Solving Urban Transit Route Design Problem using Selection Hyper-heuristics

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

Abstract

The urban transit routing problem (UTRP) focuses on finding efficient travelling routes for vehicles in a public transportation system. It is one of the most significant problems faced by transit planners and city authorities throughout the world. This problem belongs to the class of difficult combinatorial problems, whose optimal solution is hard to find with the complexity that arises from the large search space, and the number of constraints imposed in constructing the solution. Hyper-heuristics have emerged as general-purpose search techniques that explore the space of low level heuristics to improve a given solution under an iterative framework. In this work, we evaluate the performance of a set of selection hyper-heuristics on the route design problem of bus networks, with the goal of minimising the passengers' travel time, and the operator's costs. Each selection hyper-heuristic is empirically tested on a set of benchmark instances and statistically compared to the other selection hyper-heuristics to determine the best approach. A sequence-based selection method combined with the great deluge acceptance method achieved the best performance, succeeding in finding improved results in much faster run times over the current best known solutions.

Keywords: Transportation, Optimisation, Routing, Meta-heuristics

1. Introduction

The ever-increasing use of private transportation throughout the cities of the world is resulting in unacceptable levels of congestion, pollution, and

*corresponding author

environmental, social and economic cost. This has led to move towards improving public transportation services and encouraging citizens to use them more. The design of public transit systems is a complicated task that needs to satisfy the requirements of many stakeholders with conflicting needs, including passengers who are looking for a comfortable, reliable and fast service, and the transportation companies who aim to provide an adequate services within reasonable costs while achieving maximum profit.

Ceder and Wilson (1986) identified five main stages in the development of public transit: (1) network design (2) frequency setting (3) timetable development (3) bus scheduling, and (5) driver scheduling. In this work we apply hyper-heuristics to the network design element referred to as urban transit routing problem (UTRP), which focuses on finding efficient routes in a public transportation system.

Public transit route planning works with a pre-defined road network that has pre-defined pick-up, and drop-off locations, with the goal of finding efficient and feasible routes that satisfy all passenger demand, and reduce travel times, transfers, and costs. The vast search space and the many constraints involved means that it can be a challenging task to attain feasible solutions especially when the network size increases.

Meta-heuristics have enjoyed some success on versions of the UTRP, with genetic algorithms (GAs) as a particularly popular choice (Chakroborty and Wivedi, 2002; Fan and Machemehl, 2006; Fan and Mumford, 2010; Mumford, 2013). However with the very long run times involved for population-based methods for this problem, and the complicated parameter setting and tuning needed in most cases, applications tend to be limited to relatively small instances. In this paper 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.

Hyper-heuristics are general methodologies that mix and control a set of pre-defined low level heuristics while solving computationally difficult problems (Burke et al., 2013). There are two main classes of hyper-heuristics identified in the literature (Burke et al., 2010), *selection* hyper-heuristics which select from a known set of low level heuristics, and *generation* hyper-heuristics that generate new heuristics from components of existing ones. In this study, we focus on the selection class in which an initial solution is iteratively improved, passing through two stages of *selection* and *move* *acceptance*, until satisfying a termination criterion.

The present paper focusses 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 of the operators. We implement several problem-specific low level heuristics, and moreover explain in depth our applied methodology and test our set of selection hyper-heuristics on a range benchmark instances.

Section 2 summarises the previous research on this problem, outlining key approaches from the literature. Section 3 describes the UTRP in depth, explaining its objectives and constraints. The hyper-heuristic approach applied to the problem and the low level heuristics we use are explained in section 4. In section 5, our results are presented and analysed and then compared with the current best-known results. Finally section 6 presents the conclusion and future plans.

2. Related Work

2.1. 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. Mathematical programming techniques such as in (Byrne, 1975; Schéele, 1980; Vaughan, 1986; Wan and Lo, 2003; Guan et al., 2006) have been proposed to solve the UTRP. However, such methods are suitable only for very small sized instances. As a result, heuristic methods have been popular and applied in many studies. 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. Following that, Mandl (1979, 1980) applied a two stage procedure, where a feasible route set is first generated, followed by a heuristic procedure to improve the set quality. His pioneering work produced the Swiss 15 node instance, which has become a defacto benchmark used by most researchers. 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 set. Lee and Vuchic (2005) developed an iterative heuristic procedure to solve the network design and frequency setting particularly with variable demand under a total fixed demand. The objective was to decrease the total travel time through an improvement procedure on an initially generated network utilising the shortest path. A transit assignment procedure was also applied to concentrate demand on certain routes, eliminating less efficient routes.

In recent years, meta-heuristic methods have become a popular choice for solving the UTRP, with a specific focus on genetic algorithms. Pattnaik et al. (1998) 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 that time. In (Chakroborty, 2003), the author 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.

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 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. Other studies that applied GAs to the route design problem and simultaneously incorporated route frequencies can be seen in (Fan and Machemehl, 2006; Szeto and Wu, 2011; Bagloee and Ceder, 2011; Cipriani et al., 2012).

In addition to GAs, other meta-heuristic approaches have been successfully applied to the UTRP. Fan and Mumford (2010) applied hill-climbing and simulated annealing. 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. Other meta-heuristic methods applied to the UTRP include bee colony (Nikolić and Teodorović, 2013, 2014) and particle swarm optimisation (Kechagiopoulos and Beligiannis (2014)).

Several works used meta-heuristic approaches to handle real case problems for specific cities and towns. Poorzahedy and Rouhani (2007) hybridized a previously implemented ant system with concepts of meta-heuristic algorithms including genetic algorithms, tabu search, and simulated annealing to solve the network design problem. Seven hybrids were created and tested on two networks: the network of Sioux Falls, and the real network of Mashhad city in Iran that contains over 1000 nodes. Pacheco et al. (2009) applied local search and tabu search strategies to solve a bus network design problem of Burgos city in Northern Spain to minimise the trip duration and waiting time at bus stops. Other real case applications are in (Zhao and Zeng, 2006; Mauttone and Urquhart, 2009).

Several books and review papers have presented a more comprehensive coverage of previous literature in public transit planning. The interested reader can refer to (Ceder, 2016; Guihaire and Hao, 2008; Kepaptsoglou and Karlaftis, 2009; Schöbel, 2012; Farahani et al., 2013; Ibarra-Rojas et al.,

Table 1: 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)

2015).

There are some clear limitations with most previous approaches to solving the UTRP problem. The lack of benchmark data for the problem is a serious issue, and many researchers implemented methods that are highly specific to given towns or cities. Furthermore, these instances are not publicly available so we cannot judge the generality of the applied methods. Other researchers who implemented and tested their methods on benchmark instances used Mandl's 15 node benchmark, which is a very small instance. We cannot judge the performance of a method in terms of scalability based on such a small network. If we consider GA approaches in particular, with a population of perhaps several hundred solutions to maintain, the run-time for a road network of 100 vertices or above can be measured in days rather than hours. To the best of the authors' knowledge, the use of hyper-heuristics for the UTRP remains 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 UTRP with minimal adaptation. (ii) The success of hyper-heuristics in solving several NP-hard optimisation problems generally and complex routing problems specifically (Table 1). (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 with the scalability of the instances.

2.2. Selection Hyper-heuristics

The motivation behind hyper-heuristics is to develop methods that are more generally applicable than other implementations of search methodologies. When using hyper-heuristics, the search is achieved at a more abstract level (i.e. the space of low level heuristics), instead of the space of problem solutions. 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. Burke et al. (2010) classified hyper-heuristics based on the nature of the heuristics 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 is the focus of our study.

The traditional selection hyper-heuristic framework consists of two consecutive processes: 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 until meeting a termination condition. We will now outline the set of selection and move acceptance methods used in our study.

Most of the simple selection methods were identified in (Cowling et al., 2000). 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 in our study 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 in our study are: Simulated Annealing (SA), Great Deluge (GD) and Late Acceptance (LA).

Simulated annealing 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: $p_t = e^{-\frac{\Delta f}{\Delta F(1-\frac{t}{T})}}$, 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.

Great deluge accepts all improved solutions, and the 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 formula: $\tau_t = f_0 + \Delta F \times (1 - \frac{t}{T})$, 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).

2.2.1. 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 our 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 1.

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

Algorithm 1: Sequence-based selection hyper-heuristic

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

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 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).

3. Problem Description

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

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

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

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

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

A route $r_a = \{v_{i_1}, v_{i_2}, \ldots, v_{i_q}\}$, where $v_{i_k} \in \{v_1, v_2, \ldots, v_n\}$, is defined as a simple path in the graph connecting a set of edges. A route network $R = \{r_a : 1 \leq a \leq |R|\}$, is a connected set of routes, and a subgraph of the transport network. It should contain all the vertices present in the transport network, and a subset of its edges. The route network is what actually represents a solution to our model. We will also define the *transit network*, which is the network constructed during the evaluation of the route network. During the evaluation, each node that corresponds to a transfer

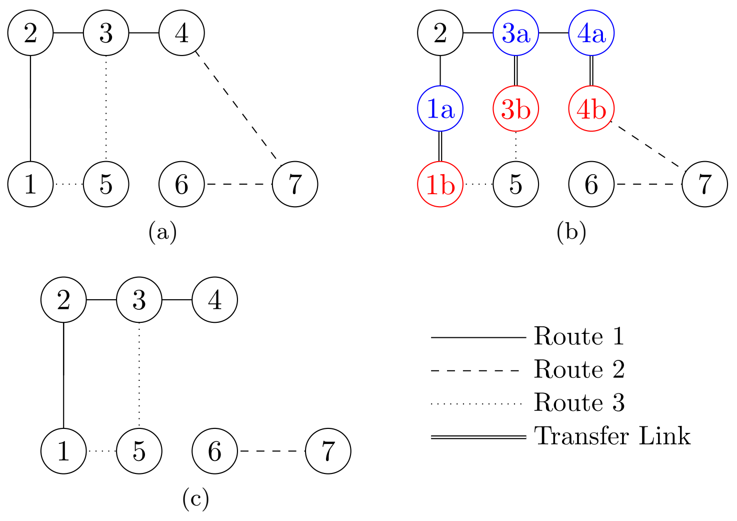


Figure 1: (a): Feasible route network containing three routes. (b): Transit network showing the duplication of nodes and transfer edges connecting the duplicates. (c): Infeasible disconnected network

point between two routes is duplicated, and the two duplicates are connected through a transfer edge (Figure 1). The size of the route network can grow as much as an order of magnitude during evaluation, depending on the number of routes, their lengths and connectivity. Mathematically, let E_a be the set of edges of route r_a. $G' = \{V', E'\}$ is the transit graph in which $V' \subseteq R \times V$. We define the node $y_{r_a i}$ as a pair consisting of a route r_a, and a vertex, $v_i \in V$. Let $(y_{r_a i}, y_{r_b j}) \in E'$ be the edge from node $y_{r_a i}$ to node $y_{r_b j}$ in the transit graph. Consequently, we define two types of edges: (i) *transport edges* (E'_1) correspond to in-vehicle travel links between two nodes within the same route, and (ii) *transfer edges* (E'_2) correspond to transfers from one route to another.

$$E'_1 = \bigcup_{r_a \in R} \{(y_{r_a i}, y_{r_a j}) : (v_i, v_j) \in E_a\} \tag{1}$$

$$E'_2 = \bigcup_{v_i \in V} \{(y_{r_a i}, y_{r_b i}) : v_i \in r_a \cap r_b\} \tag{2}$$

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

To evaluate the passenger cost for a route network $R = \{r_1, r_2, \ldots, r_{|R|}\}$, the relevant transit network is built during the evaluation to incorporate both in-vehicle travel time, and transfer penalty. The passenger cost represents the average travel time of a single passenger when travelling in the network between source and destination. The minimum journey to travel between the two vertices v_i, v_j in the route network (R) is given by the shortest path $\alpha_{ij}(R)$ from node $\{y_{r_a i} : v_i \in r_a\}$ to node $\{y_{r_b j} : v_j \in r_b\}$ in the transit network, including both transport edges (Equation 1) and transfer edges (Equation 2). The passenger cost is the total travel time made by all passengers who travel from their source to destination using the shortest path, over the entire demand served by the network (Mumford, 2013):

$$C_p(R) = \frac{\sum_{i,j=1} d_{ij}\alpha_{ij}(R)}{\sum_{i,j=1} d_{ij}} \tag{3}$$

The transport network operator aims to reduce the total cost of the system, while satisfying at least the minimum level of service quality. The operator costs include the fleet size required to satisfy the demand, the total distance travelled by the vehicles, the costs of maintenance, and the drivers employment costs. To simplify the operator costs, we use the sum of the cost (in time) for traversing all the routes in one direction (Mumford, 2013):

$$C_o(R) = \sum_{r_a \in R} \sum_{(v_i,v_j) \in E_a} t_{ij} \tag{4}$$

The fleet size is also an important consideration for the operator. We will not include it as an objective for the meantime but will describe a simple way to calculate the total fleet size required to cover the entire network demand. This could be simply done by dividing the total length of the routes (multiplied by 2 to represent travelling in the opposite direction) by the route headway (equals 10 considering an average waiting time of 5). Following these calculations, if we considered that the total length of the routes is equal to the lower bounds of the operator cost (Mumford, 2013), the smallest network in the data set will require ($\frac{2\times 63}{10} = 13$) buses and the largest network will require 186 buses.

We also use other performance indicators commonly quoted in the literature (Chakroborty and Wivedi, 2002; Fan and Mumford, 2010; Mumford, 2013; Kılıç and Gök, 2014) to evaluate the route sets more comprehensively.

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

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

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

We have previously mentioned the complexity of the UTRP, and the difficulty of incorporating real world assumptions and constraints. For this reason, we have applied a set of constraints to simplify our model, in order to allow us to focus on a general methodology for network design that can be compared to previous work. Our assumptions are listed below (Mumford, 2013):

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

For the scope of this work, we will also assume that there are sufficient vehicles traversing each route, and with enough capacity to cover the demand between all the bus stops. The frequency of the routes though can be calculated using a demand assignment procedure (Arbex and da Cunha, 2015) to determine high demand routes and transfer points and a simple heuristic procedure (Szeto and Wu, 2011) to redistribute the total available fleet (calculated as mentioned in this section) between the routes, allowing larger number of buses to be assigned to busier routes.

4. Hyper-heuristics for Urban Transit Route Design Problem

The main aim of this study is to implement a selection hyper-heuristic framework with different combinations of selection and move acceptance methods and apply it to benchmark instances with the goal of finding the best possible routes from passenger and operator perspectives in the shortest possible time. In this section we present an overview of the methodology applied to achieve this.

4.1. Evaluation Method

A solution S is in the form of a two-dimensional vector, and a candidate solution is evaluated using the following equation:

$$f(S) = \alpha F + \beta C_p + \gamma C_o + \tag{5}$$

where α, β and γ are constants used to weight the three components of the objective function; C_p and C_o are calculated using Equations 3 and 4 respectively; and F represents the feasibility of the solution and it computes to what extent the solution meets the five problem constraints presented in Section 3. If any of these constraints is violated at any position in the route set, it is penalised by increasing its value by one. For example for each missing node in the route set, the constraint concerned with including all the vertices is increased by one. Similarly, for the other four constraints with each individual violation. F is then calculated as the sum of these constraint violations. A route set is only accepted if this sum is zero. Note that F, C_p and C_o can be treated as separate objectives in a multi-objective formulation or combined into a single objective as in Equation 5. In the present paper the objectives are considered separately in the main, simply to facilitate comparisons with published results produced by other state-of-the-art methods. However it is clear that compromise solutions between objectives will be required in practice, and for this reason we include a brief illustration of how our hyper-heuristic approach can be extended to produce such a set of solutions in Section 5.5

4.2. Initial Solutions

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

4.3. Hyper-heuristics

The constructed initial solution is introduced as the current solution (S) to the hyper-heuristic framework. The selection method applies a single heuristic, or a sequence of heuristics (in the case of SSHH) to S, generating a new solution S', which is evaluated using Equation 5 with the following parameters settings: α is set to ∞ to ensure non-feasible solutions are always rejected, β and γ are set based on whether the solution is being evaluated from the passenger or the operator perspective. $\beta = 1$ and $\gamma = 10^{-10}$ for the passenger perspective, and vice versa for the operator. The low level of 10^{-10} provides an effective tie-breaker. Following evaluation the acceptance method is applied, which determines whether S' will replace S or not. In this work we are evaluating the performance of several selection hyper-heuristics made up by alternative pairings between the various selection and the various acceptance methods. More specifically we pair the following selection methods: Simple Random (SR), Random Descent (RD), Random Permutation (RP), Random Permutation Descent (RPD), Greedy Selection (GR), Sequence-based Selection (SS); with the move acceptance methods: Only Improve (OI), Improve or Equal (IE), Late Acceptance (LA), Great Deluge (GD), Simulated Annealing (SA). In LA after performing a series of experiments with different memory sizes for the circular queue L, we have set this value to 40.

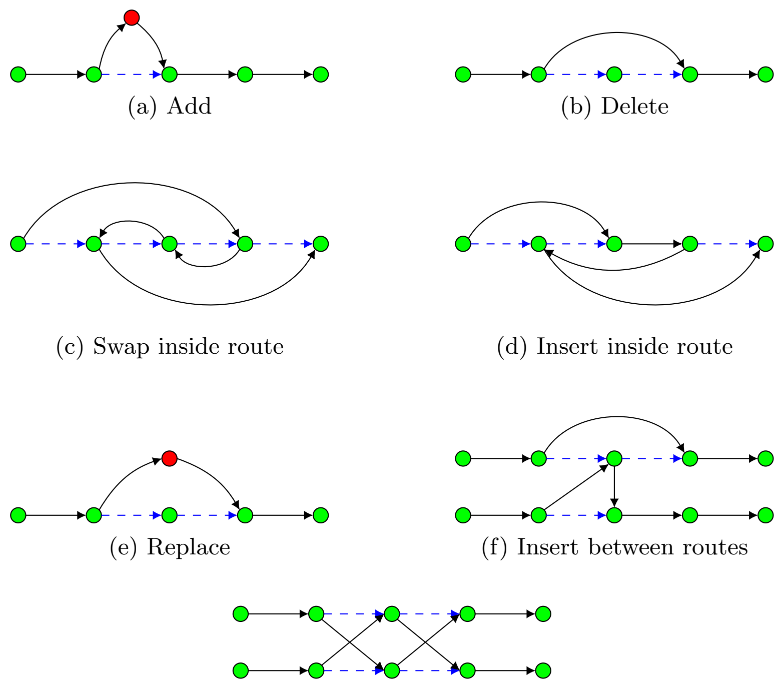
4.4. Low Level Heuristics

The hyper-heuristic controls a set of seven low level heuristics to improve the quality of a given route set (see Figure 2).

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

4.5. Problem Instances

In this study, we have used Mandl's benchmark with route set sizes 4, 6, 7, 8 considering each a separate problem. These variants are commonly used in the literature (Mumford, 2013; Chew et al., 2013). We have also used the four benchmark instances published in (Mumford, 2013) which are large size instances based loosely on the number of routes and the connectivity of



(a) Add

(b) Delete

(c) Swap inside route

(d) Insert inside route

(e) Replace

(f) Insert between routes

(g) Swap between routes

Figure 2: Low level heuristics set description. Straight arcs are edges in the route, dashed arcs are edges removed after applying the low level heuristic, curved arcs are edges added after applying the low level heuristic

bus network maps of real cities: one in China (Yubei), and two in the UK (Cardiff, and Brighton). These instances provide a variety of network sizes which is necessary to assess the scalability of our methods. The features of our datasets are provided in Table 2.

5. Empirical Results

5.1. Experimental Setup

We carried out our experiments in two phases: The first round of experiments evaluates the solution from the passenger perspective. In these experiments Equation 5 is used to evaluate the solution with the following parameters setting: $\alpha = \infty$, $\beta = 1$ and $\gamma = 10^{-10}$. In the second round of experiments the solution is evaluated from the operator perspective using the settings: $\alpha = \infty$, $\beta = 0$ and $\gamma = 1$ for 80% of the run time, and for the rest of the search time β is set to 10^{-10}. The reason for this is to focus on

Table 2: Features of our dataset

Instance	Number of vertices, edges	Number of routes	Number of vertices per route (min - max)	Average transit network size
Mandl4	15, 21	4	2 - 8	29
Mandl6	15, 21	6	2 - 8	29
Mandl7	15, 21	7	2 - 8	29
Mandl8	15, 21	8	2 - 8	29
Mumford0	30, 90	12	2 - 15	86
Mumford1	70, 210	15	10 - 30	231
Mumford2	110, 385	56	10 - 22	715
Mumford3	127, 425	60	12 - 25	930

improving the routes based on the operator qualities instead of wasting the search time with the complex calculations of the passenger objective.

The experiments were conducted on a device with the following specifications: Intel Core i5 at 2.30GHz with memory of 8GB. Each selection hyper-heuristic is run for ten trials on each instance and terminates after the run time elapses. The run time is set to vary according to the instance size by adding thirty seconds for each node.

5.2. Passenger Perspective

Tables 3 and 4 show the results from the passenger prescriptive experiments in terms of the average travel time for a single passenger (measured in minutes) for all the selection hyper-heuristics averaged over the ten trials. The minimum and the maximum values have also been recorded. The Kruskal-Wallis test is performed with 95% confidence level to compare the pairwise statistical variations in the performance between two algorithms. The following notations are used: Given two algorithms X versus Y, $>$ ($<$) denotes that $X(Y)$ performs better than $Y(X)$, and this variation is statistically significant, $\geq$ ($\leq$) denotes that $X(Y)$ performs slightly better than $Y(X)$, but the performance is not statistically significant, and $=$ denotes that X and Y perform equally. The values associated with these notations in the tables represent the number of times a particular hyper-heuristic is statistically significant, not statistically significant, or equally performing against the other selection hyper-heuristics in the tested set. The average number of iterations is also reported using the following notations: m refers to the number of iterations in millions, and k refers to the number in thousands.

It can be observed from the results the success of sequence-based selection method (SS), outperforming other selection methods regardless of the

Table 3: Results of the thirty selection hyper-heuristics from the passenger perspective for Mandl instances. Best values per each instance are highlighted in bold

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

Table 4: Results of the thirty selection hyper-heuristics from the passenger perspective for Mumford instances. Best values per each instance are highlighted in bold

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

move acceptance. This observation applies for all instances. Figure 3 shows the performance variation of our applied selection methods giving advantage for SS. The non-deterministic acceptance methods were more successful. Simulated annealing achieved the best results in all Mandl variants, delivering an improved performance that is statistically significant. Great deluge found the best minimum results in all Mandl instances. In the larger instances late acceptance and great deluge were the most successful. However LA found slightly better averages and minimum results compared to GD. Given these observations we carried out the next round of experiments from the operator perspective using three selection hyper-heuristics (SS-SA, SS-GD, SS-LA).

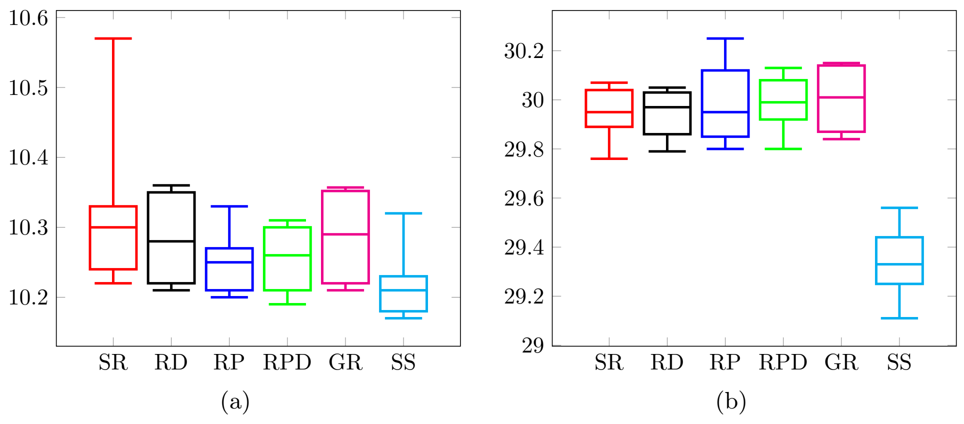


Figure 3: Box plots from 10 runs for all selection methods combined with GD acceptance method for (a) Mandl6 instance, and (b) Mumford3 instance. Values in Y axis show the average travel time and the lower boxes represent the best selection methods

5.3. Operator Perspective

Table 5 summarises the results of the experiments from the operator perspective using the average objective function value over the ten trials, the minimum values, and the pair-wise statistical performance. Comparing these results with the lower bounds for the operator cost published in (Mumford, 2013), all three selection hyper-heuristics succeeded in finding the lower bound in the Mandl problem variants and the Mumford0 instance on each of the ten trials. Referring to the table, GD is the most successful, scoring better averages and minimum values with a statistically significant performance compared to SA and LA. On the other hand, the best approach from the passengers' perspective is not so obvious. For this reason another round of experiments was conducted using longer run times.

Table 5: Results of the three best selection hyper-heuristics from the operator perspective. Best averages and minimum values per instance are highlighted in bold

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

5.4. Longer Runs

In this series of experiments we have given each of the three hyper-heuristics longer running times to observe whether improved performance can be obtained if there is more time to modify routes. We increased the running time by a factor of ten (i.e. the largest instance in the set will run for ten hours and ten times the number of iterations), and performed a two runs on each instance: one from the passenger perspective, and one from the operator perspective. According to Table 6 the results of these runs revealed the success of GD from the operator perspective similar to the short run time experiments, and from the passenger perspective GD performed the best in Mandl problems, Mumford0 and Mumford1 instances.

In the short run time experiments LA was successful in the larger instances and was slightly better than GD. This success also continues in the longer time experiments with insignificant performance difference in comparison to GD.

5.5. Obtaining Multiple Solutions

The previous experiments focused on finding the best possible route sets from passenger or operator perspectives separately. For practical use on real world public transit systems however, a compromise between the needs of the conflicting stakeholders will be required. To demonstrate how this can be achieved, another round of experiments has been carried out using SS-GD (the best performing algorithm). To ensure balance and fairness between the two objectives, their values have been normalised to 1 using the following parameters setting in Equation 5: $\alpha = \infty$, $\beta = \frac{1}{C_{Pinit}}$, $\gamma = \frac{1}{C_{oinit}}$ where C_{Pinit}, C_{oinit} equals the passenger and operator costs of the initial solution respectively. Several weight settings were then chosen to give a spread of compromise solutions.

We have tested this approach on Mandl instance (with six routes) by running several experiments each for a duration equal to the short run time

Table 6: Results of the long runs experiments from passenger and operator perspectives. Best values are highlighted in bold

Instance	C_p	C_o	C_p	C_0	C_p	C_o
	SS-SA		SS-GD		SS-LA	
			Passenger Perspective			
Mandl4	**10.482**	148	**10.482**	148	10.576	140
Mandl6	10.187	216	**10.179**	212	10.321	186
Mandl7	10.119	214	**10.103**	250	10.150	232
Mandl8	10.086	251	**10.080**	272	10.095	260
Mumford0	14.157	725	**14.093**	722	14.218	734
Mumford1	21.961	2073	**21.699**	1956	22.096	2010
Mumford2	25.554	5276	25.196	5257	**25.001**	5480
Mumford3	28.261	5807	28.056	6119	**27.894**	6217
			Operator Perspective			
Mandl4	13.8754	**63**	14.6718	**63**	13.8754	**63**
Mandl6	13.4804	**63**	14.2832	**63**	14.3571	**63**
Mandl7	13.6763	**63**	14.4438	**63**	14.8645	**63**
Mandl8	14.2158	**63**	14.7938	**63**	15.0572	**63**
Mumford0	24.814	**94**	26.320	**94**	28.475	**94**
Mumford1	42.922	414	39.452	**408**	35.269	437
Mumford2	42.356	1436	46.865	**1330**	41.188	1508
Mumford3	44.771	1877	46.054	**1746**	42.569	1758

with different weights combination per run and the results are plotted in Figure 4 along with the best results for the passenger and operator acquired previously. Clearly computing more compromise solutions will increase the cumulative run time of the optimisation. On the other hand, we do not require the vast populations generally needed to maintain diversity for evolutionary algorithms.

5.6. Analysis of SS-GD

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

Figure 5 shows the average utilisation rate for each low level heuristic for Mandl6 and Mumford3 instances after running each for a single run under SS-GD from both passenger and operator perspectives, considering only the applications of the low level heuristics that end the active sequence and improve over the best solution (i.e. Seq = end).

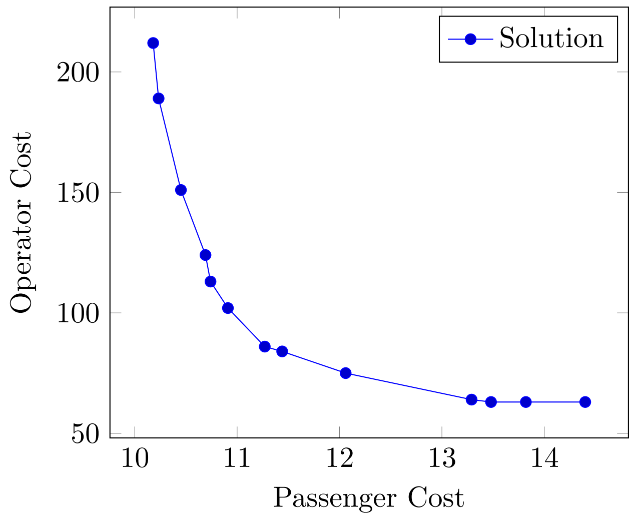
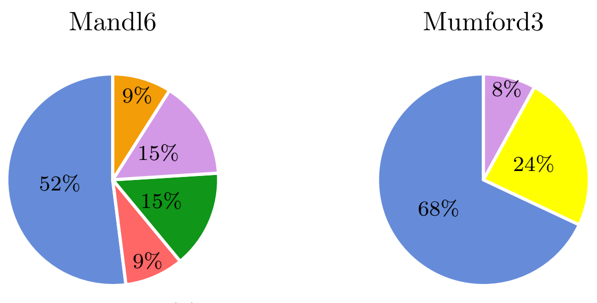


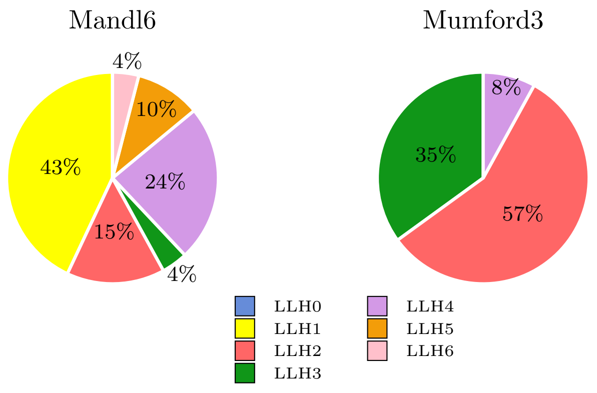
Figure 4: A plot showing a number of solutions between the best passenger and operator results in Mandl6 instance. Each point represents a different solution with different weight values.

From the passenger perspective, the add low level heuristic (LLH0) was the most successful in both instances, achieving the most contribution in the best solutions. Other low level heuristics that were also successful are insert (LLH3), replace (LLH4) in Mandl6, and delete (LLH1) in Mumford3. In contrast, the add low level heuristic was the least successful from the operator perspective, and most of the contribution was achieved by the delete (LLH1), swap (LLH2), and insert (LLH3) low level heuristics.

Figure 6 shows the transition, and the sequence construction frequency matrices, again for Mandl6 and Mumford3. A few interesting observations can be made. From the passenger perspective, we note from the sequence construction matrix that on the whole low level heuristics have a higher probability to end the sequence than continue it, which indicates that low level heuristics that contribute strongly to producing the best solutions, perform this success individually in sequences of length one. Specifically the add low level heuristic (LLH0) in Mumford3 which is the most successful in the set by referring to Figure 5, works almost independently. Yet from the transition matrix, some good sequences with a longer length than one can be identified, such as the combination of insert to different route (LLH5), and insert on the same route (LLH3) low level heuristics with the add low level heuristic (LLH0) in Mandl6. From the operator perspective, similarly the most successful low level heuristics (i.e. LLH1, LLH2, LLH3) operate best



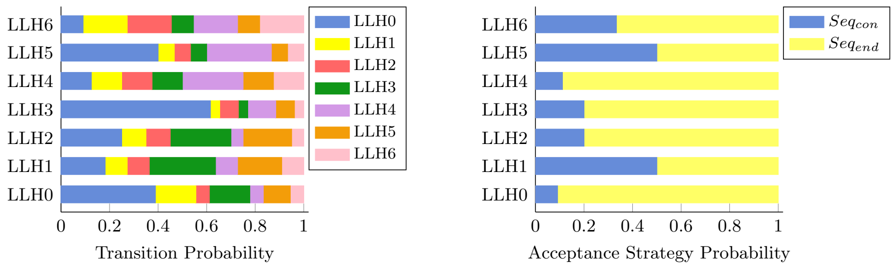
(a) Passenger perspective



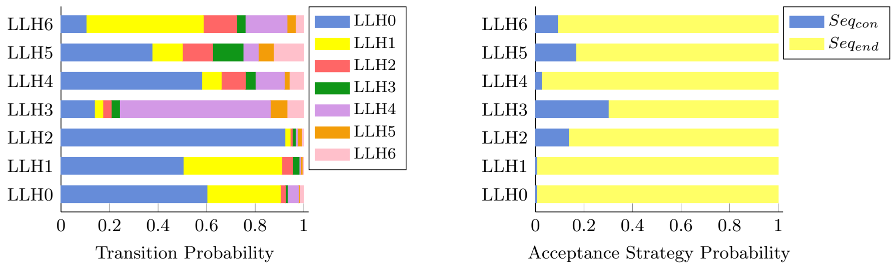
(b) Operator perspective

Figure 5: Average utilisation rate for each low level heuristic considering the invocations that generated improvements on the best solution in Mandl6 and Mumford3 instances

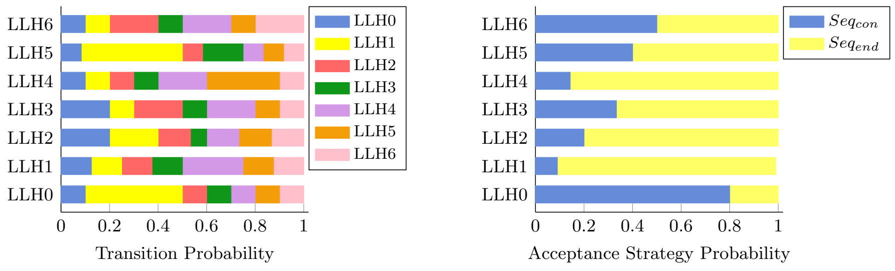
individually in both instances. In Mandl6, and Mumford3 the add low level heuristic tends to work better in sequences longer than one from the sequence construction matrix, unlike its behaviour in the passenger perspective. One of these sequences is the combination of add and delete low level heuristics in Mandl6. These observations show the intelligence of the *SS* method in identifying good sequences and understanding the relationships between low level heuristics in two different instances. The *SS* method has the advantage of intelligently revealing how the low level heuristics operate. Some low level heuristics may have a high utilisation rate, yet they achieves this with the support of other low level heuristics that might seem to have low contribution to the best solutions, but are necessary part in making the success of the high utilisation low level heuristics.



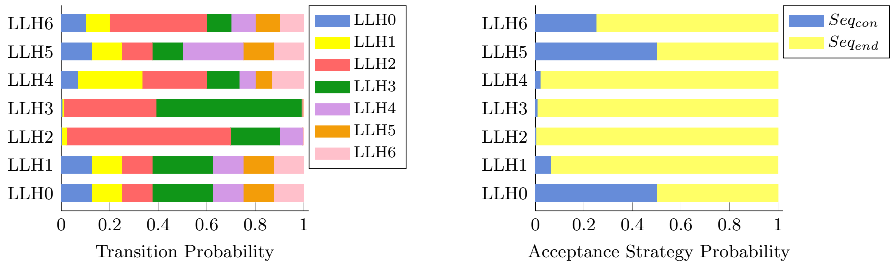
(a) Mandl6 Passenger



(b) Mumford3 Passenger



(c) Mandl6 Operator



(d) Mumford3 Operator

Figure 6: Transition and sequence construction frequency matrices for Mandl6 and Mumford3

Table 7: Passenger perspective results compared to other approaches

Instance	Parameter	Mandl (1979)	Chakroborty and Wivedi (2002)	Fan and Mumford (2010)	Mumford (2013)	Chew et al. (2013)	John et al. (2014)	Kılıç and Gök (2014)	SS-GD
Mandl4	Passenger	12.90	11.90	11.37	10.57	10.50	-	10.56	**10.48**
	Operator	-	-	147	149	150	-	137	148
	d_0	69.49	86.86	**93.26**	90.43	91.84	-	91.33	91.84
	d_1	29.93	12.00	**6.74**	9.57	8.61	-	8.16	8.15
	d_2	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	-	**0.00**	**0.00**
	d_{un}	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	-	**0.00**	**0.00**
Mandl6	Passenger	-	10.30	10.48	10.27	10.21	10.25	10.29	**10.18**
	Operator	-	-	215	221	224	212	216	212
	d_0	-	86.04	91.52	95.38	96.79	-	95.5	**97.17**
	d_1	-	13.96	8.48	4.56	3.21	-	4.5	**2.82**
	d_2	-	**0.00**	**0.00**	0.06	**0.00**	-	**0.00**	**0.00**
	d_{un}	-	**0.00**	**0.00**	**0.00**	**0.00**	-	**0.00**	**0.00**
Mandl7	Passenger	-	10.15	10.42	10.22	10.16	-	10.23	**10.10**
	Operator	-	-	231	264	239	-	274	250
	d_0	-	89.15	93.32	96.47	98.01	-	97.04	**98.84**
	d_1	-	10.85	6.36	3.34	1.99	-	2.83	**1.15**
	d_2	-	**0.00**	0.32	0.19	**0.00**	-	0.13	**0.00**
	d_{un}	-	**0.00**	**0.00**	**0.00**	**0.00**	-	**0.00**	**0.00**
Mandl8	Passenger	-	10.46	10.36	10.17	10.11	-	10.20	**10.08**
	Operator	-	-	283	291	256	-	298	272
	d_0	-	90.38	94.54	97.56	99.04	-	97.37	**99.16**
	d_1	-	9.62	5.46	2.31	0.96	-	2.63	**0.83**
	d_2	-	**0.00**	**0.00**	0.13	**0.00**	-	**0.00**	**0.00**
	d_{un}	-	**0.00**	**0.00**	**0.00**	**0.00**	-	**0.00**	**0.00**
Mumford0	Passenger	-	-	-	16.05	-	15.40	14.99	**14.09**
	Operator	-	-	-	759	-	745	707	722
	d_0	-	-	-	63.20	-	-	69.73	**88.74**
	d_1	-	-	-	35.82	-	-	30.03	**11.25**
	d_2	-	-	-	0.98	-	-	0.24	**0.00**
	d_{un}	-	-	-	**0.00**	-	-	**0.00**	**0.00**
Mumford1	Passenger	-	-	-	24.79	-	23.91	23.25	**21.69**
	Operator	-	-	-	2038	-	1861	1956	1956
	d_0	-	-	-	36.60	-	-	45.10	**65.75**
	d_1	-	-	-	52.42	-	-	49.08	**34.18**
	d_2	-	-	-	10.71	-	-	5.76	**0.07**
	d_{un}	-	-	-	0.26	-	-	0.06	**0.00**
Mumford2	Passenger	-	-	-	28.65	-	27.02	26.82	**25.19**
	Operator	-	-	-	5632	-	5461	5027	5257
	d_0	-	-	-	30.92	-	-	33.88	**56.68**
	d_1	-	-	-	51.29	-	-	57.18	**43.26**
	d_2	-	-	-	16.36	-	-	8.77	**0.05**
	d_{un}	-	-	-	1.44	-	-	0.17	**0.00**
Mumford3	Passenger	-	-	-	31.44	-	29.50	30.41	**28.05**
	Operator	-	-	-	6665	-	6320	5834	6119
	d_0	-	-	-	27.46	-	-	27.56	**50.41**
	d_1	-	-	-	50.97	-	-	53.25	**48.81**
	d_2	-	-	-	18.79	-	-	17.51	**0.77**
	d_{un}	-	-	-	2.81	-	-	1.68	**0.00**

5.7. Comparison with Other Approaches

We compared our results from the passenger and operator perspectives with the state-of-the-art methods. We used the long run results of SS-GD representing our best results. Referring to Tables 7 and 8 our method found the best average travel times in all Mandl instances as well as the best d_0, d_1, d_2 in all cases except in Mandl4 instance. In Mumford's instances, our approach outperformed the methods in (Mumford, 2013; John et al., 2014; Kılıç and Gök, 2014) in terms of the average travel time, and d_0, d_1, d_2 values scoring zero percentage for unsatisfied demand in all cases. From the

Table 8: Operator perspective results compared to other approaches

Instance	Parameter	Mumford (2013)	Chew et al. (2013)	John et al. (2014)	SS-GD
Mandl6	Operator	**63**	**63**	**63**	**63**
	Passenger	15.13	13.88	13.48	14.28
	d_0	70.91	70.91	-	62.23
	d_1	25.5	25.50	-	27.16
	d_2	2.95	2.95	-	9.57
	d_{un}	0.64	0.64	-	1.028
Mumford0	Operator	111	-	95	**94**
	Passenger	32.40	-	32.78	26.32
	d_0	18.42	-	-	14.61
	d_1	23.40	-	-	31.59
	d_2	20.78	-	-	36.41
	d_{un}	37.40	-	-	17.37
Mumford1	Operator	568	-	462	**408**
	Passenger	34.69	-	39.98	39.45
	d_0	16.53	-	-	18.02
	d_1	29.06	-	-	29.88
	d_2	29.93	-	-	31.90
	d_{un}	24.66	-	-	20.19
Mumford2	Operator	2244	-	1875	**1330**
	Passenger	36.54	-	32.33	46.86
	d_0	13.76	-	-	13.63
	d_1	27.69	-	-	23.58
	d_2	29.53	-	-	23.94
	d_{un}	29.02	-	-	38.82
Mumford3	Operator	2830	-	2301	**1746**
	Passenger	36.92	-	36.12	46.05
	d_0	16.71	-	-	16.28
	d_1	33.69	-	-	24.87
	d_2	33.69	-	-	26.34
	d_{un}	20.42	-	-	32.44

operator perspective, we succeeded in finding the lower bound in Mandl's four problems and in Mumford0 instance. Our approach also found the best results in Mumford1, Mumford2 and Mumford3 instances. This comparison proves the success of hyper-heuristics on this problem, outperforming the previously reported results using GA approaches. Cooper et al. (2014) reported that the implementation of John et al. (2014) required 44 hours to run Mumford3 instance, and to improve this, a parallel implementation of the algorithm is required. Kılıç and Gök (2014) required more than eight hours to initialise route sets and run a simple hill climbing algorithm in Mumford3 instance. Hyper-heuristic was able to find new best solutions after running Mumford3 for a single hour.

6. Conclusion

The goal of hyper-heuristics is to raise the level of generality by using methods that are easy-to-implement, cheap-to-maintain, yet deliver excellent performance on different problem domains. We have applied selection hyper-heuristics to the complex problem of urban transit network design. Thirty selection hyper-heuristics combining several known selection and move acceptance methods were tested and applied on a set of benchmark instances and their performances were compared to determine the best algorithm. Our analysis showed the success of the sequence-based selection method combined with great deluge acceptance method, outperforming other selection hyper-heuristics in both passenger and operator objectives. The hyper-heuristic approach which has been applied for this particular problem for the first time was very successful, beating the current known state-of-the art results in a very reasonable run times. Our future plan is to continue working on this research by expanding the ideas mentioned on the present paper. We will further explore the hyper-heuristic multi-objective model introduced here, with the goal of obtaining useful compromise solutions cheaply in terms of run times. Furthermore, we will focus our future efforts on data extracted from publically available sources for real road networks, bus routes and passenger demand, rather than simply use artificially created instances. Later on, we propose to incorporate frequency setting and fleet size limitation as well as other operational constraints.

References

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.

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.

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.

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.

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.

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.

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.

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.

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

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.

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.

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.

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

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.

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

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.

Guihaire, V., Hao, J.-K., 2008. Transit network design and scheduling: A global review. Transportation Research Part A: Policy and Practice 42 (10), 1251–1273.

Ibarra-Rojas, O., Delgado, F., Giesen, R., Muñoz, J., 2015. Planning, operation, and control of bus transport systems: A literature review. Transportation Research Part B: Methodological 77, 38–75.

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.

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.

Kepaptsoglou, K., Karlaftis, M., 2009. Transit route network design problem: review. Journal of Transportation Engineering 135 (8), 491–505.

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.

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

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 & Operations Research 36 (8), 2440–2449.

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.

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.

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.

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.

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

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

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

Schéele, S., 1980. A supply model for public transit services. Transportation Research Part B: Methodological 14 (1-2), 133–146.

Schöbel, A., 2012. Line planning in public transportation: models and methods. OR Spectrum 34 (3), 491–510.

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.

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

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

Vaughan, R., 1986. Optimum polar networks for an urban bus system with a many-to-many travel demand. Transportation Research Part B: Methodological 20 (3), 215–224.

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.

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.

Zhao, F., Zeng, X., 2006. Simulated annealing–genetic algorithm for transit network optimization. Journal of Computing in Civil Engineering 20 (1), 57–68.