Solving Urban Transit Route Design Problem using Selection Hyper-heuristics

Supplementary Materials: This document provides background information and additional materials related to this paper. It forms part of the PhD thesis prepared by Leena

Leena Ahmed*[a], Christine Mumford[a], Ahmed Kheiri[b]

[a]*Cardiff University, School of Computer Science*
Queens building, 5 The Parade, Roath, Cardiff, CF24 3AA, UK.
email: {AhmedLH, MumfordCL}@cardiff.ac.uk
[b]*Lancaster University, Department of Management Science*
Lancaster, LA1 4YX, UK. email: a.kheiri@lancaster.ac.uk

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 Laporte (2009). 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.

*corresponding author

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 Burke et al. (2013) 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.

2. Research Motivation and Contributions

Meta-heuristics have enjoyed some success on versions of the UTRP, with genetic algorithms (GAs) as a particularly popular choice Fan and Machemehl (2006a); Fan and Mumford (2010); Mumford (2013). 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 John (2016). 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. (2014) 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 work, 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 Bilgin et al. (2006); Özcan et al. (2008). 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 key research questions we are addressing 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?

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?

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?

4. How can we generalise the application of hyper-heuristics in different domains of complex routing problems and prove its effectiveness and computational efficiency?

   The main contributions of this work 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.

3. 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 we will further explore the time and computational complexity associated with some optimisation problems, and describe the algorithms suitable for solving them.

3.1. 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 (1998), 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 (2003), 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 3.2), 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.

3.2. $\mathcal{NP}$-Hard and $\mathcal{NP}$-Complete Problems

The foundations of the computational complexity theory were put down by Cook (1971) and Karp (1972), 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 nondeterministically 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

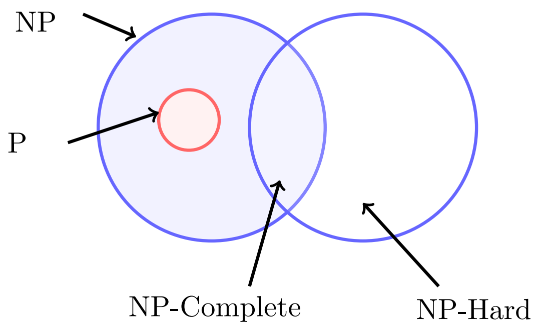


Figure 1: Relationship between P, NP, NP-Hard, and NP-Complete problems

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 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.

3.3. 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: $g_i(x) \leq 0;\ \ i = 1 \ldots, n$, 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 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), 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.

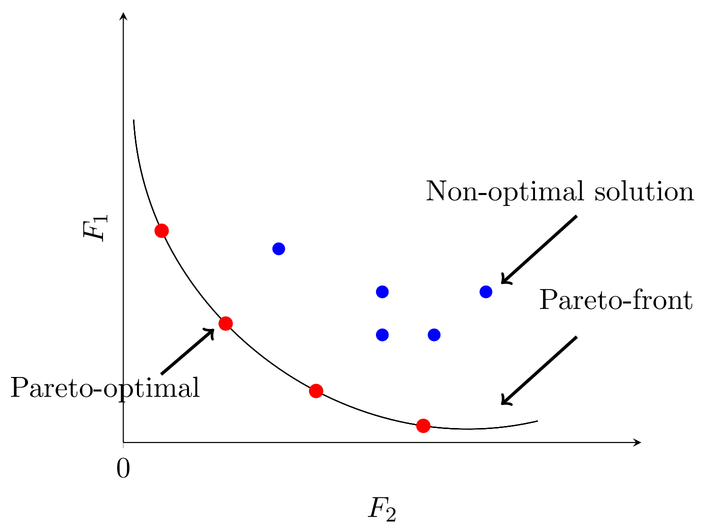


Figure 2: Pareto-front and Pareto optimal solutions

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).

4. Graph Structure

As mentioned in section 3.1, 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 paper.

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.

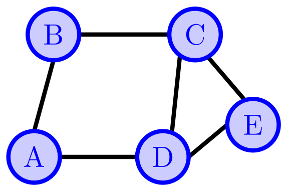
A *simple graph* is defined as a graph that is undirected, has at most one edge between any two pair of vertices, and has no self loops. A loop exists in a graph when there is at least one edge in the edge set that does not connect two distinct pair of vertices (i.e., it connects an edge to itself, $e = (u, u)$). 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 3.

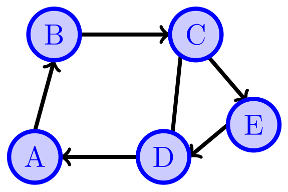
A graph G' is defined as the subgraph of the graph G, if it contains a subset of its vertices $V' \subseteq V$, and a subset of its edges $E' \subseteq E$. Of a particular interest is the *induced subgraph*, in which a specific subset of the vertices is selected, along with the set of edges that connected 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 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 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).

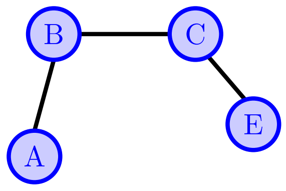
A weighted graph, is the graph where each edge is associated with a weight value (figure 3 (f)). This graph is particularly useful in applications where the shortest path or the distance between pairs of vertices should be calculated.



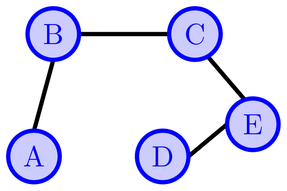
(a) Undirected graph



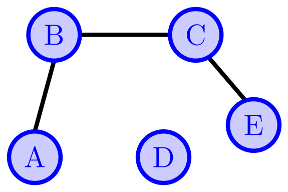
(b) Directed graph



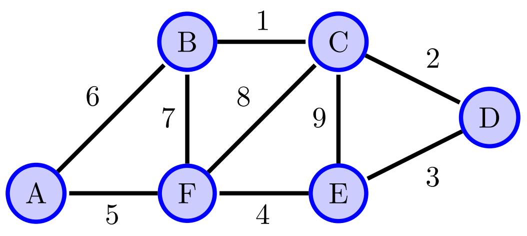
(c) Induced subgrraph



(d) Spanning subgraph



(e) Unconnected graph



(f) Undirected weighted graph

Figure 3: Demonstration of different types of graphs

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 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).

5. 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 (1959) 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 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 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*,

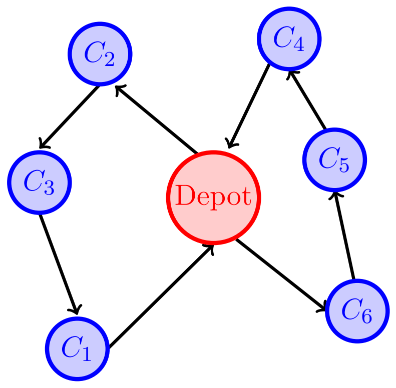


Figure 4: Solution to the basic VRP

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 5.2 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 $(d_i \leq Q \quad \forall c_i \in C)$.

5.1. 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 Laporte (1992). 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 4, 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 (1959) 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.

5.2. 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 5, 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) Russell and Urban (2008) in which violating the time windows is allowed but associated with a penalty, and the VRP with hard time windows (VRPHTW) Miranda and Conceição (2016) in which the time windows must be respected.

- The Multi-Period VRP Problem (MPVRP) (see Campbell and Wilson (2014)): 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: 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) Berbeglia et al. (2007); Parragh et al. (2008): 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) (Dror and Trudeau, 1989): 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) Montoya-Torres et al. (2015): In this variant, it is assumed that there are multiple depots from which customers can be served.

- The Multiple Trips Vehicle Routing Problem (MTVRP) Cattaruzza et al. (2014): 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.

6. 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 (1986) 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 Farahani et al. (2013). 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.

6.1. 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 Fan (2009); John (2016) 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.

6.2. 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.

7. 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 5). In the following sections we will be discussing each of these categories in further detail.

7.1. Exact Mathematical Approaches

Mathematical approaches rely on mathematical formulations for the design of the objective function and its constraints. One of the best known

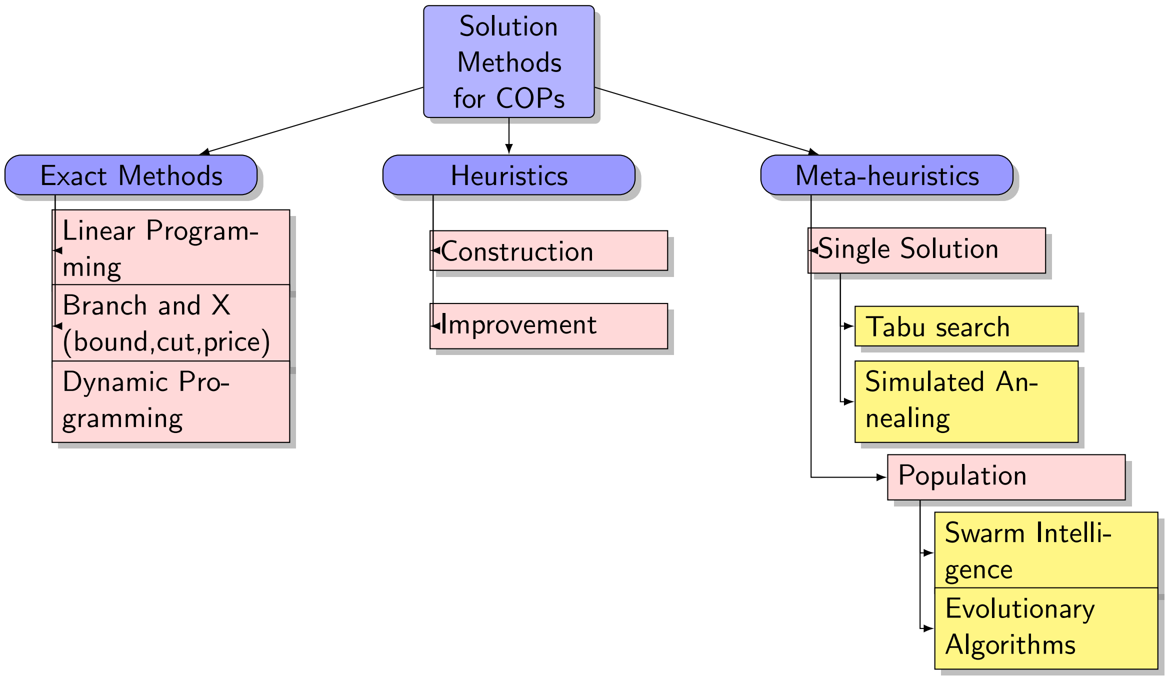


Figure 5: Solution methodologies for CO problems

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 Bellman (1952), and Branch and Bound (B&B) Lawler and Wood (1966). 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 (2003) 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.

7.2. 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 (1984) 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.

7.3. 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 (2013): "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 (2013) 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.

7.3.1. 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) Boussaïd et al. (2013).

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 Beasley et al. (1993)

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;
```

7.3.2. 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 Blum and Merkle (2008). 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) Dorigo et al. (1996) which mimics the way ants search food in nature, Bee Colony Optimisation (BCO) Lucic and Teodorovic (2001) inspired by the movement of bees during nectar collection process, and Particle Swarm Optimisation (PSO).

7.3.3. 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 Aarts and Lenstra (2003) 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) Lourenço et al. (2019) 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 (1986) 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 Feo and Resende (1995) 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.

8. 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.

8.1. 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 (2016) 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 (2003) 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 (1967), 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 (1975) 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 (1976) 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 Chang and Schonfeld (1991, 1993); Chien and Schonfeld (1997).

Mathematical programming approaches were attempted in the UTRP as early as 1970. Van Oudheusden et al. (1987) 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. (1988) 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 (1998) 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 (2003) 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. (2006) 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. (2007) 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.

8.2. Heuristic Methods

Probably, Patz (1925) 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 (1967), 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. (1979) 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 (1977) 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 (1979, 1980) 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 (1986) 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 (1991, 1995) 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 (2005) 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 (Simonis, 1981). 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.

8.3. Meta-heuristic Approaches

8.3.1. 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. (1998a) 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 (2003) using a simultaneous route and frequency coded model.

Chakroborty and Wivedi (2002) 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 Chakroborty (2003), 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 (2006a), 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 (2011) 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. (2012) 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 (2013) 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. (2013) 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 (2013) and Fan and Mumford (2010), where they reported improved results.

The work in John et al. (2014) is built upon the work of Mumford (2013) 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. (2014) to improve its efficiency in terms of run times. In Heyken Soares et al. (2019) the algorithm of John et al. (2014) 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. (2014) 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 (2015) 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 (2020) 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 NS-GAII 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 Ahmed et al. (2019b).

Other studies along with those listed above that applied GA to the route design problem/route design and frequency determination are given in table 1.

8.3.2. 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. (2012) 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 (2007) 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 (2013) 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 Nikolić and Teodorović (2014) the authors extended their work to simultaneously determine the frequencies on the designed routes.

Kechagiopoulos and Beligiannis (2014) 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. (2019) 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 (2019) 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.

8.3.3. Single Solution based Meta-heuristics

Fan and Machemehl (2006b) 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 (2010) 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 (2008) 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 (2009) 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 (2014) 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.

8.4. 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 (Walter, 2010). One example is the work by Pacheco et al. (2009), 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. (2012) 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. (Ahmed et al., 2019a; Bagloee and Ceder, 2011; Bielli et al., 2002; Heyken Soares et al., 2019)). However, the results of these studies have not been verified in real-world planning processes.

8.5. 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

Table 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. (1998b)	1998b	Total travel time	Routes, Frequencies	GA
2	Chakroborty and Wivedi (2002)	2002	Passenger and Operator costs	Routes	GA
3	Bielli et al. (2002)	2002	Multiobjective	Routes	GA
4	Tom and Mohan (2003)	2003	Passenger and operator costs	Routes, Frequencies	GA
5	Ngamchai and Lovell (2003)	2003	Passenger and operator costs	Routes, Frequencies	GA
6	Agrawal and Mathew (2004)	2003	Generalised travel cost and operator cost	Routes and frequencies	Parallel GA
7	Fan and Machemehl (2006a)	2006a	Passenger and operator costs	Routes, Frequencies	GA
8	Fan and Machemehl (2006b)	2006b	Passenger and operator costs	Routes, Frequencies	SA
9	Yang et al. (2007)	2007	maximum direct travellers per unit length	Routes and stops	Parallel ACO
10	Fan and Machemehl (2008)	2008	Passenger and operator costs and unsatisifed demand	Routes and frequencies	Tabu Search
11	Fan (2009)	2009	Total travel distance and transfers	Routes	SA
12	Yu et al. (2010)	2010	Total distance and transfers	Routes and frequencies	GA
13	Bagloee and Ceder (2011)	2011	Minimum passenger discomfort	Routes and frequencies	GA and decomposition
14	Szeto and Wu (2011)	2011	Total travel time	Routes and frequencies	GA
15	Cipriani et al. (2012)	2012	Passenger and operator costs	Routes and frequencies	Parallel GA
16	Chew and Lee (2012)	2012	Passenger cost	Routes	GA
17	Mumford (2013)	2013	Passenger and operator costs	Routes	EA
18	Chew et al. (2013)	2013	Passenger and operator costs	Routes	GA
19	Nikolić and Teodorović (2013)	2013	total travel time	Routes	BCO
20	Afandizadeh et al. (2013)	2013	Passenger and Operator costs	Routes	GA
21	John et al. (2014)	2014	Passenger and operator costs	Routes	NSGAII
22	Cooper et al. (2014)	2014	Passenger and operator costs	Routes	Parallel GA
23	Nayeem et al. (2014)	2014	Passenger costs	Routes	GA
24	Nikolić and Teodorović (2014)	2014	Passenger and operator cost, unsatisfied demand	Routes and frequencies	BCO
25	Kılıç and Gök (2014)	2014	Passenger and operator costs	Routes	Tabu Search
26	Amiripour et al. (2014)	2014	total waiting time	Routes	GA
27	Kechagiopoulos and Beligiannis (2014)	2014	Passenger and operator costs	Routes	PSO
28	Arbex and da Cunha (2015)	2015	Passenger and operator costs	Routes and frequencies	GA
29	Zhao et al. (2015)	2015	Passenger cost and unsatisfied demand	Routes and frequencies	Memetic Algorithm
30	Owais et al. (2015)	2015	Passenger and operator costs	Routes and frequencies	GA
31	Nayeem et al. (2018)	2018	Multi-objective	Routes	EA
32	Buba and Lee (2018)	2018	Passenger cost and unsatisfied demand	Routes and frequencies	Differential Evolution (DE)
33	Buba and Lee (2019)	2019	Passenger and operator costs	Routes and frequencies	DE-PSO
34	Heyken Soares et al. (2019)	2019	Passenger and operator costs	Routes	NSGAII
35	Islam et al. (2019)	2019	Minimise travel time and maximise satisified demand	Routes	Stochastic Beam Search (SBS)
36	Fan et al. (2019)	2019	Average passenger travel time and number of transfers	Routes	Flower Pollination Algorithm (FPA)
37	Jha et al. (2019)	2019	Travel time and operator cost	Routes and frequencies	MMOPSO/GA
38	Duran et al. (2019)	2019	Total travel time and CO_2 emmisions	Routes and frequencies	GA
39	Mahdavi Moghaddam et al. (2019)	2019	Passenger and operator costs	Routes and frequencies	GA
40	Duran-Micco et al. (2020)	2020	Total travel time and CO_2 emissions	Routes and frequencies	Memetic Algorithm (MA)
41	Yang and Jiang (2020)	2020	Passenger and operator costs	Routes	NSGAII
42	Chai and Liang (2020)	2020	Passenger travel time and number of vehicles	Routes and frequencies	NSGAII
43	Liang et al. (2020)	2020	Passenger and operator costs	Routes and frequencies	Cooperative coevolutinary MOEA

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.

9. 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 (Dantzig and Ramser, 1959) 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 (Clarke and Wright, 1964) 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 Toth and Vigo (2002); Laporte (1992); Campbell and Wilson (2014); Laporte (2009).

The CVRP problem has been tackled by several approaches and algorithms including exact mathematical approaches, with branch-and-cut (BC) algorithms (Augerat et al., 1995; Lysgaard et al., 2004), and algorithms based on the set partitioning formulation (Balinski and Quandt, 1964; Fukasawa et al., 2006) being the most successful. One of the first attempts to describe an exact BC algorithm for solving the CVRP is in Augerat et al. (1995), 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 (1981) 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 (1964), 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 (2005) 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 Prins (2004); Mester and Bräysy (2007); Nagata (2007); Nagata and Bräysy (2009).

For the version of CVRP with Time Windows (CVRPTW), exact methods have been successful for cases with up to 100 customers (Kolen et al., 1987), 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 (Solomon, 1987; Potvin and Rousseau, 1993; Russell, 1995), and other studies that applied meta-heuristics such as genetic algorithms, ant colony, tabu search, and simulated annealing are in (Belhaiza et al., 2014; Cheng and Wang, 2009; Ding et al., 2012; Tavakkoli-Moghaddam et al., 2011).

In Beltrami and Bodin (1974), 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. (2013) 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. (2015) 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. (2008) 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 (Campbell and Wilson, 2014) for more literature on the PVRP.

Mirzaei and Wøhlk (2017) 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 (Cattaruzza et al., 2014) 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 (Gu et al., 2019) 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. (Zhang and Chen, 2014) 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 Dror and Trudeau (1989, 1990). After these studies, most of the subsequent work focused on meta-heuristics or hybrid schemes. One example is the work of Archetti and Speranza (2012) who applied a tabu search algorithm, and Boudia et al. (2007) used a genetic algorithm combined with a local search procedure. Hybrid algorithms have then grown in popularity, examples are found in (Archetti et al., 2008; Cheng and Wang, 2009). Many exact models have also been proposed on this problem, and one example is the study in (Archetti et al., 2011). For further literature on SDVRP we refer to (Archetti and Speranza, 2012).

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. (2010) 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 (Xu and Chiu, 2001), 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. Kovacs et al. (2012) 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. Pillac et al. (2013) 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. Xie et al. (2017) 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.

10. 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 (1963) 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. (2000) 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. (2010) 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 study.

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 [2] is selected and applied to an initial created solution at each iteration, and a decision is made if the new solution

[2]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.

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 Özcan et al. (2008): 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 6. 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 Bilgin et al. (2006), 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.

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 $LLH = [llh_1, llh_2, \dots, llh_{|LLH|}]$ 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 $S_b$;
```

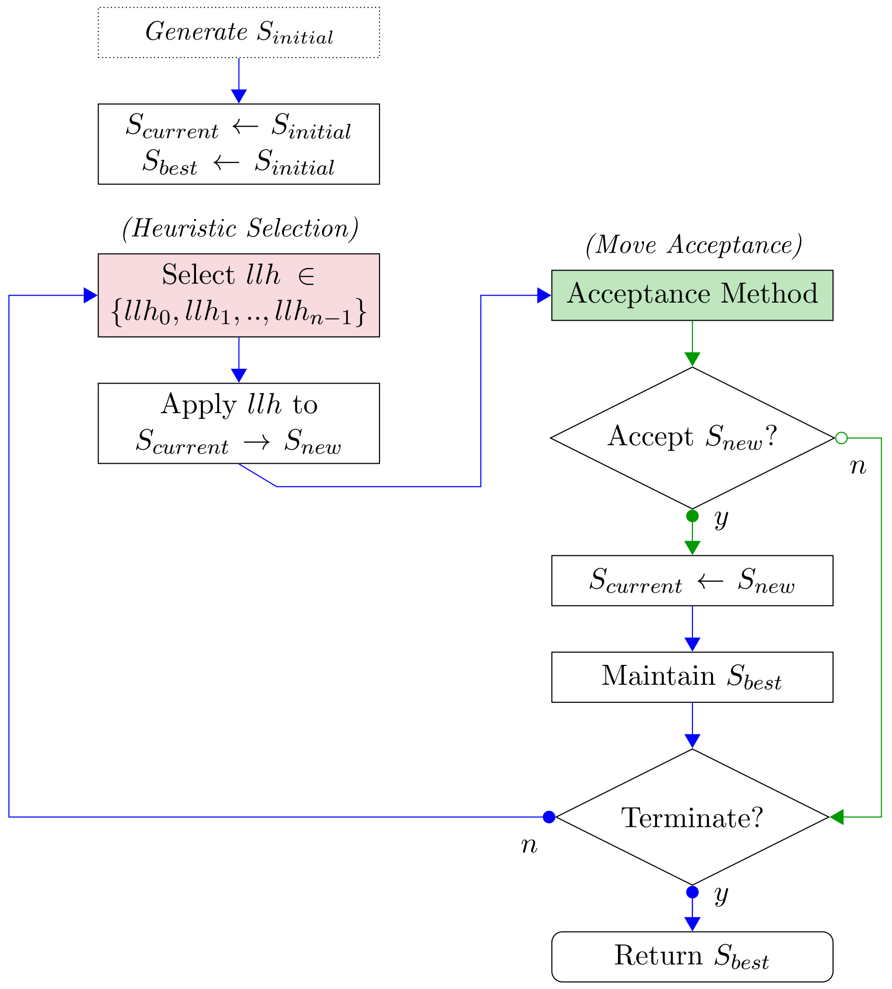

Generate $S_{initial}$
$S_{current} \leftarrow S_{initial}$
$S_{best} \leftarrow S_{initial}$
(Heuristic Selection)
Select $llh \in \{llh_0, llh_1, .., llh_{n-1}\}$
Apply llh to $S_{current} \rightarrow S_{new}$
(Move Acceptance)
Acceptance Method
Accept S_{new}?
n
y
$S_{current} \leftarrow S_{new}$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

Figure 6: A generic selection hyper-heuristic framework. The green arrows represent the acceptance component

10.1. Classification of Selection Hyper-heuristics

Hyper-heuristic algorithms have been classified into sub-categories based on a number of criteria. Burke et al. (2010) 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 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 Burke et al. (2013). Perturbative hyper-heuristics work on complete solutions, while the constructive process partially built solutions.

Drake et al. (2019), 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.

10.2. 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 Bilgin et al. (2006). 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 Kheiri (2020).

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 Özcan et al. (2012); Nareyek (2003); Burke et al. (2003). 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 11.1.

10.3. 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 6). 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. (2002) 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. Sabar and Kendall (2015) 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. Lei et al. (2015) 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. Hsiao et al. (2012) 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. Lehrbaum and Musliu (2012) 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).

11. 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 study were identified in Cowling et al. (2000). 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 paper 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 7.3.3) has been applied as a move acceptance component in selection hyper-heuristics by several studies Bilgin et al. (2006); Kalender et al. (2013) and has proved to be successful. In Bilgin et al. (2006), simulated annealing was used with the probability of accepting worsening moves given by the formula:

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

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 (1993). 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:

$$\tau_t = f_0 + \Delta F \times (1 - \frac{t}{T}) \tag{2}$$

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 Burke and Bykov (2008). 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 Burke and Bykov (2008). 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 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:

$$obj(S_{new}) \leq obj(s_{best}) + fr \times obj(S_{best}) \tag{3}$$

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

11.1. 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 study, as one of our selection methods, we have utilised a scheme inspired by the hidden Markov model (HMM) Kheiri and Keedwell (2017) 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

```
Let $S, S', S_b$ be candidate, new and best solutions, respectively;
Let $Tran$ be the transition matrix;
Let $Seq$ be the sequence construction matrix;
Let $[lh_0, llh_1, llh_2, \ldots, llh_{n-1}]$ be the low level heuristics set;
Let $HeuristicsSequence$ be the application sequence of low level
  heuristics;
$HeuristicsSequence \leftarrow [\ \ ]$;
$curr \leftarrow$ SelectRandomly$[0, 1, 2, \ldots, n-1]$;
$HeuristicsSequence$.add($llh_{curr}$);
repeat
    $next \leftarrow$ SelectNext($Tran, curr$);
    $HeuristicsSequence$.add($llh_{next}$);
    $Status \leftarrow$ ComputeStatus($Seq, next$);
    if $Status = end$ then
        $S' \leftarrow$ Apply($HeuristicsSequence, S$);
        if $S'$ isBetterThan $S_b$ then
            $S_b \leftarrow S'$;
            Update($Tran, Seq$);
        end
        $S \leftarrow$ Accept($S, S'$);
        $HeuristicsSequence$.clear();
    end
    $curr \leftarrow next$;
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

	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 7: 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.

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 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 Kheiri and Keedwell (2015, 2017).

12. Hyper-heuristics in Routing Problems

Since the development of the hyper-heuristics framework, it has been utilised for solving various important combinatorial optimisation problems such as timetabling Bilgin et al. (2006); Ahmed et al. (2015), personnel

Table 2: Some selected routing problems in which hyper-heuristics were used as solution methodologies

Problem Domain	Reference
Ready-mix concrete delivery	Misir et al. (2011)
Dynamic capacitated vehicle routing	Garrido and Castro (2012)
Capacitated vehicle routing	Marshall et al. (2015)
Dial-a-ride with time window	Urra et al. (2015)
Periodic vehicle routing	Chen et al. (2016)
Vehicle routing with cross-docking	Yin et al. (2016)
Inventory routing problem	Kheiri (2020)

scheduling Cowling and Chakhlevitch (2003) routing Pisinger and Ropke (2007); Garrido and Castro (2012); Walker et al. (2012), Bin Packing Ross et al. (2002), and Constraint Satisfaction Terashima-Marín et al. (2008). VRPs are area in which hyper-heuristics proved to be particularly successful, and have been applied to solve different variants of VRPs (table 2). Pisinger and Ropke (2007) 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 (2009) presented 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 Garrido and Castro (2012) 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. (2012) 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 (Kheiri et al., 2019).

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 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.

13. Methods for Solving Multi-objective Optimisation Problems

13.1. Evolutionary Algorithms

Evolutionary algorithms are a broad category of population-based metaheuristics 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 7.3.1) 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

Konak et al. (2006). 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) Fonseca et al. (1993), Strength Pareto Evolutionary Algorithm (SPEA) Zitzler and Thiele (1999), and Fast Non-dominated Sorting Genetic Algorithm (NSGAI and NSGAII) Deb et al. (2002).

13.2. 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:

$$\sum_{i=1}^{k} w_i f_i(x) \tag{4}$$

Where $w_i \geq 0$, and $\sum_{i=1}^{k} w_i = 1$. Ideally, the weight values in equation 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 Grodzevich and Romanko (2006). 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 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 8). Therefore, multi-objective frameworks based on the weighted sum approach have difficulties in finding solutions over a non-convex trade-off surface Konak et al. (2006).

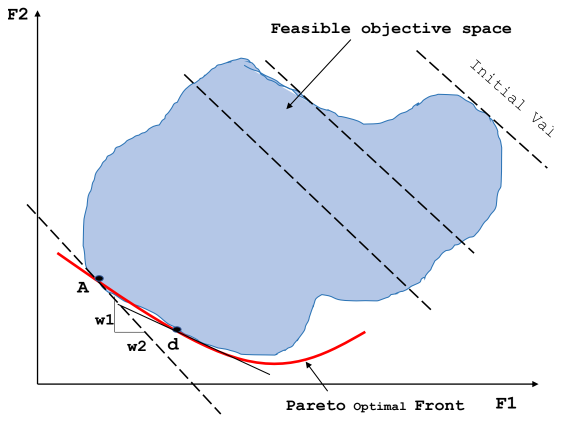
13.3. The Applied Weighted Sum Method

Throughout this paper, 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.

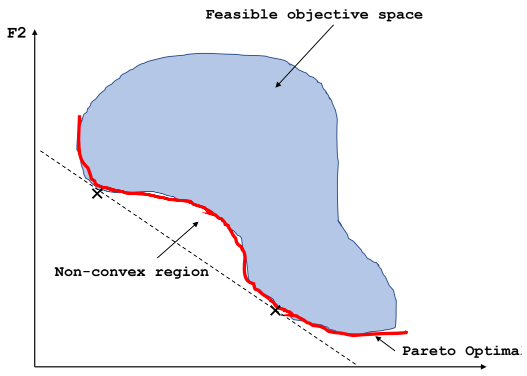
References

Aarts, E., Lenstra, J. K., 2003. Local search in combinatorial optimization. Princeton University Press.

Afandizadeh, S., Khaksar, H., Kalantari, N., 2013. Bus fleet optimization using genetic algorithm a case study of mashhad. International Journal of Civil Engineering 11 (1), 43–52.



(a) Weighted sum approach in a minimisation problem



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

Figure 8: Illustration of the weighted sum approach in a minimisation problem and in non-convex Pareto front

Agrawal, J., Mathew, T. V., 2004. Transit route network design using parallel genetic algorithm. Journal of Computing in Civil Engineering 18 (3), 248–256.

Ahmed, L., Heyken Soares, P., Mumford, C., Mao, Y., 2019a. 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. ACM, pp. 1102–1110.

Ahmed, L., Mumford, C., Kheiri, A., 2019b. Solving urban transit route design problem using selection hyper-heuristics. European Journal of Operational Research 274 (2), 545–559.

Ahmed, L. N., Özcan, E., Kheiri, A., 2015. Solving high school timetabling problems worldwide using selection hyper-heuristics. Expert Systems with Applications 42 (13), 5463–5471.

Alonso, F., Alvarez, M. J., Beasley, J. E., 2008. 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.

Amiripour, S. M., Ceder, A. A., Mohaymany, A. S., 2014. Designing large-scale bus network with seasonal variations of demand. Transportation Research Part C: Emerging Technologies 48, 322–338.

Arbex, R. O., da Cunha, C. B., 2015. Efficient transit network design and frequencies setting multi-objective optimization by alternating objective genetic algorithm. Transportation Research Part B: Methodological 81, 355–376.

Archetti, C., Bianchessi, N., Speranza, M. G., 2011. A column generation approach for the split delivery vehicle routing problem. Networks 58 (4), 241–254.

Archetti, C., Jabali, O., Speranza, M. G., 2015. Multi-period vehicle routing problem with due dates. Computers & Operations Research 61, 122–134.

Archetti, C., Speranza, M. G., 2012. Vehicle routing problems with split deliveries. International transactions in operational research 19 (1-2), 3–22.

Archetti, C., Speranza, M. G., Savelsbergh, M. W., 2008. An optimization-based heuristic for the split delivery vehicle routing problem. Transportation Science 42 (1), 22–31.

Augerat, P., Belenguer, J. M., Benavent, E., Corberán, A., Naddef, D., Rinaldi, G., 1995. Computational results with a branch and cut code for the capacitated vehicle routing problem. IMAG.

Baaj, M. H., Mahmassani, H. S., 1991. An AI-based approach for transit route system planning and design. Journal of Advanced Transportation 25 (2), 187–209.

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

Bagloee, S. A., Ceder, A. A., 2011. Transit-network design methodology for actual-size road networks. Transportation Research Part B: Methodological 45 (10), 1787–1804.
URL `http://dx.doi.org/10.1016/j.trb.2011.07.005`

Balinski, M. L., Quandt, R. E., 1964. On an integer program for a delivery problem. Operations research 12 (2), 300–304.

Barra, A., Carvalho, L., Teypaz, N., Cung, V.-D., Balassiano, R., 2007. Solving the transit network design problem with constraint programming.

Beasley, D., Bull, D. R., Martin, R. R., 1993. An overview of genetic algorithms: Part 1, fundamentals. University computing 15 (2), 56–69.

Belhaiza, S., Hansen, P., Laporte, G., 2014. A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers & Operations Research 52, 269–281.

Bellman, R., 1952. On the theory of dynamic programming. Proceedings of the National Academy of Sciences of the United States of America 38 (8), 716.

Beltrami, E. J., Bodin, L. D., 1974. Networks and vehicle routing for municipal waste collection. Networks 4 (1), 65–94.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. Top 15 (1), 1–31.

Bielli, M., Caramia, M., Carotenuto, P., 2002. Genetic algorithms in bus network optimization. Transportation Research Part C: Emerging Technologies 10 (1), 19–34.

Bilgin, B., Özcan, E., Korkmaz, E. E., 2006. An experimental study on hyper-heuristics and exam timetabling. In: International Conference on the Practice and Theory of Automated Timetabling. pp. 394–412.

Blum, C., Merkle, D., 2008. Swarm intelligence. Swarm Intelligence in Optimization; Blum, C., Merkle, D., Eds, 43–85.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM computing surveys (CSUR) 35 (3), 268–308.

Boudia, M., Prins, C., Reghioui, M., 2007. An effective memetic algorithm with population management for the split delivery vehicle routing problem. In: International Workshop on Hybrid Metaheuristics. Springer, pp. 16–30.

Boussaïd, I., Lepagnot, J., Siarry, P., 2013. A survey on optimization metaheuristics. Information sciences 237, 82–117.

Buba, A. T., Lee, L. S., 2018. A differential evolution for simultaneous transit network design and frequency setting problem. Expert Systems with Applications 106, 277–289.

Buba, A. T., Lee, L. S., 2019. Hybrid differential evolution-particle swarm optimization algorithm for multiobjective urban transit network design problem with homogeneous buses. Mathematical Problems in Engineering 2019.

Burke, E. K., Bykov, Y., 2008. A late acceptance strategy in hill-climbing for exam timetabling problems. In: PATAT 2008 Conference, Canada.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R., 2013. Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society 64 (12), 1695–1724.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J. R., 2010. A classification of hyper-heuristic approaches. In: Handbook of Metaheuristics. Springer, pp. 449–468.

Burke, E. K., Kendall, G., Soubeiga, E., 2003. A tabu-search hyperheuristic for timetabling and rostering. Journal of heuristics 9 (6), 451–470.

Bussieck, M., 1998. Optimal lines in public rail transport. Ph.D. thesis, Citeseer.

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

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

Campbell, A. M., Wilson, J. H., 2014. Forty years of periodic vehicle routing. Networks 63 (1), 2–15.

Cattaruzza, D., Absi, N., Feillet, D., Vigo, D., 2014. An iterated local search for the multi-commodity multi-trip vehicle routing problem with time windows. Computers & Operations Research 51, 257–267.

Ceder, A., 2016. Public transit planning and operation: Modeling, practice and behavior. CRC press.

Ceder, A., Wilson, N. H., 1986. Bus network design. Transportation Research Part B: Methodological 20 (4), 331–344.

Chai, S., Liang, Q., 2020. An improved nsga-ii algorithm for transit network design and frequency setting problem. Journal of Advanced Transportation 2020.

Chakroborty, P., 2003. Genetic algorithms for optimal urban transit network design. Computer-Aided Civil and Infrastructure Engineering 18 (3), 184–200.

Chakroborty, P., Wivedi, T., 2002. Optimal route network design for transit systems using genetic algorithms. Engineering Optimization 34 (1), 83–100.

Chang, S. K., Schonfeld, P. M., 1991. Multiple period optimization of bus transit systems. Transportation Research Part B: Methodological 25 (6), 453–478.

Chang, S. K., Schonfeld, P. M., 1993. Welfare maximization with financial constraints for bus transit systems. Transportation Research Record (1395).

Chen, Y., Mourdjis, P., Polack, F., Cowling, P., Remde, S., 2016. Evaluating hyperheuristics and local search operators for periodic routing problems. In: European Conference on Evolutionary Computation in Combinatorial Optimization. Springer, pp. 104–120.

Cheng, C.-B., Wang, K.-P., 2009. Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm. Expert Systems with Applications 36 (4), 7758–7763.

Chew, J. S. C., Lee, L. S., 2012. A genetic algorithm for urban transit routing problem. In: International Journal of Modern Physics: Conference Series. Vol. 9. World Scientific, pp. 411–421.

Chew, J. S. C., Lee, L. S., Seow, H. V., 2013. Genetic algorithm for biobjective urban transit routing problem. Journal of Applied Mathematics.

Chien, S., Schonfeld, P., 1997. Optimization of grid transit system in heterogeneous urban environment. Journal of Transportation Engineering 123 (1), 28–35.

Cipriani, E., Gori, S., Petrelli, M., 2012. Transit network design: A procedure and an application to a large urban area. Transportation Research Part C: Emerging Technologies 20 (1), 3–14.
URL `http://dx.doi.org/10.1016/j.trc.2010.09.003`

Clarke, G., Wright, J. W., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations research 12 (4), 568–581.

Cook, S. A., 1971. The complexity of theorem-proving procedures. In: Proceedings of the third annual ACM symposium on Theory of computing. pp. 151–158.

Cooper, I. M., John, M. P., Lewis, R., Mumford, C. L., Olden, A., 2014. Optimising large scale public transport network design problems using mixed-mode parallel multi-objective evolutionary algorithms. In: IEEE Congress on Evolutionary Computation (CEC),. IEEE, pp. 2841–2848.

Cordeau, J.-F., Laporte, G., Pasin, F., Ropke, S., 2010. Scheduling technicians and tasks in a telecommunications company. Journal of Scheduling 13 (4), 393–409.

Cowling, P., Chakhlevitch, K., 2003. Hyperheuristics for managing a large collection of low level heuristics to schedule personnel. In: Evolutionary Computation, 2003. CEC'03. The 2003 Congress on. Vol. 2. IEEE, pp. 1214–1221.

Cowling, P., Kendall, G., Han, L., 2002. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600). Vol. 2. IEEE, pp. 1185–1190.

Cowling, P., Kendall, G., Soubeiga, E., 2000. A hyperheuristic approach to scheduling a sales summit. In: International Conference on the Practice and Theory of Automated Timetabling. Springer, pp. 176–190.

Dantzig, G. B., Ramser, J. H., 1959. The truck dispatching problem. Management science 6 (1), 80–91.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation 6 (2), 182–197.

Ding, Q., Hu, X., Sun, L., Wang, Y., 2012. An improved ant colony optimization and its application to vehicle routing problem with time windows. Neurocomputing 98, 101–107.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26 (1), 29–41.

Drake, J. H., Kheiri, A., Özcan, E., Burke, E. K., 2019. Recent advances in selection hyper-heuristics. European Journal of Operational Research.

Dror, M., Trudeau, P., 1989. Savings by split delivery routing. Transportation Science 23 (2), 141–145.

Dror, M., Trudeau, P., 1990. Split delivery routing. Naval Research Logistics (NRL) 37 (3), 383–402.

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

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

Duran, J., Pradenas, L., Parada, V., 2019. Transit network design with pollution minimization. Public Transport 11 (1), 189–210.

Duran-Micco, J., Vermeir, E., Vansteenwegen, P., 2020. Considering emissions in the transit network design and frequency setting problem with a heterogeneous fleet. European Journal of Operational Research 282 (2), 580–592.

Fan, L., 2009. Metaheuristic methods for the urban transit routing problem. Ph.D. thesis, Cardiff University.

Fan, L., Chen, H., Gao, Y., 2019. An improved flower pollination algorithm to the urban transit routing problem. Soft Computing, 1–10.

Fan, L., Mumford, C. L., 2010. A metaheuristic approach to the urban transit routing problem. Journal of Heuristics 16 (3), 353–372.

Fan, W., Machemehl, R. B., 2006a. Optimal transit route network design problem with variable transit demand: genetic algorithm approach. Journal of Transportation Engineering 132 (1), 40–51.

Fan, W., Machemehl, R. B., 2006b. Using a Simulated Annealing Algorithm to Solve the Transit Route Network Design Problem. Journal of Transportation Engineering 132(2).

Fan, W., Machemehl, R. B., 2008. A tabu search based heuristic method for the transit route network design problem. In: Computer-aided Systems in Public Transport. Springer, pp. 387–408.

Farahani, R. Z., Miandoabchi, E., Szeto, W. Y., Rashidi, H., 2013. A review of urban transportation network design problems. European Journal of Operational Research 229 (2), 281–302.

Feo, T. A., Resende, M. G., 1995. Greedy randomized adaptive search procedures. Journal of global optimization 6 (2), 109–133.

Fisher, H., 1963. Probabilistic learning combinations of local job-shop scheduling rules. Industrial scheduling, 225–251.

Fisher, M. L., Jaikumar, R., 1981. A generalized assignment heuristic for vehicle routing. Networks 11 (2), 109–124.

Fonseca, C. M., Fleming, P. J., et al., 1993. Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization. In: Icga. Vol. 93. Citeseer, pp. 416–423.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., Werneck, R. F., 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical programming 106 (3), 491–511.

Garrido, P., Castro, C., 2009. Stable solving of cvrps using hyperheuristics. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation. pp. 255–262.

Garrido, P., Castro, C., 2012. A flexible and adaptive hyper-heuristic approach for (dynamic) capacitated vehicle routing problems. Fundamenta Informaticae 119 (1), 29–60.

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

Grodzevich, O., Romanko, O., 2006. Normalization and other topics in multi-objective optimization.

Gu, W., Cattaruzza, D., Ogier, M., Semet, F., 2019. Adaptive large neighborhood search for the commodity constrained split delivery vrp. Computers & Operations Research 112, 104761.

Guan, J., Yang, H., Wirasinghe, S. C., 2006. Simultaneous optimization of transit line configuration and passenger line assignment. Transportation Research Part B: Methodological 40 (10), 885–902.

Heyken Soares, P., Mumford, C. L., Amponsah, K., Mao, Y., 2019. An adaptive scaled network for public transport route optimisation. Public Transport 11 (2), 379–412.

Holroyd, E., 1967. 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.

Hsiao, P.-C., Chiang, T.-C., Fu, L.-C., 2012. A vns-based hyper-heuristic with adaptive computational budget of local search. In: 2012 IEEE Congress on Evolutionary Computation. IEEE, pp. 1–8.

Islam, K. A., Moosa, I. M., Mobin, J., Nayeem, M. A., Rahman, M. S., 2019. A heuristic aided stochastic beam search algorithm for solving the transit network design problem. Swarm and Evolutionary Computation 46, 154–170.

Jha, S. B., Jha, J. K., Tiwari, M. K., 2019. 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.

John, M. P., 2016. Metaheuristics for designing efficient routes & schedules for urban transportation networks. Ph.D. thesis, Cardiff University.

John, M. P., Mumford, C. L., Lewis, R., 2014. An improved multi-objective algorithm for the urban transit routing problem. In: European Conference on Evolutionary Computation in Combinatorial Optimization. Springer, pp. 49–60.

Kalender, M., Kheiri, A., Özcan, E., Burke, E. K., 2013. A greedy gradient-simulated annealing selection hyper-heuristic. Soft Computing 17 (12), 2279–2292.

Karp, R. M., 1972. Reducibility among combinatorial problems. In: Complexity of computer computations. Springer, pp. 85–103.

Kechagiopoulos, P. N., Beligiannis, G. N., 2014. Solving the urban transit routing problem using a particle swarm optimization based algorithm. Applied Soft Computing 21, 654–676.

Kheiri, A., 2020. Heuristic sequence selection for inventory routing problem. Transportation Science 54 (2), 302–312.

Kheiri, A., Dragomir, A. G., Mueller, D., Gromicho, J., Jagtenberg, C., van Hoorn, J. J., 2019. Tackling a vrp challenge to redistribute scarce equipment within time windows using metaheuristic algorithms. EURO Journal on Transportation and Logistics, 1–35.

Kheiri, A., Keedwell, E., 2015. A sequence-based selection hyper-heuristic utilising a hidden Markov model. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. ACM, pp. 417–424.

Kheiri, A., Keedwell, E., 2017. 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.

Kılıç, F., Gök, M., 2014. A demand based route generation algorithm for public transit network design. Computers & Operations Research 51, 21–29.

Kolen, A. W., Rinnooy Kan, A., Trienekens, H. W., 1987. Vehicle routing with time windows. Operations Research 35 (2), 266–273.

Konak, A., Coit, D. W., Smith, A. E., 2006. Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering & System Safety 91 (9), 992–1007.

Kovacs, A. A., Parragh, S. N., Doerner, K. F., Hartl, R. F., 2012. Adaptive large neighborhood search for service technician routing and scheduling problems. Journal of scheduling 15 (5), 579–600.

Lampkin, W., Saalmans, P., 1967. 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.

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

Laporte, G., 2009. Fifty years of vehicle routing. Transportation Science 43 (4), 408–416.

Lawler, E. L., Wood, D. E., 1966. Branch-and-bound methods: A survey. Operations research 14 (4), 699–719.

Lee, Y.-J., Vuchic, V. R., 2005. Transit network design with variable demand. Journal of Transportation Engineering 131 (1), 1–10.

Lehrbaum, A., Musliu, N., 2012. A new hyperheuristic algorithm for cross-domain search problems. In: International Conference on Learning and Intelligent Optimization. Springer, pp. 437–442.

Lei, Y., Gong, M., Jiao, L., Zuo, Y., 2015. A memetic algorithm based on hyper-heuristics for examination timetabling problems. International Journal of Intelligent Computing and Cybernetics 8 (2), 139–151.

Liang, M., Wang, W., Dong, C., Zhao, D., 2020. A cooperative coevolutionary optimization design of urban transit network and operating frequencies. Expert Systems with Applications 160, 113736.

Lourenço, H. R., Martin, O. C., Stützle, T., 2019. Iterated local search: Framework and applications. In: Handbook of metaheuristics. Springer, pp. 129–168.

Lucic, P., Teodorovic, D., 2001. Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence. In: Preprints of the TRISTAN IV triennial symposium on transportation analysis. pp. 441–445.

Lysgaard, J., Letchford, A. N., Eglese, R. W., 2004. A new branch-and-cut algorithm for the capacitated vehicle routing problem. Mathematical Programming 100 (2), 423–445.

Mahdavi Moghaddam, S. H., Rao, K. R., Tiwari, G., Biyani, P., 2019. Simultaneous bus transit route network and frequency setting search algorithm. Journal of Transportation Engineering, Part A: Systems 145 (4), 04019011.

Mandl, C., 1979. Applied network optimization. Operations Research and Industrial Engineering. Academic Press.

Mandl, C. E., 1980. Evaluation and optimization of urban public transportation networks. European Journal of Operational Research 5 (6), 396–404.

Marshall, R. J., Johnston, M., Zhang, M., 2015. Hyper-heuristic operator selection and acceptance criteria. In: European Conference on Evolutionary Computation in Combinatorial Optimization. Springer, pp. 99–113.

Mauttone, A., Urquhart, M. E., 2009. A route set construction algorithm for the transit network design problem. Computers and Operations Research 36 (8), 2440–2449.

Mester, D., Bräysy, O., 2007. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. Computers & Operations Research 34 (10), 2964–2975.

Miranda, D. M., Conceição, S. V., 2016. The vehicle routing problem with hard time windows and stochastic travel and service time. Expert Systems with Applications 64, 104–116.

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

Misir, M., Vancroonenburg, W., Verbeeck, K., Berghe, G. V., 2011. A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete. In: Proceedings of the Metaheuristics International Conference. pp. 289–298.

Montoya-Torres, J. R., Franco, J. L., Isaza, S. N., Jiménez, H. F., Herazo-Padilla, N., 2015. A literature review on the vehicle routing problem with multiple depots. Computers & Industrial Engineering 79, 115–129.

Morgan, M. J., Mumford, C. L., 2005. Capacitated vehicle routing: perturbing the landscape to fool an algorithm. In: 2005 IEEE Congress on Evolutionary Computation. Vol. 3. IEEE, pp. 2271–2277.

Mumford, C. L., 2013. New heuristic and evolutionary operators for the multi-objective urban transit routing problem. In: IEEE Congress on Evolutionary Computation (CEC), 2013. IEEE, pp. 939–946.

Nagata, Y., 2007. Edge assembly crossover for the capacitated vehicle routing problem. In: European Conference on Evolutionary Computation in Combinatorial Optimization. Springer, pp. 142–153.

Nagata, Y., Bräysy, O., 2009. Edge assembly-based memetic algorithm for the capacitated vehicle routing problem. Networks: An International Journal 54 (4), 205–215.

Nareyek, A., 2003. Choosing search heuristics by non-stationary reinforcement learning. In: Metaheuristics: Computer decision-making. Springer, pp. 523–544.

Nayeem, M. A., Islam, M. M., Yao, X., 2018. Solving transit network design problem using many-objective evolutionary approach. IEEE Transactions on Intelligent Transportation Systems 20 (10), 3952–3963.

Nayeem, M. A., Rahman, M. K., Rahman, M. S., 2014. Transit network design by genetic algorithm with elitism. Transportation Research Part C: Emerging Technologies 46, 30–45.

Ngamchai, S., Lovell, D. J., 2003. Optimal time transfer in bus transit route network design using a genetic algorithm. Journal of Transportation Engineering 129 (5), 510–521.

Nikolić, M., Teodorović, D., 2013. Transit network design by bee colony optimization. Expert Systems with Applications 40 (15), 5945–5955.

Nikolić, M., Teodorović, D., 2014. A simultaneous transit network design and frequency setting: Computing with bees. Expert Systems with Applications 41 (16), 7200–7209.

Owais, M., Osman, M. K., Moussa, G., 2015. Multi-objective transit route network design as set covering problem. IEEE Transactions on Intelligent Transportation Systems 17 (3), 670–679.

Özcan, E., Bilgin, B., Korkmaz, E. E., 2008. A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis 12 (1), 3–23.

Özcan, E., Misir, M., Ochoa, G., Burke, E. K., 2012. A reinforcement learning: great-deluge hyper-heuristic for examination timetabling. In: Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends. IGI Global, pp. 34–55.

Pacheco, J., Alvarez, A., Casado, S., González-Velarde, J. L., 2009. A tabu search approach to an urban transport problem in northern spain. Computers & Operations Research 36 (3), 967–979.

Papadimitriou, C. H., Steiglitz, K., 1998. Combinatorial optimization: algorithms and complexity. Courier Corporation.

Parragh, S. N., Doerner, K. F., Hartl, R. F., 2008. A survey on pickup and delivery problems. Journal für Betriebswirtschaft 58 (1), 21–51.

Pattnaik, S., Mohan, S., Tom, V., 1998a. Urban bus transit route network design using genetic algorithm. Journal of Transportation Engineering 124 (4), 368–375.

Pattnaik, S. B., Mohan, S., Tom, V., 1998b. Urban bus route network design using genetic algorithm. Reviewed by the Urban Transportation Division 37 (3), 24.

Patz, A., 1925. Die richtige auswahl von verkehrslinien bei großen strassenbahnnetzen. Verkehrstechnik 50, 51.

Pearl, J., 1984. Intelligent search strategies for computer problem solving. Addision Wesley.

Pillac, V., Gueret, C., Medaglia, A. L., 2013. A parallel matheuristic for the technician routing and scheduling problem. Optimization Letters 7 (7), 1525–1535.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Computers & operations research 34 (8), 2403–2435.

Poorzahedy, H., Rouhani, O. M., 2007. Hybrid meta-heuristic algorithms for solving network design problem. European Journal of Operational Research 182 (2), 578–596.

Potvin, J.-Y., Rousseau, J.-M., 1993. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. European Journal of Operational Research 66 (3), 331–340.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 31 (12), 1985–2002.

Rahimi-Vahed, A., Crainic, T. G., Gendreau, M., Rei, W., 2013. A path relinking algorithm for a multi-depot periodic vehicle routing problem. Journal of heuristics 19 (3), 497–524.

Ross, P., Schulenburg, S., Marín-Bläzquez, J. G., Hart, E., 2002. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation. pp. 942–948.

Russell, R. A., 1995. Hybrid heuristics for the vehicle routing problem with time windows. Transportation science 29 (2), 156–166.

Russell, R. A., Urban, T. L., 2008. Vehicle routing with soft time windows and erlang travel times. Journal of the Operational Research Society 59 (9), 1220–1228.

Sabar, N. R., Kendall, G., 2015. Population based monte carlo tree search hyper-heuristic for combinatorial optimization problems. Information Sciences 314, 225–239.

Simonis, C., 1981. Optimierung von Omnibuslinien. Berichte des Instituts für Stadtbauwesen (26).

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

Sonntag, H., 1977. Linienplanung im öffentlichen Personennahverkehr. Ph.D. thesis, Technical University Berlin.

Sörensen, K., Glover, F., 2013. Metaheuristics. Encyclopedia of operations research and management science 62, 960–970.

Szeto, W. Y., Wu, Y., 2011. A simultaneous bus route design and frequency setting problem for Tin Shui Wai, Hong Kong. European Journal of Operational Research 209 (2), 141–155.

Tavakkoli-Moghaddam, R., Gazanfari, M., Alinaghian, M., Salamatbakhsh, A., Norouzi, N., 2011. 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.

Terashima-Marín, H., Ortiz-Bayliss, J. C., Ross, P., Valenzuela-Rendón, M., 2008. Hyper-heuristics for the dynamic variable ordering in constraint satisfaction problems. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation. pp. 571–578.

Tom, V., Mohan, S., 2003. Transit route network design using frequency coded genetic algorithm. Journal of Transportation Engineering 129 (2), 186–195.

Toth, P., Vigo, D., 2002. The vehicle routing problem. SIAM.

Urra, E., Cubillos, C., Cabrera-Paniagua, D., 2015. A hyperheuristic for the dial-a-ride problem with time windows. Mathematical Problems in Engineering 2015.

van Nes, R., Hamerslag, R., Immers, L., 1988. The design of public transport networks. Vol. 1202. National Research Council, Transportation Research Board.

Van Oudheusden, D., Ranjithan, S., Singh, K., 1987. The design of bus route systems—an interactive location-allocation approach. Transportation 14 (3), 253–270.

Walker, J. D., Ochoa, G., Gendreau, M., Burke, E. K., 2012. Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In: International conference on learning and intelligent optimization. Springer, pp. 265–276.

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

Wan, Q. K., Lo, H. K., 2003. A mixed integer formulation for multiple-route transit network design. Journal of Mathematical Modelling and Algorithms 2 (4), 299–308.

Xie, F., Potts, C. N., Bektaş, T., 2017. Iterated local search for workforce scheduling and routing problems. Journal of Heuristics 23 (6), 471–500.

Xu, J., Chiu, S. Y., 2001. Effective heuristic procedures for a field technician scheduling problem. Journal of Heuristics 7 (5), 495–509.

Yang, J., Jiang, Y., 2020. Application of modified nsga-ii to the transit network design problem. Journal of Advanced Transportation 2020.

Yang, Z., Yu, B., Cheng, C., 2007. A parallel ant colony algorithm for bus network optimization. Computer-Aided Civil and Infrastructure Engineering 22 (1), 44–55.

Yin, P.-Y., Lyu, S.-R., Chuang, Y.-L., 2016. Cooperative coevolutionary approach for integrated vehicle routing and scheduling using cross-dock buffering. Engineering Applications of Artificial Intelligence 52, 40–53.

Yu, B., Yang, Z., Yao, J., 2010. Genetic algorithm for bus frequency optimization. Journal of Transportation Engineering 136 (6), 576–583.

Yu, B., Yang, Z.-z., Jin, P.-h., Wu, S.-h., Yao, B.-z., 2012. Transit route network design-maximizing direct and transfer demand density. Transportation Research Part C 22, 58–75.
URL http://dx.doi.org/10.1016/j.trc.2011.12.003

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

Zhao, H., Jiang, R., et al., 2015. The memetic algorithm for the optimization of urban transit network. Expert Systems with Applications 42 (7), 3760–3773.

Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE transactions on Evolutionary Computation 3 (4), 257–271.