until the initial route set contains the required number of routes. Although the construction stage reduces the feasibility constraint violations to a minimum, still it does not guarantee a feasible route set is obtained. For this reason, a repair stage is applied when needed. This stage consists of a simple hyper-heuristic combining *simple random* selection and *improve or equal* acceptance. This hyper-heuristic evaluates the route set with the same parameter settings as the construction stage and terminates immediately after finding a feasible solution. The initial solution generation procedure is demonstrated in algorithm 5.

4.2.3 Hyper-heuristics

The constructed initial solution is introduced as the current solution (S) to the hyper-heuristic framework. The selection method applies a single heuristic, or a sequence of heuristics (in the case of a sequence based selection method) to S, generating a new solution S', which is evaluated using Equation 4.1 with the following parameters settings: α is set to ∞ to ensure non-feasible solutions are always rejected, β and γ are set based on whether the solution is being evaluated from the passenger or the operator perspective.

Algorithm 5: Initial Solution Construction

```
1  Let $\mathbf{Paths}$ be the pool of shortest paths.;
2  Let $\mathbf{route_c}, \mathbf{path_b}$ be a route under construction, and the best path respectively.;
3  Let $\mathbf{noRoutes}, \mathbf{noPaths}$ be the number of routes in the route set and the number of
     paths in the pool of shortest paths respectively.;
4  Let $\mathbf{S}, \mathbf{S'}, \mathbf{S_b}$ represent the current, new, and best solution respectively.;
5  Let $\mathbf{obj}, \mathbf{obj'}, \mathbf{Obj_b}$ be the current, new and best objective respectively.;
6  Construction Stage;
7  $Paths \leftarrow$ CalculateDijkstra($n$);
8  $obj_b = \infty$;
9  for $i \leftarrow 1$ to $noRoutes$ do
10     for $j \leftarrow 1$ to $noPaths$ do
11         $route_{c_i} \leftarrow Paths_j$ ;
12         $Obj \leftarrow$ EvaluateSolution($S$);
13         if $Obj$ isBetterThan $Obj_b$ then
14             $Obj_b \leftarrow Obj$;
15             $path_b \leftarrow Paths_j$
16         end
17     end
18     $route_{c_i} \leftarrow path_b$ ;
19 end
20 Repair Stage ;
21 $S_b \leftarrow S$;
22 repeat
23     $llh \leftarrow$ RandomlySelect($LLH$);
24     $S' \leftarrow apply(llh, S)$ ;
25     if $Accept(S, S')$ then
26         $S \leftarrow S'$
27     end
28     $S_b \leftarrow$ UpdateBestSolution($S$)
29 until EvaluateSolution($S_b$) = 0;
30 return $S_b$
```

$\beta = 1$ and $\gamma = 10^{-10}$ for the passenger perspective, and vice versa for the operator. The low level of 10^{-10} provides an effective tie-breaker. Following evaluation the acceptance method is applied, which determines whether S' will replace S or not. In this work we are evaluating the performance of several selection hyper-heuristics made up by alternative pairings between various selection and acceptance methods. More specifically we pair the following selection methods: Simple Random (SR), Random Descent (RD), Random Permutation (RP), Random Permutation Descent (RPD), Greedy Selection (GR), Sequence-based Selection (SS); with the move acceptance methods: Only Improve (OI), Improve or Equal (IE), Late Acceptance (LA), Great Deluge (GD), Simulated Annealing

(SA).

We conducted the parameter tuning manually by performing a series of quick experiments on Mandl and Mumford3 instances (i.e. the smallest and the largest instance in our tested dataset). We run each experiment for five minutes on the selected instances, each experiment with different parameters values. From these series of experiments, the parameter values that give the best result were selected. According to this, in Great Deluge (GD) and Simulated Annealing (SA), the best known objective for calculating the maximum change in the objective function was set to zero. In Late Acceptance (LA), we tested different memory sizes for the circular queue L ranging between 10 to 100. Finally, we have set this value to 40 which provided the best passenger and operator costs for both instances.

4.2.4 Low Level Heuristics

The hyper-heuristic controls a set of seven low level heuristics to improve the quality of a given route set. All the low level heuristics are mutational, and they perform basic operations to mutate a given route set.(see Figure 4.1).

- **LLH0**: Selects a random route and a random position in this route and adds a random node into this position.
- **LLH1**: Selects a random route and a random node in this route and deletes this node.
- **LLH2**: Selects a random route and two random nodes in this route and swaps the two nodes.
- **LLH3**: Selects a random route a random node and a random position in this route. The selected node is inserted into that position.
- **LLH4**: Selects a random route and a random node and replaces this node with another random node.
- **LLH5**: Selects two random routes a random node and a random position. The node on the first route is inserted into the second position on the second route.
- **LLH6**: Selects two random routes and a random node on each route and swaps the two nodes.

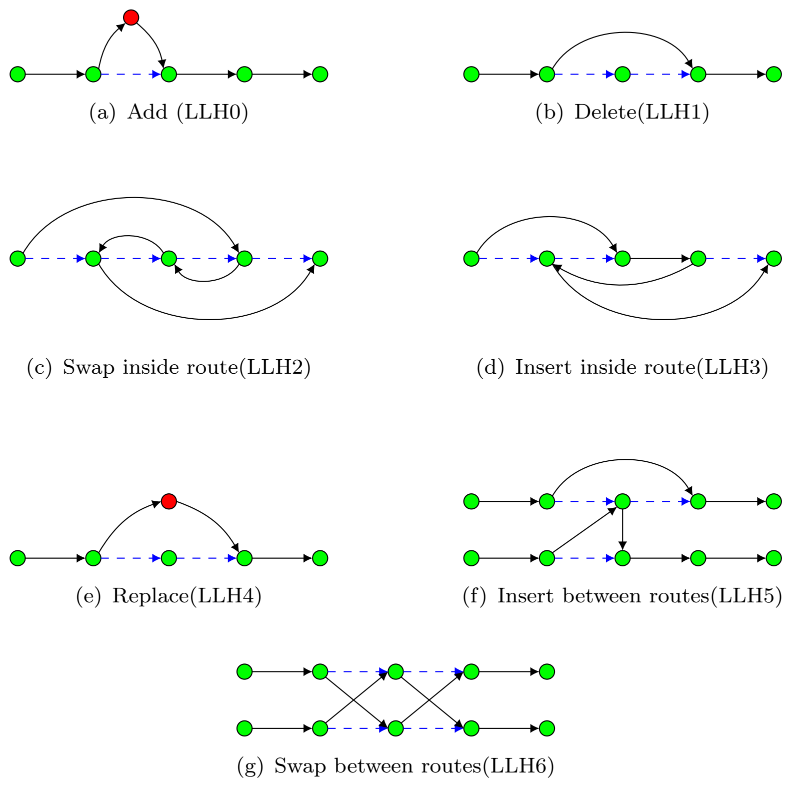

(a) Add (LLH0)
(b) Delete(LLH1)
(c) Swap inside route(LLH2)
(d) Insert inside route(LLH3)
(e) Replace(LLH4)
(f) Insert between routes(LLH5)
(g) Swap between routes(LLH6)

FIGURE 4.1: Low level heuristics set description. Straight arcs are edges in the route, dashed arcs are edges removed after applying the low level heuristic, curved arcs are edges added after applying the low level heuristic

4.2.5 Problem Instances

In this study, we have used Mandl's benchmark instance [73], and the four benchmark instances published in [6]. Mandl benchmark instance is considered the defacto benchmark for solving the UTRP, despite its size and layout which does not represent a real world transportation network. Mandl's instance contains only 15 nodes, which is very small compared to real-world networks that can contain hundreds or thousands of nodes. Until recently, this was the only publicly available instance that comes with all the related information of the network layout, demand data, and travel times.

In [6], a new data set of four instances was published to aid researchers to test their methods on fairly large size instances. These instances were generated using user defined parameters for determining the network vertices, edges, and the lower and upper bound on the demand. The number of routes and the connectivity of Mumford data set is loosely based on bus network maps of real cities: one in China (Yubei), and two in the UK (Cardiff, and Brighton). This benchmark set provides a variety of network sizes which is necessary to assess the scalability of an algorithm, and therefore it has been a

Table 4.1: Features of our dataset

Instance	Number of vertices, edges	Number of routes	Vertices per route (min, max)	Average transit network size
Mandl4	15, 21	4	2, 8	4*(2+8)/2 = 20
Mandl6	15, 21	6	2, 8	30
Mandl7	15, 21	7	2, 8	35
Mandl8	15, 21	8	2, 8	40
Mumford0	30, 90	12	2, 15	102
Mumford1	70, 210	15	10, 30	300
Mumford2	110, 385	56	10, 22	896
Mumford3	127, 425	60	12, 25	1110

subject for application by many studies due to the practical size of its instances. Some example studies that tested this data set can be found in [31, 83, 85, 87]. The details of the generation procedure of Mumford data set are published in the doctoral thesis of Fan [28].

The above described instances have symmetrical travel time and demand data, and therefore fits our problem model and assumptions. The features of our data sets are provided in Table 4.1. We have used Mandl's benchmark with route set sizes 4, 6, 7, and 8 considering each a separate problem. These variants are commonly used in the literature [6, 82].

4.3 Experimental Results

We carried out our experiments in two phases. The first round of experiments evaluates the solution from the passenger perspective. This phase focuses on generating route sets that serve the passenger needs, by providing the best possible travel time from origin to destination with the lowest number of transfers. In these experiments Equation 4.1 is used to evaluate the solution with the following parameter setting: $\alpha = \infty$, $\beta = 1$ and $\gamma = 10^{-10}$. In the second round of experiments the solution is evaluated from the operator perspective to generate route sets focusing on the requirements of the operator using the settings: $\alpha = \infty$, $\beta = 0$ and $\gamma = 1$ for 80% of the run time, and for the rest of the search time β is set to 10^{-10}. The reason for this is to allow the hyper-heuristic to focus on improving the routes based on the operator qualities, instead of wasting the search time with the complex and time consuming operations related to the calculation of the passenger objective.

The experiments were conducted on a device with the following specifications: Intel Core i5 at 2.30GHz with memory of 8GB. Each selection hyper-heuristic is run for ten trials on each instance and terminates after the run time elapses. The run time is set to vary according to the instance size by adding thirty seconds for each node. According to these settings, the largest instance in the set, Mumford3 will require one hour on average per trial and ten hours for the ten trials per selection hyper-heuristic. Since we are testing thirty selection hyper-heuristics, performing ten trials was reasonable giving the time required for the single trial and the number of selection hyper-heuristics that we are testing.

4.3.1 Passenger Perspective

Tables 4.2 and 4.3 show the results from the passenger prescriptive experiments in terms of the average travel time for a single passenger (measured in minutes) for all the selection hyper-heuristics averaged over the ten trials. The minimum and the maximum values have also been recorded. The Kruskal-Wallis stastical test is performed with 95% confidence level to compare the pairwise statistical variations in the performance between two algorithms. The following notations are used: Given two algorithms X versus Y, $>$ ($<$) denotes that $X(Y)$ performs better than $Y(X)$, and this variation is statistically significant, $\geq$ ($\leq$) denotes that $X(Y)$ performs slightly better than $Y(X)$, but the performance is not statistically significant, and $=$ denotes that X and Y perform equally. The values associated with these notations in the tables represent the number of times a particular hyper-heuristic is statistically significant, not statistically significant, or equally performing against the other selection hyper-heuristics in the tested set. The average number of iterations is also reported using the following notations: m refers to the number of iterations in millions, and k refers to the number in thousands.

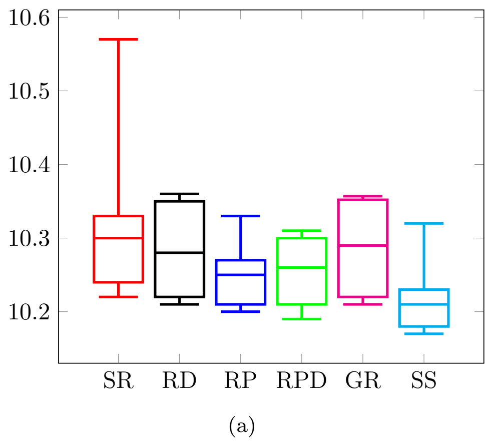
The success of the sequence-based selection method (SS) can be observed from the results, outperforming other selection methods regardless of the move acceptance. This observation applies for all instances. Figure 4.2 shows the performance variation of our applied selection methods when combined with GD acceptance giving an advantage for SS. With regard to the move acceptances, the non-deterministic acceptance methods were more successful. Simulated annealing achieved the best results in all Mandl variants, delivering an improved performance that is statistically significant. Great deluge found the best minimum results in all Mandl instances. Note that there is no proof of optimally for any of the recorded results in Mandl variants. The only observation is

TABLE 4.2: Results of the thirty selection hyper-heuristics from the passenger perspective for Mandl instances. Best values per each instance are highlighted in bold

		MANDL4								MANDL6								MANDL7								MANDL8							
MA	SM	avg	std	min	>	<	≥	≤	Iter	avg	std	min	>	<	≥	≤	Iter	avg	std	min	>	<	≥	≤	Iter	avg	std	min	>	<	≥	≤	Iter
OI	SR	10.891	0.273	10.716	0	12	0	17	39m	10.510	0.140	10.309	0	6	5	18	17m	10.480	0.113	10.316	0	12	3	14	13m	10.360	0.114	10.190	0	7	5	17	10m
	RD	10.831	0.270	10.523	0	6	1	21	40m	10.555	0.207	10.322	0	7	2	20	18m	10.491	0.106	10.344	0	13	2	14	13m	10.407	0.070	10.325	0	16	2	11	11m
	RP	10.736	0.247	10.515	0	6	6	17	37m	10.457	0.111	10.294	0	6	9	14	16m	10.472	0.107	10.328	0	12	4	13	14m	10.457	0.100	10.272	0	13	0	16	11m
	RPD	10.718	0.152	10.515	0	6	10	13	380m	10.645	0.260	10.438	0	12	0	17	18m	10.500	0.323	10.278	0	9	1	19	13m	10.417	0.118	10.242	0	8	1	20	10m
	GR	10.778	0.208	10.611	0	7	5	17	5m	10.513	0.104	10.369	0	11	4	14	2m	10.504	0.089	10.346	0	13	0	16	1m	10.383	0.125	10.227	0	8	3	18	1m
	SS	10.606	0.065	10.510	0	6	18	4	57m	10.346	0.109	10.190	0	**0**	15	14	28m	10.264	0.074	10.185	0	6	15	8	23m	10.194	0.086	10.094	0	**0**	15	14	16m
IE	SR	10.821	0.231	10.641	0	8	3	18	39m	10.558	0.105	10.383	0	11	1	17	17m	10.371	0.121	10.206	0	7	10	12	13m	10.278	0.073	10.166	0	7	9	13	11m
	RD	10.831	0.270	10.523	0	6	1	21	40m	10.501	0.131	10.336	0	9	6	14	17m	10.392	0.127	10.246	0	8	7	14	13m	10.278	0.054	10.157	0	7	8	14	10m
	RP	10.735	0.247	10.515	0	6	7	16	36m	10.473	0.139	10.312	0	6	7	16	18m	10.426	0.116	10.209	0	7	6	16	13m	10.278	0.113	10.167	0	7	7	15	10m
	RPD	10.721	0.158	10.515	0	6	9	14	37m	10.417	0.077	10.327	0	8	12	9	16m	10.453	0.112	10.306	0	11	5	13	13m	10.363	0.094	10.194	0	7	4	18	11m
	GR	10.784	0.208	10.611	0	7	4	18	5m	10.517	0.110	10.349	0	9	3	17	2m	10.387	0.094	10.245	0	8	8	13	1m	10.283	0.101	10.128	0	3	6	20	1m
	SS	10.606	0.065	10.510	0	6	18	4	58m	10.314	0.055	10.216	1	**0**	15	13	28m	10.234	0.065	10.168	2	3	15	9	22m	10.168	0.064	10.102	1	1	16	11	16m
LA	SR	10.665	0.065	10.523	0	6	14	9	38m	10.444	0.148	10.241	0	6	10	13	19m	10.281	0.103	10.130	0	**0**	13	16	13m	10.212	0.095	10.112	0	1	12	16	10m
	RD	10.725	0.099	10.619	0	7	8	14	39m	10.391	0.106	10.244	0	6	13	10	18m	10.352	0.117	10.229	0	7	11	11	14m	10.200	0.074	10.106	1	1	13	14	10m
	RP	10.708	0.081	10.617	0	7	11	11	40m	10.464	0.100	10.239	0	6	8	15	17m	10.291	0.117	10.177	0	5	12	12	14m	10.261	0.089	10.171	0	7	11	11	11m
	RPD	10.683	0.084	10.608	0	6	12	11	39m	10.439	0.137	10.248	0	6	11	12	18m	10.376	0.121	10.137	0	**0**	9	20	14m	10.275	0.127	10.164	0	7	10	12	11m
	GR	10.672	0.119	10.572	0	6	13	10	5m	10.347	0.079	10.240	0	6	14	9	2m	10.265	0.070	10.195	0	7	14	8	2m	10.204	0.076	10.109	1	1	12	15	2m
	SS	10.608	0.071	10.510	0	6	17	6	62m	10.272	0.082	10.184	0	**0**	20	9	27m	10.188	0.048	10.114	8	**0**	14	7	21m	10.155	0.052	10.090	2	**0**	**20**	7	19m
GD	SR	10.604	0.050	10.533	1	6	19	3	46m	10.302	0.103	10.229	0	4	17	8	23m	10.191	0.046	10.127	6	**0**	15	8	17m	10.158	0.045	10.105	2	1	17	9	14m
	RD	10.597	0.074	10.500	1	6	**20**	2	45m	10.280	0.064	10.216	3	**0**	16	10	23m	10.227	0.071	10.146	0	1	**19**	9	17m	10.157	0.041	10.123	2	2	18	7	14m
	RP	10.587	0.044	10.489	2	1	**20**	6	46m	10.256	0.043	10.208	5	**0**	17	7	22m	10.235	0.054	10.137	4	**0**	12	13	18m	10.162	0.049	10.113	2	2	16	9	14m
	RPD	10.627	0.049	10.560	1	6	14	8	47m	10.260	0.042	10.194	7	**0**	14	8	23m	10.227	0.037	10.182	5	6	15	3	17m	10.155	0.033	10.097	4	**0**	17	8	14m
	GR	10.623	0.048	10.518	1	6	15	7	6m	10.291	0.051	10.211	3	**0**	15	11	3m	10.228	0.029	10.192	5	7	13	4	2m	10.174	0.047	10.114	2	2	14	11	1m
	SS	10.521	0.050	**10.482**	6	**0**	17	6	72m	10.212	0.043	**10.180**	6	**0**	**21**	2	33m	10.135	0.034	**10.101**	13	**0**	14	2	24m	10.098	0.023	**10.069**	11	**0**	17	1	19m
SA	SR	**10.484**	0.005	**10.482**	**23**	**0**	6	**0**	38m	10.222	0.005	10.213	**16**	**0**	7	6	18m	10.154	0.013	10.133	16	**0**	8	5	14m	10.113	0.007	10.099	12	**0**	14	3	11m
	RD	10.486	0.005	**10.482**	22	**0**	6	1	39m	10.220	0.009	10.200	15	**0**	9	5	18m	10.151	0.016	10.128	16	**0**	9	4	14m	10.121	0.008	10.106	11	1	12	5	11m
	RP	10.488	0.007	**10.482**	22	**0**	4	3	39m	10.218	0.009	10.196	**16**	**0**	9	4	18m	10.154	0.014	10.139	15	1	8	5	14m	10.118	0.011	10.103	11	1	13	4	11m
	RPD	10.487	0.006	**10.482**	22	**0**	5	2	39m	10.216	0.010	10.196	15	**0**	11	3	18m	10.149	0.010	10.136	17	**0**	9	3	14m	10.117	0.009	10.106	11	1	14	3	11m
	GR	10.489	0.006	**10.482**	22	**0**	3	4	5m	10.207	0.008	10.191	**16**	**0**	12	1	2m	10.134	0.011	10.117	17	**0**	11	1	2m	10.105	0.006	10.096	15	**0**	12	2	1m
	SS	10.489	0.007	**10.482**	22	**0**	2	5	67m	**10.202**	0.009	10.190	**16**	**0**	13	**0**	26m	**10.122**	0.008	10.110	**19**	**0**	10	**0**	21m	**10.093**	0.005	10.084	**23**	**0**	6	**0**	15m

TABLE 4.3: Results of the thirty selection hyper-heuristics from the passenger perspective for Mumford instances. Best values per each instance are highlighted in bold

		MUMFORD0								MUMFORD1								MUMFORD2								MUMFORD3							
MA	SM	avg	std	min	>	<	≥	≤	Iter	avg	std	min	>	<	≥	≤	Iter	avg	std	min	>	<	≥	≤	Iter	avg	std	min	>	<	≥	≤	Iter
	SR	14.611	0.087	14.496	0	15	3	11	2m	22.302	0.160	22.086	0	**0**	6	23	1m	27.241	0.101	27.103	0	22	0	7	440k	30.254	0.115	30.105	0	13	2	14	385k
	RD	14.621	0.080	14.488	0	15	2	12	2m	22.294	0.116	22.100	0	**0**	9	20	1m	27.161	0.100	26.982	0	19	3	7	438k	30.280	0.071	30.121	0	13	0	16	387k
OI	RP	14.563	0.113	14.380	0	3	5	21	2m	22.314	0.153	22.085	0	**0**	3	26	1m	27.166	0.113	26.999	0	19	2	8	448k	30.245	0.091	30.138	0	16	3	10	386k
	RPD	14.702	0.145	14.501	0	16	0	13	2m	22.308	0.164	22.113	0	**0**	5	24	1m	27.235	0.112	27.070	0	21	1	7	445k	30.204	0.053	30.149	0	17	4	8	368k
	GR	14.606	0.137	14.352	0	2	4	23	300k	22.372	0.086	22.253	0	2	2	25	243k	27.135	0.199	26.859	0	17	4	8	63k	30.197	0.106	30.053	0	10	6	13	55k
	SS	14.669	0.136	14.467	0	14	1	14	2m	22.580	0.179	22.280	0	3	0	26	2m	26.915	0.150	26.705	0	9	8	12	254k	29.732	0.095	29.609	8	2	15	4	242k
	SR	14.409	0.071	14.287	0	**0**	7	22	2m	22.241	0.158	22.021	0	**0**	15	14	1m	26.654	0.135	26.473	4	3	11	11	522k	29.963	0.084	29.836	2	5	14	8	470k
	RD	14.449	0.150	14.322	0	**0**	6	23	2m	22.207	0.104	22.097	0	**0**	19	10	1m	26.692	0.113	26.488	5	3	9	12	519k	29.964	0.118	29.834	0	5	14	10	475k
IE	RP	14.403	0.081	14.310	0	**0**	9	20	2m	22.282	0.100	22.169	0	**0**	10	19	1m	26.705	0.109	26.546	4	4	7	14	514k	29.945	0.103	29.848	2	5	17	5	474k
	RPD	14.362	0.073	14.275	0	**0**	12	17	2m	22.224	0.152	22.051	0	**0**	17	12	1m	26.695	0.094	26.560	5	4	7	13	520k	29.982	0.089	29.803	5	4	7	13	466k
	GR	14.407	0.099	14.281	0	**0**	8	21	310k	22.234	0.100	22.118	0	**0**	16	13	250k	26.694	0.112	26.532	5	4	8	12	3k	29.964	0.146	29.730	1	4	14	10	3k
	SS	14.395	0.119	14.234	0	**0**	10	19	2m	22.399	0.138	22.205	0	**0**	1	28	1m	26.258	0.134	26.120	19	1	8	1	305k	29.399	0.165	29.087	21	1	6	1	292k
	SR	14.285	0.110	**14.118**	4	**0**	20	5	2m	22.095	0.071	22.000	**2**	**0**	26	1	2m	26.382	0.089	26.262	11	2	14	2	814k	29.747	0.078	29.626	9	2	12	6	730k
	RD	14.312	0.088	14.138	4	**0**	15	10	2m	**22.075**	0.096	21.955	**2**	**0**	**27**	**0**	2m	26.442	0.091	26.259	11	2	11	5	818k	29.701	0.100	29.549	14	1	11	3	725k
LA	RP	14.301	0.106	14.160	0	**0**	23	6	2m	22.180	0.102	22.012	0	**0**	24	5	2m	26.402	0.065	26.268	14	2	10	3	832k	29.745	0.107	29.528	9	1	13	6	721k
	RPD	14.271	0.061	14.178	5	**0**	20	4	2m	22.124	0.095	22.007	1	**0**	25	3	2m	26.428	0.090	26.307	11	2	12	4	819k	29.731	0.083	29.580	9	2	15	3	722k
	GR	**14.234**	0.066	14.158	4	**0**	**25**	**0**	351k	22.179	0.093	22.083	0	**0**	25	4	450k	26.593	0.088	26.483	8	3	9	9	168k	29.904	0.079	29.806	6	4	14	5	3k
	SS	14.346	0.102	14.198	1	**0**	15	13	2m	22.120	0.119	**21.906**	0	**0**	**27**	2	2m	**25.661**	0.073	**25.523**	**29**	**0**	0	**0**	589k	**28.811**	0.089	**28.710**	**29**	**0**	0	**0**	598k
	SR	14.312	0.100	14.176	3	**0**	17	9	5m	22.296	0.151	22.013	0	**0**	8	21	3m	26.529	0.126	26.374	8	2	13	6	651k	29.960	0.090	29.762	5	4	12	8	511k
	RD	14.347	0.096	14.168	4	**0**	11	14	5m	22.271	0.117	22.036	0	**0**	12	17	3m	26.588	0.072	26.475	10	3	8	8	640k	29.977	0.083	29.795	5	4	8	12	501k
GD	RP	14.319	0.090	14.219	0	**0**	18	11	5m	22.195	0.126	22.053	0	**0**	23	6	3m	26.553	0.129	26.417	8	2	12	7	640k	29.958	0.137	29.800	0	4	**18**	7	498k
	RPD	14.302	0.084	14.205	4	**0**	18	7	5m	22.197	0.092	22.089	0	**0**	22	7	3m	26.569	0.140	26.315	5	2	14	8	641k	29.997	0.092	29.808	2	4	9	14	504k
	GR	14.310	0.070	14.229	4	**0**	17	8	604k	22.255	0.089	22.132	0	**0**	14	15	505k	26.618	0.125	26.468	5	3	11	10	88k	30.019	0.114	29.848	0	5	10	14	72k
	SS	14.270	0.070	14.123	**6**	**0**	20	3	5m	22.216	0.082	22.076	0	**0**	18	11	3m	26.067	0.130	25.888	26	1	2	**0**	524k	29.330	0.122	29.117	24	1	4	**0**	365k
	SR	14.352	0.031	14.300	4	**0**	10	15	2m	22.311	0.108	22.176	0	**0**	4	25	2m	26.938	0.158	26.703	0	9	5	15	627k	30.272	0.091	30.109	0	13	1	15	576k
	RD	14.373	0.036	14.319	4	**0**	7	18	2m	22.265	0.062	22.157	0	**0**	13	16	2m	26.910	0.101	26.773	2	12	7	8	625k	30.202	0.184	29.988	0	9	5	15	565k
SA	RP	14.355	0.044	14.299	4	**0**	9	16	2m	22.276	0.074	22.187	0	**0**	11	18	2m	26.897	0.145	26.590	1	8	9	11	648k	30.083	0.158	29.832	0	5	9	15	575k
	RPD	14.345	0.043	14.291	4	**0**	13	12	2m	22.298	0.090	22.188	0	**0**	7	22	2m	26.931	0.131	26.777	0	12	7	10	636k	30.185	0.202	29.910	0	8	7	14	557k
	GR	14.257	0.071	14.139	4	**0**	23	2	361k	22.201	0.105	22.026	0	**0**	21	8	348k	26.937	0.081	26.793	2	12	4	11	85k	30.159	0.111	29.973	0	9	8	12	77k
	SS	14.250	0.052	14.137	**6**	**0**	22	1	2m	22.201	0.084	22.072	0	**0**	20	9	2m	26.344	0.112	26.173	11	1	**15**	2	346k	29.499	0.148	29.246	21	1	5	2	327k



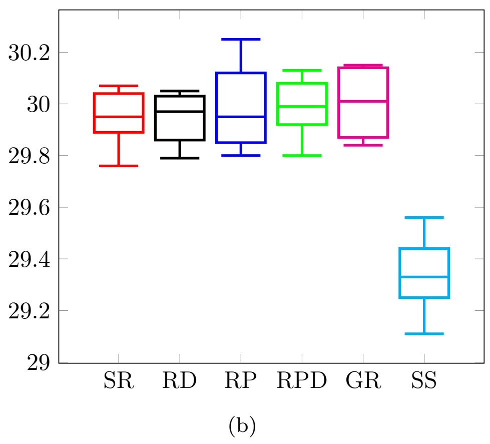


FIGURE 4.2: Box plots from 10 runs for all selection methods combined with GD acceptance method for (a) Mandl6 instance, and (b) Mumford3 instance. Values in Y axis show the average travel time and the lower boxes represent the best selection methods

MANDL4, where all the selection methods combined with SA found a best minimum of 10.482, and the same value was the best minimum found by SS combined with GD. This implies that it could possibly be an optimal solution. In the larger instances late acceptance and great deluge were the most successful. However LA found slightly better averages and minimum results compared to GD. Given these observations, we carried out the next round of experiments from the operator perspective using three selection hyper-heuristics (SS-SA, SS-GD, SS-LA) which combined the best selection method with the most successful move acceptance methods.

4.3.2 Operator Perspective

Table 4.4 summarises the results of the experiments from the operator perspective using the average objective function value over the ten trials, the minimum values, and the pair-wise statistical performance. Comparing these results with the lower bounds for the operator cost published in [6], all three selection hyper-heuristics succeeded in finding the lower bound in the Mandl problem variants and the Mumford0 instance on each of the ten trials. Referring to the table, GD is the most successful, scoring better averages and minimum values with a statistically significant performance compared to SA and LA.

Giving our previous observations from the passenger perspective trials, we notice that GD performed consistently well in all the instances on the data set, in contrast to LA

Table 4.4: Results of the three best selection hyper-heuristics from the operator perspective. Best averages and minimum values per instance are highlighted in bold

	SS-SA					SS-GD					SS-LA			
Instance	avg	std	min	Iter	v.s	avg	std	min	Iter	v.s	avg	std	min	Iter
Mandl(4,6,7,8)	**63.0**	0.00	**63**	350m	=	**63.0**	0.00	**63**	350m	=	**63.0**	0.00	**63**	350m
Mumford0	**94.0**	0.00	**94**	424m	=	**94.0**	0.00	**94**	417m	=	**94.0**	0.00	**94**	423m
Mumford1	423.9	3.75	419	485m	<	**414.1**	4.90	**406**	467m	>	434.8	6.08	427	488m
Mumford2	1761.1	26.96	1722	209m	<	**1438.2**	39.25	**1382**	194m	>	2007.0	36.51	1952	214m
Mumford3	1966.7	16.13	1944	182m	≤	**1935.4**	45.71	**1881**	182m	>	2607.3	16.78	2577	195m

that performed well only on the larger instances, and SA which achieved success on Mandl problem variants. Combining this with the operator perspective results, it can be concluded that GD combined with SS is the best approach in both perspectives. Having discovered the best selection hyper-heuristics, we then extended the run times simply to gauge whether further improvements are possible, rather than to make additional comparisons between the three methods. This way, we can also strengthen our assumption for selecting SS-GD as our best selection hyper-heuristic.

4.3.3 Longer Runs

In this series of experiments we have given each of the three hyper-heuristics longer running times to observe whether improved performance can be obtained if there is more time to modify routes. We increased the running time by a factor of ten (i.e. the largest instance in the set will run for ten hours and ten times the number of iterations), and performed two runs on each instance: one from the passenger perspective, and one from the operator perspective. According to Table 4.5 the results of these runs revealed the success of GD from the operator perspective in all instances similar to the short run time experiments, and from the passenger perspective GD performed the best in Mandl problems, Mumford0 and Mumford1 instances. In Mumford2, and Mumford3 it was also successful in scoring competitive results.

4.3.4 Obtaining Multiple Solutions

The previous experiments focused on finding the best possible route sets from passenger or operator perspectives separately. For practical use on real world public transit systems however, a compromise between the needs of the conflicting stakeholders will be required. To demonstrate how this can be achieved, another round of experiments

Table 4.5: Results of the long runs experiments from passenger and operator perspectives. Best values are highlighted in bold

Instance	C_p	C_o	C_p	C_o	C_p	C_o
	SS-SA		SS-GD		SS-LA	
	Passenger Perspective					
Mandl4	**10.482**	148	**10.482**	148	10.576	140
Mandl6	10.187	216	**10.179**	212	10.321	186
Mandl7	10.119	214	**10.103**	250	10.150	232
Mandl8	10.086	251	**10.080**	272	10.095	260
Mumford0	14.157	725	**14.093**	722	14.218	734
Mumford1	21.961	2073	**21.699**	1956	22.096	2010
Mumford2	25.554	5276	25.196	5257	**25.001**	5480
Mumford3	28.261	5807	28.056	6119	**27.894**	6217
	Operator Perspective					
Mandl4	13.8754	**63**	14.6718	**63**	13.8754	**63**
Mandl6	13.4804	**63**	14.2832	**63**	14.3571	**63**
Mandl7	13.6763	**63**	14.4438	**63**	14.8645	**63**
Mandl8	14.2158	**63**	14.7938	**63**	15.0572	**63**
Mumford0	24.814	**94**	26.320	**94**	28.475	**94**
Mumford1	42.922	414	39.452	**408**	35.269	437
Mumford2	42.356	1436	46.865	**1330**	41.188	1508
Mumford3	44.771	1877	46.054	**1746**	42.569	1758

has been carried out using SS-GD (the best performing algorithm according to the experiments results). To ensure balance and fairness between the two objectives, their values have been normalised to 1 using the following parameters setting in Equation 4.1: $\alpha = \infty$, $\beta = \frac{1}{C_{Pinit}}$, $\gamma = \frac{1}{C_{oinit}}$ where C_{Pinit}, C_{oinit} equals the passenger and operator costs of the initial solution respectively. Several weight settings were then chosen to give a spread of compromise solutions.

We have tested this approach on Mandl instance (with six routes) by running several experiments each for a duration equal to the short run time with different weights combination per run and the results are plotted in Figure 4.3 along with the best results for the passenger and operator acquired previously. Clearly computing more compromise solutions will increase the cumulative run time of the optimisation. On the other hand, we do not require the vast populations generally needed to maintain diversity for evolutionary algorithms.

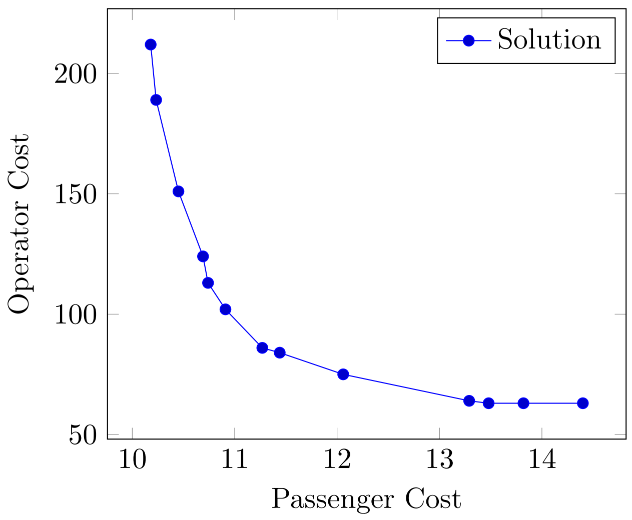


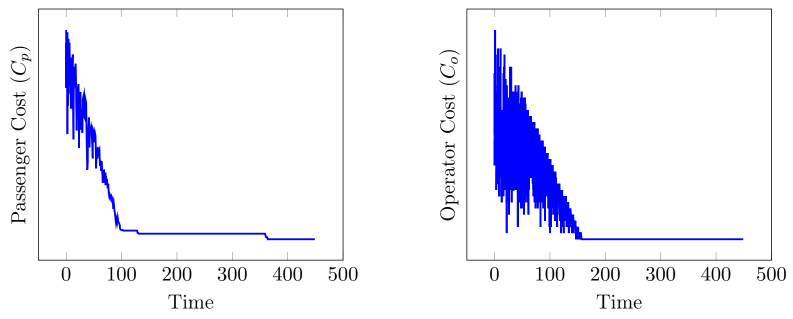
FIGURE 4.3: A plot showing a number of solutions between the best passenger and operator results in Mandl6 instance. Each point represents a different solution with different weight values.

4.3.5 Analysis of SS-GD

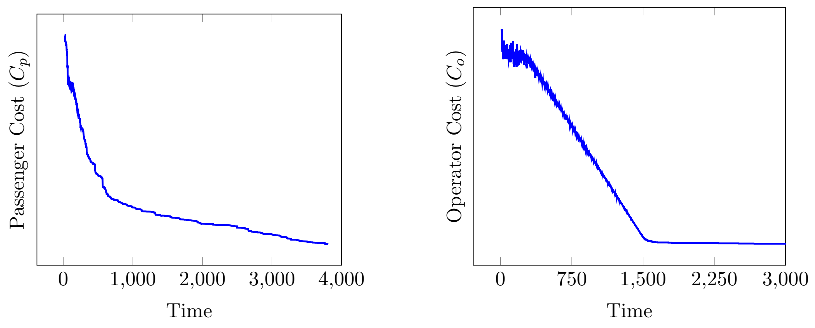
Based on the extensive experiments carried out we have chosen SS-GD as our best performing algorithm. The following analysis is performed on Mandl6 and Mumford3 instances, representing the smallest and largest networks in our dataset. We took Mandl6 further into analysis as its the most common variant of Mandl's problem addressed in the literature.

Figure 4.4 shows the change of the cost value over time for Mandl6 and Mumford3 instances after running each for a single short run. The cost has been recorded every 100 iterations from the passenger perspective and every 10^4 iterations from the operator perspective. From the passenger perspective, we can observe the quick variation in Mandl6 between worsening and improved solutions while the cost is dropping rapidly at the first stage of the search. In less than half of the search time, we notice that the variation stabilised indicating that the hyper-heuristic found the best solution quickly due to the small size of the instance and the simpler calculations which allow more iterations in short time. In Mumford3, similarly the cost value varies between worsening and improved solutions while dropping linearly. It can be noticed that there is a continuous improvement in the cost until the end of the search, indicating the ability to find better solutions with longer run times.

From the operator perspective the two instances showed similar behaviour, this could be referred to the simplicity of the operator calculations which can be performed easily



(a) Mandl6



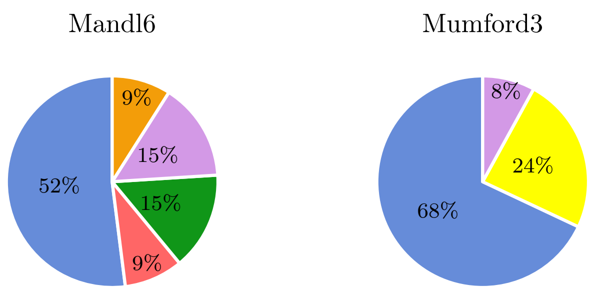
(b) Mumford3

FIGURE 4.4: Operator and passenger costs change over time in (a) Mandl6 and (b) Mumford3 instances

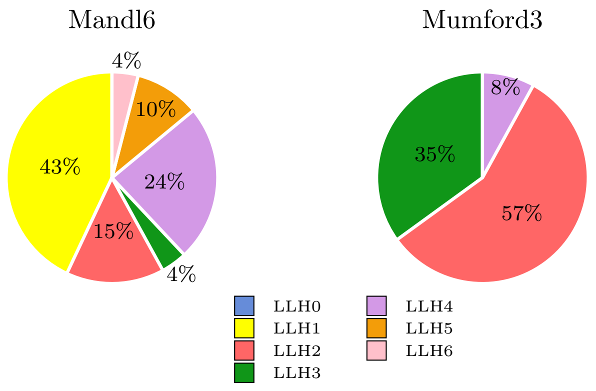
even if the instance size is large. In both instances the hyper-heuristic was able to find the best possible solution in less than half of the search time. The threshold value at this stage becomes less than the best solution making GD works similarly to only improve acceptance method.

Figure 4.5 shows the average utilisation rate for each low level heuristic for Mandl6 and Mumford3 instances after running each for a single run under SS-GD from both passenger and operator perspectives, considering only the applications of the low level heuristics that end the active sequence and improve over the best solution (i.e. $Seq = \text{end}$).

From the passenger perspective, the add low level heuristic (LLH0) was the most successful in both instances, achieving the most contribution in the best solutions. Other low level heuristics that were also successful are insert (LLH3), replace (LLH4) in Mandl6, and delete (LLH1) in Mumford3. In contrast, the add low level heuristic was the least successful from the operator perspective, and most of the contribution was achieved by the delete (LLH1), swap (LLH2), and insert (LLH3) low level heuristics. One possible explanation for this, is that the operator perspective focuses on building short routes that reduces the overall travelled distance. The delete low level heuristic works on achieving



(a) Passenger perspective



(b) Operator perspective

FIGURE 4.5: Average utilisation rate for each low level heuristic considering the invocations that generated improvements on the best solution in Mandl6 and Mumford3 instances

this, and the swap and insert low level heuristics improve these routes further by randomly mutating the nodes on the routes. In contrast to the passenger perspective, where adding more nodes to the routes increases the chances for direct trips.

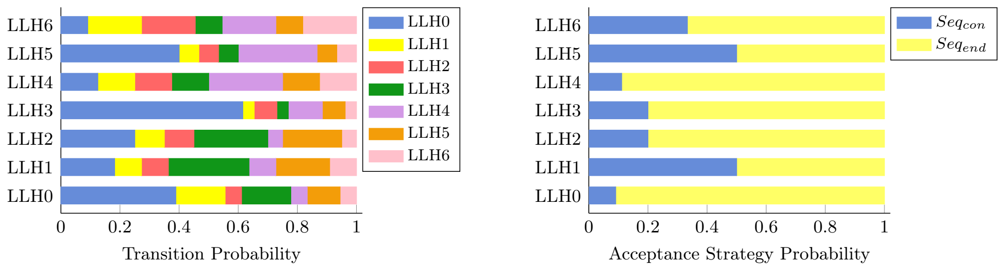
Figure 4.6 shows the transition, and the sequence construction frequency matrices, again for Mandl6 and Mumford3. A few interesting observations can be made. From the passenger perspective, we note from the sequence construction matrix that on the whole, the low level heuristics have a higher probability to end the sequence than continue it, which indicates that low level heuristics that contribute effectively to producing the best solutions, perform this success individually in sequences of length one. Specifically the add low level heuristic (LLH0) in Mumford3 which is the most successful in the set by referring to Figure 4.5, works almost independently. Yet from the transition matrix, some good sequences with a longer length than one can be identified, such as the combination of insert to different route (LLH5), and insert on the same route (LLH3) low level heuristics with the add low level heuristic (LLH0) in Mandl6. From the operator perspective, similarly the most successful low level heuristics (i.e. LLH1, LLH2, LLH3) operate best individually in both instances. In Mandl6, and Mumford3 the add low level heuristic tends to work better in sequences longer than one from the sequence construction matrix, unlike its behaviour in the passenger perspective. One of these sequences is the combination of add and delete low level heuristics in Mandl6. These observations show the intelligence of the SS method in identifying good sequences and understanding the relationships between low level heuristics in two different instances and two different evaluation approaches. The SS method has the advantage of intelligently revealing how the low level heuristics operate. Some low level heuristics may have a high utilisation rate, yet they achieve this with the support of other low level heuristics that might seem to have low contribution to the best solutions, but are necessary part in making the success of the high utilisation low level heuristics.

4.3.6 Comparison with Other Approaches

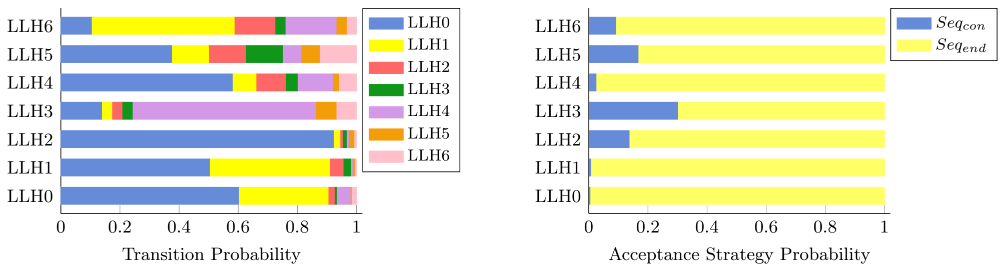
We compared our results from the passenger and operator perspectives with the state-of-the-art methods from the literature. We used the long run results of SS-GD representing our best results. Referring to Tables 4.6 and 4.7 our method found the best average travel times in all Mandl problem variants as well as the best d_0, d_1, d_2 in all cases except in Mandl4 instance, where Fan [28] found a better result for d_0. In Mumford's instances, our approach outperformed the methods in [6, 31, 83] in terms of the average travel time, and d_0, d_1, d_2 values scoring zero percentage for unsatisfied demand in all cases. From the operator perspective, we succeeded in finding the lower bound in Mandl's four problems and in Mumford0 instance. Our approach also found the best results in Mumford1, Mumford2 and Mumford3 instances. This comparison proves the success of hyper-heuristics on this problem, outperforming the previously reported results using GA approaches. Cooper et al. [8] reported that the implementation of John et al. [83] required 44 hours to run Mumford3 instance, and to improve this, a parallel implementation of the algorithm is required. Kılıç and Gök [31] required more than eight hours to initialise route sets and run a simple hill climbing algorithm in Mumford3 instance. Hyper-heuristic was able to find new best solutions after running Mumford3 for a single hour.

4.4 Summary

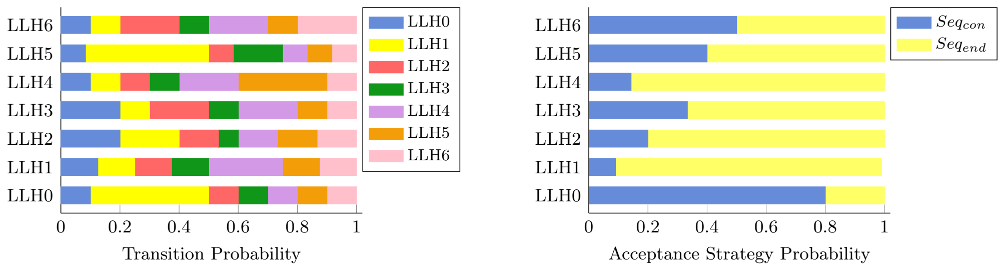
In this chapter we described our application of selection hyper-heuristics to the complex problem of the urban transit network design (UTRP). Thirty selection hyper-heuristics



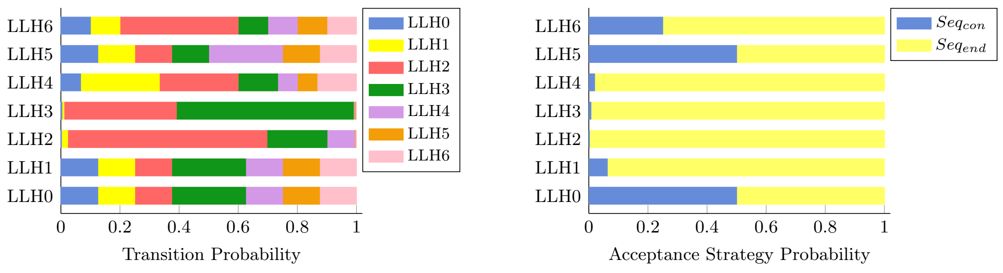
(a) Mandl6 Passenger



(b) Mumford3 Passenger



(c) Mandl6 Operator



(d) Mumford3 Operator

FIGURE 4.6: Transition and sequence construction frequency matrices for Mandl6 and Mumford3

TABLE 4.6: Passenger perspective results compared to other approaches

Instance	Parameter	Mandl [73]	Chakroborty and Wivedi [3]	Fan and Mumford [5]	Mumford [6]	Chew et al. [82]	John et al. [83]	Kılıç and Gök [31]	SS-GD
Mandl4	Passenger	12.90	11.90	11.37	10.57	10.50	-	10.56	**10.48**
	Operator	-	-	147	149	150	-	137	148
	d_0	69.49	86.86	**93.26**	90.43	91.84	-	91.33	91.84
	d_1	29.93	12.00	**6.74**	9.57	8.61	-	8.16	8.15
	d_2	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	-	**0.00**	**0.00**
	d_{un}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	-	**0.00**	**0.00**
Mandl6	Passenger	-	10.30	10.48	10.27	10.21	10.25	10.29	**10.18**
	Operator	-	-	215	221	224	212	216	212
	d_0	-	86.04	91.52	95.38	96.79	-	95.5	**97.17**
	d_1	-	13.96	8.48	4.56	3.21	-	4.5	**2.82**
	d_2	-	**0.00**	**0.00**	0.06	**0.00**	-	**0.00**	**0.00**
	d_{un}	-	**0.00**	**0.00**	**0.00**	**0.00**	-	**0.00**	**0.00**
Mandl7	Passenger	-	10.15	10.42	10.22	10.16	-	10.23	**10.10**
	Operator	-	-	231	264	239	-	274	250
	d_0	-	89.15	93.32	96.47	98.01	-	97.04	**98.84**
	d_1	-	10.85	6.36	3.34	1.99	-	2.83	**1.15**
	d_2	-	**0.00**	0.32	0.19	**0.00**	-	0.13	**0.00**
	d_{un}	-	**0.00**	**0.00**	**0.00**	**0.00**	-	**0.00**	**0.00**
Mandl8	Passenger	-	10.46	10.36	10.17	10.11	-	10.20	**10.08**
	Operator	-	-	283	291	256	-	298	272
	d_0	-	90.38	94.54	97.56	99.04	-	97.37	**99.16**
	d_1	-	9.62	5.46	2.31	0.96	-	2.63	**0.83**
	d_2	-	**0.00**	**0.00**	0.13	**0.00**	-	**0.00**	**0.00**
	d_{un}	-	**0.00**	**0.00**	**0.00**	**0.00**	-	**0.00**	**0.00**
Mumford0	Passenger	-	-	-	16.05	-	15.40	14.99	**14.09**
	Operator	-	-	-	759	-	745	707	722
	d_0	-	-	-	63.20	-	-	69.73	**88.74**
	d_1	-	-	-	35.82	-	-	30.03	**11.25**
	d_2	-	-	-	0.98	-	-	0.24	**0.00**
	d_{un}	-	-	-	**0.00**	-	-	**0.00**	**0.00**
Mumford1	Passenger	-	-	-	24.79	-	23.91	23.25	**21.69**
	Operator	-	-	-	2038	-	1861	1956	1956
	d_0	-	-	-	36.60	-	-	45.10	**65.75**
	d_1	-	-	-	52.42	-	-	49.08	**34.18**
	d_2	-	-	-	10.71	-	-	5.76	**0.07**
	d_{un}	-	-	-	0.26	-	-	0.06	**0.00**
Mumford2	Passenger	-	-	-	28.65	-	27.02	26.82	**25.19**
	Operator	-	-	-	5632	-	5461	5027	5257
	d_0	-	-	-	30.92	-	-	33.88	**56.68**
	d_1	-	-	-	51.29	-	-	57.18	**43.26**
	d_2	-	-	-	16.36	-	-	8.77	**0.05**
	d_{un}	-	-	-	1.44	-	-	0.17	**0.00**
Mumford3	Passenger	-	-	-	31.44	-	29.50	30.41	**28.05**
	Operator	-	-	-	6665	-	6320	5834	6119
	d_0	-	-	-	27.46	-	-	27.56	**50.41**
	d_1	-	-	-	50.97	-	-	53.25	**48.81**
	d_2	-	-	-	18.79	-	-	17.51	**0.77**
	d_{un}	-	-	-	2.81	-	-	1.68	**0.00**

combining several known selection and move acceptance methods were tested and applied on Mandl benchmark instance and Mumford data set and their performances were compared statistically to determine the best algorithm. After a series of short and long time experiments from the perspective of operator and passenger, the analysis showed the success of the sequence-based selection method combined with Great Deluge (GD) acceptance method, outperforming other selection hyper-heuristics in both passenger and operator objectives. We showed That the hyper-heuristic approach which has been applied for this particular problem for the first time was very successful, beating the known state-of-the art results in very reasonable run times.

Table 4.7: Operator perspective results compared to other approaches

Instance	Parameter	Mumford [6]	Chew et al. [82]	John et al. [83]	SS-GD
Mandl6	Operator	**63**	**63**	**63**	**63**
	Passenger	15.13	13.88	13.48	14.28
	d_0	70.91	70.91	-	62.23
	d_1	25.5	25.50	-	27.16
	d_2	2.95	2.95	-	9.57
	d_{un}	0.64	0.64	-	1.028
Mumford0	Operator	111	-	95	**94**
	Passenger	32.40	-	32.78	26.32
	d_0	18.42	-	-	14.61
	d_1	23.40	-	-	31.59
	d_2	20.78	-	-	36.41
	d_{un}	37.40	-	-	17.37
Mumford1	Operator	568	-	462	**408**
	Passenger	34.69	-	39.98	39.45
	d_0	16.53	-	-	18.02
	d_1	29.06	-	-	29.88
	d_2	29.93	-	-	31.90
	d_{un}	24.66	-	-	20.19
Mumford2	Operator	2244	-	1875	**1330**
	Passenger	36.54	-	32.33	46.86
	d_0	13.76	-	-	13.63
	d_1	27.69	-	-	23.58
	d_2	29.53	-	-	23.94
	d_{un}	29.02	-	-	38.82
Mumford3	Operator	2830	-	2301	**1746**
	Passenger	36.92	-	36.12	46.05
	d_0	16.71	-	-	16.28
	d_1	33.69	-	-	24.87
	d_2	33.69	-	-	26.34
	d_{un}	20.42	-	-	32.44

Chapter 5

Hyper-heuristics for Solving Real-world Applications of the Urban Transit Routing Problem

In this chapter we describe the application of hyper-heuristics to two problems. The first problem addresses the application of the SSHH algorithm combined with Great Delluge acceptance (SS-GD) to a larger scale and more complex version of the route design problem. In this version we tackle the presence of specific terminal nodes in the transport network from which buses are restricted to start and end their journeys. We design an initialisation procedure for the route network based on the passengers' demand information, and implement a set of low level heuristics to handle the terminal nodes at routes ends. Furthermore, we apply our selection hyper-heuristic algorithm to a new set of instances recently introduced in the work of Heyken Soares et al. [84] with real-world characteristics and size. The instances were generated using a novel generation procedure which aims to scale down a real world transportation network, yet preserves the vital characteristics of the network layout. We test our selection hyper-heuristic algorithm on this set of instances, and show that our results top the best results found by the application of NSGAII and extracted real-world route sets.

The second part of this chapter discusses the application of the SSHH algorithm on a real planning application that involves the optimisation of bus transport routes by the commercial transport planning software "Visum". The hyper-heuristic is integrated with Visum software through interface procedures, and applied to two optimisation problems, one of them being a city-size network. We demonstrate that hyper-heuristics are able to effectively reduce the passenger and operator objectives on both benchmark and real-size networks.

5.1 Optimising Bus Routes with Fixed Terminal Nodes: Comparing Hyper-heuristics with NSGAII on Realistic Transportation Networks

5.1.1 The UTRP with Terminal Nodes

We have previously discussed a simplified model for the UTRP in chapter 2, which was applied to test the selection hyper-heuristics for solving the UTRP in chapter 4. Recalling, the model aims to a find a feasible route set from a predefined transport network with known pick-up/drop-off locations in order to satisfy the entire network demand and reduce passenger and operator expenditures. A feasible route set is defined by feasibility criteria which must be satisfied in order to deem the solution feasible. The feasibility conditions were defined in section 2.4.3 and were used to design the feasibility test as described in section 4.2.1 of the previous chapter.

In this work we introduce an additional constraint into this network model, that restricts the start and end points of bus journeys to specific points named terminals. Identifying end points for bus journeys is essential when solving the routing design problem in an urban context, to provide u-turn possibilities for buses. However, adding this condition creates extra complexity by making it more difficult to construct feasible solutions. Figure 5.1 illustrates a “legal” connected network according to the feasible network definitions in chapters 2, and 4. The same network becomes infeasible when three terminal points are introduced (green) making one of the routes invalid with an incorrect end terminal (node 4). This effect becomes more profound with the increase in network size, or the decrease in the number of valid terminals.

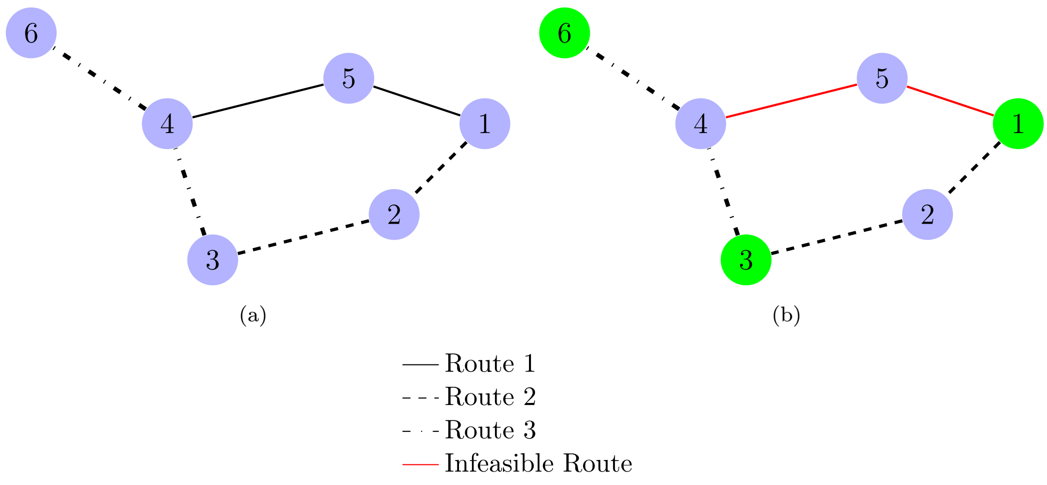


Figure 5.1: Feasible route network (a) becomes infeasible (b) by introducing three terminal points (green)

Very few models in the literature incorporate terminals, especially for large instances. However, Pattnaik et al. [79] solved the network design problem for a small network representing parts of Madras city in India, using genetic algorithms (GA) in two phases: first a heuristic procedure is applied to generate a set of candidate routes and then the GA is applied in the second phase. Their candidate route set generation procedure is based on the demand matrix, route set constraints and designer's knowledge. The procedure involves finding the shortest path between every origin and destination pair which are selected from a set of terminal points. The designer identifies the terminal points by taking the network layout into consideration. Szeto and Wu [80] solved the bus network design of the suburban area of Tin shui Wai in Hong Kong using a network model of 28 nodes, where trips originate from specific terminal points and end at one of five destination nodes. Seven terminal points are specified in their network model and a GA incorporating a frequency setting heuristic is used to solve the route design problem and determine bus frequencies. Amiripour et al. [108] tackled the bus network design problem by considering seasonal variation in the demand to provide a convenient bus service throughout the year. A GA has been applied to solve their model by testing it on two small benchmark instances and a real case study in the city of Mashhad in Iran. In the larger network of Mashahd, several terminal points have been identified by testing their turning possibilities and performing K-shortest path between pairs of terminal points to create feasible routes.

We will describe here our hyper-heuristic solution to design routes for a new published data set of instances generated following the procedure described in [84], which scales a real-world street network into a size manageable by optimisation algorithms while preserving its vital characteristics. These instances include additional information which indicates whether a node is a terminal or not. We will further describe our applied procedures and the instances set in the following sections.

5.1.2 Problem Formulation

In this problem we adopted the UTRP model described in chapter 2, and also applied in our methodology in chapter 4, with some minor variations due to the presence of new information describing terminal nodes.

The road network comprises a set of stops connected by road segments. This can be mapped into an undirected graph $G = \{V, E\}$, where the graph vertices $V = \{v_1, v_2, \ldots, v_n\}$ are access points (i.e. bus stops), and the graph edges $E = \{e_1, e_2, \ldots, e_m\}$ are direct transport links. Some of the vertices are identified as terminal points $U = \{u_1, u_2, \ldots, u_k\}$, such that $U \subseteq V$. These terminal points allow u-turns to make the reverse trip in the opposite direction. A public transport route r_a, according to these definitions, is a path in the graph that connects a set of vertices, and starts and finishes at a terminal point $r_a = u_j, v_{i_2}, \ldots, v_{i_{q-1}}, u_k$.

Similarly, the route network is the model solution, and its evaluation requires demand and travel time information which are represented by symmetrical two dimensional matrices. The terminal points are identified using one dimensional vector $U_{n\times 1}$, where n is the number of vertices in the road network and each entry u_i has a value of one if v_i is a valid terminal, or zero otherwise. The feasibility of route set R is guided by the constraints defined in section 2.4.3, in addition to the following constraint: $\forall r_a \in R : v_{i_1}, v_{i_q} \in U$.

5.1.3 Optimisation Procedure

In this section we describe the methodology applied to optimise the route set in our data set with terminal node information, starting with the creation of a high quality (and feasible) initial route set, followed by the optimisation procedure using selection hyper-heuristics.

5.1.3.1 Creating an Initial Route Set Using a Heuristic Construction Procedure

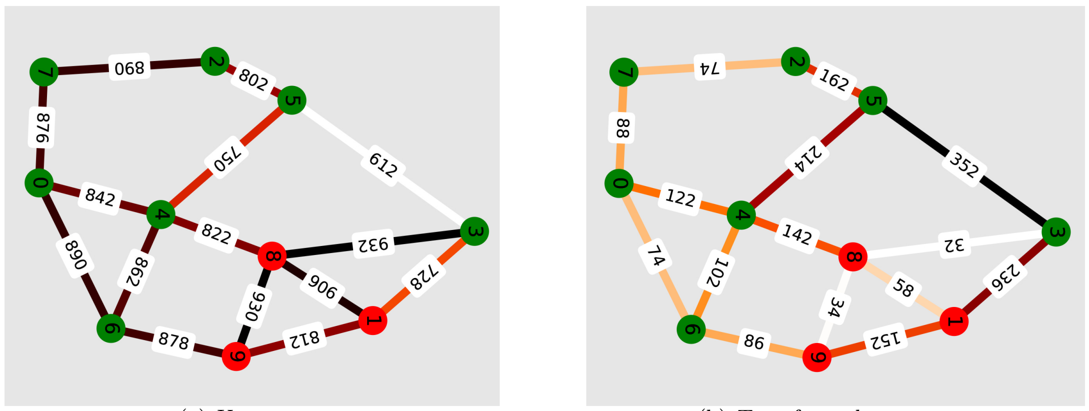
Heuristic construction procedures are widely used in previous research of the UTRP to generate high quality initial route sets. Some approaches use shortest path algorithms to generate candidate solutions, and improvement heuristics are applied in further step to improve their quality. Examples of this approach are in [74, 75, 83]. Other approaches use shortest path algorithms to create a pool of routes from which a set of routes is constructed based on a defined set of rules. Examples of this method are given in [79, 82]. The importance of an efficient route generation algorithm is significant and can lead to high quality end results. Kılıç and Gök [31] argued that the initial generation has a higher impact on the quality of the final route set than the local search procedures, and they pointed out that using the edge usage statistics can provide excellent guidance during the route set construction.

In our proposed initial route set generation procedure, we use the property of edge usage statistics to find a palette of routes from which a single initial solution is constructed. Our method utilises the edge usage which is defined by the total demand that use this edge during their travel. We attempt to include the highest usage edges in the final route set.

The initial generation procedure produces an initial route set based on the following parameters: the demand matrix, the terminal points vector, the road network graph, the predetermined number of routes in the route set, and the minimum and maximum length of each route (in terms of the number of nodes). Using this information, an initial route set is generated guided by the demand matrix to ensure that as much of the demand as possible is routed along its shortest travel time path. The initialisation algorithm involves the following steps:

- Produce an edge usage graph guided by demand and shortest travel time path information.
- Create a pool of candidate routes.
- Construct a route set from the candidate route pool.

Assuming the passenger prefers to travel along his/her shortest travel time path, the shortest path between every pair of nodes in the road network is calculated. It is then



(a) Usage map

(b) Transformed usage map

FIGURE 5.2: The usage map and the transformed usage map: darker colour = high demand edges, lighter colour = lower demand edges. Green vertices = terminal vertices, red vertices = non-terminal vertices.

an easy matter to create a “shortest-path-usage map" by adding up the total demand travelling along each edge in the network, assuming all travellers are able to traverse their shortest paths[1]. An example of such a map is displayed in figure(5.2(a)), using the Clifton instance (will be described in section). In the diagram the edge labels represent the total demand along each link. A similar approach for calculating the edges usages has been used in [31].

Next we perform a simple transformation on this map to convert the usages into distances so that the largest usage becomes the shortest distance and vice-versa. This is done by subtracting the usage on each edge from some arbitrary large number. We have chosen to use the total demand for the whole network for this purpose. Figure (5.2(b)) demonstrates the transformed usage map using the upper bound of the demand for Clifton (i.e equals 964). In this case the highest usage (i.e from node 3 to node 8) becomes the shortest distance (964 - 932 = 32). Our approach here differs from [31] where they calculate probabilities to select edges based on their usage value.

The transformed usage map is then used to generate routes for the routes pool, which will later be used as a palette from which to select routes for the initial route set. The algorithm will iterate through pairs of terminal nodes and create routes by performing shortest path computations based only on the transformed usage map. In this way the algorithm will generate routes that include the busiest edges, and each of these will enter the pool as a candidate route, provided its route length lies between the minimum and maximum allowed. However, to guarantee that the pool of routes covers all of the

[1]The demand of each edge is aggregated in the two directions of travelling.

nodes in the network, it is necessary to include some less busy links. To achieve this, the shortest path algorithm iterates several times between all pairs of terminal nodes. After each iteration, the weights of the transformed usage map are updated by slightly increasing the ones that correspond to edges selected by the route generation procedure and included in one of the routes in the routes pool. This encourages the shortest path algorithm to look for alternative paths that may include undiscovered nodes. The weight values are increased by multiplying them with a very small value which have been tuned to 1.1 after a series of trials. The iterations terminate after the inclusion of all the network nodes in the candidate pool.

The final step is to construct a legal route set from the route pool, by selecting them one at a time, without replacement. The first route in the route set is randomly chosen from the pool. Then the number of unseen nodes with respect to the route set under construction (currently including one route) is calculated for every candidate route in the pool. The candidate route that has the highest number of nodes that are not yet included in the route set and has at least one node in common with the first route is selected as the second route. The third route is chosen similarly while guaranteeing it has at least one node in common with one of the first two routes to ensure connectivity of the route set. If all the nodes have been included and the route set has not yet reached the predetermined limit for the number of routes, the algorithm completes the route set by selecting the first route in the pool. This process continues until $|N|$ routes are constructed and all the nodes are included in the route set.

5.1.3.2 Objectives and Evaluation

In this problem we tackled the same objectives described in chapter 2 and was applied in our work on chapter 4. These objectives are the passenger (C_p) and the operator (C_o) objectives described respectively by the two formulae 2.3, 2.4. The assumption in the passenger cost is that the passenger always chooses the shortest path for their journey, and it incorporates the in-vehicle travel time, the waiting time, and the transfer time. The waiting time and the penalty for making a transfer are combined as a single time set to 5 minutes.

$$f(S) = \alpha \frac{C_p(S)}{C_p(S_0)} + \gamma \frac{C_o(S)}{C_o(S_0)} \tag{5.1}$$

Equation 5.4 is used to evaluate a given solution, where $C_p(S_0)$ and $C_o(S_0)$ are the initial values for the passenger and operator objectives respectively, and α and γ are parameters to determine how to direct the optimisation by either focusing on optimising one of the objectives, or balancing them. For example setting γ to a very small value while α is equal to 1 or larger will generate route sets biased towards the passenger perspective, and vice versa. To find route sets balancing both objectives the difference between the two parameters α γ, should be nearly zero.

To analyse candidate route sets more extensively, the following parameters are used to calculate the percentages of demand satisfied by direct (i.e. zero transfers) and indirect trips (i.e. one or two transfers): d_0, d_1, d_2, d_{un}. The demand that requires three transfers or more is considered unsatisfied (d_{un}). To calculate these parameters, it is assumed that the passenger always prefer the route with the fewest transfers, if there exist more than one shortest path between two points.

5.1.3.3 Optimising Route Sets Using Selection Hyper-heuristics

In this work We have applied the winning algorithm according to the results and analysis presented in chapter 4 which is the selection hyper-heuristic combining the SSHH selection method and Great Deluge acceptance method. We will refer to this algorithm by SS-GD.

At the start of the optimisation, the initial solution (S_{init}) built using the initial route generation method described in section 5.1.3.1, is introduced to the hyper-heuristics as the current solution (S_{curr}) and the sequence of heuristics constructed by SSHH is applied to S_{curr} to generate a new solution (S_{new}). The feasibility of S_{new} is tested, a single violation in any of the problem constraints results in rejecting this solution (e.g. if at least one of the terminals of any route in the route set is not valid). In this case a new sequence of heuristics is constructed and applied to generate a new solution. If S_{new} is feasible, it is evaluated using equation 5.4 and the parameters are set to determine which objective the optimisation is focusing on. For example to optimise the route set by balancing the two objectives, the parameters α and γ are both set to 1, or one of them can be slightly increased to favour one of the objectives. To generate route sets optimised from one of the perspectives of passenger or operator, one of the parameters is set to 1 and the other to a very small value (e.g 10^{-4}).

After evaluating S_{new}, and if it is better than the best known solution, the best solution is updated and the sequence of heuristics is rewarded by increasing the probability of selecting this sequence again. The Great Deluge (GD) move acceptance decides on the acceptance of S_{new} by comparing it to S_{curr}. If S_{new} is accepted, the value of S_{curr} is updated to the value of S_{new}. The optimisation terminates when the time set by the user elapses.

5.1.3.4 Low Level Heuristics

The low level heuristics set has been carefully designed to to ensure setting the correct route terminals. They also ensure that nodes are placed in the right positions where they are directly connected with the neighbouring nodes according to the adjacency relationships defined by the travel time matrix [2]. Our complete list of low level heuristics is presented below:

LLH0: **Add**. Selects a random route and a random position in the route. A node is selected and added in this position.

LLH1: **Delete**. Selects a random route and a random position in the route. The node in this position is deleted.

LLH2: **Replace**. Selects a random route and a random position. A node is selected to replace the node in this position.

LLH3: **Swap**. Selects a random route and two random positions and swaps the nodes in these positions.

LLH4: **Shift**. Selects a random route and two random positions. The node in the first position is inserted into the second position.

LLH5: **Add terminal**. Selects a random route and a random terminal node and inserts it into one of the route terminals by randomly selecting one of them.

LLH6: **Reverse**. Selects a random route and two random positions and reverses the order of nodes between these positions.

LLH7: **Crossover**. Selects two random routes and a random position on each route and splits the route in this position. Two different routes are created by swapping the parts of the two routes.

[2]The adjacency relations (i.e. the presence of direct connection) between nodes in the transport network can be derived easily from the travel time matrix as follows: the entries in the travel time matrix has either two values, a positive value indicating the time required to travel between two nodes, or ∞ indicating the absence of direct connection between two nodes

LLH8: **Delete(Add) nodes**. Selects a random route and adds a number of nodes at the route terminal or deletes a number of nodes from the route until the route reaches the maximum or minimum length.

LLH9: **Replace route**. Selects a random route and deletes it. A build procedure is then applied to construct a new route by finding the shortest path between two randomly selected terminal nodes. The deleted route is replaced by the new constructed route.

5.1.4 Nottingham Data set

In this study we used a set of instances based on different parts of the urban area of Nottingham city in the UK (figure 5.3). The instances vary in size: the largest covering the entire study area and the smallest representing only the small Clifton area in Nottingham. All instances are generated from official street and census data of the year 2011. The procedure effectively reduces the street network to a graph size tractable by optimisation algorithms while maintaining the characteristics of the street network layout to ensure they are reflected sufficiently in the instances.

This novel instances generation procedure was presented in the work of [84], where they applied it to generate the largest instance in the set, the Nottingham instance. The other smaller instances in the data set were generated using the same procedure and were presented in our work [122] to test the selection hyper-heuristic algorithm against NSGAII performance on a varying size of instances. Since this procedure is not an essential part of our work, we will summarise its steps very briefly, and for the complete description the reader can refer to [84].

The first step in the generation procedure is to select the streets available for bus travel in the study area and construct a street map. This is done based on official street classifications and the positions of existing bus stops. After that, the positions of the nodes are determined by placing initial nodes at all junctions and intersections of the street map. In cases where initial nodes are closer to each other than a defined distance, they are replaced by a new node half way between the positions of the original nodes. The resulting set of nodes do not represent concrete stop locations, but more precisely routing points which define the course of the bus route. It is assumed that vehicles travel on a path defined by these nodes, and stop at defined locations along the way.

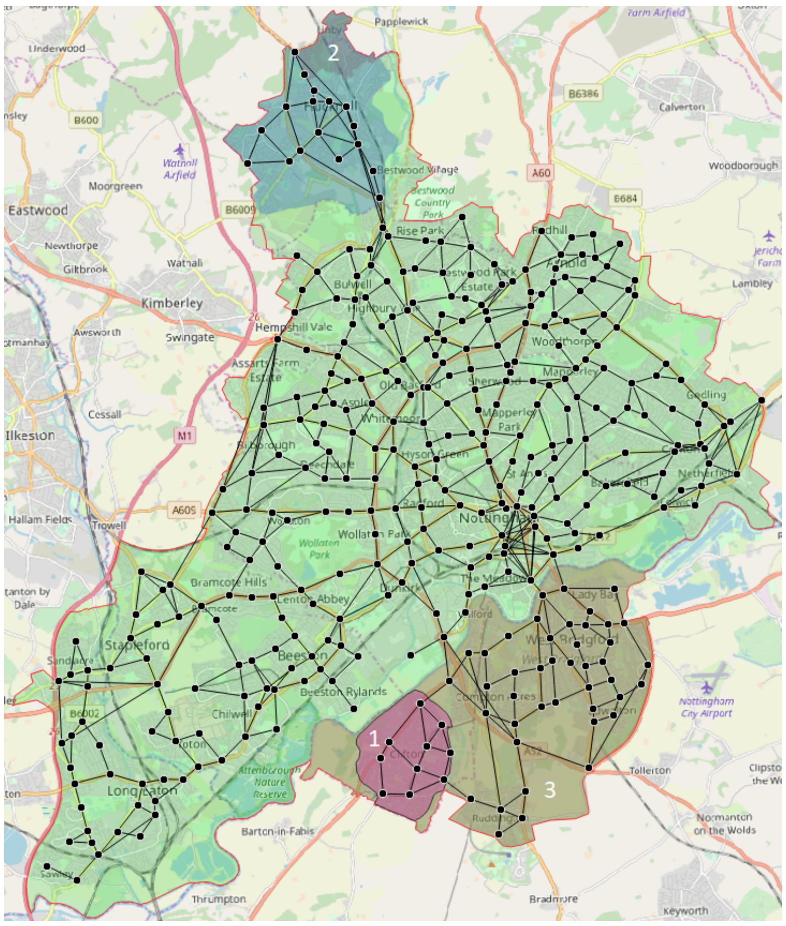


FIGURE 5.3: Map of the study area together with nodes and network edges generated with the method described in this section. The colours and numbers indicate the areas of the instances: 1: Clifton (red), 2: Hucknall (blue), 3: South of Trent (brown) and 4: Nottingham (green). The instances Hucknall and South of Trent are subsets of the Nottingham instance and Clifton is a subset of South of Trent.

In order to ensure that the results of the optimisation are directly comparable with the performance achieved by the real world bus routes, the instance should only include the nodes that are present in the paths of the real routes. The real bus routes are extracted from UK 2011 National Transport Data Repository (NPTDR) where bus journeys are stored in the form of journey patterns. Therefore the initial nodes determined by the previous step are filtered out to exclude the nodes that are not present in the real bus routes.

A number of nodes need to be designated as terminals representing potential start and end points of routes where buses can turn around. These nodes are identified by projecting the real world journey patterns on the generated street map to determine at which

Table 5.1: Features of the data set

Instance	No. of vertices/edges	No. of routes	No.of vertices per route (min/max)	No.of terminal nodes
Clifton	10, 15	4	2, 8	7
Hucknall	17, 28	5	2, 9	10
South of Trent	54, 86	18	2, 13	25
Nottingham	376, 656	69	3, 45	159

locations the actual bus journeys begin and end, and specify the nodes at these locations as terminal nodes.

The travel times associated with the network edges are defined by calculating the shortest paths between pairs of nodes and the demand between pairs of nodes was extracted from travel to work data from 2011 UK census. It gives the number of commuters between different census zones, and can be converted into a matrix of passengers travelling between different nodes by assigning zones demand to the network nodes.

Table 5.1 summarises the features of the data set. Note that the instances generation procedure ensures the production symmetrical demand and travel time matrices matching the problem description (section 5.1.2). The Journey patterns used in generating the real route sets are also modified to satisfy the problem constraints, in order to ensure fair comparison to the optimisation results.

The problem objectives are highly sensitive to the route set parameters, therefore they should be carefully set to ensure route set feasibility while considering the stakeholders needs. For example having a large number of particularly long routes is not beneficial to operators because longer travel distances require more vehicles and staff. On the other hand short routes increase the numbers of vehicle transfers for passengers. Sufficient routes should be present to cover the entire network nodes while maintaining the connectivity of the routes.

For the larger instances Nottingham and South of Trent, the route set parameters are determined from the real route sets, to ensure the optimisation results are fairly compared against them. The number of routes is the same as the extracted real world route set, while the maximum number of nodes is 10% longer than the longest real world route to give the optimisation algorithm freedom to slightly extend the existing routes. The minimum length is one node less than the shortest real world route. The parameters of the two smaller instances, Hucknall and Clifton, have been tuned to ensure route set coverage and connectivity while giving good initial results for both objectives. The tuning has been performed by testing different combinations of the following parameters:

the maximum route length, the minimum route length, and the number of routes in the route set. The combination that found a feasible solution with the best passenger and operator objectives was selected as parameters for these instances. For Example, we found that two routes were not sufficient in Clifton instance to find a feasible route set that covers the entire network nodes. Also choosing four routes resulted in a better passenger cost than three routes without significant impact on the operator cost.

5.1.5 NSGAII Optimisation

NSGAII is an elitist non-dominated sorting MOEA algorithm used very widely in solving multi-objective optimisation problems. It has been shown that this algorithm is successful in converging to near optimal Pareto-front compared with other Pareto-based methods [193]. The idea is to generate a parent population of size N_{pop} and use it to generate an offspring population of size N_{pop} through crossover and mutation operations. The parent and the offspring populations are combined to produce a population of size $2 \cdot N_{pop}$ from which the population for the next generation is selected by applying non-dominated sorting algorithm and crowding distance and choosing the first N_{pop} solutions of the sorted population. The NSGAII was applied in [83] to solve the UTRP problem in Mandl's benchamrk and Mumford data set, where they developed a specific set of mutation operators which proved its success by finding new best results in some of the tested instances. Heyken Soares et al. [84] based their NSGAII algorithm implementation on [83] and made some changes to adapt to the use of terminal nodes .

We will compare the set of solutions generated by our hyper-heuristic algorithm with the NSGAII Pareto-front constructed by the algorithm in [84]. To summarise this algorithm, a crossover operator generates an offspring route set from pairs of parent route sets, where the routes from the two parents are selected alternately such that the proportion of unseen vertices in the offspring is maximised. The generated offspring route set has then a certain chance to undergo mutation. For the mutation, one of the following mutation operators is selected randomly: delete nodes, add nodes, exchange two routes, replace route, merge two routes. These mutation operations are similar to the ones implemented in [83], but have been adapted to the presence of terminal vertices. The NSGAII algorithm minimises the same objectives as the hyper-heuristics (i.e. passenger and operator costs), and is also subject to the same set of constraints described in section 5.1.2. In the following sections, we will be demonstrating the results of the hyper-heuristic, and further compare it to the NSGAII performance and to the real-world route sets of the instances Nottingham and south of Trent.

5.1.6 Experimental Results

The experiments were conducted by applying SS-GD to each instance of the data set following three scenarios: 1) from the perspective of passenger 2) from the perspective of operator 3) balancing the two objectives. This is achieved by setting the parameters in equation 5.4 as follows: to generate route sets biased toward the passenger (operator) objective α (γ) is set to 10^{-4} while the other parameter is set to 1. Whilst for balancing the two objectives α is set to 2 and γ to 1. The three scenarios are applied to each instance, and for each scenario the hyper-heuristic is run for 10 trials, each terminating after a specific time period. The running length of each trial increases with the instance size by adding thirty seconds to each node, thus the smallest instance run for five minutes and the largest for three hours.

5.1.6.1 SS-GD Results

Table 5.2 summarises the results of SS-GD in all the instances from the perspective of passenger, operator, and balancing the two objectives measured in minutes for the average passenger travel time and the routes length. The results are reported using the average of the ten trials, the minimum and maximum results in the ten trials, and the standard deviation. The average number of iterations of SS-GD is also reported in the table.

From these results, we can observe the variation between the two objectives in the passenger and operator perspective trials. While one of the objectives significantly improves, the other drastically worsen, and this observation applies in all instances. The balanced trials make a notable improvement in both objectives simultaneously in all instances, providing route sets that are more suitable for a real-world problem. The passenger and the operator perspective trials attempt to find the best possible solution for each of the objectives, making these solutions far from representing a real-world solution, and more suitable for finding lower bounds for the passenger and operator costs. We also notice that SS-GD performed equally well in all instances despite their varying sizes, and succeeded in decreasing the passenger and the operator objectives significantly from their initial values.

TABLE 5.2: Results of SS-GD in all instances from the passenger, operator, and balanced perspective. The results are reported using the average of the ten trials, the maximum and minimum objective values, and the number of iterations.

		Passenger Perspective					Operator Perspective					Balanced Perspective				
Instance		min	max	avg	std	Iter	min	max	avg	std	Iter	min	max	avg	std	Iter
Clifton	C_p	3.11	3.22	3.14	0.034	6083367	7.69	7.69	7.69	0.000	14955130	3.89	4.31	4.26	0.131	8564069
	C_o	40.31	50.68	45.31	3.711	6083367	14.91	14.91	14.91	0.000	149551230	20.64	24.77	21.10	1.299	8564069
Hucnall	C_p	4.43	5.19	4.80	0.233	9064538	12.37	13.91	13.35	0.587	18382757	5.37	6.49	5.71	0.431	17360516
	C_o	59.03	71.74	65.91	3.963	9064538	26.25	26.25	26.25	0.000	18382757	29.99	38.65	35.12	2.857	17360516
South Of Trent	C_p	6.94	7.51	7.07	0.164	641158	20.61	25.68	23.00	1.541	5524759	8.06	8.71	8.43	0.212	2715845
	C_o	277.40	309.33	286.49	8.813	641158	80.05	85.85	82.18	1.912	5524759	110.99	141.78	121.79	9.590	2715845
Nottingham	C_p	11.00	11.23	11.12	0.072	44191	28.17	43.74	35.37	5.417	1505483	12.47	12.72	12.60	0.086	160402
	C_o	1960.29	2155.32	2060.18	56.976	44191	564.23	673.29	619.88	29.483	1505483	974.95	1103.27	1030.09	40.322	160402

We can see that the standard deviation was insignificant in most of the cases, reflecting the consistency of the results found by the ten trials. In the two smaller instances Clifton, and Hucknall, the same operator cost was recorded by the ten trials in the operator perspective. This implies that it can possibly be an optimal solution for the operator cost of these instances. However, since we do not have an exact lower bounds or optimal solutions for any of the instances in this data set, we cannot guarantee this proposition.

5.1.6.2 Comparison of SS-GD and NSGAII

Table 5.3 summarises and compares the results of SS-GD against NSGAII from the perspective of passenger and operator measured in minutes for the average passenger travel time and the total routes length. The minimum result in the 10 trials is compared to the best result found by NSGAII from the perspective of passenger and operator. The average of the 10 trials is also recorded. Figure 5.4 plots the results of SS-GD with the evaluation results of the final population which forms a clear Pareto front. SS-GD results are taken from four key positions: the best result from the passenger perspective, the best result from the operator perspective, the most passenger friendly and the most operator friendly route sets in the 10 trials that balance the two objectives.

From results in table 5.3 and the plots, it can be clearly seen that SS-GD outperforms NSGAII from the passenger and operator perspectives in all instances. In fact, the best passenger results for SS-GD in all instances not only succeeded in improving the passenger average travel time, but also the operator cost has improved. The best operator results for SS-GD also improve significantly over NSGAII in all instances, especially the largest instance Nottingham, although NSGAII could find better average travel times for passengers in this case. The compromise solutions (i.e. balanced perspective) of SS-GD are also very successful. Comparing these solutions to the solutions of NSGAII with the same passenger objective, SS-GD is successful in finding much improved costs for the operator, and this observation applies for all instances. The greatest success is witnessed in the largest instance of Nottingham, where the most passenger friendly route set in the compromise solutions is better than the best passenger result found by NSGAII, while the operator cost is improved by more than 50%. Also comparing the run time of these algorithms for the largest instance Nottingham, NSGAII requires more than a week to generate a final population of Pareto solutions. SS-GD is much faster in producing a single solution of high quality compared to NSGAII in a single run, which takes only

Table 5.3: Comparison between the best results from the perspectives of passenger and operator between SS-GD and NSGAII for each instance.

Instance	Objective	SS-GD		NSGAII
Passenger Perspective				
		min	avg	
Clifton	C_p	**3.11**	3.14	3.30
	C_o	50.67	45.31	54.65
Hucknall	C_p	**4.42**	4.80	4.56
	C_o	65.40	65.91	58.64
South of Trent	C_p	**7.07**	7.20	7.31
	C_o	278.17	275.35	303.75
Nottingham	C_p	**11.00**	11.11	12.44
	C_o	2105.06	2060.18	2325.87
Operator Perspective				
Clifton	C_p	7.69	7.69	8.61
	C_o	**14.91**	14.91	17.01
Hucknall	C_p	12.36	13.34	8.43
	C_o	**26.24**	26.24	26.96
South of Trent	C_p	22.00	23.43	18.55
	C_o	**82.32**	84.30	99.83
Nottingham	C_p	43.74	35.36	19.77
	C_o	**564.23**	619.88	741.83

three hours as mentioned previously. This can be clearly seen in the best passenger results of Nottingham instance where SS-GD was able to reduce the passenger travel time by 1 minute and offer better operator costs compared to NSGAII in an individual run of three hours. In the smaller instances, SS-GD run time was also better requiring only few minutes to find solutions that improve over the best passenger and operator solutions found by NSGAII.

5.1.6.3 Comparison with Real World Route Sets

In this section we compare the optimisation results with the real world routes for the two largest instances: Nottingham and South of Trent. Extracting real bus routes (i.e. journeys) is important to compare to the optimisation results and validating the efficiency of the optimisation algorithm. The real world bus routes are the operating routes in the city of Nottingham from the year 2011 extracted from the National Public Transport Data Repository (NPTDR). The journey patterns from the NPTDR were utilised to construct the real world route sets by applying a set of filtering criteria in order to

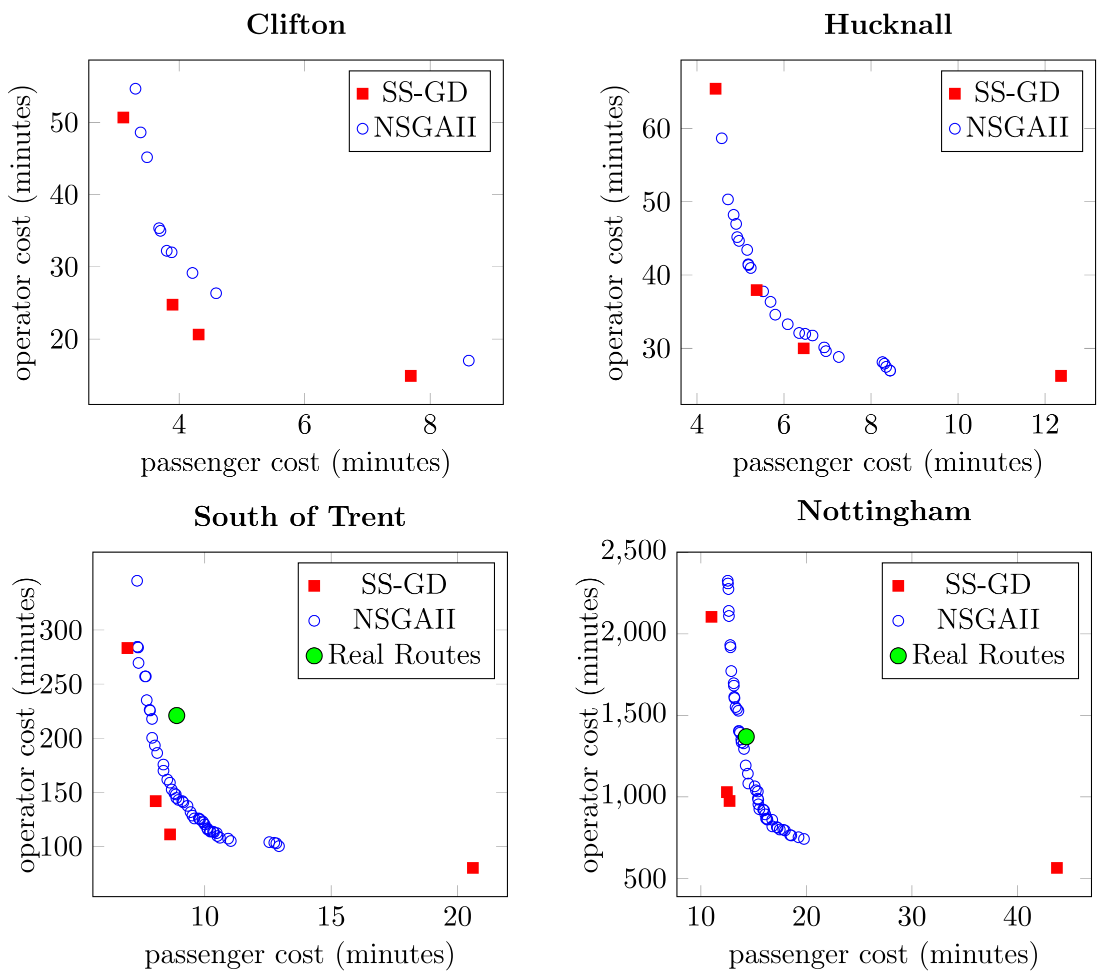


Figure 5.4: SS-GD results plotted against the evaluation results of the final population. The blue dots are the results of the population evaluation and the red dots are SS-GD results.

make these routes directly comparable to the results of our optimisation algorithm (i.e. more details in [84]). Table 5.4 summarises the results of this comparison. For this comparison we have selected the best results of SS-GD from the passenger and operator perspectives, and the most passenger friendly, and the most operator friendly route sets from the balanced trials. The percentages demonstrate the increase (positive sign), or the decrease (negative sign) made by SS-GD on the metrics of the passenger and operator costs and the statistics of the demand over the real world routes.

The results of Nottingham instance from the table shows a clear success of the balanced perspective results. In the most passenger and most operator friendly route sets, a high improvement rate was found in both objectives. The demand statistics show improvement in the directness of trips and the unsatisfied demand percentage has also decreased.

The plots in figure 5.4 indicate that SS-GD is able to provide improved solutions over the real routes, given that there are Pareto points (red) that clearly dominate the real route positions (green). Taking the Nottingham instance as an example, the real routes offer a single passenger an average travel time of 14.3 minutes and the summed route length for the entire route set in minutes is 1369. The best result from the passenger perspective found by SS-GD (11.00) decreased the average travel time by 3 minutes, while the average routes length increased by almost 50% (2105). On the other hand the most passenger friendly route set in the compromise route sets (12.46) improved the average travel time of the real routes by 2 minutes, and the routes length improved by almost 25% (1029). Also trip directness is enhanced by decreasing the percentage of passengers needing two transfers from 14% to 10% while increasing the percentage of direct travellers from 30% to 33%.

In South of Trent instance, the same success is noticed. The best passenger results improved the passenger cost by almost two minutes, and the most passenger friendly route set improved the passenger cost, and at the same time the operator cost was improved by 40%. The percentage of direct trips also improved by 3%.

5.2 Public Transport Network Optimisation in PTV Visum Using Selection Hyper-heuristics

Despite the success in the field of automatic public transport routes optimisation, there is a vast gap between the often purely academic studies and the application of their findings in real world planning processes. One reason for this might be the differences in data requirements between the algorithms used in the UTRP research and the commonly used planning tools. Researchers working on the UTRP apply their design algorithms to abstract models that simplify many aspects of real world transport networks, while models built with professional transport modelling software packages such as PTV's Visum [197] or INRO's Emme [198] have a high degree of detail in terms of the street network layout and infrastructure, and the travel demand data.

In this work, we bridge the gap between the two worlds of theoretical research on the UTRP and the real-world transportation planning, focusing specifically on Visum transportation modelling software. We apply selection hyper-heuristics to optimise the public transport routes in Visum, utilising interface procedures which have been implemented

Table 5.4: Comparison between the real world route sets and SS-GD best results from passenger and operator and balanced perspectives for Nottingham and South of Trent instances

metric	Real routes	SS-GD best		SS-GD balanced	
Nottingham					
C_P	14.3	11.00	−23.1%	12.46	−12.8%
C_O	1369	2105.1	+53.7%	1029.2	−24.8%
d_0	30.6%	46.6%	+16.0%	33.3%	+2.7%
d_1	54.1%	51.0%	−3.1%	55.4%	+1.3%
d_2	13.9%	2.3%	−11.6%	10.7%	−3.2%
d_{un}	1.4%	0.001%	−1.4%	0.53%	−0.87%
C_P	14.3	41.4	189%	12.71	−11.1%
C_O	1369	583.8	−57.4%	974.9	−28.8%
d_0	30.6%	11.5%	−19.1%	31.7%	+1.1%
d_1	54.1%	14.6%	−39.5%	54.8%	+0.7%
d_2	13.9%	15.1%	+1.2%	12.6%	−1.3%
d_{un}	1.4%	58.6%	+57.2%	0.77%	−0.63%
South of Trent					
C_P	8.89	7.07	−20.4%	8.23	−7.4%
C_O	220.9	286.1	+29.5%	134.1	−39.2%
d_0	50.8%	69.1%	+18.3%	53.5%	+2.7%
d_1	46.6%	30.2%	−16.4%	41.6%	−5.0%
d_2	2.5%	0.62%	−1.8%	4.8%	+2.5%
d_{un}	0.0%	0.0%	0.0%	0.02%	+0.02%
C_P	8.89	22.0	+147.4%	8.57	−3.6%
C_O	220.9	82.3	−62.74%	122.3	−44.6%
d_0	50.8%	27.6%	−23.2%	51.8%	+1.0%
d_1	46.6%	21.1%	−25.5%	42.4%	−4.2%
d_2	2.5%	18.0%	+15.5%	5.6%	+3.2%
d_{un}	0.0%	33.1%	+33.1%	0.07%	+0.07%

based on an extensive study on the UTRP and Visum network structures and their differences. The interface procedures aim to properly translate the network components between the two models, and hyper-heuristics are used as an optimisation tool to optimise Visum public transport routes through the interface model while taking advantage of Visum analysis tools to evaluate a given candidate solution.

In the following sections, we will briefly summarise the features of Visum transportation modelling software and the key differences between its network model, and the UTRP model explained in the previous chapters. We focus on the description of the hyper-heuristics implementation to optimise Visum public transport routes through the interface procedures. The details of the interface procedures can be found in our work [199].

5.2.1 Visum Transportation Modelling Software

5.2.1.1 History and Features

Visum is a macroscopic transport modelling software package that allows planners to analyse and plan robust models for transportation systems and to develop advanced transport strategies and solutions. It is currently used world-wide by traffic analysts with a set of powerful analysis tools, a smart and simple graphical user interface, and a set of complex built-in demand assignment procedures. Additionally, Visum provides a specific functionality for public transport to help analyse and evaluate an existing or a proposed public transport service from the passenger or operator perspectives. Visum is a product of PTV company based in Karlsruhe, Germany, and has been available for commercial use since the late 90's. The software helps in determining the impacts of existing or planned transport system, which can include both private and public transport lines. According to [200], the following transit features are supported by Visum:

- Visum offers a network model compatible with the Geographic Information System (GIS) as well as passengers information systems, and vehicle and crew scheduling systems. This made it possible to combine between geographical link network data, and timetable data in an integrated network model.
- Visum provides a fare model to estimate revenues from tickets sales. This model includes both zone-based and distance-based fares.
- Visum include many features that aids the design process, and help planners to test different scenarios (e.g., drawing line routes on the screen).
- Visum includes a set of assignment procedures which use search algorithms. This helps the planner to find the impacts on passengers by calculating essential performance indicators (e.g. journey time, waiting time, number of transfers, frequency of vehicles arrival) and travel costs for each origin-destination pair.
- Visum offers a unique feature of storing all passengers routes during the assignment. This feature helps in analysing and predicting passengers behaviours, and accordingly the estimations of fares and ticket sales.
- Visum supports both private and public transport, and can help in the design of public transport networks of several modes including train, bus, and tram networks.

In addition to the variety of analysis tools, one important feature that facilitated the use of Visum in this work, is the possibility to control it via python scripts over Visum COM-API. The Visum COM-API library provides a number of interface functions that allow controlling Visum via scripts and extract any required information. These scripts formed the basis for implementing the interface procedures.

5.2.1.2 Differences between Visum and the UTRP Network Models

As described in chapter 2, almost all previous approaches to the UTRP choose to represent the available street (or rail) network as undirected graph $G = \{V, E\}$, with the vertices V of the graph representing access and interchange points, and the edges E representing the direct connections between them. A public transport line (route) is represented by a set of directly connected vertices and is assumed bi-directional. The main advantage of such graphs for solving the UTRP, is that it allow us to limit the possible solutions to those with routes made up of directly connected vertices. This excludes many possible flawed solutions and thereby drastically reduces the search space.

The Visum network model is also based on a graph structure however, more detailed compared to the simple UTRP graph. The graph in a Visum network model is composed of a set of links connected through nodes. Links in Visum represent street, or rail segments and are usable by certain transportation modes. Each link is composed of two network objects for each direction of travelling. Nodes at the beginning and end of each link represent the positions of intersections and junctions in the network.

The PuT (i.e. public transport) interchange points in Visum which correspond to vertices in the UTRP model are defined in three layers: the level of stop points, stop areas, and stops. The highest level is the stops which incorporate several stop areas, and the middle layer are the stop areas which include several stop points. Similarly, PuT lines in Visum are also defined in a hierarchical fashion. At the top reside the line which represent a single PuT line belonging to one transportation system. The lowest layer is the line route, where each line aggregates two line routes for each direction of travelling.

From the above description, the key differences between the Visum and the UTRP network models, is that Visum PuT lines (i.e. routes) are directed, unlike in the simple UTRP model in which routes are assumed bi-directional. The second key difference is the representation of the interchange points in Visum as several layers of directed stop points and stops. This has made interfacing between UTRP algorithms and a macroscopic transport modelling software like Visum, a real challenge.

Our work in [199] handles these structural differences by implementing interface procedures that translate Visum directed line routes into undirected routes as in the UTRP model, and translating these routes back into Visum as line routes. Hyper-heuristics are then applied to optimise the translated bi-directional routes, and the interface is used to implement the changed routes into Visum for evaluation. There are two main interface procedures implemented in [199]: the first extracts a UTRP graph from a Visum network model by finding the adjacency relations between the vertices. This graph will be the basis for routes alteration during the optimisation. The second interface procedure translates lists of vertices into directed lists of stop points, one for each direction of travelling. This procedure is applied at each iteration of the optimisation algorithm to implement the changed routes in Visum for evaluation. Further detail of how the interface procedures work on the transformation between the two network models are in [199].

5.2.2 Selection hyper-heuristics for Optimising Visum Public Transport Lines

The motivation of this work is to use interface procedures integrated with hyper-heuristic as an optimisation tool to optimise Visum public transport routes, and use Visum capabilities and tools for conducting the evaluation.

There are many advantages of the selection hyper-heuristic framework that makes it a good candidate for application to this work. First, it is a single point based framework, meaning it only requires a single initial solution. This allows us to extract the existing public transport network from a given Visum network model and use it as the initial solution. Second, maintaining a single solution while improving it iteratively during the search, makes the the interaction with Visum through the interface procedures straightforward. Also the relatively short run time of hyper-heuristic methods significantly adds to their attractiveness.

Before integrating the optimisation process, there are vital questions that need to be answered: is the optimisation going to be accomplished on the level of lines or line routes in Visum? Which level of interchange points in Visum is going to represent the vertices in the UTRP network? Optimising the line routes individually can lead to significant deviations between the line routes that belong to the same line. This would not be desirable in practice. Therefore, we have chosen to perform the optimisation at the level of lines. We have also chosen the stops in Visum to represent the vertices of the UTRP, since they are undirected network elements, and therefore they better suit the bi-directional assumption of the routes in the UTRP model.

In this work, we have tested two selection methods: simple random (SR), and sequence based selection hyper-heuristic (SSHH). Both selection methods were combined with improve or equal (IE) acceptance method. The reason for applying these selection and move acceptance components, is that we wanted to test a relatively simple selection hyper-heuristics in this study to facilitate the interaction with Visum and to reduce the total time for running and testing multiple selection hyper-heuristics due to the complex objective evaluation procedures performed by Visum which require a significant computation time for each iteration compared to the evaluation performed in our previously described UTRP models. We have selected SR selection method because it provides a reference for comparison with other more complex selection methods. Furthermore, the SSHH method has proved its success on the UTRP over other simple selection methods as outlined in chapter 4, and therefore it was worth investigating in this study.

The optimisation process begins with initialising a set of routes from the Visum network model. This is done through the interface procedure which transforms Visum line routes into lists of stops (vertices) to construct an initial solution (S_{init}). The initial solution (S_{init}) is introduced to the hyper-heuristic as the current solution (S_{curr}), and the iterative optimisation of the single initial solution begins. One iteration of the SSHH algorithm is depicted in figure 5.5. Depending on the selection mechanism, either a single heuristic is selected, or a sequence of heuristics is selected and applied to S_{curr} to create the new solution S_{new} which is tested for its feasibility. If S_{new} is not feasible, it is rejected and a new heuristic/heuristics sequence is selected. Otherwise, S_{new} is converted into lists of stop points through the interface procedures and implemented in the Visum network model for evaluation.

With the necessary information generated in Visum, the objective function $f(S_{new})$ is calculated. If $f(S_{new}) \leq f(S_{curr})$, S_{new} replaces S_{curr} and becomes the new basis for finding new solutions. In case of the SSHH, the relevant values in the transition and the sequence construction matrices are updated. The hyper-heuristic iterates in generating new solutions, building them into Visum and evaluating them until a predetermined termination condition is met (will described in section 5.2.3).

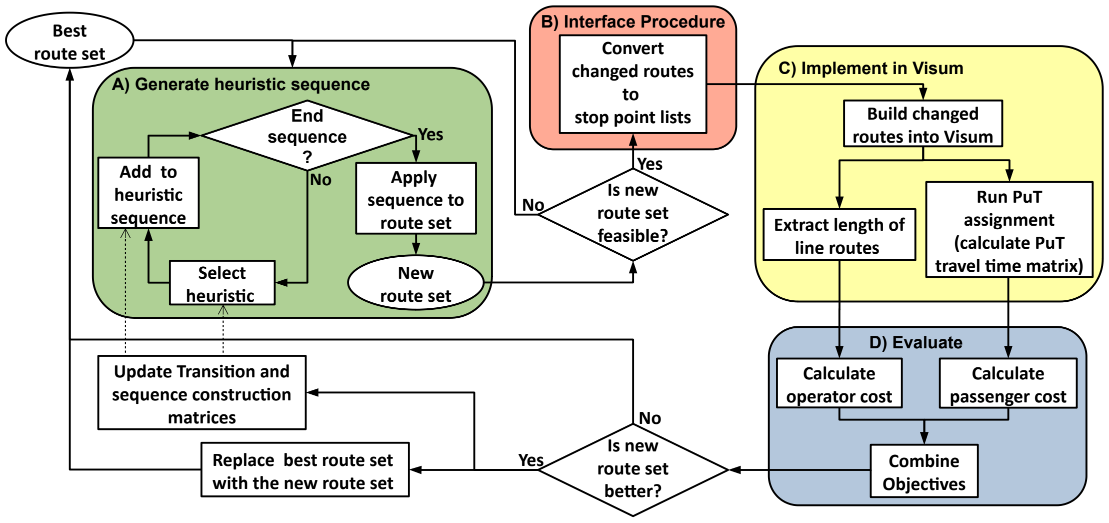

Best route set
A) Generate heuristic sequence
End sequence ?
Yes
No
Add to heuristic sequence
Apply sequence to route set
Select heuristic
New route set
B) Interface Procedure
Convert changed routes to stop point lists
Yes
Is new route set feasible?
No
C) Implement in Visum
Build changed routes into Visum
Extract length of line routes
Run PuT assignment (calculate PuT travel time matrix)
D) Evaluate
Calculate operator cost
Calculate passenger cost
Combine Objectives
Is new route set better?
No
Yes
Update Transition and sequence construction matrices
Replace best route set with the new route set

Figure 5.5: Description of one iteration of the SSHH algorithm application in the global optimisation. Each iteration begins with box A: The generation of sequence of heuristics and applying it to the current route set to create a new route set. The new route set is tested for its feasibility. If the new route set is feasible it is converted to stop point lists (Box B). The stop point lists are implemented in Visum as line routes and other necessary information for the evaluation are extracted (Box C). The evaluation includes combining the objectives of passenger and operator costs (Box D).

5.2.2.1 Low Level Heuristics

All the low level heuristics have been designed to follow the adjacency relations when performing operations. If applying the operation on the selected routes and positions would create invalid connections, new routes and positions are selected instead. This increases the chance of generating feasible solutions. The list of the applied low level heuristics is given below, it is also demonstrated by figure 5.6:

- **LLH0 (Add)**: Selects a random route and a random position in this route. A new vertex is selected and added in this position.
- **LLH1 (Delete)**: Selects a random route and random position and deletes the vertex in this position.
- **LLH2 (Swap Inside Route)**: Selects a random route and two random positions. The two vertices in these positions swap with each other.
- **LLH3 (Insert Inside Route)**: Selects a random route and two random positions. The vertex in the first position is inserted in the second position.
- **LLH4 (Swap Between Routes)**: Selects two random routes and two random positions on each of them. The vertices in these positions swap with each other.

- **LLH5 (Insert Between Routes)**: Selects two random routes and two random positions on each of them. The vertex in the first position of the first route is inserted in the second position of the second route.
- **LLH6 (Replace)**: Selects a random route and a random position. The vertex in this position is replaced by another selected vertex.
- **LLH7 (Exchange)**: Selects two random routes and splits them at a common vertex. The parts of the two routes are exchanged to create two new routes. If the selected routes do not have a common vertex, a new pair of routes is selected.
- **LLH8 (Extend Route)**: Selects a random route and adds vertices to the end of the route until reaching another terminal.
- **LLH9 (Reduce Route)**: Selects a random route and deletes vertices starting from the last vertex in the route until reaching another terminal node.

5.2.2.2 Feasibility and Evaluation

In section 5.2.2, we have demonstrated in the optimisation process that the solution must be feasible to be evaluated, otherwise the solution will be rejected. The feasibility constraints are defined here with respect to Visum network model. For instance, backtracks and cycles are tolerated in Visum, while they are considered a common violation in most of the UTRP models. Other constraints are similar to the common UTRP model, such as the start and end of each route at a terminal, following the adjacency relations in the ordering of vertices on the routes, and restricting the routes length within defined limits.

Two evaluation methods were used in our optimisation model: the global optimisation and the local optimisation. The former uses a travel time matrix generated from the perceived journey time [3]. The latter accesses the vehicle loads on selected links.

Global Optimisation

Global optimisation is the method used by the vast majority of the UTRP studies: an objective function that aggregates information from the entirety of the system. We have chosen to use the sum of two relatively simple components for our objective function.

[3]The perceived journey time defines all the time costs associated with a certain path. Such time costs include: in-vehicle travel time, transfers time between stop points, and transfers penalties.

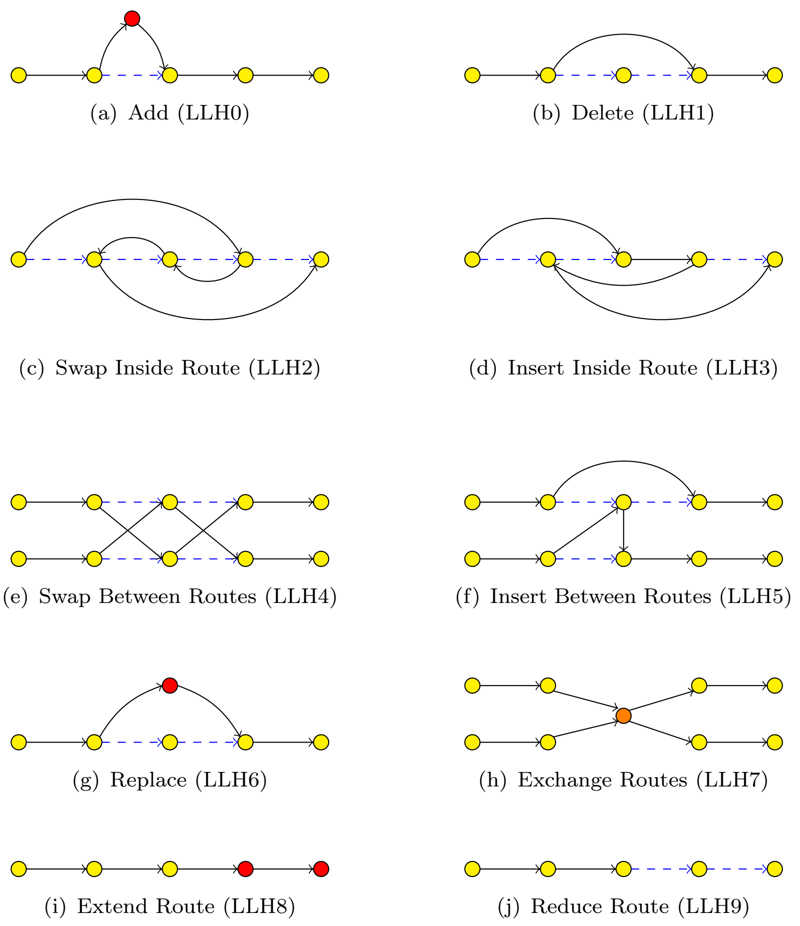

(a) Add (LLH0)
(b) Delete (LLH1)
(c) Swap Inside Route (LLH2)
(d) Insert Inside Route (LLH3)
(e) Swap Between Routes (LLH4)
(f) Insert Between Routes (LLH5)
(g) Replace (LLH6)
(h) Exchange Routes (LLH7)
(i) Extend Route (LLH8)
(j) Reduce Route (LLH9)

FIGURE 5.6: Low-level heuristics set description. Straight arcs are edges in the route or added after applying the heuristic, dashed arcs are edges removed after applying the heuristic, red nodes are nodes added after applying the heuristic.

The first objective is to reduce the passenger cost (i.e. the average perceived journey time of passengers). It is given by the following equation:

$$C_P(S) = \frac{\sum_{i,j=1}^{|Z|} D_{i,j} \cdot \Theta_{i,j}(S)}{\sum_{i,j=1}^{|Z|} D_{i,j}} \tag{5.2}$$

where $D_{i,j}$ is the PuT travel demand from a zone i to a zone j, and $|Z|$ is the total number of zones [4]. $\Theta_{i,j}$ is the shortest perceived journey time from zone i to zone j using the PuT network defined by the solution S.

The second objective is the reduction of the operator costs. We have used a simple approximation for the operators expenditures given by the total sum of travel times for

[4]A standard practice is macroscopic simulation softwares is to aggregate the demand at the level of zones.

travelling all the line routes in the PuT network:

$$C_O(S) = \sum_{i=1}^{|lr|} \tau_i(S) \tag{5.3}$$

where τ_i is the total travel time of line route i and $|lr|$ is the total number of line routes.

The two objectives are combined into a single objective function in the form of a weighted normalised sum given by the following formula:

$$f_{global}(S) = \alpha \frac{C_P(S)}{C_P(S_0)} + \beta \frac{C_O(S)}{C_O(S_0)} \tag{5.4}$$

where S_0 is the initial solution. The two weighting factors α and β can be adjusted in relation to one another to generate solutions that are more favourable for either operators or passengers and can take values in the range $[0 - 1]$.

Local Optimisation

The local optimisation represents a special optimisation experiment that takes advantage of Visum tools and capabilities to access results at a very localised level. The objective is to minimise the load of private cars on the selected links given by:

$$C_L(S) = \sum_{i=1}^{|L|} \nu_i(S) \tag{5.5}$$

$\nu_i(S)$ is the load of private cars on link $i \in L$ while the travellers in the network can choose between travelling via the public transport network defined by solution S, or by private cars.

5.2.3 Empirical Results

5.2.3.1 Test on Small Instance

For this set of experiments, we used a transport model from Viusm quick start tutorial which is loosely based on the small town of Pfullingen, Germany. The network model is relatively small, containing 652 nodes and 1782 links, 81 zones, 35 stops and stop points, and only five bus lines. The optimisation in these experiments is based on the global evaluation method described in the above section. The termination condition is defined by the number of successful iterations. A successful iteration consists of generating a new feasible solution, implementing it in Visum and evaluating it. The number of successful iterations before the hyper-heuristic terminates is set to 20000 iterations. In these set of experiments, we tested the two selection methods SR and SSHH in three distinctive scenarios: the passenger perspective, the operator perspective, and the balanced perspective. Each of these scenarios is defined by a different set of parameters in equation 5.4: For the operator perspective, effectively only the operator cost was considered as we set $\alpha = 10^{-6}$ and $\beta = 1 - 10^{-6}$. The opposite in the passenger perspective, where the focus is set on the passenger cost by setting $\alpha = 1 - 10^{-6}$ and $\beta = 10^{-6}$. In the balanced perspective, we create a balance between the two objectives by setting both parameters to $\alpha = \beta = 0.5$.

Figure 5.8 displays the change of the average passenger cost C_p(green line), average operator cost C_O (blue line), and the combined objective f_{global} (black line) calculated by equation 5.4. For each of the three scenarios the passenger and operator costs have been normalised using their initial values for better interpretation of their performance. The averages are calculated from the ten runs for each successful iteration.

From the figures, it is clear that from either the passenger or the operator perspective, the objective that the optimisation is focusing on decreases rapidly from its initial value, while the other objective increases. This improvement starts to slow down at the 2000 iterations stage. In the case of balancing the two objectives, both the passenger and operator costs show similar behaviour by dropping quickly at the beginning of the search and slowing down after 2000 iterations. This can be referred to how IE works. At the start of the search improvements can easily be found over the initial solution, but as the search progresses we require more diversification by accepting some of the worsening solutions which the IE functionality lacks. The SSHH selection method was more successful in improving the operator cost, while both selection methods reduced the passenger cost at a similar rate.

Table 5.5 summarises the results of the passenger and operator costs for the ten runs normalised and averaged. The best results and the standard deviation are also recorded. From this table, the most notable improvement is in the passenger perspective with a reduction of 20% in the passenger cost from the initial values, although this was at the expense of significantly increasing the operator costs. The operator perspective runs reduced the operator cost on average by almost 7%, and the balanced runs recorded an improvement of nearly 5% on the operator cost while the passenger cost is also improved by 3%. This implies that the approach of balancing the two objectives in this instance,

Table 5.5: Results from passenger, operator, and balanced configurations for the two selection hyper-heuristics. The results are normalised and averaged over the ten runs. The standard deviation and the best results are also recorded.

		SR-IE			SSHH-IE		
Scenario	Obj	avg	std	min	avg	std	min
Passenger	C_p	0.81	0.005	0.80	0.81	0.006	0.80
	C_o	6.96	1.07	5.12	7.49	1.55	6.46
Operator	C_p	1.03	0.004	1.02	1.03	0.004	1.02
	C_o	0.96	0.014	0.93	0.96	0.014	0.93
Balanced	C_p	0.98	0.003	0.97	0.96	0.008	0.96
	C_o	0.98	0.014	0.96	0.96	0.008	0.96

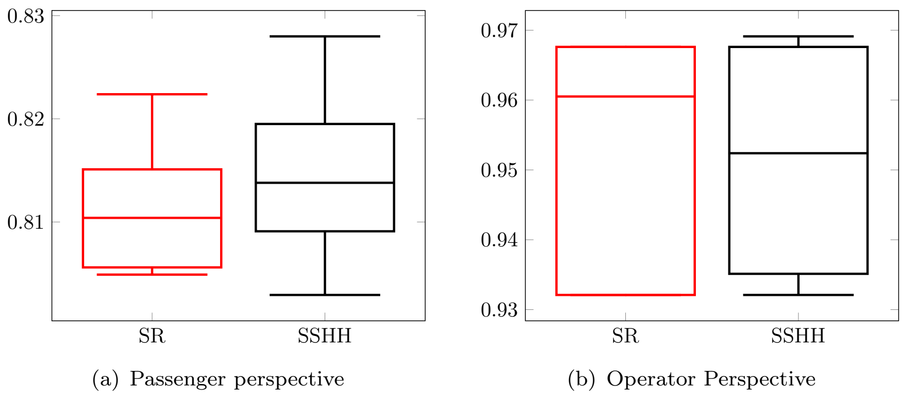
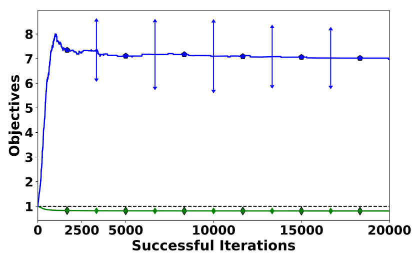


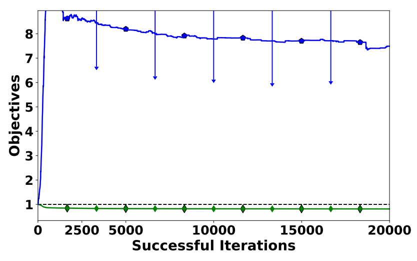
Figure 5.7: Box plots from 10 runs for the SSHH and SR selections combined with IE acceptance for the small network test case from the passenger and operator perspectives. Values in Y axis are the normalised passenger and operator costs.

is more successful than the operator perspective, with the operator cost decreasing at a similar rate, while the passenger cost has also improved.

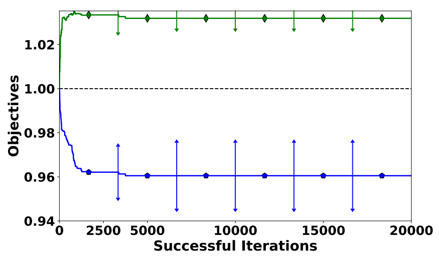
The performance difference between the two selection hyper-heuristics is very small as can be seen from the table. Figure 5.7 shows the performance variation between the two selection methods. SR was more successful in the passenger perspective scoring better average, while the SSHH found the best minimum result. The SSHH was clearly dominating in the operator perspective with better average and minimum values. The results from the table and figure 5.7 do not give a clear advantage for either of the selection methods. However, there were two observations made during the experiments: the SSHH selection method was able to improve more in fewer iterations compared to SR, and this fact is critical in working with larger networks. Second, the SSHH recorded better individual results for the runs in many cases, especially from the operator perspective and found the best minimum in the passenger perspective. Based on these



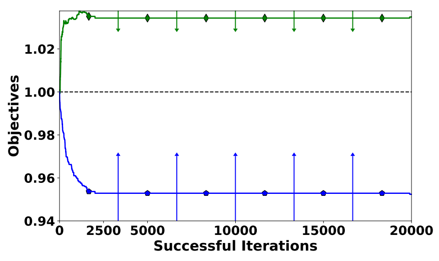
(a) SR passenger perspective



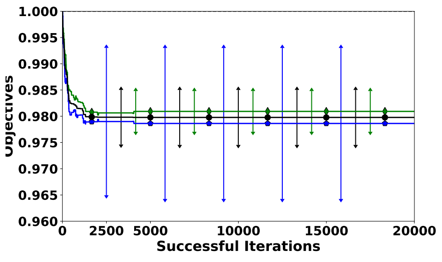
(b) SSHH passenger perspective



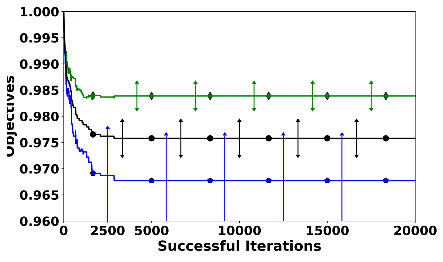
(c) SR operator perspective



(d) SSHH operator perspective



(e) SR balanced perspective



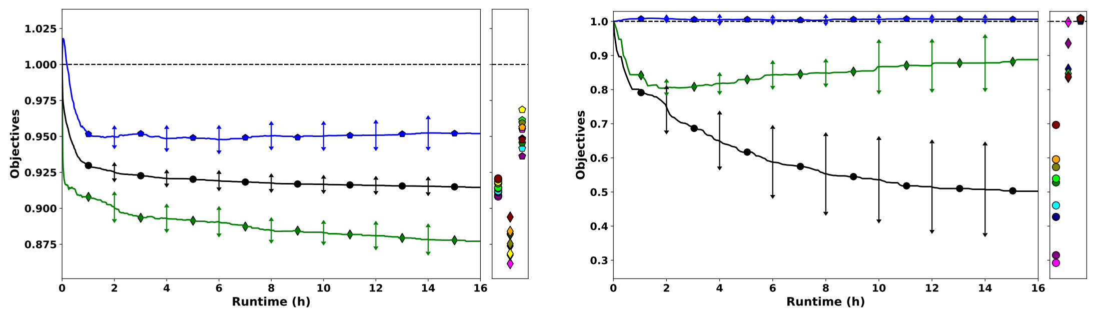
(f) SSHH balanced perspective

FIGURE 5.8: Results of the global optimisation on a small network model for three scenarios: passenger perspective, operator perspective, and balancing the two objectives using two selection hyper-heuristics (SR and SSHH). Each figure displays the development of the normalised passenger objective C_P averaged for ten runs (green with rectangles) the normalised averaged operator objective C_O (blue with pentagons) and the averaged combined optimisation function f_{global} (black with circles). The averages were calculated for every successful iteration from ten independent runs. The bars in the middle represent the standard deviation between the runs.

facts we have selected the SSHH to be applied in the next set of experiments on a larger network model.

5.2.3.2 Application on City Size Network and Local Optimisation

In order to show the validity of the proposed concepts in a larger scale, another set of experiments has been performed on a network model originating from a real-world



(a) Global Optimisation Halle (b) Local Optimisation Halle

FIGURE 5.9: Results for PuT line global (a) and local (b) optimisation with SSHH on a city-sized network. Displayed is the development of the normalised average passenger objective C_P (green with rectangles), the operator objective C_O (blue with pentagons), the combined optimisation function f_{global} (a), and the average reduction in the local objective C_L (b)(black with circles). The averages were calculated in steps of one minute from ten independent runs. The bars show the standard deviation between the runs. The markers at the right side bar show the distribution of the final values of the individual runs after 16 hours of run time (also shown in table 5.6). Each marker represents the final value of either f_{global} (C_L) (circles), C_P (rectangles), or C_O (pentagons) for each run. Each colour uniquely identifies one of the ten experiments.

planning process. The model was generated in the 1990s for the city of Halle, Germany. It is made up of 1934 nodes, 4832 links, 81 zones, 288 stops and 313 stop points, and in total 41 PuT lines of which 18 are bus lines. Although this model has been modified over time, its size and layout are still sufficient to represent a real-world network model.

The optimisation in this model is only conducted on bus lines, and the termination condition is set to run time rather than successful iterations, where each experiment is run for 16 hours before it terminates. This was done for practical reasons, and resulted in an average of 8919 successful iterations (i.e. on average 6.6 seconds per successful iteration). For the local optimisation, the same network model of Halle is used, and the links of an important connector street in the city were selected for the purpose of reducing the load of private cars. The termination criterion is also set to time, and each experiment is run for 16 hours. However, due to the more complex sequence of procedures required for the evaluation, the average length of a successful iteration during the experiments was 83.8 seconds, leading to an average number of successful iterations of 765. The experiments were performed on a device with the following specifications: Intel i3-4150 3.50GHz Dual Core CPU and 16GB RAM.

For the global optimisation experiments on this transport network, Ten runs were applied using the SSHH with the balanced configuration. This configuration was chosen as an example of a planning process which requires a compromise between passengers and

TABLE 5.6: Final values of ten runs with global and local SSHH optimisation on Halle network. Values normalised on initial values and minimal values are highlighted in bold.

	1	2	3	4	5	6	7	8	9	10	Avg
					Global Optimisation						
f_{global}	0.912	0.917	0.914	0.911	0.919	0.914	0.909	**0.908**	0.921	0.920	0.915
C_P	0.882	0.875	0.868	0.874	0.868	0.882	0.882	**0.862**	0.894	0.884	0.877
C_O	0.942	0.959	0.961	0.949	0.969	0.945	**0.936**	0.955	0.948	0.956	0.952
					Local Optimisation						
C_L	**0.292**	0.426	0.315	0.528	0.461	0.573	0.596	0.539	0.697	0.596	0.502
C_P	0.998	0.860	0.936	0.847	1.049	0.836	0.837	0.848	0.839	**0.833**	0.888
C_O	1.001	**1.00**	1.005	1.005	1.006	1.008	1.01	1.009	1.009	1.008	1.006

operators. Figure 5.9(a) shows the development of the average normalised passenger C_P objective (green with rectangles), the operator objective C_O (blue with pentagons), and the combined optimisation function f_{global} (black with circles). It can be observed that at the early stages of the search, the passenger objective steadily decreases, while the very early solutions show an increase in the operator cost. However, over the search time, the operator cost drops below its initial values but hovers around a value of 0.95, unlike the passenger cost which continues to drop until the end time of the search for most runs. After 16 hours, the passenger cost reaches on average a value of 0.877.

Table 5.6 displays the passenger objective C_P, operator objective C_O, and the global optimisation function f_{global} results for each individual run. The average of the runs is also recorded. The right side bar in figure 5.9 shows the distribution of this results, with different marker colours for each objective. We see that while for the operator cost C_O, the reduction is between 3.1% and 6.4%, for the passenger cost C_P larger reductions between 10.6% and 13.8% are achieved. Interestingly, the two runs with the highest reduction in C_O and C_P, respectively, are also the two best runs in terms of combined reduction of both objectives. The run reducing C_O by 6.4% reduced C_P by 11.8%, and the run which reduced C_P by 13.8%, also reduced C_O by 4.5%. This shows that improvements in both objectives are not mutually exclusive. It should be noted, that no global optimum could be found. The fact that the standard deviation between the runs does not change much over the latter half of the search indicates that this will not change much more even if the run time is increased.

In the local optimisation, ten independent runs were performed and the results are displayed in figure 5.9. The data lines show the time development of the average of the global passenger objective C_P (green with rectangles), and the operator objective C_O (blue with pentagons), respectively. The black line with circles shows the average of the local objective C_L. The bars show the standard deviation between the different runs. It can be seen from the figure that the main objective of these experiments, which is reducing the load of private cars on the selected links has been successfully achieved with an average reduction of C_L to 50.7% from its initial value. We also see a broad spread of solutions as the standard deviation between the results starts to increase with the progression of search time. For the least successful run, C_L was reduced by 30.3% while for the most successful one, the car load on the selected links is reduced by 70.8%, more than two thirds of the initial value. Further, we see that the average value of C_P drops at the beginning of the search, even more than that which occurred in the global optimisation in some cases. This suggests that the initial reductions of C_L are the result of decreasing the average travel time in general for all passengers and in turn increasing the attractiveness of public transport, causing a reduction in the use of private cars. The values of C_O are very similar in all the runs without any significant development. All runs end up with an increase in C_O between a minimum of 0.003% and a maximum of 0.96%.

In the experiments described above, the number of iterations before the SSHH terminates was limited by the time factor. In the global optimisation, the iterations were up to 9000 and in the local optimisation and due to the complex evaluation functions involved, the iterations were limited to around 900. We still see from the results and despite the short run times that the SSHH algorithm performed well on this problem. This reflects the efficiency and robustness of the involved online learning mechanisms, and the ability of this algorithm to learn in short duration to optimise a highly complex and large networks.

5.2.4 Summary

In this chapter, we proposed two problems to which the SSHH algorithm was applied. The first problem involved solving a version of the UTRP with terminal nodes information on a newly published set of instances with real world characteristics and size. An initial heuristic generation algorithm guided by demand information was implemented to ensure generating high quality route sets that obey the problem constraints. The SS-GD hyper-heuristic is tested on the new data set with specific implementation tailored to the presence of terminal points and compared to the solutions generated by NSGAII genetic algorithm and to the real bus routes used by local bus companies. Comparisons show the success of SS-GD in finding solutions better than NSGAII in all the instances from the perspective of passenger and operator. Also SS-GD was able to improve the existing routes service for both passengers and operators, showing a great potential for handling complicated and real world versions of the UTRP in very short run times compared to genetic algorithms.

The second application of the SSHH algorithm was the optimisation of the public transport routes in the transport modelling software package "Visum". The optimisation algorithm was integrated with a set of interface procedures that translate the network components between the UTRP network model and the Visum network model. Visum tools and capabilities were also utilised to conduct the evaluation during the optimisation. The SSHH algorithm and the simple random selection were applied and tested using two different optimisation modes: the global optimisation and the local optimisation. The results of the global and local optimisation showed the validity of hyper-heuristics in a small as well as a city-sized example network. In both cases high reduction rates in the passenger and operator costs are achieved simultaneously. Additionally, the local optimisation was tested on the city size network, reducing the rate of private car users on the targeted streets by up to 70%. This work opens a wide range of opportunities in real world transportation planning by integrating efficient optimisation algorithms such as selection hyper-heuristics with the capabilities of a transport modelling software such as Visum, therefore allow us to solve versions of the UTRP that are more useful and applicable to real world transportation systems planners.

Chapter 6

Population-Based Hyper-heurstic for the Delivery and Installation of Equipment

In this chapter we tackle a rich VRP problem integrating a capacitated vehicle routing problem with time windows (CVRPTW), and a service technician routing and scheduling problem (STRSP) for delivering various equipment based on customers' requests, and the subsequent installation by a number of technicians (Previously described in chapter 2 (section 2.5)). The main objective is to reduce the overall costs of hired resources, and the total transportation costs of trucks/technicians. The problem was the topic of the fourth edition of the VeRoLog Solver Challenge in cooperation with the ORTEC company. Our contribution to this research is the development of a novel hyper-heuristic algorithm to solve the problem based on a population of solutions. Experimental results on two datasets of small and real-world size revealed the success of the hyper-heuristic approach in finding optimal solutions in a shorter computational time, when compared to the results of an exact model specifically developed for this problem. The results of the large size dataset were also compared to the results of the eight finalists in the competition and were found to be competitive proving the potential of our developed hyper-heuristic framework. In this chapter we will provide the problem description and the data sets used, our hyper-heuristic methodology, and our full results, analysis, and the comparisons with the results of the developed mathematical model.

6.1 Description of the Problem

The real-world problem from VeRoLog Solver Challenge 2019 can formally be stated as follows. There are a number of locations $L = \{l_1, l_2, \dots l_L\}$ distributed at different geographical coordinates, each representing the home location of a customer, technician, and a depot [1]. The distance between any two locations (i.e., customer/technician/depot) l_i and l_j is given by d_{l_i,l_j}. A depot is located at $l_1 \in L$ where all trucks journeys start and end and all the machines to be delivered are located at the start of the planning horizon. A number of customers $Cr = \{cr_1, cr_2, \dots cr_{|Cr|}\}$ spread at different locations, each customer $cr_i \in Cr$ is located at a certain home location $l_{cr_i} \in L$ and has a request or a number of requests to be satisfied. The purpose is to respond to customers' requests by delivering machines and getting them installed by a technician within a defined time horizon $T = \{1, \dots A\}$ of A consecutive days.

An unlimited number of identical trucks (i.e. vehicles) $K = \{k_1, k_2, \dots\}$ can be hired to transport the machines to the customers. They are located at the depot each with a maximum capacity C. Also, a number of machines $M = \{m_1, m_2 \dots m_{|M|}\}$ are available to be delivered to the customers at their request, and there are different types of machines each identified by its size expressed in the same size unit as the truck capacity and a penalty value. The machines are all located in the depot, with enough machines to satisfy all the demand. A set of customers requests $R = \{r_1, r_2 \dots r_{|R|}\}$ should be satisfied. The requests are known at the start of the planning period. A single request $r_j = \{l_{cr_i}, w_j, m_k, n_k\}$ asks for one machine type $m_k \in M$, of quantity $n_k \in \mathbb{N}$, for exactly one customer at location l_i, and w_j is the associated time window of request r_j. w_j is specified by the earliest and the latest day $[e_j, l_j] \subseteq T$ to deliver request r_j. A request of the same type of machines cannot be split and should be delivered by the same truck, and if a customer requires another machine type, a separate request is made. Each truck journey on a day should start and end at the depot location l_1 and can carry different types of machines to satisfy several customers' requests, where request r_j occupies c_j of the truck capacity, and a single request should not exceed its maximum capacity C. The truck can return back to the depot location multiple times during the day to pick up more machines. Also, there is a limit D on the maximum distance a single truck can travel per day. It does not take any time to load a machine at the depot or to unload a machine at a customer.

[1]According to the description provided by the competition, each location has a unique identified ID, and the depot is always located at ID=1. A technician home location can be based at the depot, also a customer and a technician can share the same home location.

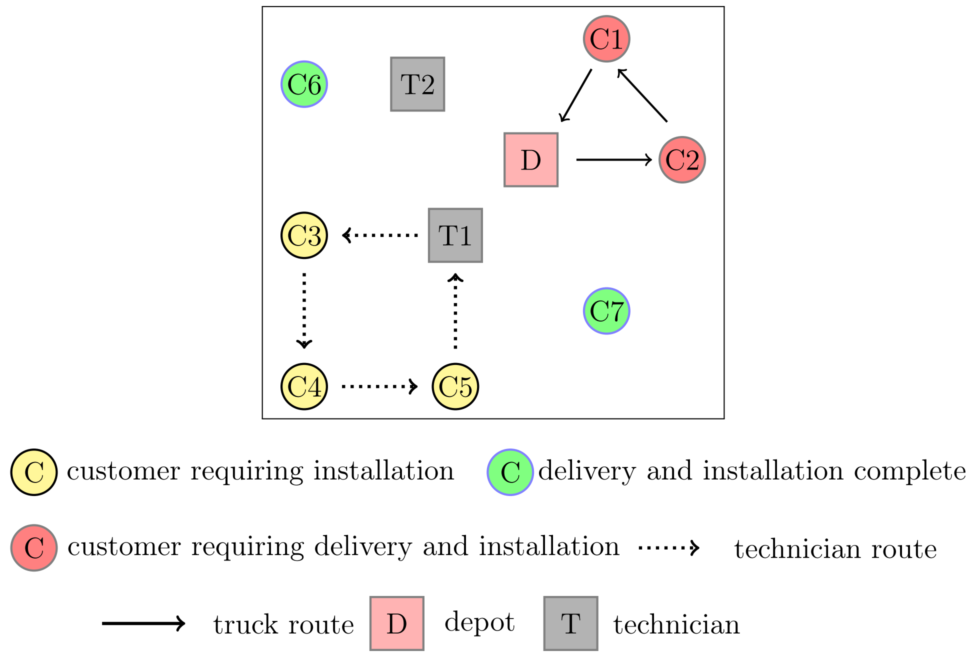

C1
C2
C3
C4
C5
C6
C7
T1
T2
D
C customer requiring installation
C delivery and installation complete
C customer requiring delivery and installation
technician route
truck route
D depot
T technician

FIGURE 6.1: Description of the customers states during the planning horizon. Dashed arrows display technician routes and solid arrows display truck routes

There is a fixed number of technicians $S = \{s_1, s_2 \dots s_{|S|}\}$ who are responsible for installing the delivered machines at the customer location the day after the delivery. Each technician $s_n \in S$ is located at a certain home location $l_{s_n} \in L$. A technician's daily route starts and ends at his/her home location, and like trucks, there is a maximum distance D_{s_n} that technician s_n can travel per day. In addition, there is a maximum limit N_{s_n} on the number of requests technician s_n can carry each day, where carrying out a request means installing all the machines for that request. The technician can maximally work for 5 consecutive days, and must have two days off if he/she has worked for 5 consecutive days.

Each technician has a skill set for installing certain types of machines. a_{s_n,m_j} refers to technician $s_n \in S$ installing machine $m_j \in M$, and is equal 1 if the technician is eligible to install this machine, and zero otherwise. Installing a machine does not take any time. A technician is described with the following entry:
$s_n = \{H_{s_n}, D_{s_n}, N_{s_n}, \{a_{s_n m_1}, a_{s_n m_2} \dots a_{s_n m_{|M|}}\}\}$ referring respectively to the technician home location, maximum travel distance per day, maximum number of installations per day, and which machines they have the skill to install.

The last point to mention, is that the technician should install a delivered machine as soon as possible after the delivery, and for each delayed installation of request r_j, a penalty CL_j is added to the cost, where each machine type has a different penalty value.

The main objective is to reduce the overall costs associated with trucks, technicians and idle machines costs. The trucks/technicians total cost is constituted of the following parts: the cost of hiring a truck/technician per day, the cost of hiring a truck/technician within the planning horizon T, and the cost per unit of distance for the travelling of truck/technician. The distance between coordinates (x_1, y_1) and (x_2, y_2) is defined as the ceiling of the Euclidean distance, $\lceil\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\rceil$. In addition, there is the cost for penalising idle machines that remain without installation for more than one day. This penalty cost is dependant on which machine it is and the number of days it was idle. The objective function is described by the following equation:

$$min \sum_{k=1}^{K} CV_k + \sum_{t=1}^{T} \sum_{k=1}^{K} CVU_{kt} + \sum_{t=1}^{T} \sum_{k=1}^{K} CVTd_{l_i l_j tk}$$
$$+ \sum_{s=1}^{S} CT_s + \sum_{t=1}^{T} \sum_{s=1}^{S} CTU_{st} + \sum_{t=1}^{T} \sum_{s=1}^{S} CTTd_{l_i l_j ts} + \sum_{i \in R} Cl_i b_i$$

Where

CV_k: the cost of using truck (vehicle) $k \in K$ in the planning horizon. This cost is equal zero for truck k if not hired during the planning horizon.

CVU_{kt}: the cost of using truck k on day t in the planning horizon. This cost is equal to zero for truck k if its not hired on day t.

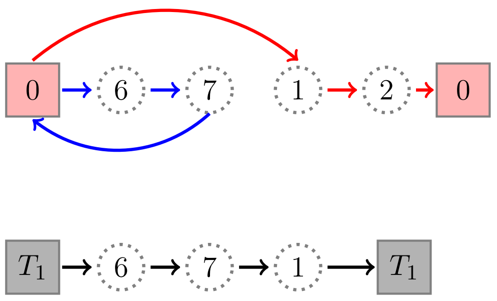
$CVTd_{l_i l_j tk}$: the cost of travelling unit distance by truck k on day t from location l_i to request l_j. This cost is equal zero for truck k if it has not travelled from location l_i to location l_j.

CT_s: the cost of hiring technician $s \in S$ during the planning horizon. This cost is equal zero for technician s if not hired during the planning horizon.

CTU_{st}: the cost of hiring technician s on day t in the planning horizon. This cost is equal zero for technician s if not hired on day t.

$CTTd_{ijts}$: the cost of travelling unit distance by technician s on day t from location l_i to location l_j. This cost is equal zero for technician s if he/she has not travelled from location l_i to location l_j.

$CL_i b_i$: the penalty of request i remaining without installation. b_i is the number of days request i remained without installation.



(a)

Day = 1
Number of Trucks = 2
1 6 7 0 1 2
2 3 4 5
Number of Technicians = 0

Day = 2
Number of Trucks = 1
1 8 10 0 9
Number of Technicians = 1
1 6 7 1

(b)

FIGURE 6.2: Solution example, and the routes of truck 1 in day 1, and technician 1 in day 2. The red and blue arcs represent two different tours. The pink and grey coloured boxes represent the depot and technician T_i respectively

The total cost is calculated as the weighted sum of the costs of travelling/hiring of the trucks and technicians during the planning horizon. The weights of the objective function components are defined with each instance, and their values vary depending on which cost the optimisation should be focusing to minimise.

6.2 Solution Format

A solution gives, for each day in the planning horizon, the routes followed by each truck/technician. Assuming that $\{1, 6, 7, 0, 1, 2\}$ is the route of a single truck in one of the planning days. The first element in the route '1' is the truck ID, followed by the requests ID's that this truck served. The ID '0' refers to the depot, and it means that truck 1 visited the depot after serving requests '6' and '7' and was loaded to serve requests '1' and '2'. Each series of requests before the truck goes back to the depot is named "tour". In this route, there are two tours given as $\{6, 7\}$ and $\{1, 2\}$. The start and end of the truck journey at the depot is not explicitly written in this route format.

The technician routes follow a similar pattern to the trucks routes, starting with the technician ID, followed by the ID's of the requests that this technician served. Also, the start and end of the technician journey at their home location is not explicitly mentioned in the solution.

6.3 Problem Instances

We have used two datasets of instances, the *small* dataset has been developed for this work, and the *hidden* dataset, was used in the competition to evaluate the participants' algorithms in the VeRoLog Solver Challenge. Each of the instances provides different types of information such as the weights of the objective function components, the maximum truck capacity, the number of days in the planning horizon, and the maximum travel distance allowed by each truck. The details of the requests, locations given as x, y coordinates (i.e. depot, technicians homes, customers) and technicians are also given. The characteristics of these datasets are provided in Table 6.1. The *small* dataset, which includes instances of sizes varying between 6 to 16 requests, is developed specifically for this work [2]. It is essential to test our developed hyper-heuristic approach on instances with different characteristics and ensure that it scales. It is also important to compare its performance to the exact model by its ability of finding optimal solutions in a short duration of time.

The hidden dataset was used to assess the performance of the competitors algorithms and rank the finalists in the restricted resource challenge[3]. This dataset contains instances of large sizes up to 900 requests. The number of different types of machines vary between 3 and 7 in each instance, and the number of technicians range from 25 to 125. The highest variation can be found in the costs of using trucks and technicians, distance costs, and the costs per day for using trucks and technicians. These values range from 10 to 100,000. We refer to [32] for a comprehensive description of the problem and the formal challenge rules[4].

6.4 Hyper-heuristics Methodology of CVRP for Delivery and Installation of Machines

In this section we describe the hyper-heuristic framework applied to this problem and discuss solution initialisation and representation and the low level heuristics set.

[2]A mathematical model has been developed for this problem and the small instances dataset was essential to test this model which only works on instances of such sizes.

[3]The solvers of the finalists were run on the hidden dataset for a limited computational times determined by the challenge rules.

[4]A detailed description of the challenge and the datasets is also provided here: https://verolog2019.ortec.com/

TABLE 6.1: Characteristics of the small and hidden instances

Instance	Days	Truck Capacity	Truck Max Distance	Truck Distance Cost	Truck Day Cost	Truck Cost	Technician Distance Cost	Technician Day Cost	Technician Cost	Machines	Locations	Requests	Technicians
Small_01	4	18	1090	10000	1000	1000	10000	10000	10	3	5	6	5
Small_02	5	18	905	100	10000	10	1000	100	100	4	8	8	10
Small_03	6	18	2000	10000	100	100	100	1000	1000	5	10	10	15
Small_04	7	18	2000	10000	10	1000	10000	10000	10000	6	9	12	20
Small_05	4	18	2000	1000	10000	10000	100000	100	100000	7	10	14	25
Small_06	5	18	2000	10	1000	100000	10	1000	10	3	10	16	5
Small_07	6	18	2000	100000	100	10000	10000	100	100	4	6	6	10
Small_08	7	18	1045	1000	10	1000	100000	10	1000	5	8	8	15
Small_09	4	18	970	10	100	100000	1000	100	10000	6	8	10	20
Small_10	5	18	2000	100	100000	10	100000	10	100000	7	10	12	25
Small_11	6	18	1195	100000	1000	100000	100000	100000	10	3	9	14	5
Small_12	7	18	980	1000	10000	1000	10000	1000	100	4	9	16	10
Small_13	4	18	2000	100000	10000	10000	100	10	1000	5	4	6	15
Small_14	5	18	465	100	1000	100	10	10	10000	6	8	8	20
Small_15	6	18	620	10	1000	10	1000	100000	100000	7	8	10	25
Small_16	7	18	775	100000	100000	100	10	100000	10	3	7	12	5
Small_17	4	18	2000	10	10000	10	1000	100000	100	4	9	14	10
Small_18	5	18	1135	100000	100000	100000	100	1000	1000	5	13	16	15
Small_19	6	18	830	100	100000	100000	100	1000	10000	6	8	6	20
Small_20	7	18	2000	1000	100	10000	100	10000	100000	7	9	8	25
Small_21	4	18	2000	100	100000	10000	10	100000	10	3	8	10	5
Small_22	5	18	980	1000	10	100	1000	10000	100	4	11	12	10
Small_23	6	18	2000	10000	10	100	10	100	1000	5	10	14	15
Small_24	7	18	960	10000	10	1000	100000	10000	10000	6	12	16	20
Small_25	4	18	2000	10	100	10	10000	10	100000	7	10	6	25
Hidden_01	15	18	1620	1000	1000	100	100	10	10	3	54	150	25
Hidden_02	25	18	1350	10	100000	1000	100000	1000	100	4	112	300	50
Hidden_03	35	18	1060	10000	1000	100000	100	1000	1000	5	163	450	75
Hidden_04	45	18	1040	10	10000	100	10	10000	10000	6	217	600	100
Hidden_05	55	18	2000	1000	100000	10000	100000	100000	100000	7	270	750	125
Hidden_06	15	18	1205	100	10	10	10	10	10	3	306	900	25
Hidden_07	25	18	980	1000	100000	1000	10000	1000	100	4	59	150	50
Hidden_08	35	18	1030	10000	100	100000	1000	100	1000	5	116	300	75
Hidden_09	45	18	2000	1000	100	1000	100000	100	10000	6	167	450	100
Hidden_10	55	18	950	10	10000	100000	1000	10000	100000	7	220	600	125
Hidden_11	15	18	2000	10000	10000	10	100000	100	10	3	254	750	25
Hidden_12	25	18	1405	10000	100	10000	1000	100000	100	4	310	900	50
Hidden_13	35	18	2000	100000	100	1000	10000	100000	1000	5	68	150	75
Hidden_14	45	18	1430	10000	1000	100000	100	100000	10000	6	117	300	100
Hidden_15	55	18	1350	1000	100	10	10	10	100000	7	173	450	125
Hidden_16	15	18	1170	100	100000	100	1000	10000	10	3	205	600	25
Hidden_17	25	18	2000	100000	10000	100	10000	100	100	4	260	750	50
Hidden_18	35	18	1435	100	100000	1000	10	10	1000	5	313	900	75
Hidden_19	45	18	1010	100000	10	10	10000	100000	10000	6	60	150	100
Hidden_20	55	18	1205	10	10	100	10000	1000	100000	7	125	300	125
Hidden_21	15	18	1230	100	1000	10000	1000	100	10	3	154	450	25
Hidden_22	25	18	1500	10	10	10000	100	1000	100	4	206	600	50
Hidden_23	35	18	1100	100000	1000	100000	100	10000	1000	5	266	750	75
Hidden_24	45	18	1290	100000	10	10	10	10	10000	6	317	900	100
Hidden_25	55	18	1160	100	10000	10000	100000	10000	100000	7	68	150	125

6.4.1 Population-based Hyper-heuristic Framework (POHH)

Following the description of the selection hyper-heuristic framework in chapter 3, most of the previously proposed solution methodologies in selection hyper-heuristics utilise a single solution during the search process and iteratively improve it, while some other methodologies adopt the idea of using a population of solutions during the search as a whole, or during some part of it. Our proposed framework is based on a population of solutions from which one of them will be selected and applied to a selection hyper-heuristic at each step in the search. We are motivated in this work to use a population of solutions as we believe that this provides diversity in the search and allows better exploration of new areas in the search space. Unlike the UTRP in which the population-based algorithms add a significant computational time with large instances due to the complex and costly operations involved in the evaluation. The evaluation step in this problem is relatively fast, thus applying selection hyper-heuristics on a multiple solutions setting does not necessarily mean that it will have the same impact as in the UTRP. For instance, we observed that on the large instances in the hidden dataset with more than 300 locations, an average of five million iterations were achieved on a single minute of run time, while on Mumford3 instance, less than one million iterations were achieved in one hour run time.

The process starts by initialising a number of solutions using a generation method to create an initial population $pop = \{sol_1, sol_2, \dots sol_{pop_{size}}\}$. A number of selection hyper-heuristics $HH = \{hh_1, hh_2, \dots hh_n\}$ combining different selection and move acceptance methods are implemented.

A solution sol_i and a selection hyper-heuristic hh_j are randomly selected from *pop* and HH respectively, where sol_i will serve as $S_{current}$ to hh_j. The selection hyper-heuristic hh_j selects a heuristic (or sequence of heuristics) and applies it to $S_{current}$ to create new solution S_{new}, which is checked for feasibility, and rejected if it is not feasible (i.e. violates at least one of the problem constraints listed in section 6.1). If S_{new} is feasible it will be evaluated and the decision of its acceptance is made by the move acceptance component of hh_j. The best found solution S_{best} is replaced by S_{new} if it is better. The iteration between the selection and acceptance components continues until the termination criteria are satisfied, that is until the global time limit is exceeded or when there is no improvement on the best obtained solution for a certain number of iterations.

After hh_j terminates, S_{best} is checked against the best found global solution S_{global} and replaces it if it is better. Afterwards, S_{best} is shuffled by randomly selecting and applying

Algorithm 6: Algorithm of the population based hyper-heuristic

```
1  Let $S_{current}, S_{new}, S_{best}, S_{global}$ be current, new, best and global solutions respectively;
2  Let $HH = [hh_1, hh_2, \dots, hh_n]$ be the combinations of selection hyper-heuristics;
3  Let $pop = [sol_1, sol_2, \dots, sol_{pop_{size}}]$ be the solutions in the population;
4  Let $LLH = [llh_1, llh_2, \dots, llh_{|LLH|}]$ be the set of low level heuristics;
5  $pop \leftarrow$ InitialGeneration();
6  repeat
7      $sol_i \leftarrow$ SelectRandomly($pop$);
8      $S_{current} \leftarrow sol_i$;
9      $S_{best} \leftarrow S_{current}$;
10     $hh_j \leftarrow$ SelectRandomly($HH$);
11     repeat
12         $llh \leftarrow Select(hh_j, LLH)$;
13         $S_{new} \leftarrow$ ApplyLLH($llh, S_{current}$) ;
14         if $Accept(hh_j, S_{new}, S_{current})$ then
15             $S_{current} = S_{new}$;
16         end
17         if $S_{current}$ $isBetterThan$ $S_{best}$ then
18             $S_{best} \leftarrow S_{current}$;
19         end
20     until $TerminationCriteria$;
21     if $S_{best}$ $isBetterThan$ $S_{global}$ then
22         $S_{global} \leftarrow S_{best}$;
23     end
24     $sol_i \leftarrow S_{best}$;
25     Shuffle($sol_i$);
26 until $timeLimit$;
27 return $S_{global}$;
```

a series of low level heuristics. The number of steps to shuffle a solution is tuned by the user, and is constant during the search. This shuffling is necessary in order to avoid the possibility of early convergence and to refresh the population. Next, a solution from the population and a selection hyper-heuristic are randomly selected and the same steps mentioned above are repeated. This iterates until the specified time for running an instance passes. Algorithm 6 outlines the applied framework.

For this framework we have tested a total of eight selection hyper-heuristics combining the selection methods: Simple Random (SR), Sequence-based Selection Hyper-heuristic (SS), and the move acceptance methods: Record-to-Record (RR), Naïve acceptance (Naïve), Great Deluge (GD), and Simulated Annealing (SA). We have chosen to apply two different selection methods for testing our approach, one represents the simplest form of selection, and the other is based on the concepts of online learning and selecting sequences of heuristics. On the other hand, all the move acceptance methods are non deterministic and accept moves that worsen solutions to help in escaping local optima.

6.4.2 Solution Representation and Feasibility

The described hyper-heuristic framework requires initialising a number of solutions to build a population, and this is achieved using an initial generation method. Each solution is composed of two main components: truck visits, and technician visits. The truck visits component corresponds to the schedule of trucks during the planning horizon which can be modelled as four levels: days, trucks dispatched on each day, tours performed by each truck, and the requests to deliver on each tour. Similarly, the technician visits correspond to the technicians' schedule composed of three levels: days, technicians scheduled on each day, and visits performed by each technician. The initial generation method randomly generates these schedules, while ensuring the final constructed solution is feasible. The main focus of the initial generation method is the feasibility of the solution and not its quality.

The feasibility of the solution must also be maintained during the hyper-heuristic operation. A feasibility test is implemented to ensure that the problem constraints still hold after each application of low level heuristic(s). A single violation of any of these constraints results in rejecting the solution. This test prevents the evaluation of too many infeasible solutions which can consume valuable search time.

6.4.3 Low Level Heuristics

The hyper-heuristic controls a total of twenty five low level heuristics to improve the quality of a given initial solution. These low level heuristics perform swap and insert operations for requests in truck and technician routes. Low level heuristics are restricted, as needed, to only produce routes that respect some of the constraints. For example, some low level heuristics perform operations between different days in the planning period; in this case if the operation involves delivery requests, the time windows of these requests must be respected and any installations that as a consequence violate the time windows constraints must be rescheduled. If it involves installation requests, the delivery of these requests must be ensured at least the day before.

- **LLH0**: selects a random day, a random truck route and a random tour, and swaps any two randomly selected requests on this tour.
- **LLH1**: selects a random day, a random truck route, and two different random tours, and swaps any two randomly selected requests on each tour.
- **LLH2**: selects a random day, two different random truck routes, two random tours from each route, and swaps two randomly selected requests from each tour.
- **LLH3**: selects two different random days, two random truck routes from each day, and two random tours from each route, and swaps two randomly selected requests from each tour.
- **LLH4**: selects a random day, a random technician scheduled on this day, and swaps two randomly selected requests of this technician.
- **LLH5**: selects a random day, two different random technicians scheduled on this day, and swaps two randomly selected requests of these technicians.
- **LLH6**: selects two different random days, and two random technicians, and swaps two randomly selected requests of these technicians.
- **LLH7**: selects a random day, a random truck route, a random tour, and two random positions on this tour. The request on the first position is inserted into the second position.
- **LLH8**: selects a random day, a random truck route, two different random tours on the selected route, and a random position on each tour. The request on the first position of the first tour, is inserted into the second position of the second tour.
- **LLH9**: selects a random day, two different random truck routes, a random tour on each route, and a random position on each tour. The request on the first position is inserted into the second position.
- **LLH10**: selects two different random days, a random truck route on each day, a random tour on each route, and a random position on each tour. The request on the first position is inserted into the second position.
- **LLH11**: selects a random day, a random technician scheduled on this day, and two random positions on the technician route. The request on the first position is inserted into the second position.

- **LLH12**: selects a random day, two different random technicians, and a random position on each technician route, and inserts the request on the first position into the second position.
- **LLH13**: selects two different random days, a random technician on each day, and a random position on each technician route. The request on the first position is inserted into the second position.
- **LLH14**: selects a random day, two different random truck routes, and a random tour on each route, and swaps the two selected tours.
- **LLH15**: selects two different random days, a random truck route on each day, and a random tour on each route, and swaps the two selected tours.
- **LLH16**: selects a random day, two different random truck routes, and a random position on each route. The tour on the first position is inserted into the second position.
- **LLH17**: selects two different random days, a random truck route on each day, and a random position on each route. The tour on the first position is inserted into the second position.
- **LLH18**: selects a random day, two different random truck routes, and two random positions. A block of consecutive requests starting at the first position is swapped with another block of requests starting at the second position. The size of the block is randomly selected.
- **LLH19**: selects two different random days, a random truck route on each day, and two random positions on each route. A block of visits starting at each of the positions are swapped with each other.
- **LLH20**: selects a random day and two different random technicians, and swaps two blocks of requests for these technicians.
- **LLH21**: selects two random different days and two random technicians from each day, and swaps two blocks of requests of these technicians.
- **LLH22**: selects a random day, and two different random truck routes. A block of requests is moved from the first route to the second route at randomly selected positions.

- **LLH23**: selects two different random days and two random truck routes. A block of requests is moved from the first route to the second route at a randomly selected positions.
- **LLH24**: selects a random day and two different random technicians, and moves a block of requests from the first technician to the second technician.
- **LLH25**: selects two different random days and two random technicians, and moves a block of requests from the first technician to the second technician.

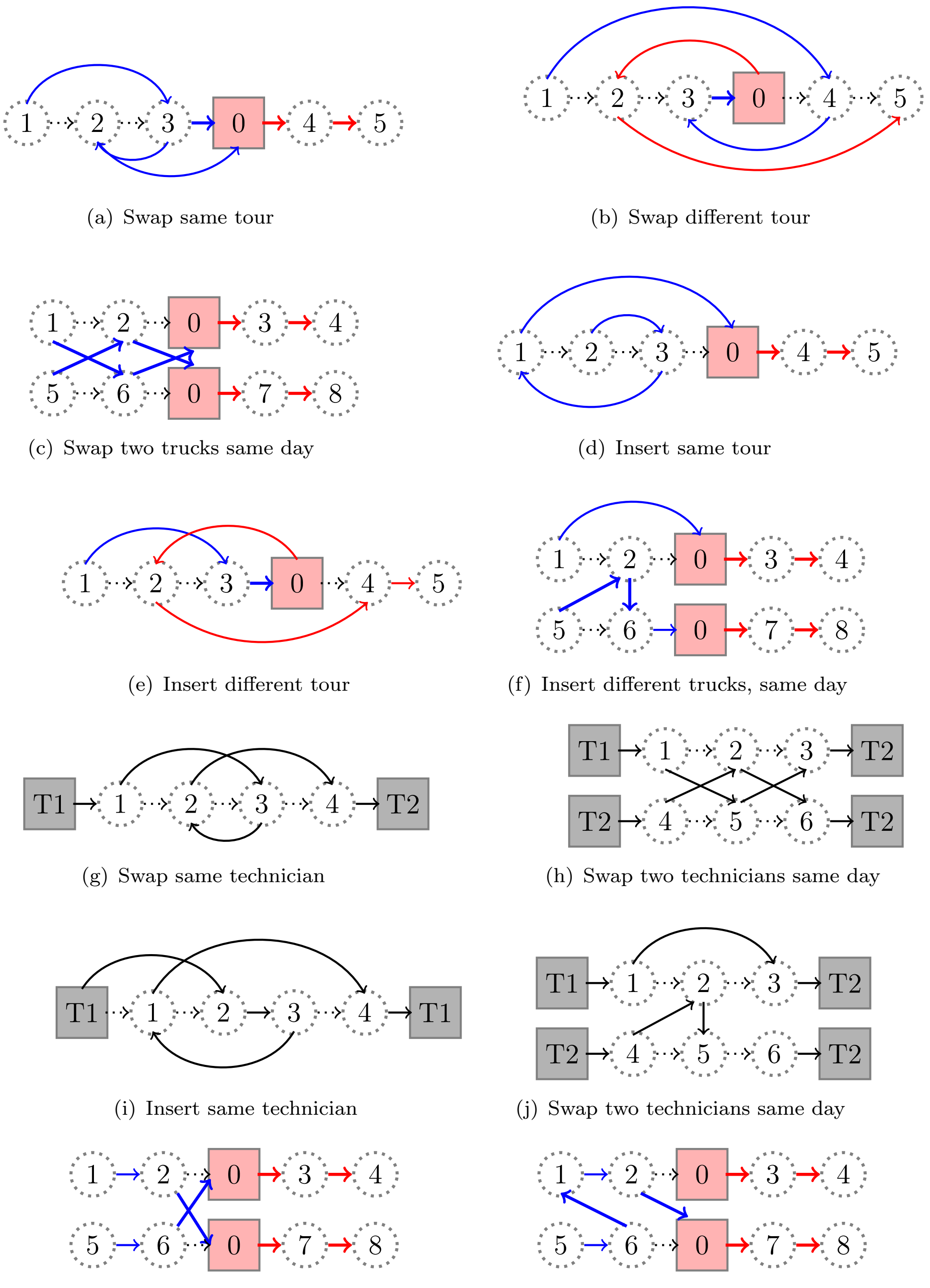
6.5 Experimental Results

The experiments for the heuristic method on the hidden dataset were performed on a device with the specifications: Intel Core i5 at 2.3GHz with memory of 8GB. On both datasets, the experiments were designed according to the competition rules, which required nine runs per instance with nine different random seeds also determined by the competition rules. The run time for each instance in both datasets was also calculated according to the competition rules, where it has been specified that each instance is run for a limited time on the user machine calculated with the formula: $T_{limit} = fb \times (10 + |R|)$, where T_{limit} is the time limit for running an instance according to the user local machine, fb is a factor calculated by a benchmark tool provided by the competition to estimate the equivalent time on any machine compared to the organisers core machine, and $|R|$ is the number of delivery requests in the instance. The POHH algorithm involves several design parameters that are set the by the user. To tune these parameters, a manual approach is followed, where a series of extensive experiments were performed to fine tune the design parameters. We arrived at a combination of parameter values that resulted in a relatively better performance across a subset of public instances [5]. The values of these parameters are shown in table 6.2.

6.5.1 Results on the Small Dataset

The small dataset was tested on an exact mathematical model, specifically formulated for this problem, and on our hyper-heuristic method. The details of the mathematical

[5]The parameters tuning was tested on another set of instances provided by the competition named "Late dataset". This set was published by the challenge organisers to help competitors test their solvers before submitting them. The characteristics of this dataset are similar to the hidden dataset.



(a) Swap same tour

(b) Swap different tour

(c) Swap two trucks same day

(d) Insert same tour

(e) Insert different tour

(f) Insert different trucks, same day

(g) Swap same technician

(h) Swap two technicians same day

(i) Insert same technician

(j) Swap two technicians same day

(k) Swap tours same day

(l) Insert tour same day

FIGURE 6.3: Some selected low level heuristic descriptions. Blue and red arrows represent two different tours. Dashed arrows are edges removed by the application of the low level heuristic

Table 6.2: The algorithm parameters and the chosen values

Parameter	Tuning
Population size (pop_{size})	2-5
Limit on iterations without improvement	10^5
Naïve acceptance probability	0.1
RR factor (fr) starting value	10
Number of iterations to shuffle solutions	10

model are found in our work [201]. Table 6.3 provides the results of the small instances dataset for the exact and the hyper-heuristic method. For the exact model, the upper and lower bounds are provided for each instance. The lower bound indicates that an optimal solution was not found for a particular instance. The results of the hyper-heuristic experiments are reported in terms of the minimum and maximum objective values achieved in the nine runs, the average of the nine runs and the standard deviation. The time in seconds is the time that was required to find the reported results by the exact model, and the time limit for each run of POHH. The time was normalised to its equivalent in the standard machine using the calibration tool provided by the competition.

From Table 6.3 we can directly compare the performance of our algorithm to the results of the exact model in terms of finding optimal solutions and the time required to find them. By comparing the results of the exact model to the minimum value in the nine runs, POHH was able to find the optimal solutions in twelve instances. These are the same set of instances where the exact model successfully found an optimal solution . In the other cases where no optimal solution was proved by the exact model, the POHH algorithm either found the same upper bound or better than this upper bound in seven instances cases. A feasible solution for Small_09 was found, while the exact model failed to find any feasible solution for this instance. Although the POHH algorithm was able to find the exact upper bound found by the exact model in many instances cases, we cannot argue with certainty that this solution is the optimal solution of these instances. In terms of run time, POHH achieved improved run times in most of the cases. The exact model in many instances required more than 3000 seconds to find a solution, while POHH required less than 30 seconds on these instances. The run time for the small instances was calculated using the competition rules and ranged between 16 to 26 seconds. The POHH algorithm was able to find better results in some instances than the lower bound found by the exact model in this short duration of time.

Table 6.3: Summary of the results. The table shows upper bound (or optimum), lower bound (if not optimum), percentual deviation, solution time (in second)

	CPLEX				Gurobi				POHH				
Instance	Upper Bound	Lower Bound	Deviation	Time	Upper Bound	Lower Bound	Deviation	Time	Min	Max	Avg	Std	Time
Small_01	36,016,020			5	36,016,020			0	36,016,020	36,016,020	36,016,020	0	16
Small_02	1,786,430			10	1,786,430			2	1,786,430	1,786,520	1,786,440	30	18
Small_03	32,085,900	14,442,700	55	1,800	32,085,800	21,461,179	33	1,800	32,085,800	32,086,000	32,085,922	83	20
Small_04	18,816,347	14,825,629	21	1,807	18,816,347			1,797	18,816,347	18,816,555	18,816,393	92	22
Small_05	128,025,356	112,613,264	12	1,800	128,124,700	112,490,056	12	1,800	128,014,700	128,015,356	128,015,034	237	24
Small_06	180,524	154,910	14	1,805	180,524	171,931	5	1,800	180,524	180,524	180,524	0	26
Small_07	239,350,700			92	239,350,700			27	239,350,700	239,350,700	239,350,700	0	16
Small_08	66,877,075	34,378,156	49	1,800	66,877,075			408	66,877,075	66,878,040	66,877,826	426	18
Small_09	Infeasible Solution			1,800	Infeasible Solution			1,800	1,073,410	1,073,510	1,073,421	33	20
Small_10	133,887,780	133,873,627	0	1,800	133,887,780			270	133,887,780	133,887,790	133,887,789	3	22
Small_11	537,212,505	341,632,540	36	1,800	537,613,730	363,037,030	32	1,800	537,212,505	537,213,820	537,212,866	577	24
Small_12	16,209,060	6,570,854	59	1,800	16,205,455	6,528,676	60	1,800	16,203,435	16,205,395	16,204,949	581	26
Small_13	221,514,230			202	221,514,230			367	221,514,230	221,514,230	221,514,230	0	16
Small_14	211,980	101,866	52	1,802	212,005	113,930	46	1,800	211,980	212,080	212,003	35	18
Small_15	2,148,165			122	2,148,165			33	2,148,165	2,151,980	2,149,312	1,442	20
Small_16	333,416,760	138,661,467	58	1,800	333,416,760	158,241,503	53	1,800	333,416,760	333,422,700	333,418,473	2,203	22
Small_17	758,620	615,961	19	1,810	758,620	660,245	13	1,800	758,620	758,620	758,620	0	24
Small_18	480,118,510	194,470,467	59	1,842	479,818,890	219,958,625	54	1,800	479,817,740	479,821,390	479,819,453	969	26
Small_19	877,960			7	877,960			0	877,960	878,320	878,080	180	16
Small_20	1,763,630	1,413,949	20	1,802	1,763,630			528	1,763,630	1,766,945	1,764,367	1,462	18
Small_21	1,074,748	893,161	17	1,802	1,074,748			1,255	1,074,748	1,074,748	1,074,748	0	20
Small_22	8,326,840	6,880,399	17	1,802	8,326,840	7,542,245	9	1,800	8,326,840	8,326,840	8,326,840	0	22
Small_23	35,499,570	23,517,476	34	1,802	35,499,570	27,966,620	21	1,800	35,499,539	35,499,580	35,499,563	19	24
Small_24	182,073,280	130,142,469	29	1,805	182,073,280	132,738,156	27	1,800	182,072,650	182,099,930	182,076,697	8,778	26
Small_25	3,499,190			785	3,499,190			10	3,499,190	3,499,190	3,499,190	0	16

6.5.2 Results on Hidden Dataset

As mentioned previously, the hidden dataset was used to assess the competitors' algorithms in the restricted resources challenge, and according to the results of this challenge, eight teams were selected as finalists. We have followed the same competition rules as the other teams who were qualified as finalists with this set of experiments, to fairly compare and justify our results to the finalists of the competition.

Table 6.4 summarises the results of the top six teams, including our hyper-heuristic approach, for each instance ranked based on their best found solution from best to worst. For each instance, the results are reported for the best six teams out of nine using the average of the nine runs and the minimum objective value.

Considering the minimum and average objective values obtained over nine runs for each hidden instance, POHH is relatively competitive between the finalists. The POHH achieved a position in the top six in 15 instances out of 25. For 11 out of 25 instances, including: Hidden_02, Hidden_07, Hidden_09, Hidden_11, Hidden_13, Hidden_16, Hidden_17, Hidden_19, Hidden_21, Hidden_22, Hidden_25, POHH performs better than *at least* half of the finalist approaches in terms of average objective value. Except the instances Hidden_13 and Hidden_17, the same phenomena is observed with those instances with respect to the minimum objective values. The best achieved results are found on instances Hidden_07, Hidden_11, Hidden_16, and Hidden_21, where POHH is ranked the fourth based on both the average and the minimum objective values. These instances are all of different sizes: 150, 450, 600, and 750 requests, reflecting the ability of POHH to perform well on instances with varying characteristics and complexities.

We have also ranked our approach amongst the eight finalists using the same method used in the competition. A ranking score is calculated per instance for each submitted solver by removing the two best and worst solutions found by this solver. We then take the average objective value of these five solutions as score for the algorithm, and rank all methods accordingly. The average of all ranking scores for the instances represents the final mean rank of the solver, which was used to order the competitors from the first to the last. Figure 6.4 displays the ranking of the POHH algorithm among the eights finalists based on this method. It is clearly seen that our method was able to produce results competitive with the finalists by achieving a better final mean rank than three teams, and an insignificant difference from the ranks of the third team [39], fourth, and fifth teams.

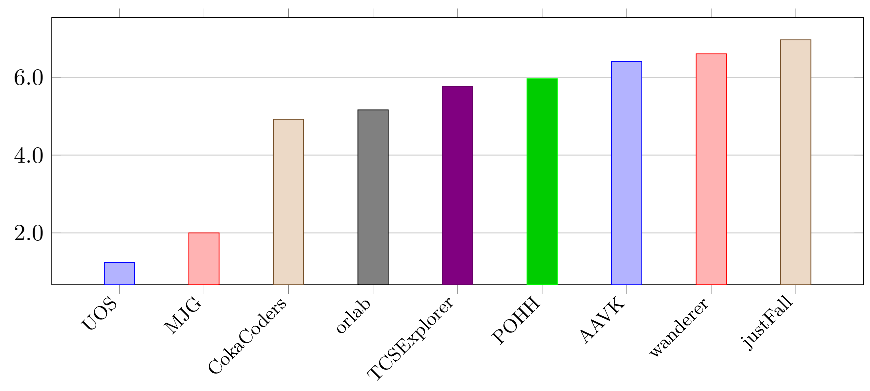


FIGURE 6.4: Mean ranks of the finalists teams and the POHH algorithm computed according to the competition rules

Although the proposed algorithm did not succeed in improving any current best known solutions on hidden instances, it performed well (see Section 6.5.1) on small instances with few requests, and the results on the hidden instances are considered reasonably good.

6.5.3 Performance Analysis of POHH

Figure 6.5 visualises the six sample instances, including Small_01, 03, 06 and Hidden_01, 03, 06 that we used for the analysis purposes, reflecting the varying characteristics of each instance. These instances were arbitrarily chosen to represent varying sizes from each of the data sets. Yellow locations indicated by small circles have technicians, and they might or might not have requests. Green locations have requests, but no technicians. The location of the depot is coloured in light blue, which might have technicians. Each location is shown with different sizes (diameters). The large diameter is used when the location is open. The depot is open throughout the planning horizon and each request location is open on the days specified. The figure shows only the beginning of day 1 of each instance. Figure 6.5 also shows semi-transparent green circles around the locations with technicians. The radius of these circles is equal to the half of the maximum daily distance of the corresponding technician. Because these circles are semitransparent, overlap leads to a darker colour, making visually clear which customer locations are within reach of few or many technicians. One may notice from the clustering, for example, as for Hidden_06, that the locations are making the shape of the Netherlands. This implies that these are indeed based on real-world data of locations offered by ORTEC.

TABLE 6.4: The performance comparison of the finalists and POHH, based on the average, and minimum of the objective values over 9 runs for each instance. The top six methods per each instance are reported and best values are highlighted in bold

Instance	Team	Min	Avg	Instance	Team	Min	Avg	Instance	Team	Min	Avg
Hidden_01	UOS	**67303070**	**67347510.0**	Hidden_10	UOS	**31425420**	**31615172.8**	Hidden_19	UOS	**4379062260**	**4379503211.1**
	MJG	67539160	67768841.7		MJG	32134970	32406070		MJG	4379599620	4384265171
	CokaCoders	67936410	68239128.3		TCSExplorer	35602350	36296800		CokaCoders	4385514720	4416094113
	AAVK	68049230	68497476.7		CokaCoders	41951125	44037471.7		orlab	4390908575	4397116298
	orlab	68059355	68531090.6		orlab	42557550	43110900.6		POHH	4400633465	4433562786
	TCSExplorer	68067705	68669280		POHH	44757390	46971126.1		justFall	4415787735	4460775575
Hidden_02	UOS	**866780485**	**884328838.3**	Hidden_11	UOS	**4052063633**	**4143464703.2**	Hidden_20	UOS	**126020890**	**127117758.9**
	MJG	878698335	887583730		MJG	4104065729	4150825522		MJG	128220285	130004861.7
	TCSExplorer	940755820	966644283.9		TCSExplorer	4490010535	4573692086		TCSExplorer	138750815	141489971.1
	CokaCoders	978877270	992970729.3		POHH	4792051226	4947603379		CokaCoders	143190180	146078606.1
	POHH	999703125	1070465586		justFall	4933454022	5124240633		orlab	164537160	167708532.2
	justFall	1178709770	1289905508		AAVK	5050880606	5176963402		POHH	171503820	177538151.1
Hidden_03	UOS	**1353070685**	**1356206683.3**	Hidden_12	UOS	**2985079895**	**2988919325.0**	Hidden_21	UOS	**33041255**	**33217240.6**
	MJG	1358875910	1363989363		MJG	2985608315	2989299754		MJG	33313460	33871058.9
	CokaCoders	1362273745	1374383989		CokaCoders	2998137785	3062709553		TCSExplorer	35525950	36048057.8
	orlab	1365243450	1373092383		orlab	3020441765	3030703193		AAVK	38119685	38845431.7
	AAVK	1388162220	1394947955		wanderer	3056711365	3059281502		POHH	38347205	38839638.3
	justFall	1389700390	1407156949		justFall	3072299875	3086846959		justFall	40036785	41366946.1
Hidden_04	MJG	**5354045**	**5382928.3**	Hidden_13	UOS	**5237955246**	**5239090235.3**	Hidden_22	MJG	**6642535**	**6677298.3**
	UOS	5407020	5470823.3		MJG	5243444173	5270924784		UOS	6700250	6750354.4
	TCSExplorer	5782480	5973581.1		CokaCoders	5251322598	5267097465		TCSExplorer	6986000	7020108.9
	AAVK	6053945	6122060		orlab	5273032383	5300095233		AAVK	7473115	7575487.8
	orlab	6258020	6409908.3		justFall	5284251162	5322430964		POHH	7671390	7831162.8
	POHH	6750245	6812897.2		POHH	5289271553	5312240754		orlab	7777350	7896450.6
Hidden_05	UOS	**2420668237**	**2438943861.4**	Hidden_14	UOS	**1378863050**	**1380531068.9**	Hidden_23	CokaCoders	**22341696590**	22374478803
	MJG	2461219726	2472452759		orlab	1385370725	1392172054		UOS	22342274615	**22368503380.6**
	CokaCoders	2793126594	2824641910		CokaCoders	1385514390	1387323213		MJG	22473152460	22489158018
	TCSExplorer	2873811043	2900849794		MJG	1386071905	1389987112		orlab	22564706605	22644343334
	wanderer	3463496082	3463496082		AAVK	1394884865	1403511603		justFall	22803650615	22946341903
	orlab	3489050112	3582195247		justFall	1399908710	1414358900		AAVK	22946775355	23021877311
Hidden_06	MJG	**32919590**	**32950587.2**	Hidden_15	UOS	**163646890**	**163861015.0**	Hidden_24	UOS	**31350467425**	**31373781566.1**
	UOS	33035255	33122941.7		orlab	164986370	165539965.6		CokaCoders	31386137725	31441905918
	orlab	33243100	33310187.2		MJG	165442970	165746732.2		orlab	31457461925	31554867014
	AAVK	33537475	33642695		AAVK	166559140	167109473.3		MJG	31502515080	31579529472
	TCSExplorer	33730430	33798883.1		wanderer	167271785	167271785		justFall	31915258830	32102001066
	wanderer	33799140	33799140		TCSExplorer	168523765	168889354.4		wanderer	31945701195	31947545553
Hidden_07	UOS	**102098250**	**102289614.4**	Hidden_16	MJG	**52730075**	**52860556.7**	Hidden_25	UOS	**549505255**	**549854756.1**
	MJG	102375745	103005780.6		UOS	53232615	53561470.6		MJG	552735110	563054914.4
	CokaCoders	107548420	110818258.9		TCSExplorer	58750440	59286328.3		CokaCoders	586771940	605819273.9
	POHH	108198340	109634304.4		POHH	59861645	60499887.8		TCSExplorer	611102855	621490488.9
	TCSExplorer	108308895	110359084.4		justFall	61176575	63036133.3		POHH	625293110	652332916.1
	justFall	121495535	132836498.3		AAVK	61626085	63624953.9		orlab	659488500	678896693.9
Hidden_08	UOS	**728784055**	**729462820.6**	Hidden_17	UOS	**27301086592**	**27322051463.0**				
	MJG	729588325	732733357.2		MJG	27387159376	27426339743				
	CokaCoders	734907040	737740395		CokaCoders	27396284273	27996989611				
	orlab	735203675	737068948.3		orlab	27486127713	27525238097				
	AAVK	743077790	750319370.6		justFall	27717856052	27822517277				
	justFall	746651030	757387192.2		POHH	27867305050	27914747560				
Hidden_09	UOS	**1692627537**	**1713909634.1**	Hidden_18	MJG	**52602450**	**52658013.3**				
	MJG	1732570744	1746683152		UOS	52921995	53040623.3				
	CokaCoders	1884282890	1939897718		AAVK	54778115	55313427.2				
	TCSExplorer	1964621385	2033485372		TCSExplorer	54906655	55136475				
	POHH	2375666512	2439427249		justFall	55108940	58268698.3				
	justFall	2509944603	2639808947		POHH	55585100	56052686.1				

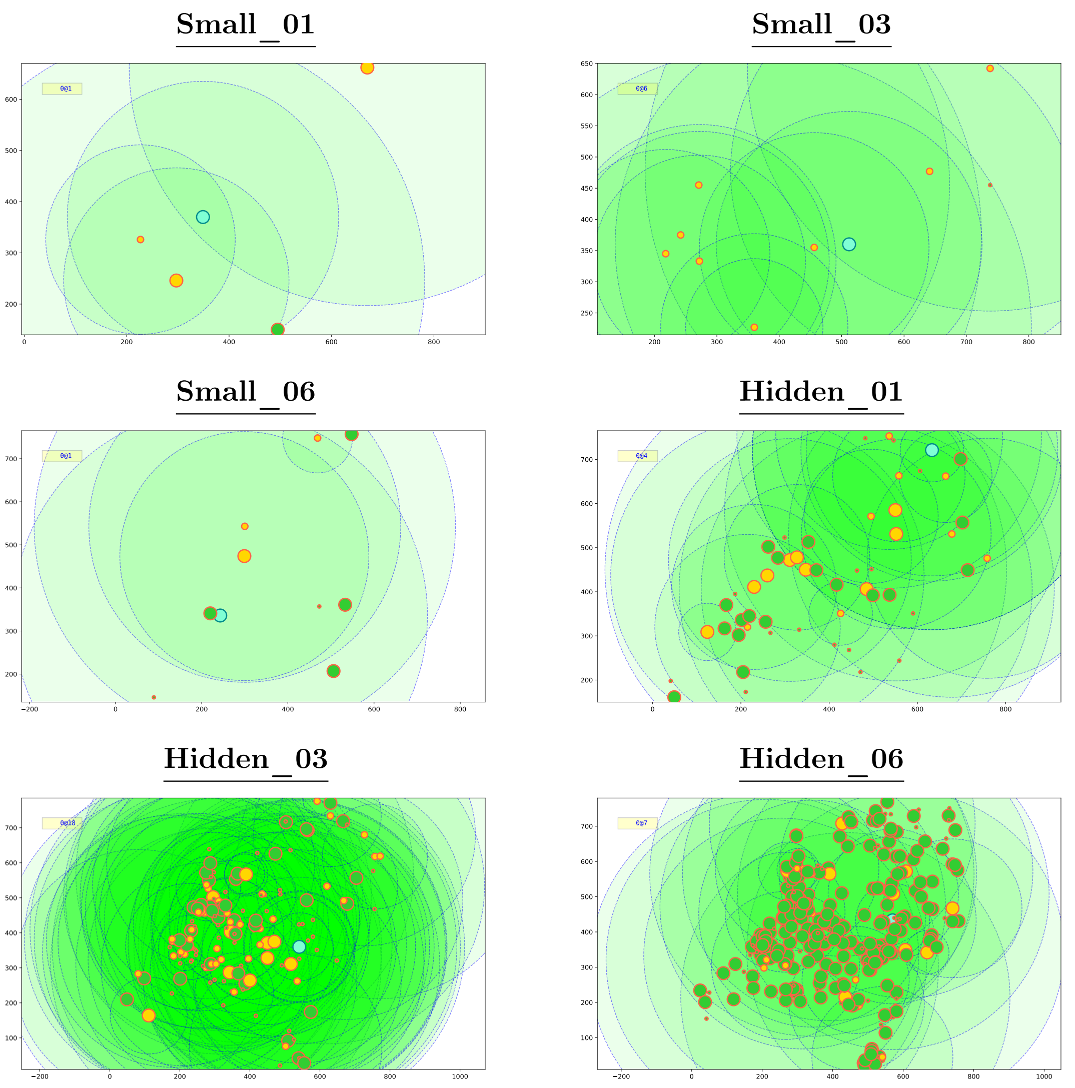


FIGURE 6.5: Visualisation of sample instances

Although Figure 6 gives a rough picture of those instances (e.g. some instances are more limiting in terms of number and action radius of technicians), we must emphasise that it does not describe a given problem instance fully. For example, the importance of violating a given constraint is not depicted and, as we mentioned before, penalties for violating the different constraints can differ substantially as a part of the cost function.

The pie charts in Figure 6.6 depict the utilisation rates of the different selection hyper-heuristics applied in our framework. The utilisation is calculated in terms of the ratio between the number of times a selection hyper-heuristic was successful in finding an improved solution over the best global solution to the total number of improvements made by all the selection hyper-heuristics in the duration of run time.

There are particular selection hyper-heuristic methods that clearly performed better than the rest in making improvements to the solutions during the search process. The incorporation of the RR-based selection hyper-heuristics in the POHH appears to play a key role of solving the problem in a relatively effective manner, in particular SS-RR which performed equally well in the small and hidden datasets. SR-GD was very successful in the larger size instances, where 50% of the improvement rate was achieved by SR-GD in Hidden_06 that has 900 requests. The least successful selection hyper-heuristics are the ones combined with the simulated annealing acceptance. The utilisation of SR-SA was very insignificant ($\simeq 0\%$) in all instances, except for a small improvement rate of 5% in Small_03. Also, SS-SA did not make much contribution in terms of improvement for the hidden set. The naïve acceptance is more successful for the small instances than the larger ones, but only when it is combined with the sequence based selection method.

There seems to be a variation in the performance between the selection hyper-heuristics in instances with different complexities, and we cannot generalise that a certain combination of a selection and move acceptance methods would be successful in every instance in this problem. An interesting idea would be to embed a high-level intelligent control mechanism that can observe these variations, and apply the components of selection hyper-heuristics accordingly during the search time, similar to the online selection methods. The random selection criteria that we apply currently in our framework was able to find 'reasonable' results as reported, and we expect that the suggested improvements in the selection mechanism could yield even better results.

6.5.4 Performance Comparison to the Constituent Hyper-heuristics

Another round of experiments have been conducted by applying the eight selection hyper-heuristics employed in our framework independently on the instances displayed in Figure 6.5. Each selection hyper-heuristic was run for nine times using the same rules to calculate the run time of an instance set by the competition and described in section 6.5. The results are displayed in table 6.5 using the best and average objective values from the nine runs of each individual selection hyper-heuristic, along with the associated standard deviation. The best minimum and average objectives on each instance is highlighted in bold. The Mann–Whitney–Wilcoxon test is performed with a 95% confidence level in order to compare pairwise performance variations of two given algorithms statistically. The following notations are used: (i) '+' denotes that our algorithm (POHH) is better

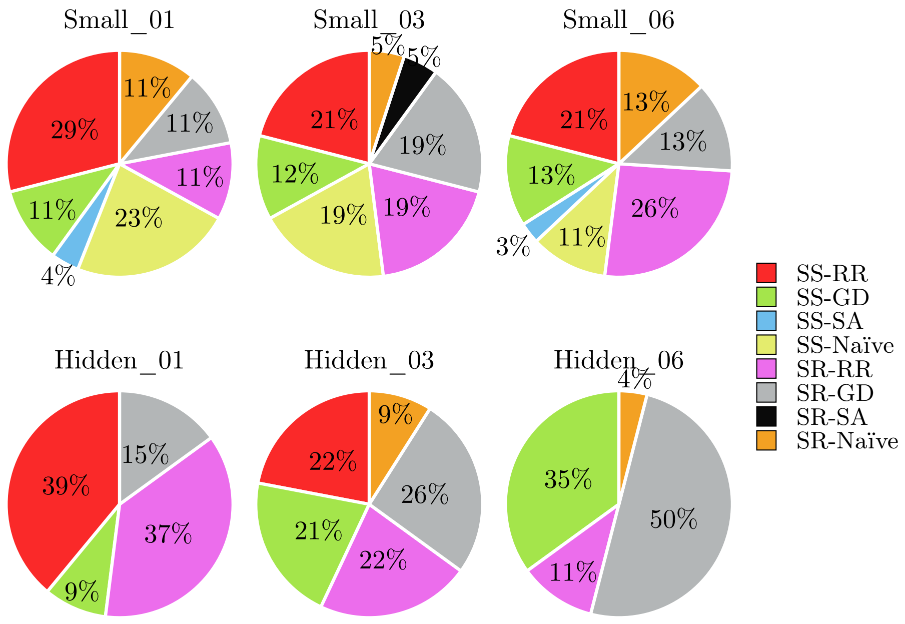


FIGURE 6.6: Average utilisation rate

and this performance variance is statistically significant, (ii) '-' denotes that the performance of POHH is worse and this performance variance is statistically significant, (iii) '=' indicates that there is no statistical significant between the two methods.

The POHH algorithm performed statistically better than each of the individual selection hyper-heuristics used in our framework for all instances, except Hidden_03, where the two methods SR-RR and SS-RR found slightly better averages and performed statistically better. Other than those two cases, the POHH algorithm found the best averages and minimum values in all instances, performing exceptionally better in particular on the largest instance of Hidden_06. This provides evidence for the success of two proposals: 1) utilising multiple solutions allows better exploration of the search space and therefore more possibilities for further improvement in new areas in the search space, instead of focusing on a single solution which means that the improvement will intensify for a specific area of the search space; 2) applying a sequence of selection hyper-heuristics to a solution might be useful in utilising the varying performances of these selection hyper-heuristics and their characteristics. For instance switching from simple selection to online learning by sequences might potentially improve the solution by randomly selecting and applying a heuristic that can find an improvement, and was rather applied as part of a sequence. The opposite is also true in finding the strength of some heuristics that can perform better by being part of a sequence instead of applying them individually. The combined application of these varying characteristics can lead the search into different directions that can yield further improvement. An important future direction to this algorithm would be how to intelligently apply these hyper-heuristic components, to understand the combinations and the sequence of their application that can find the best improvement.

6.6 Summary

In this chapter, we tackled a complex VRP problem which was the subject of the fourth edition of the VeRoLog solver challenge (2019). The challenge consisted of a novel VRP problem comprising two interdependent stages: a capacitated VRP problem with time windows (CVRPTW) for delivering various equipment to customers on their requests, and a service technician routing and scheduling problem (STRSP) for the installation of the delivered equipment. We propose a hyper-heuristic approach and apply it to the set of instances supplied by the competition organisers, and also to another small set of generated test instances. We introduce a novel population-based hyper-heuristic algorithm (POHH) to solve the problem, which was proved to be successful compared to the results of an exact model specifically formulated for this problem, insofar as optimal solutions where found in shorter computational times. Additionally, the POHH algorithm results were compared to the results of the eight finalists of the competition. Our analysis showed that the proposed POHH algorithm performs better than the constituent hyper-heuristics when tested individually for most of the instances.

Table 6.5: The performance comparison of POHH, SS-RR, SS-GD, SS-SA, SS-Naïve, SR-RR, SR-GD, SR-SA and SR-Naïve based on the average (Avg), associated standard deviation (Std), minimum (Min) of the objective values over 9 trials and the pairwise average performance comparison of POHH vs (SS-RR, SS-GD, SS-SA, SS-Naïve, SR-RR, SR-GD, SR-SA and SR-Naïve) based on Mann-Whitney-Wilcoxon for each instance produced by each approach. The hyper-heuristic producing the best value for Avg and Min per each instance are highlighted in bold

	Small_01				Small_03				Small_06			
Method	vs	Avg	Std	Min	vs	Avg	Std	Min	vs	Avg	Std	Min
POHH	=	**36016020**	0	**36016020**	=	**32085922**	83	**32085800**	=	**180524**	0	**180524**
SS-RR	=	**36016020**	0	**36016020**	+	32087251	858	**32085800**	+	247213	49513	**180524**
SS-GD	=	**36016020**	0	**36016020**	+	32094142	3088	32089068	+	200911	2055	198515
SS-SA	=	36016433	620	**36016020**	+	32114432	9290	32098239	+	211280	3317	206996
SS-Naïve	=	**36016020**	0	**36016020**	+	32087868	1110	32086696	+	183922	649	182888
SR-RR	+	36016847	620	**36016020**	+	32086500	533	**32085800**	+	280221	0	280221
SR-GD	=	36016219	395	**36016020**	+	32092212	1004	32090760	+	199464	2466	196666
SR-SA	+	36016732	554	**36016020**	+	32109815	5690	32104880	+	210099	2883	205800
SR-Naïve	=	**36016020**	0	**36016020**	+	32086480	462	32085900	+	183029	1056	182161
	Hidden_01				Hidden_03				Hidden_06			
Method	vs	Avg	Std	Min	vs	Avg	Std	Min	vs	Avg	Std	Min
POHH	=	**68683339**	290545	**68151265**	=	1410411798	5353659	1399418890	=	**34157264**	106666	**34008705**
SS-RR	+	69197055	311273	68917615	-	1389617007	3467539	**1381431185**	+	36409477	75941	36313420
SS-GD	+	85782696	778892	84218865	+	1815339912	13783473	1791026650	+	41742650	251435	41235155
SS-SA	+	86554467	657440	85333250	+	1802310946	18417576	1773264280	+	42003434	158875	41697785
SS-Naïve	+	73793282	565531	72767395	+	1537797289	6427024	1529124090	+	36526647	154146	36315960
SR-RR	+	69209684	153049	69060100	-	**1387767927**	2077527	1385115295	+	36291840	47266	36224495
SR-GD	+	86004059	466746	85462365	+	1811403039	8342226	1800385550	+	41963413	197981	41518920
SR-SA	+	86552397	462097	85483775	+	1812705973	4911482	1805213785	+	42044408	139089	41796340
SR-Naïve	+	74076952	437930	73144995	+	1527445028	5356630	1515517920	+	36719875	63682	36626675

Chapter 7

Conclusion

7.1 Summary of Work

The goal of hyper-heuristics is to raise the level of generality by using methods that are easy-to-implement, cheap-to-maintain, yet deliver excellent performance in different problem domains. In this thesis, we explored a selection hyper-heuristic framework on two versions of complex routing problems: the Urban Transit Route Design Problem (UTRP), and the VeRoLog challenge 2019 vehicle routing problem for the delivery and installation of equipment. We focused on specific concepts of the hyper-heuristic framework and their contribution in improving solution quality and run time. Some of these concepts include: online learning during selection, the selection of sequences of heuristics rather than individual heuristics, the role of single-point based optimisation in improving run times, and optimising multiple solutions during the search. These concepts have been extensively tested and analysed in the two routing problems using benchmark instances as well as larger instances with many real-world features.

In chapter 2 and 3, we introduced the basic concepts of the optimisation of difficult combinatorial problems such as VRPs and defined the two routing applications that are the core of the thesis. A comprehensive survey of the previous research of the UTRP revealed some drawbacks in the commonly applied meta-heuristic methods, these are the long run times associated with large instances, the lack of general methodologies for solving instances of real world size, and the rare of studies that applied their algorithms on real planning processes. We also introduced our selection hyper-heuristic framework, explaining its features and its differences from other meta-heuristic algorithms such as its general applicability to several problems and instances with minimal adaption, and the separation from any problem specific information by the domain barrier. We explained online learning in hyper-heuristics selection using our proposed sequence based method inspired by the transition between different states in the Hidden Markov Model. We also outlined hyper-heuristics contribution in the domain of optimisation of complex routing problems which is a field where hyper-heuristics proved its success and has been widely applied. We also demonstrate that the implementation of hyper-heuristics on the UTNDP is not yet investigated in the literature, and therefore the studies proposed in this thesis provide the foundation for applying this framework on the UTNDP. In the next paragraphs, we will provide a summary of how we have addressed the research questions listed in chapter 1.

RQ1: How can a selection hyper-heuristic being a single-point based framework succeed in overcoming the run time issues in population-based methods while delivering high quality solutions in small as well as large size instances

In chapter 4, we introduced our novel implementation of a selection hyper-heuristic algorithm to solve the complex Urban Transit Route Design Problem (UTRP). Thirty selection hyper-heuristics combining several known selection and move acceptance methods were tested and applied on a set of benchmark instances and their performances were compared statistically to determine the best algorithm. After a series of experiments from the perspective of passenger and operator and the statistical comparisons, the sequence-based selection method combined with the great deluge acceptance method was the most successful, outperforming other selection hyper-heuristics in both passenger and operator objectives and proving the effectiveness of our applied online selection method. The hyper-heuristic approach which has been applied to this particular problem for the first time was very successful, beating the current known state-of-the art results in a very reasonable run times. We have also shown that the run times of hyper-heuristics in the larger instances were very short compared to other population-based meta-heuristics, where we compared our results against key studies that applied population-based approaches and found that hyper-heuristics can perform in significantly less run time to provide a single solution of high quality to large size instances.

RQ2: How can we extend our implementation of the hyper-heuristic framework to be applied on more complex versions of the UTRP and on instances with real-world size and characteristics

In chapter 5, the developed selection hyper-heuristic algorithm described in chapter 4 was modified and adopted in two applications. The first is the application of the best hyper-heuristic algorithm combining sequence based selection and the Great Deluge acceptance (SS-GD) to a more complex version of the UTRP that restricts the the start and end of the routes to specific nodes (i.e., terminals) in the road network. We illustrated the complexity of this version and the difficulty of finding feasible solutions compared to the version described in chapter 4. Therefore careful design of the algorithm is required to avoid generating too many infeasible solutions and to handle the presence of U-turn points. An initialisation method based on the demand information is also proposed. This method aims to include the edges with highest demand in the routes of the initial solution, therefore improving its quality. Moreover, a specific set of low level heuristics is implemented to preform operations while preserving the correct terminal constraints. The algorithm was tested on a new set of instances representing the extended urban area of Nottingham city. These instances were generated using the procedure described in [84], and have real world size and characteristics, thus were ideal to test our methods. The SS-GD algorithm was tested using several scenarios to find the best solutions from the passenger and operator perspectives, and a set of solutions effectively balancing the two objectives. A comparison of the SS-GD results using the three scenarios with the Pareto front generated using a NSGAII framework revealed that our algorithm was successful in finding solutions that clearly dominate the Pareto solutions generated by the NSGAII. Furthermore, a comparison of the route sets generated by our algorithm was conducted against real world route sets extracted from the operating bus routes in Nottingham city. We showed that our algorithm was also successful in improving the existing real routes, where high reductions were observed in the average passenger travel time and the percentage of direct travellers also improved. The findings of this study prove that the hyper-heuristic approach which was tested as a proof of concept on benchmark instances and succeeded in beating well known population-based meta-heuristics, can also be modified and implemented on instances of real-world size and complexity while continuing its success in generating high quality solutions from passengers and operators perspective.

RQ3: How can we bridge the gap between academic versions of the UTRP and real-world transportation systems planning by integrating the algorithms used to solve the UTRP theoretically with a commercial software package used by transportation systems planners

In the second part of chapter 5, we tested hyper-heuristic on the optimisation of public transport routes in the transport modelling software Visum. Visum software functions were utilised to implement interface procedures that translate the differences between its network model and the commonly applied UTRP model, and to design the evaluation functions. Selection hyper-heuristics were tested using the simple random selection and the sequence-based selection combined with Improve or Equal (IE) acceptance. Our algorithms were integrated with the interface functions to translate Visum directed routes into undirected routes suitable for applying our algorithm, and to implement the optimised routes in a suitable format in the Visum network for evaluation. The objectives were to minimise the passenger and the operator costs, where three optimisation scenarios were applied by normalising the objectives and adjusting their weight values: the passenger, the operator, and the balanced perspectives. Two optimisation experiments were conducted on two networks from Visum training examples, one of them represents a real city: the global optimisation and the local optimisation. In the global optimisation experiments, the three optimisation scenarios were tested on a small network using the simple random and the sequence based selection methods. The experiments showed the success of hyper-heuristics in achieving high reduction rates in both passenger and operator costs. The performance difference between the two selection methods was insignificant, although we preferred to apply the sequence based selection in the larger network experiments as we observed that SSHH was able to find better solutions in fewer iterations, which is crucial for testing larger size networks. In the global optimisation experiments on a city-size network, hyper-heuristics continued the success with reductions in the operator cost up to 6%, and in the passenger up to 13%. In this network, we applied only the balanced optimisation configuration, which is more suitable for a real world planning process to generate route sets balancing the two costs. Additionally, the local optimisation was tested on the city-size network, reducing the rate of private car users on the targeted streets by up to 70%. This work shows the possibility to use UTRP algorithms in real planning operations that are of practical use to transportation systems planners, opening a wide range of opportunities to future research towards real-world transportation systems planning using the algorithms that were exclusively used in pure academic versions.

RQ4: How can we generalise the application of hyper-heuristics on different domains of complex routing problems and prove its effectiveness and computational efficiency

The final chapter discussed our solution methodology using a population-based hyper-heuristic algorithm (POHH) which was deployed to solve the VeRoLog solver challenge 2019 problem. The challenge consisted of a novel VRP comprising two interdependent stages: a Capacitated VRP with Time Windows (CVRPTW) for delivering various equipment to customers on their request, and a Service Technician Routing and Scheduling Problem (STRSP) for the installation of the delivered equipment. We described the problem and its constraints as proposed in the official challenge, and demonstrated the complexity that lays in the interconnection between the two routing stages. The objectives to be minimised are the total costs of the hiring and travelling of the trucks/technicians in a single day and during the planning period. Several data sets of varying sizes and complexities were tested, which was essential to show the scalablity of our approach. The small size data set was designed specifically for the proposed work, while the large data set named "Hidden data set" was supplied by the competition organisers to test and rank the competitors algorithms. We proposed a novel problem-independent population-based hyper-heuristic algorithm (POHH). The algorithm maintains a number of solutions during the search, and a sequence of constituent selection hyper-heuristics together with a large set of low level heuristics which are applied to one solution at a time. The constituent hyper-heuristics have been proven to tackle a wide range of problem domains [9, 202, 203].

The experiments were designed according to the competition rules in order to ensure fairness when comparing our results with the results of the competition finalists. The results from the first set of experiments on the small data set were compared with the results of a mathematical model developed specifically for the problem. The results showed that the POHH algorithm was able to find an optimal solution in all the instances where an optimal solution was actually found by the exact model. Furthermore, the POHH was able to find feasible solutions when the exact model failed to do so. These results were achieved in notably less computational time compared to the exact model. The comparison with the finalists results in the Hidden data set also showed that our algorithm is competitive. The POHH achieved a position in the top six in 15 instances out of 25 and found a mean rank that is better than three of the eight finalists. Finally, a statistical comparison between the results of the POHH on a selected set of instances and the results of the constituent selection hyper-heuristics each run individually has shown that the POHH results were statistically better in most of the instances. This proves that the approach of utilising multiple solutions during the search, and applying a sequence of selection hyper-heuristics to a single solution is successful and helps in better exploration of the search space. In this work we showed that the hyper-heuristic framework succeeds in finding improved results in short running times to such complex routing problem and that the success of the framework continues to other routing problems in addition the UTRP, which proves its general applicability to various complex routing applications.

7.2 Future Work

There are many future directions by which the work proposed in this thesis can be extended and improved, and here we discuss some of them. The application of the selection hyper-heuristic framework on the UTRP was based on a simplified model assuming bi-directional routes and symmetrical travel times and demand. The inital purpose was to use a simplified and a commonly applied model from the literature to prove that our approach, which was applied for the first time in the UTRP, is successful, and to allow other researchers to directly compare to our results. Now that we have proved that our algorithm is successful with efficient computational times in large scale instances, more realistic features and assumptions can be added to this model. In terms of the objectives, we can utilise more complex and effective functions in the calculations. In the operator cost for example, we used a simple formula in all the UTRP versions handled in the thesis. There are ways to represent this cost more realistically which will require information about the fleet size and the capacity of each vehicle. Furthermore, we have put some simplified assumptions for the waiting times by merging the waiting and the transfer time in a single cost equal to 5 minutes. The waiting times are associated with the routes' headways and the frequency of vehicle arrivals which were not incorporated in the design of our model, although we put forward some basic proposals for how we can achieve this. Our model can be extended to optimise the frequencies simultaneously with the routes, and accordingly calculate the waiting times in a more realistic way. This will require the implementation of a demand assignment model to determine the passengers flow in the network similar to the studies [86, 91, 94, 112]. Currently, our assumption that the demand between any origin-destination points is static and does not change throughout the day, and that the demand is symmetric. For future research, the variations of the demand throughout the day or in different seasons [108] can be further investigated and modelled as well as considering asymmetric demand. Additionally, the passenger route choice is based solely on the assumption that the passenger always selects the shortest travel path. Improving the simulation of the passenger route choices is an interesting future direction to add more realism in our model. It is also essential to propose our methods to transport planners, and real operating bus companies in the city of Nottingham to prove that our algorithm works competitively on more recent data of bus routes.

We pointed out previously that the passenger cost is the most expensive function in the evaluation, as it includes complicated and computationally expensive operations such as the graph expansion and performing an all-pairs shortest path algorithm on the expanded graph. We found out that using Dijkstra shortest path algorithm with priority queue implementation saves a significant computational time compared to applying Floyd Warshall algorithm. Recently, Lewis [204] investigated the implementation of shortest path algorithms, focusing on graphs where penalties are incurred at the vertices. Two versions of Dijkstra algorithm were introduced that operate on the original unexpanded graph. These methods can be tested in the future in the evaluation of the passenger objective, as it might have a significant impact in time saving during the evaluation.

Our work on the integration of hyper-heuristics and Visum transport modelling software is a promising direction in real transportation systems planning. We used a relatively simple set of low level heuristics to prove the concepts proposed in this work. More complex operations to mutate the routes can be implemented and we can observe whether any improvement in the results can be achieved. Especially interesting would be to include heuristics which add and delete routes to make it possible to vary the number of routes. Moreover, the SS selection can be tested with threshold acceptance methods to see if this can lead to any improvements. Also, it would be worthwhile to explore and improve the process of optimising routes of multiple transport modes.

The VeroLog solver challenge problem is a real life routing application based on real data of ORTEC company based on the Netherlands. Our algorithm may have potential for use on real data for other delivery and installation companies based in the UK, following adjustments according to the nature of their application and constraints.

With regard to the selection hyper-heuristic algorithms proposed, the SSHH algorithm has proven its success in more than one problem domain and was more competitive than other simple selection methods. There are some design aspects that can be looked into to improve this algorithm. For instance, the history of the sequences performance given by the matrices can be saved globally, instead of re-initialising the matrices on each application on a new instance. This way, the successful sequences will be identified and used instantly on hidden domains. There is scope for further research into several aspects of the POHH algorithm proposed in chapter 6. One example is how to decide on which hyper-heuristic strategies to include into our population-based algorithm. We observed in the analysis of the POHH that different combinations of selection and move acceptance methods had varying performances on instances with different sizes and complexities. An interesting modification of the algorithm is to embed an online learning mechanism to capture these variations, and to decide which components of selection and move acceptance will perform better on a particular instance. The random selection criteria that we apply currently gives fairly good results as reported in chapter 6, and we expect that a more intelligent mechanism that controls the selection of the hyper-heuristic components during the search will improve the algorithm performance.

Additionally, the algorithm does not have the ability to learn from history which strategy performs well. Thus, it may be beneficial to exclude certain strategies altogether to speed-up the algorithm.

Bibliography

[1] Gilbert Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4): 408–416, 2009.

[2] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

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

[4] 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.

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

[6] Christine L Mumford. New heuristic and evolutionary operators for the multi-objective urban transit routing problem. In *IEEE Congress on Evolutionary Computation (CEC), 2013*, pages 939–946. IEEE, 2013.

[7] Matthew P John. *Metaheuristics for designing efficient routes & schedules for urban transportation networks.* PhD thesis, Cardiff University, 2016.

[8] 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 *IEEE Congress on Evolutionary Computation (CEC),*, pages 2841–2848. IEEE, 2014.

[9] Burak Bilgin, Ender Özcan, and Emin Erkan Korkmaz. An experimental study on hyper-heuristics and exam timetabling. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 394–412, 2006.

[10] Ender Özcan, Burak Bilgin, and Emin Erkan Korkmaz. A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1):3–23, 2008.

[11] Leena Ahmed, Christine Mumford, and Ahmed Kheiri. Solving urban transit route design problem using selection hyper-heuristics. *European Journal of Operational Research*, 274(2):545–559, 2019.

[12] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity.* Courier Corporation, 1998.

[13] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3): 268–308, 2003.

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

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

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

[17] Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, 1992.

[18] Robert A Russell and Timothy L Urban. Vehicle routing with soft time windows and erlang travel times. *Journal of the Operational Research Society*, 59(9):1220–1228, 2008.

[19] Douglas Moura Miranda and Samuel Vieira Conceição. The vehicle routing problem with hard time windows and stochastic travel and service time. *Expert Systems with Applications*, 64:104–116, 2016.

[20] Ann Melissa Campbell and Jill Hardin Wilson. Forty years of periodic vehicle routing. *Networks*, 63(1):2–15, 2014.

[21] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31, 2007.

[22] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[23] Moshe Dror and Pierre Trudeau. Savings by split delivery routing. *Transportation Science*, 23(2):141–145, 1989.

[24] Jairo R Montoya-Torres, Julián López Franco, Santiago Nieto Isaza, Heriberto Felizzola Jiménez, and Nilson Herazo-Padilla. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, 79: 115–129, 2015.

[25] Diego Cattaruzza, Nabil Absi, Dominique Feillet, and Daniele Vigo. An iterated local search for the multi-commodity multi-trip vehicle routing problem with time windows. *Computers & Operations Research*, 51:257–267, 2014.

[26] Avishai Ceder and Nigel H.M. Wilson. Bus network design. *Transportation Research Part B: Methodological*, 20(4):331–344, 1986. ISSN 0191-2615. doi: http://dx.doi.org/10.1016/0191-2615(86)90047-0.

[27] Reza Zanjirani Farahani, Elnaz Miandoabchi, Wai Yuen Szeto, and Hannaneh Rashidi. A review of urban transportation network design problems. *European Journal of Operational Research*, 229(2):281–302, 2013.

[28] Lang Fan. *Metaheuristic methods for the urban transit routing problem.* PhD thesis, Cardiff University, 2009.

[29] Joaquín de Cea Ch, R Henry Malbran, et al. Demand responsive urban public transport system design: Methodology and application. *Transportation Research Part A: Policy and Practice*, 42(7):951–972, 2008.

[30] 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.

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

[32] Joaquim Gromicho, Pim van't Hof, and Daniele Vigo. The verolog solver challenge 2019. *Journal on Vehicle Routing Algorithms*, pages 1–3, 2019.

[33] Tony Wauters, Túlio Toffolo, Jan Christiaens, and Sam Van Malderen. The winning approach for the verolog solver challenge 2014: the swap-body vehicle routing problem. *Proceedings of ORBEL29*, 2015.

[34] Martin Josef Geiger. On an effective approach for the coach trip with shuttle service problem of the verolog solver challenge 2015. *Networks*, 69(3):329–345, 2017.

[35] Wout Dullaert, Joaquim Gromicho, Jelke van Hoorn, Gerhard Post, and Daniele Vigo. The verolog solver challenge 2016–2017. *Journal on Vehicle Routing Algorithms*, 1(1):69–71, 2018.

[36] Ahmed Kheiri, Alina G Dragomir, David Mueller, Joaquim Gromicho, Caroline Jagtenberg, and Jelke J van Hoorn. Tackling a vrp challenge to redistribute scarce equipment within time windows using metaheuristic algorithms. *EURO Journal on Transportation and Logistics*, pages 1–35, 2019.

[37] Jean-François Cordeau, Gilbert Laporte, Federico Pasin, and Stefan Ropke. Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409, 2010.

[38] Heechul Bae and Ilkyeong Moon. Multi-depot vehicle routing problem with time windows considering delivery and installation vehicles. *Applied Mathematical Modelling*, 40(13-14):6536–6549, 2016.

[39] Caroline J Jagtenberg, Oliver J Maclaren, Andrew J Mason, Andrea Raith, Kevin Shen, and Michael Sundvick. Columnwise neighborhood search: A novel set partitioning matheuristic and its application to the verolog solver challenge 2019. *Networks*, 76(2):273–293, 2020.

[40] Benjamin Graf. Adaptive large variable neighborhood search for a multiperiod vehicle and technician routing problem. *Networks*, 76(2):256–272, 2020.

[41] Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716, 1952.

[42] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

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

[44] Judea Pearl. Intelligent search strategies for computer problem solving. *Addision Wesley*, 1984.

[45] Kenneth Sörensen and Fred Glover. Metaheuristics. *Encyclopedia of operations research and management science*, 62:960–970, 2013.

[46] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information sciences*, 237:82–117, 2013.

[47] David Beasley, David R Bull, and Ralph Robert Martin. An overview of genetic algorithms: Part 1, fundamentals. *University computing*, 15(2):56–69, 1993.

[48] Christian Blum and Daniel Merkle. Swarm intelligence. *Swarm Intelligence in Optimization; Blum, C., Merkle, D., Eds*, pages 43–85, 2008.

[49] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[50] Panta Lucic and Dusan Teodorovic. Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence. In *Preprints of the TRISTAN IV triennial symposium on transportation analysis*, pages 441–445, 2001.

[51] Emile Aarts and Jan Karel Lenstra. *Local search in combinatorial optimization.* Princeton University Press, 2003.

[52] Helena Ramalhinho Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search: Framework and applications. In *Handbook of metaheuristics*, pages 129–168. Springer, 2019.

[53] Fred Glover. Future paths for integer programming and links to ar tifi cial intelli g en ce. *Computers operations research*, 13(5):533–549, 1986.

[54] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[55] Avishai Ceder. *Public transit planning and operation: Modeling, practice and behavior.* CRC press, 2016.

[56] VM Tom and S Mohan. Transit route network design using frequency coded genetic algorithm. *Journal of Transportation Engineering*, 129(2):186–195, 2003.

[57] EM Holroyd. The optimum bus service: a theoretical model for a large uniform urban area. In *Proceedings of the Third International Symposium on the Theory of Traffic FlowOperations Research Society of America*, 1967.

[58] 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.

[59] Bernard F Byrne. Cost minimizing positions, lengths and headways for parallel public transit lines having different speeds. *Transportation Research*, 10(3):209–214, 1976.

[60] Shyue Koong Chang and Paul M Schonfeld. Multiple period optimization of bus transit systems. *Transportation Research Part B: Methodological*, 25(6):453–478, 1991.

[61] Seong Kyu Chang and Paul M Schonfeld. Welfare maximization with financial constraints for bus transit systems. *Transportation Research Record*, (1395), 1993.

[62] Steven Chien and Paul Schonfeld. Optimization of grid transit system in heterogeneous urban environment. *Journal of Transportation Engineering*, 123(1):28–35, 1997.

[63] DL Van Oudheusden, S Ranjithan, and KN Singh. The design of bus route systems—an interactive location-allocation approach. *Transportation*, 14(3):253–270, 1987.

[64] Rob van Nes, Rudi Hamerslag, and LH Immers. *The design of public transport networks*, volume 1202. National Research Council, Transportation Research Board, 1988.

[65] Michael Bussieck. *Optimal lines in public rail transport.* PhD thesis, Citeseer, 1998.

[66] 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.

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

[68] Alexandre Barra, Luis Carvalho, Nicolas Teypaz, Van-Dat Cung, and Ronaldo Balassiano. Solving the transit network design problem with constraint programming. 2007.

[69] A Patz. Die richtige auswahl von verkehrslinien bei großen strassenbahnnetzen. *Verkehrstechnik*, 50:51, 1925.

[70] 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.

[71] D Dubois, G Bel, and M Llibre. A set of methods in transportation network synthesis and analysis. *Journal of the Operational Research Society*, 30(9):797–808, 1979.

[72] Herbert Sonntag. *Linienplanung im öffentlichen Personennahverkehr*. PhD thesis, Technical University Berlin., 1977.

[73] C.E. Mandl. *Applied network optimization*. Operations Research and Industrial Engineering. Academic Press, 1979. ISBN 9780124683501.

[74] Christoph E Mandl. Evaluation and optimization of urban public transportation networks. *European Journal of Operational Research*, 5(6):396–404, 1980.

[75] 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.

[76] 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.

[77] Young-Jae Lee and Vukan R Vuchic. Transit network design with variable demand. *Journal of Transportation Engineering*, 131(1):1–10, 2005.

[78] Carsten Simonis. Optimierung von Omnibuslinien. *Berichte des Instituts für Stadtbauwesen*, (26), 1981.

[79] 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.

[80] Wai Yuen 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.

[81] 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. ISSN 0968090X. doi: 10.1016/j.trc.2010.09.003. URL `http://dx.doi.org/10.1016/j.trc.2010.09.003`.

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

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

[84] Philipp Heyken Soares, Christine L Mumford, Kwabena Amponsah, and Yong Mao. An adaptive scaled network for public transport route optimisation. *Public Transport*, 11(2):379–412, 2019.

[85] 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.

[86] Renato Oliveira Arbex and Claudio Barbieri da Cunha. Efficient transit network design and frequencies setting multi-objective optimization by alternating objective genetic algorithm. *Transportation Research Part B: Methodological*, 81:355–376, 2015.

[87] Jie Yang and Yangsheng Jiang. Application of modified nsga-ii to the transit network design problem. *Journal of Advanced Transportation*, 2020, 2020.

[88] Leena Ahmed, Christine Mumford, and Ahmed Kheiri. Solving urban transit route design problem using selection hyper-heuristics. *European Journal of Operational Research*, 274(2):545–559, 2019. doi: https://doi.org/10.1016/j.ejor.2018.10.022.

[89] Bin Yu, Zhong-zhen Yang, Peng-huan Jin, Shan-hua Wu, and Bao-zhen Yao. Transit route network design-maximizing direct and transfer demand density. *Transportation Research Part C*, 22:58–75, 2012. ISSN 0968-090X. doi: 10.1016/j.trc.2011.12.003. URL `http://dx.doi.org/10.1016/j.trc.2011.12.003`.

[90] Hossain Poorzahedy and Omid M Rouhani. Hybrid meta-heuristic algorithms for solving network design problem. *European Journal of Operational Research*, 182 (2):578–596, 2007.

[91] Miloš Nikolić and DušAn Teodorović. Transit network design by bee colony optimization. *Expert Systems with Applications*, 40(15):5945–5955, 2013.

[92] Miloš Nikolić and Dušan Teodorović. A simultaneous transit network design and frequency setting: Computing with bees. *Expert Systems with Applications*, 41 (16):7200–7209, 2014.

[93] Panagiotis N Kechagiopoulos and Grigorios N Beligiannis. Solving the urban transit routing problem using a particle swarm optimization based algorithm. *Applied Soft Computing*, 21:654–676, 2014.

[94] Shashi Bhushan Jha, Jitendra Kumar Jha, and Manoj Kumar Tiwari. A multi-objective meta-heuristic approach for transit network design and frequency setting problem in a bus transit system. *Computers & Industrial Engineering*, 130:166–186, 2019.

[95] Ahmed Tarajo Buba and Lai Soon Lee. Hybrid differential evolution-particle swarm optimization algorithm for multiobjective urban transit network design problem with homogeneous buses. *Mathematical Problems in Engineering*, 2019, 2019.

[96] 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), 2006.

[97] 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.

[98] Antonio Mauttone and Maria E. Urquhart. A route set construction algorithm for the transit network design problem. *Computers and Operations Research*, 36(8): 2440–2449, 2009. ISSN 03050548. doi: 10.1016/j.cor.2008.09.014.

[99] S. B. Pattnaik, S. Mohan, and V.M. Tom. Urban bus route network design using genetic algorithm. *Reviewed by the Urban Transportation Division*, 37(3):24, 1998.

[100] Maurizio Bielli, Massimiliano Caramia, and Pasquale Carotenuto. Genetic algorithms in bus network optimization. *Transportation Research Part C: Emerging*

Technologies, 10(1):19–34, 2002. ISSN 0968090X. doi: 10.1016/S0968-090X(00)00048-6.

[101] 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.

[102] Jitendra Agrawal and Tom V Mathew. Transit route network design using parallel genetic algorithm. *Journal of Computing in Civil Engineering*, 18(3):248–256, 2004.

[103] Zhongzhen Yang, Bin Yu, and Chuntian Cheng. A parallel ant colony algorithm for bus network optimization. *Computer-Aided Civil and Infrastructure Engineering*, 22(1):44–55, 2007.

[104] Bin Yu, Zhongzhen Yang, and Jinbao Yao. Genetic algorithm for bus frequency optimization. *Journal of Transportation Engineering*, 136(6):576–583, 2010.

[105] 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. ISSN 01912615. doi: 10.1016/j.trb.2011.07.005. URL `http://dx.doi.org/10.1016/j.trb.2011.07.005`.

[106] Joanne Suk Chun Chew and Lai Soon Lee. A genetic algorithm for urban transit routing problem. In *International Journal of Modern Physics: Conference Series*, volume 9, pages 411–421. World Scientific, 2012.

[107] Sh Afandizadeh, H Khaksar, and N Kalantari. Bus fleet optimization using genetic algorithm a case study of mashhad. *International Journal of Civil Engineering*, 11(1):43–52, 2013.

[108] SM Mahdi Amiripour, Avishai Avi Ceder, and Afshin Shariat Mohaymany. Designing large-scale bus network with seasonal variations of demand. *Transportation Research Part C: Emerging Technologies*, 48:322–338, 2014.

[109] Hang Zhao, Rong Jiang, et al. The memetic algorithm for the optimization of urban transit network. *Expert Systems with Applications*, 42(7):3760–3773, 2015.

[110] Mahmoud Owais, Mostafa K Osman, and Ghada Moussa. Multi-objective transit route network design as set covering problem. *IEEE Transactions on Intelligent Transportation Systems*, 17(3):670–679, 2015.

[111] Muhammad Ali Nayeem, Md Monirul Islam, and Xin Yao. Solving transit network design problem using many-objective evolutionary approach. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3952–3963, 2018.

[112] Ahmed Tarajo Buba and Lai Soon Lee. A differential evolution for simultaneous transit network design and frequency setting problem. *Expert Systems with Applications*, 106:277–289, 2018.

[113] Kazi Ashik Islam, Ibraheem Muhammad Moosa, Jaiaid Mobin, Muhammad Ali Nayeem, and M Sohel Rahman. A heuristic aided stochastic beam search algorithm for solving the transit network design problem. *Swarm and Evolutionary Computation*, 46:154–170, 2019.

[114] Lang Fan, Hui Chen, and Ying Gao. An improved flower pollination algorithm to the urban transit routing problem. *Soft Computing*, pages 1–10, 2019.

[115] Javier Duran, Lorena Pradenas, and Victor Parada. Transit network design with pollution minimization. *Public Transport*, 11(1):189–210, 2019.

[116] SM Hassan Mahdavi Moghaddam, K Ramachandra Rao, G Tiwari, and Pravesh Biyani. Simultaneous bus transit route network and frequency setting search algorithm. *Journal of Transportation Engineering, Part A: Systems*, 145(4):04019011, 2019.

[117] Javier Duran-Micco, Evert Vermeir, and Pieter Vansteenwegen. Considering emissions in the transit network design and frequency setting problem with a heterogeneous fleet. *European Journal of Operational Research*, 282(2):580–592, 2020.

[118] Shushan Chai and Qinghuai Liang. An improved nsga-ii algorithm for transit network design and frequency setting problem. *Journal of Advanced Transportation*, 2020, 2020.

[119] Mingzhang Liang, Wei Wang, Changyin Dong, and De Zhao. A cooperative coevolutionary optimization design of urban transit network and operating frequencies. *Expert Systems with Applications*, 160:113736, 2020.

[120] Stefan Walter. Nachfrageorientierte liniennetzoptimierung am beispiel graz (demand orientated line optimisation at the example of graz). Master's thesis, Graz University of Technologie, 2010.

[121] Joaquín Pacheco, Ada Alvarez, Silvia Casado, and José Luis González-Velarde. A tabu search approach to an urban transport problem in northern spain. *Computers & Operations Research*, 36(3):967–979, 2009.

[122] Leena Ahmed, Philipp Heyken Soares, Christine Mumford, and Yong Mao. Optimising bus routes with fixed terminal nodes: comparing hyper-heuristics with nsgaii on realistic transportation networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1102–1110. ACM, 2019.

[123] Geoff 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.

[124] Paolo Toth and Daniele Vigo. *The vehicle routing problem.* SIAM, 2002.

[125] Ph Augerat, Jose Manuel Belenguer, Enrique Benavent, A Corberán, D Naddef, and G Rinaldi. *Computational results with a branch and cut code for the capacitated vehicle routing problem.* IMAG, 1995.

[126] Jens Lysgaard, Adam N Letchford, and Richard W Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

[127] Michel L Balinski and Richard E Quandt. On an integer program for a delivery problem. *Operations research*, 12(2):300–304, 1964.

[128] Ricardo Fukasawa, Humberto Longo, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F Werneck. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical programming*, 106(3):491–511, 2006.

[129] Marshall L Fisher and Ramchandran Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.

[130] Matthew JW Morgan and Christine L Mumford. Capacitated vehicle routing: perturbing the landscape to fool an algorithm. In *2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2271–2277. IEEE, 2005.

[131] Christian Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, 31(12):1985–2002, 2004.

[132] David Mester and Olli Bräysy. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, 34(10): 2964–2975, 2007.

[133] Yuichi Nagata. Edge assembly crossover for the capacitated vehicle routing problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 142–153. Springer, 2007.

[134] Yuichi Nagata and Olli Bräysy. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. *Networks: An International Journal*, 54(4): 205–215, 2009.

[135] Antoon WJ Kolen, AHG Rinnooy Kan, and Harry WJM Trienekens. Vehicle routing with time windows. *Operations Research*, 35(2):266–273, 1987.

[136] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[137] Jean-Yves Potvin and Jean-Marc Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331–340, 1993.

[138] Robert A Russell. Hybrid heuristics for the vehicle routing problem with time windows. *Transportation science*, 29(2):156–166, 1995.

[139] Slim Belhaiza, Pierre Hansen, and Gilbert Laporte. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers & Operations Research*, 52:269–281, 2014.

[140] Chi-Bin Cheng and Keng-Pin Wang. Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm. *Expert Systems with Applications*, 36(4):7758–7763, 2009.

[141] Qiulei Ding, Xiangpei Hu, Lijun Sun, and Yunzeng Wang. An improved ant colony optimization and its application to vehicle routing problem with time windows. *Neurocomputing*, 98:101–107, 2012.

[142] R Tavakkoli-Moghaddam, M Gazanfari, M Alinaghian, A Salamatbakhsh, and N Norouzi. A new mathematical model for a competitive vehicle routing problem with time windows solved by simulated annealing. *Journal of manufacturing systems*, 30(2):83–92, 2011.

[143] Edward J Beltrami and Lawrence D Bodin. Networks and vehicle routing for municipal waste collection. *Networks*, 4(1):65–94, 1974.

[144] Alireza Rahimi-Vahed, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. A path relinking algorithm for a multi-depot periodic vehicle routing problem. *Journal of heuristics*, 19(3):497–524, 2013.

[145] Claudia Archetti, Ola Jabali, and M Grazia Speranza. Multi-period vehicle routing problem with due dates. *Computers & Operations Research*, 61:122–134, 2015.

[146] Federico Alonso, M Jesús Alvarez, and John E Beasley. A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, 59(7):963–976, 2008.

[147] Samira Mirzaei and Sanne Wøhlk. Erratum to: A branch-and-price algorithm for two multi-compartment vehicle routing problems. *EURO Journal on Transportation and Logistics*, 6(2):185–218, 2017.

[148] Wenjuan Gu, Diego Cattaruzza, Maxime Ogier, and Frédéric Semet. Adaptive large neighborhood search for the commodity constrained split delivery vrp. *Computers & Operations Research*, 112:104761, 2019.

[149] Y Zhang and XD Chen. An optimization model for the vehicle routing problem in multi-product frozen food delivery. *Journal of applied research and technology*, 12 (2):239–250, 2014.

[150] Moshe Dror and Pierre Trudeau. Split delivery routing. *Naval Research Logistics (NRL)*, 37(3):383–402, 1990.

[151] Claudia Archetti and Maria Grazia Speranza. Vehicle routing problems with split deliveries. *International transactions in operational research*, 19(1-2):3–22, 2012.

[152] Mourad Boudia, Christian Prins, and Mohamed Reghioui. An effective memetic algorithm with population management for the split delivery vehicle routing problem. In *International Workshop on Hybrid Metaheuristics*, pages 16–30. Springer, 2007.

[153] Claudia Archetti, M Grazia Speranza, and Martin WP Savelsbergh. An optimization-based heuristic for the split delivery vehicle routing problem. *Transportation Science*, 42(1):22–31, 2008.

[154] Claudia Archetti, Nicola Bianchessi, and Maria Grazia Speranza. A column generation approach for the split delivery vehicle routing problem. *Networks*, 58(4): 241–254, 2011.

[155] Jiyang Xu and Steve Y Chiu. Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7(5):495–509, 2001.

[156] Attila A Kovacs, Sophie N Parragh, Karl F Doerner, and Richard F Hartl. Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of scheduling*, 15(5):579–600, 2012.

[157] Victor Pillac, Christelle Gueret, and Andrés L Medaglia. A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7(7): 1525–1535, 2013.

[158] Fulin Xie, Chris N Potts, and Tolga Bektaş. Iterated local search for workforce scheduling and routing problems. *Journal of Heuristics*, 23(6):471–500, 2017.

[159] Henry Fisher. Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, pages 225–251, 1963.

[160] Peter Cowling, Graham Kendall, and Eric Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 176–190. Springer, 2000.

[161] Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyper-heuristic approaches. In *Handbook of Metaheuristics*, pages 449–468. Springer, 2010.

[162] John H. Drake, Ahmed Kheiri, Ender Özcan, and Edmund K. Burke. Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 2019. ISSN 0377-2217.

[163] Ahmed Kheiri. Heuristic sequence selection for inventory routing problem. *Transportation Science*, 54(2):302–312, 2020.

[164] Ender Özcan, Mustafa Misir, Gabriela Ochoa, and Edmund K Burke. A reinforcement learning: great-deluge hyper-heuristic for examination timetabling. In *Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends*, pages 34–55. IGI Global, 2012.

[165] Alexander Nareyek. Choosing search heuristics by non-stationary reinforcement learning. In *Metaheuristics: Computer decision-making*, pages 523–544. Springer, 2003.

[166] Edmund K Burke, Graham Kendall, and Eric Soubeiga. A tabu-search hyper-heuristic for timetabling and rostering. *Journal of heuristics*, 9(6):451–470, 2003.

[167] Peter Cowling, Graham Kendall, and Limin Han. An investigation of a hyper-heuristic genetic algorithm applied to a trainer scheduling problem. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1185–1190. IEEE, 2002.

[168] Nasser R Sabar and Graham Kendall. Population based monte carlo tree search hyper-heuristic for combinatorial optimization problems. *Information Sciences*, 314:225–239, 2015.

[169] Yu Lei, Maoguo Gong, Licheng Jiao, and Yi Zuo. A memetic algorithm based on hyper-heuristics for examination timetabling problems. *International Journal of Intelligent Computing and Cybernetics*, 8(2):139–151, 2015.

[170] Ping-Che Hsiao, Tsung-Che Chiang, and Li-Chen Fu. A vns-based hyper-heuristic with adaptive computational budget of local search. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

[171] Andreas Lehrbaum and Nysret Musliu. A new hyperheuristic algorithm for cross-domain search problems. In *International Conference on Learning and Intelligent Optimization*, pages 437–442. Springer, 2012.

[172] Murat Kalender, Ahmed Kheiri, Ender Özcan, and Edmund K Burke. A greedy gradient-simulated annealing selection hyper-heuristic. *Soft Computing*, 17(12): 2279–2292, 2013.

[173] Gunter Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86–92, 1993.

[174] Edmund K Burke and Yuri Bykov. A late acceptance strategy in hill-climbing for exam timetabling problems. In *PATAT 2008 Conference, Canada*, 2008.

[175] Ahmed Kheiri and Ed Keedwell. A hidden Markov model approach to the problem of heuristic selection in hyper-heuristics with a case study in high school timetabling problems. *Evolutionary Computation*, 25(3):473–501, 2017.

[176] Ahmed Kheiri and Ed Keedwell. A sequence-based selection hyper-heuristic utilising a hidden Markov model. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 417–424. ACM, 2015.

[177] Leena N Ahmed, Ender Özcan, and Ahmed Kheiri. Solving high school timetabling problems worldwide using selection hyper-heuristics. *Expert Systems with Applications*, 42(13):5463–5471, 2015.

[178] Peter Cowling and Konstantin Chakhlevitch. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 2, pages 1214–1221. IEEE, 2003.

[179] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435, 2007.

[180] Pablo Garrido and Carlos Castro. A flexible and adaptive hyper-heuristic approach for (dynamic) capacitated vehicle routing problems. *Fundamenta Informaticae*, 119 (1):29–60, 2012.

[181] James D Walker, Gabriela Ochoa, Michel Gendreau, and Edmund K Burke. Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In *International conference on learning and intelligent optimization*, pages 265–276. Springer, 2012.

[182] Peter Ross, Sonia Schulenburg, Javier G Marín-Blázquez, and Emma Hart. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 942–948, 2002.

[183] Hugo Terashima-Marín, José Carlos Ortiz-Bayliss, Peter Ross, and Manuel Valenzuela-Rendón. Hyper-heuristics for the dynamic variable ordering in constraint satisfaction problems. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 571–578, 2008.

[184] Pablo Garrido and Carlos Castro. Stable solving of cvrps using hyperheuristics. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 255–262, 2009.

[185] Mustafa Misir, Wim Vancroonenburg, Katja Verbeeck, and Greet Vanden Berghe. A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete. In *Proceedings of the Metaheuristics International Conference*, pages 289–298, 2011.

[186] Richard J Marshall, Mark Johnston, and Mengjie Zhang. Hyper-heuristic operator selection and acceptance criteria. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 99–113. Springer, 2015.

[187] Enrique Urra, Claudio Cubillos, and Daniel Cabrera-Paniagua. A hyperheuristic for the dial-a-ride problem with time windows. *Mathematical Problems in Engineering*, 2015, 2015.

[188] Yujie Chen, Philip Mourdjis, Fiona Polack, Peter Cowling, and Stephen Remde. Evaluating hyperheuristics and local search operators for periodic routing problems. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 104–120. Springer, 2016.

[189] Peng-Yeng Yin, Sin-Ru Lyu, and Ya-Lan Chuang. Cooperative coevolutionary approach for integrated vehicle routing and scheduling using cross-dock buffering. *Engineering Applications of Artificial Intelligence*, 52:40–53, 2016.

[190] Abdullah Konak, David W Coit, and Alice E Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91 (9):992–1007, 2006.

[191] Carlos M Fonseca, Peter J Fleming, et al. Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization. In *Icga*, volume 93, pages 416–423. Citeseer, 1993.

[192] 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.

[193] 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.

[194] Oleg Grodzevich and Oleksandr Romanko. Normalization and other topics in multi-objective optimization. 2006.

[195] Michael Barbehenn. A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. *IEEE Transactions on computers*, 47(2):263, 1998.

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

[197] *PTV Visum 17 User Manual.* PTV AG, Karlsruhe, Germany, 2018.

[198] *Emme 4 User Manual.* INRO, Montreal, Canada, 2018.

[199] Philipp Heyken Soares, Leena Ahmed, Yong Mao, and Christine Mumford. Public transport network optimisation in ptv visum using selection hyper-heuristics. *Under Review*, 2020.

[200] Markus Friedrich, Thomas Haupt, and Klaus Noekel. Planning and Analyzing Transit Networks: An Integrated Approach Regarding Requirements of Passengers and Operators. *Journal of Public Transportation*, 2(4):19–39, 1999. ISSN 1077-291X. doi: 10.5038/2375-0901.2.4.2.

[201] Ahmed Kheiri, Leena Ahmed, Burak k Boyacı, Joaquim Gromicho, Christine Mumford, Ender Özcan, and Ali Selim Dirikoç. Exact and hyper-heuristic solutions for the distribution-installation problem from the verolog 2019 challenge. *Networks*, pages 1–35, 2020.

[202] Dennis Wilson, Silvio Rodrigues, Carlos Segura, Ilya Loshchilov, Frank Hutter, Guillermo López Buenfil, Ahmed Kheiri, Ed Keedwell, Mario Ocampo-Pineda, Ender Özcan, Sergio Ivvan Valdez Peña, Brian Goldman, Salvador Botello Rionda, Arturo Hernández-Aguirre, Kalyan Veeramachaneni, and Sylvain Cussat-Blanc. Evolutionary computation for wind farm layout optimization. *Renewable Energy*, 126:681–691, 2018.

[203] Ahmed Kheiri, Edward Keedwell, Michael J. Gibson, and Dragan Savic. Sequence analysis-based hyper-heuristics for water distribution network optimisation. *Procedia Engineering*, 119:1269–1277, 2015. Computing and Control for the Water Industry (CCWI2015) Sharing the best practice in water management.

[204] Rhyd Lewis. A shortest path algorithm for graphs featuring transfer costs at their vertices. In *International Conference on Computational Logistics*, pages 539–552. Springer, 2020.