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

Leena Ahmed
Cardiff University
ahmedlh@cardiff.ac.uk

Philipp Heyken-Soares
Nottingham University
philipp.heyken@nottingham.ac.uk

Christine Mumford
Cardiff University
mumfordcl@cardiff.ac.uk

Yong Mao
Nottingham University
yong.mao@nottingham.ac.uk

ABSTRACT

The urban transit routing problem (UTRP) is concerned with finding efficient travelling routes for public transportation systems. This problem is highly complex, and the development of effective algorithms to solve it is very challenging. Furthermore, realistic benchmark data sets are lacking, making it difficult for researchers to compare their problem-solving techniques with those of other researchers. In this paper we contribute a new set of benchmark instances that have been generated by a procedure that scales down a real world transportation network, yet preserves the vital characteristics of the network layout including "terminal nodes" from which buses are restricted to start and end their journeys. In addition, we present a hyper-heuristic solution approach, specially tailored to solving instances with defined terminal nodes. We use our hyper-heuristic technique to optimise the generalised costs for passengers and operators, and compare the results with those produced by an NSGAII implementation on the same data set. We provide a set of competitive results that improve on the current bus routes used by bus operators in Nottingham.

CCS CONCEPTS

• **Applied computing** → **Operations research**; **Transportation**;

KEYWORDS

Public transport, routing, benchmarks, hyper-heuristics

ACM Reference Format:
Leena Ahmed, Philipp Heyken-Soares, Christine Mumford, and Yong Mao. 2019. Optimising Bus Routes with Fixed Terminal Nodes: Comparing Hyper-heuristics with NSGAII on Realistic Transportation Networks. In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3321707.3321867

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6111-8/19/07…$15.00
https://doi.org/10.1145/3321707.3321867

1 INTRODUCTION

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. Public transportation systems design is a topic that has been addressed extensively in the literature [9, 10, 12] using a variety of models and solution methodologies. Yet the models in the literature hugely differ, and there is a lack of public benchmarks that can help researchers to effectively compare their algorithms. Moreover, current models and benchmark instances fail to properly reflect real world road network layouts or realistic operating constraints. For example, Mandl's Swiss network [16, 17] is considered the defacto benchmark until very recent time, though it only contains 15 nodes that does not represent a real network size. Another set of instances published in [18] provides larger sizes that have been generated based on user defined parameters which determine the number of vertices, edges and the upper and lower bounds of demand at each node in the network.

One of the key elements in public transportation systems planning 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.

In this work we introduce a constraint into the network model, that restricts the start and end points of bus journeys to specific points named terminals. Identifying end points for bus journeys is essential when solving the routing design problem in an urban context, to provide u-turn possibilities for buses. However, adding this condition creates extra complexity by making it more difficult to construct feasible solutions. Figure 1 illustrates a "legal" connected network according to the feasible network definitions in [2, 18]. The same network becomes infeasible when three terminal points are introduced (green) making one of the routes invalid with an incorrect end terminal (node 4). This effect becomes more profound with the increase in network size, or the decrease in the number of valid terminals.

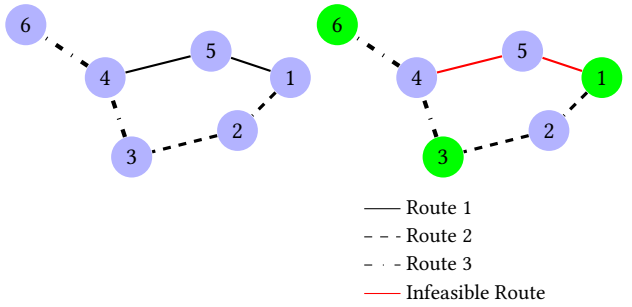


Figure 1: Feasible route network becomes infeasible by introducing three terminal points (green)

Very few models in the literature incorporate terminals, especially for large instances. However, Pattnaik et al. [19] solved the network design problem for a small network representing parts of Madras city in India, using genetic algorithms (GA) in two phases: first a heuristic procedure is applied to generate a set of candidate routes and then the GA is applied in the second phase. Their candidate route set generation procedure is based on the demand matrix, route set constraints and designer's knowledge. The procedure involves finding the shortest path between every origin and destination pair which are selected from a set of terminal points. The designer identifies the terminal points by taking the network layout into consideration. Szeto and Wu [20] solved the bus network design of the suburban area of Tin shui Wai in Hong Kong using a network model of 28 nodes, where trips originate from specific terminal points and end at one of five destination nodes. Seven terminal points are specified in their network model and a GA incorporating a frequency setting heuristic is used to solve the route design problem and determine bus frequencies. Amiripour et al. [3] tackled the bus network design problem by considering seasonal variation in the demand to provide a convenient bus service throughout the year. A GA has been applied to solve their model by testing it on two small benchmark instances and a real case study in the city of Mashhad in Iran. In the larger network of Mashahd, several terminal points have been identified by testing their turning possibilities and performing K-shortest path between pairs of terminal points to create feasible routes.

We propose in this work a new set of instances incorporating terminal point information. These instances have been generated following the procedure proposed in [11] which scales down a real world street network into a size manageable by optimisation algorithms while preserving the characteristics of the street network layout. The procedure has been applied to the urban area of Nottingham city and all the data associated with the instances including terminal points positions, travel and demand data are extracted entirely from public and open sources. This data set can be downloaded from [1], thus improving the availability of benchmarks that sufficiently reflect real world conditions. A hyper-heuristic approach which has been specifically tailored to solve this version of the problem with terminal nodes is used to optimise the route network and the results are further compared to those generated by the NSGAII genetic algorithm and also to real-world route sets extracted from the study area.

Hyper-heuristics are motivated by the idea of automating the design of heuristic methods to solve difficult computational problems [4]. There is currently a growing interest in using such cross-domain methodologies which represent a general framework applicable to several problem domains while requiring minimal adaption.

In a recent study, Ahmed et al. [2] have applied hyper-heuristics to the UTRP for the first time. A sequence selection method was used based on the hidden Markov model, in an attempt to mitigate the problems encountered by other meta-heuristic approaches, particularly population based methods such as genetic algorithms which require a huge computational time to maintain and evaluate the individuals of the population and therefore fail to solve large size networks in a reasonable run time.

The work showed the success of hyper-heuristics in delivering excellent results compared to population based methods with faster run times when tested on known benchmarks. A comparison between several selection hyper-heuristics was carried out, and the best performing approach combined a sequence based selection method (SSHH) with the great deluge acceptance method (GD). Thus SSHH with GD is adopted for the present work to find good solutions to the new Nottingham data set, and prove that hyper-heuristics work equally well on larger scale, and more complicated versions of the UTRP.

The rest of the sections describe the problem formulation, the optimisation methodology, the data set description, and finally the results and conclusions.

2 PROBLEM FORMULATION

We will use a simplified formulation for the UTRP utilised by several previous studies [2, 5, 13, 15, 18], which is the graph representation for a given road network with identified stop locations. The road network comprises a set of stops connected by road segments, this can be mapped into an undirected graph $G = \{V, E\}$, where the graph vertices $V = \{v_1, v_2, \ldots, v_n\}$ are access points (i.e. bus stops), and the graph edges $E = \{e_1, e_2, \ldots, e_m\}$ are direct transport links. Some of the vertices are identified as terminal points $U = \{u_1, u_2, \ldots, u_k\}$, such that $U \subseteq V$. These terminal points allow u-turns to make the reverse trip in the opposite direction. A public transport route, according to this graph definition, is a path in the graph that connects a set of vertices, and starts and finishes at terminal points $r = \{u_i, v_{i_1}, \ldots, v_{i_q}, u_j\}$. The *route network*, which is the solution to this model, results from devising a set of routes $R = \{r_1, r_2, \ldots r_{|N|}\}$ to form the transportation network, where $|N|$ is the total number of routes in the network. To evaluate a given route network, the following information is required for every pair of vertices in the transport network $(v_i, v_j) \in V$: the time to travel between the two vertices, and the number of passengers travelling. Travel time and demand between each pair of vertices is given in the form of two dimensional matrices (i) travel time matrix (ii) demand matrix. t_{v_i,v_j}, a single entry in the travel time matrix refers to the time in minutes required to travel from v_i to v_j and d_{v_i,v_j} in the demand matrix refers to the number of passengers travelling between v_i and v_j. The travel time matrix records travel times between directly connected nodes, with travel times of zero between a node and itself, and ∞ between pairs of nodes that are not directly connected in the graph. On the other hand, the

demand matrix records travel demand between pairs of source-destination nodes for the travellers. The two matrices are assumed to be symmetrical for simplicity, meaning that the inbound and outbound journeys along the route will have the same travelling duration. We also assume that the demand level remains the same and does not change for the duration of the day. The terminal points are identified using a one dimensional vector $Ur_{n\times 1}$, where n is the number of vertices in the road network and each entry Ur_i has a value of one if v_i is a valid terminal, or zero otherwise.

The feasibility of the solution (i.e. route network) determines to what extent the solution obeys the problem constraints. A single violation in any of the constraints results in rejecting the solution. A solution is accepted if and only if the sum of constraint violations is zero. The full list of constraints is presented below:

- The route set R includes exactly $|N|$ routes and each route is uniquely identified.
- $\forall r_i \in R$, the length of r_i in terms of the number of nodes is between a defined minimum and maximum values.
- $\forall r_i \in R$, no cycles or backtracks should be present.
- The route set taken as a whole (R) is fully connected to allow a user to reach any point in the network from any other point.
- $\forall v_i \in V$, v_i should be present in at least one route in the route set.
- $\forall r_i \in R$, the starting and ending nodes must be terminal nodes.

3 OPTIMISATION PROCEDURE

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

3.1 Creating Initial Route Set Using a Heuristic Construction Procedure

The initial generation procedure produces an initial route set based on the following parameters: the demand matrix, the terminal points vector, the road network graph, the predetermined number of routes in the route set, and the minimum and maximum length of each route (in terms of the number of nodes). Using this information, an initial route set is generated guided by the demand matrix to ensure that as much of the demand as possible is routed along its shortest travel time path. This gives the optimisation algorithm a good start. The initialisation algorithm involves the following steps: (i) Produce an edge usage graph guided by demand and shortest travel time path information. (ii) Create a pool of candidate routes. (iii) Construct a route set from the candidate route pool. Assuming the passenger prefers to travel along his/her shortest travel time path, the shortest path between every pair of nodes in the road network is calculated. It is then an easy matter to create a "shortest-path-usage map" by adding up the total demand travelling along each edge in the network, assuming all travellers are able to traverse their shortest paths[1]. An example of such a map is displayed in figure(2a), using the Clifton instance (described in section 4). In the diagram the edge labels represent the total demand along each link. A similar approach for calculating the edges usages has been used in [15].

Next we perform a simple transformation on this map to convert the usages into distances so that the largest usage becomes the shortest distance and vice-versa. This is done by subtracting the usage on each edge from some arbitrary large number. We have chosen to use the total demand for the whole network for this purpose. Figure (2b) demonstrates the transformed usage map using the upper bound for Clifton (i.e equals 964). In this case the highest usage (i.e from node 3 to node 8) becomes the shortest distance (964 - 932 = 32). Our approach here differs from [15] where they calculate probabilities to select edges based on their usage value.

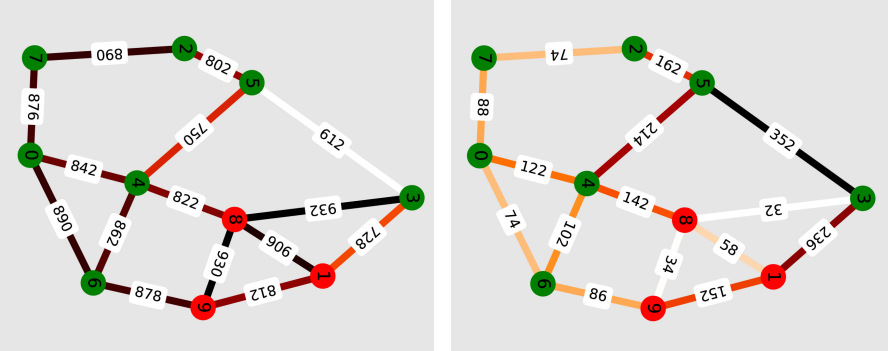
The transformed usage map is then used to generate routes for the routes pool, which will later be used as a palette from which to select routes for the initial route set. The algorithm will iterate through pairs of terminal nodes and create routes by performing shortest path computations based only on the transformed usage map. In this way the algorithm will generate routes that include the busiest edges, and each of these will enter the pool as a candidate route, provided its route length lies between the minimum and maximum allowed. However, to guarantee that the pool of routes covers all of the nodes in the network, it is necessary to include some less busy links. To achieve this, the shortest path algorithm iterates several times between all pairs of terminal nodes. After each iteration, the weights of the transformed usage map are updated by slightly increasing the ones that correspond to edges selected by the route generation procedure. This encourages the shortest path algorithm to look for alternative paths that may include undiscovered nodes. The weight values are increased by multiplying them with a very small value which have been tuned to 1.1 after a series of trials. The iterations terminate after the inclusion of all the network nodes in the candidate pool.

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

3.2 Objectives and Evaluation

The UTRP is a multi-decision problem incorporating several stakeholders with conflicting requirements. A designed transportation system should take into consideration the passengers' needs, the limited budget of the operating companies, and the rules imposed by the local authorities. In our model, the optimisation will focus on reducing the average travel time encountered by the single

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



(a) Usage map (b) Transformed usage map

Figure 2: The usage map and the transformed usage map: darker colour = high demand edges, lighter colour = lower demand edges. Green vertices = terminal vertices, red vertices = non-terminal vertices.

passenger, ensuring the passengers' convenience in reaching their destination as fast as possible with the least number of transfers, while also lowering the network operator's expenses. The following formulae are used to calculate the objectives and to evaluate a given solution:

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

$$C_o(R) = \sum_{\forall r_i \in R} \sum_{(v_i, v_j) \in r_i} t_{ij} \tag{2}$$

$$f(S) = \alpha \frac{C_p(R)}{C_{Pinitial}} + \gamma \frac{C_o(R)}{C_{Oinitial}} \tag{3}$$

Equation 1 calculates the passenger objective, where we consider reducing the average travel time of the single passenger in the network. Our assumption is that the passenger choice of route is always the path that requires minimum journey time (i.e the shortest path), thus α_{ij} in the equation denotes the shortest path between stops i and j and it incorporates the in-vehicle travel time, the waiting time, and the transfer time. The waiting time and the penalty for making a transfer are combined as a single time set to 5 minutes. The total travelling distance in the transportation network is an important consideration for the operator. Therefore we consider the cost for travelling all the routes in a single direction calculated by equation 2 as the operator objective. Equation 3 is used to evaluate a given solution, where $C_{Pinitial}$ and $C_{Oinitial}$ are the initial values for the passenger and operator objectives respectively, and α and γ are parameters to determine how to direct the optimisation by either focusing on optimising one of the objectives, or balancing them. To analyse candidate route sets more extensively, the following parameters are used to calculate the percentages of demand satisfied by direct (i.e. zero transfers) and indirect trips (i.e. one or two transfers): d_0, d_1, d_2, d_{un}. The demand that requires three transfers or more is considered unsatisfied (d_{un}). To calculate these parameters, it is assumed that the passenger prefers the route with the fewest transfers, if there is more than one shortest path between two points.

3.3 Optimising Route Sets Using Selection Hyper-heuristics

Hyper-heuristics, in contrast to other meta-heuristic techniques, control and perturb 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, giving them the advantage of easily being applied to any problem by only providing the relevant set of low level heuristics, the objective function and problem instances. The general framework of selection hyper-heuristics is that they iteratively improve a given initial solution through two processes known as selection and move acceptance methods, using the set of implemented low level heuristics until a termination condition is met. The best solution is constantly updated and returned at the end of the process.

Since the term hyper-heuristics was first introduced in [6] it has been widely used to solve difficult optimisation problems, and has been particularly popular in solving routing problems . In Ahmed et al. [2] the UTRP is solved using hyper-heuristics, by testing and comparing thirty selection hyper-heuristics. The best selection hyper-heuristic algorithm combined a sequence based selection method (SSHH) [14] with the great deluge acceptance method (GD) [8].

SSHH is an online selection method based on the hidden Markov model that constructs sequences of heuristics to apply at each decision point. A probabilistic scheme is utilised in this method to increase the chance of choosing successful sequences in later steps, and the selection method maintains and learns these sequences during the search. The great deluge acceptance method uses a threshold value to determine an acceptance range for the solutions. The threshold value equals the initial solution at the beginning of the search and decreases with time in a linear rate. At each step, the threshold value is recalculated using the following 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. Improved solutions are always accepted, while worsening solutions are accepted if their objective value is less than or equal to the calculated threshold

value at the current step. The details of the winning algorithm and the analysis are in [2]. This algorithm will be used to optimise the proposed data set and will be known by the name SS-GD in the rest of the paper.

At the start of the optimisation, the initial solution (S_{init}) built using the heuristic method described above, is introduced to the hyper-heuristics as the current solution (S_{curr}) and the sequence of heuristics constructed by SSHH is applied to S_{curr} to generate a new solution (S_{new}). The feasibility of S_{new} is tested, a single violation in any of the constraints listed in section 2 results in rejecting this solution (e.g. if at least one of the terminals of any route in the route set is not valid). In this case a new sequence of heuristics is constructed and applied to generate a new solution. If S_{new} is feasible, it is evaluated using equation 3 and the parameters are set to determine which objective the optimisation is focusing on. For example to optimise the route set by balancing the two objectives, the parameters α and γ are both set to 1, or one of them can be slightly increased to favour one of the objectives. To generate route sets optimised from one of the perspectives (i.e., passenger or operator), one of the parameters is set to 1 and the other to a very small value (e.g 10^{-4}).

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

3.4 The Low Level Operators

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

h1: **Add**. Selects a random route and a random position in the route. A node is selected and added in this position according to its adjacency relations with the neighbouring nodes.
h2: **Delete**. Selects a random route and a random position in the route. The node in this position is deleted while considering the adjacency of the neighbouring nodes of this position.
h3: **Replace**. Selects a random route and a random position. A node is selected to replace the node in this position while considering the adjacency of the neighbouring nodes with the selected node.
h4: **Swap**. Selects a random route and two random positions and swaps the nodes in these positions according to the adjacency relationships.
h5: **Shift**. Selects a random route and two random positions. The node in the first position is inserted into the second position according to the adjacency relationships.
h6: **Add terminal**. Selects a random route and a random terminal node and inserts it into one of the route terminals by randomly selecting one of them.
h7: **Reverse**. Selects a random route and two random positions and reverses the order of nodes between these positions.
h8: **Crossover**. Selects two random routes and a random position on each route and splits the route in this position. Two different routes are created by swapping the parts of the two routes.
h9: **Delete(Add) nodes**. Selects a random route and adds a number of nodes at the route terminal or deletes a number of nodes until the route reaches the maximum or minimum length.
h10: **Replace route**. Selects a random route and deletes it. A build procedure is then applied to construct a new route by finding the shortest path between two randomly selected terminal nodes. The deleted route is replaced by the new constructed route.

4 NOTTINGHAM INSTANCES

In this study we introduce a set of instances based on different parts of the urban area of Nottingham city in the UK (figure 3). The instances vary in size: the largest covering the entire study area and the smallest representing only the small Clifton area in Nottingham. All instances are generated from official street and census data of the year 2011. The procedure effectively reduces the street network to a graph size tractable by optimisation algorithms while maintaining the characteristics of the street network layout to ensure they are reflected sufficiently in the instances.

The first step in the generation procedure is to select the streets available for bus travel in the study area and construct a street map. This is done based on official street classifications[2] and the positions of existing bus stops. After that, the positions of the nodes are determined by placing initial nodes at all junctions and intersections of the street map. In cases where initial nodes are closer to each other than a defined distance, they are replaced by a new node half way between the positions of the original nodes. The resulting set of nodes do not represent concrete stop locations, but more precisely routing points which define the course of the bus route. It is assumed that vehicles travel on a path defined by these nodes, and stop at defined locations along the way.

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

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

[2]The selected streets classifications are: "A-", "B-" ,"Minor Road" and "Local Street" according to UK official road classifications. One-way streets are only included if travel in the other direction is possible on parallel streets within a short distance.

The travel times associated with the network edges are defined by calculating the shortest paths between pairs of nodes. For every nodes pair, the shortest path is found for each direction of travelling. If these paths are direct, they are averaged, and recorded as the travel time between this pair. If the paths pass by at least one other node, the connection is considered indirect and assigned to the travel time matrix as ∞. The final step is to determine the travel demand between pairs of nodes in the network. For this, travel to work data from 2011 UK census is used. It gives the number of commuters between different census zones, and can be converted into a matrix of passengers travelling between different nodes by assigning zones demand to the network nodes. It is done by assigning the zone demand to a node if the zone's centroid is not more than 400 meters away from the node position. In case that the zone centroid is close to more than one node, the travel demand is divided equally between these nodes. Table 1 summarises the features of the data set. Note that the instances generation procedure ensures producing symmetrical demand and travel time matrices

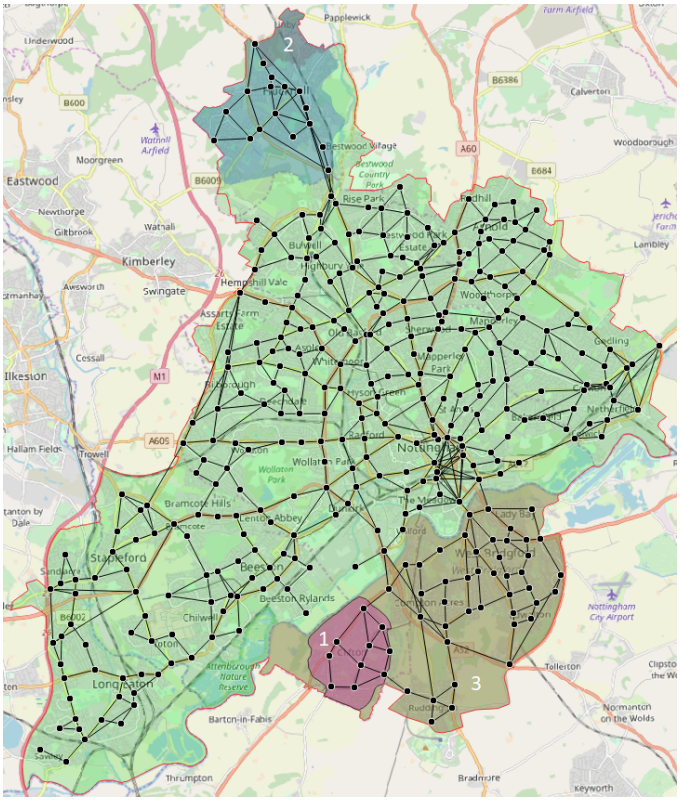


Figure 3: Map of the study area together with nodes and network edges generated with the method described in section 4. (map source: https://www.openstreetmap.org.) The colours and numbers indicate the areas of the instances: 1: Clifton (red), 2: Hucknall (blue), 3: South of Trent (brown) and 4: Nottingham (green). It should be noted that the instances Hucknall and South of Trent are subsets of the Nottingham instance and in the same way Clifton is a subset of South of Trent.

Table 1: Features of the data set

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

matching the problem description (section 2). The Journey patterns used in generating the real route sets are also modified to satisfy the problem constraints, in order to ensure fair comparison to the optimisation results.

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

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

The original description of the instances generation procedure is in [11] where the steps described above are applied to generate the larger instance of Nottingham, and the same steps are applied in this work to generate the smaller instances set. All the instances and information on how to use them can be downloaded from [1].

5 EXPERIMENTAL RESULTS

5.1 NSGAII Optimisation

NSGAII [7] is an elitist non-dominated sorting algorithm used very widely in multi-objective optimisation. The idea is to generate a parent population of size N_{pop} and use it to generate an offspring population of size N_{pop} through crossover and mutation operations. The parent and the offspring populations are combined to produce a population of size $2 \cdot N_{pop}$ from which the population for the next generation is selected by applying non-dominated sorting algorithm and crowding distance and choosing the first N_{pop} solutions of the sorted population. The NSGAII is applied in [13] to solve the UTRP problem in Mandl and Mumford data sets, and in [11] it has been tailored to adapt to the presence of terminal nodes. In this work we have used the algorithm applied in [11] that found preliminary optimised results for Nottingham instance. We will give a brief outline here for the crossover and mutation operators of this algorithm. The crossover operator generates an offspring route set from pairs of parent route sets, where the routes from the two parents are selected alternately such that the proportion of unseen vertices in the offspring is maximised. The generated offspring route set has then a certain chance to undergo mutation. For the mutation,

one of the following mutation operators is selected randomly: delete nodes, add nodes, exchange two routes, replace route, merge two routes. These operators are similar to the ones implemented in [13] with some modified to the presence of terminals. Note that the NSGAII is subject to the same constraints described in section 2, and a feasibility test after both crossover and mutation ensures that all the offspring route sets obey these constraints. Repair operators to add the missing nodes, and replace overlapping routes are also implemented to avoid rejecting too many solutions. Further, NSGAII attempts to minimise the same objectives as the hyper-heuristics, using the equations 1 and 2. The parameters for route lengths and numbers of routes in the route set are are also the same as in the hyper-heuristics experiments. We will be comparing the results of NSGAII with SS-GD in the following subsection.

5.2 Comparison of SS-GD and NSGAII

The experiments were conducted by applying SS-GD (i.e. SSHH used with great deluge, GD) to each instance of the data set following three scenarios: 1) from the perspective of passenger 2) from the perspective of operator 3) balancing the two objectives. This is achieved by setting the parameters in equation 3 as follows: to generate route sets biased toward the passenger (operator) objective α (γ) is set to 10^{-4} while the other parameter is set to 1. Whilst for balancing the two objectives α is set to 2 and γ to 1. The three scenarios are applied to each instance, and for each scenario the hyper-heuristic is run for 10 trials, each terminating after a specific time period. The running length of each trial increases with the instance size, with the smallest instance run for five minutes and the largest for three hours. The NSGAII experiments use the following sizes for the initial population: 50 for Nottingham and South of Trent, 25 for Hucknall and 10 for Clifton. The experiments are run for 200 generations for each instance.

Table 2 summarises and compares the results of SS-GD against NSGAII from the perspective of passenger and operator measured in minutes for the average passenger travel time and the total routes length. The minimum result in the 10 trials is compared to the best result found by NSGAII from the perspective of passenger and operator. The average of the 10 trials is also recorded. Figure 4 plots the results of SS-GD with the evaluation results of the final population which forms a clear Pareto front. SS-GD results are taken from four key positions: the best result from the passenger perspective, the best result from the operator perspective, the most passenger friendly and the most operator friendly route sets in the 10 trials that balance the two objectives.

From results in table 2 and the plots, it can be clearly seen that SS-GD outperforms NSGAII from the passenger and operator perspectives in all instances. In fact, the best passenger results for SS-GD not only succeeded in improving the passenger average travel time, but also the operator cost has improved. The best operator results for SS-GD also improve significantly over NSGAII in all instances, especially the largest instance Nottingham, although NSGAII could find better average travel times for passengers in this case. The compromise solutions (i.e. balancing the two objectives) of SS-GD are also very successful. Comparing these solutions to the solutions of NSGAII with the same passenger objective, SS-GD is successful in finding much improved costs for the operator, and this observation applies for all instances. The greatest success is witnessed in the largest instance of Nottingham, where the most passenger friendly route set in the compromise solutions is better than the best passenger result found by NSGAII, while the operator cost is improved by more than 50%.

Also comparing the run time of these algorithms for the largest instance Nottingham, NSGAII requires more than a week to generate a final population of Pareto solutions. SS-GD is much faster in producing a single solution of high quality compared to NSGAII in a single run, which takes only three hours, as mentioned previously. This can be clearly seen in the best passenger results of Nottingham instance where SS-GD was able to reduce the passenger travel time by 1 minute and offer better operator costs compared to NSGAII in an individual run of three hours.

Table 2: Comparison between the best results from the perspectives of passenger and operator between SS-GD and NSGAII for each instance.

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

5.3 Comparison with Real World Route Sets

In this section we compare the optimisation results with the real world routes for the two largest instances: Nottingham and South of Trent. The real world bus routes are the operating routes in the city of Nottingham from the year 2011 extracted from the national public transport data repository (NPTDR) as described in section 4.

The plots in figure 4 indicate that SS-GD is able to provide improved solutions over the real routes, given that there are Pareto points (red) that clearly dominate the real route positions (green). Taking the Nottingham instance as an example, the real routes offer a single passenger an average travel time of 14.3 minutes and the summed route length for the entire route set in minutes is 1369. The best result from the passenger perspective found by SS-GD

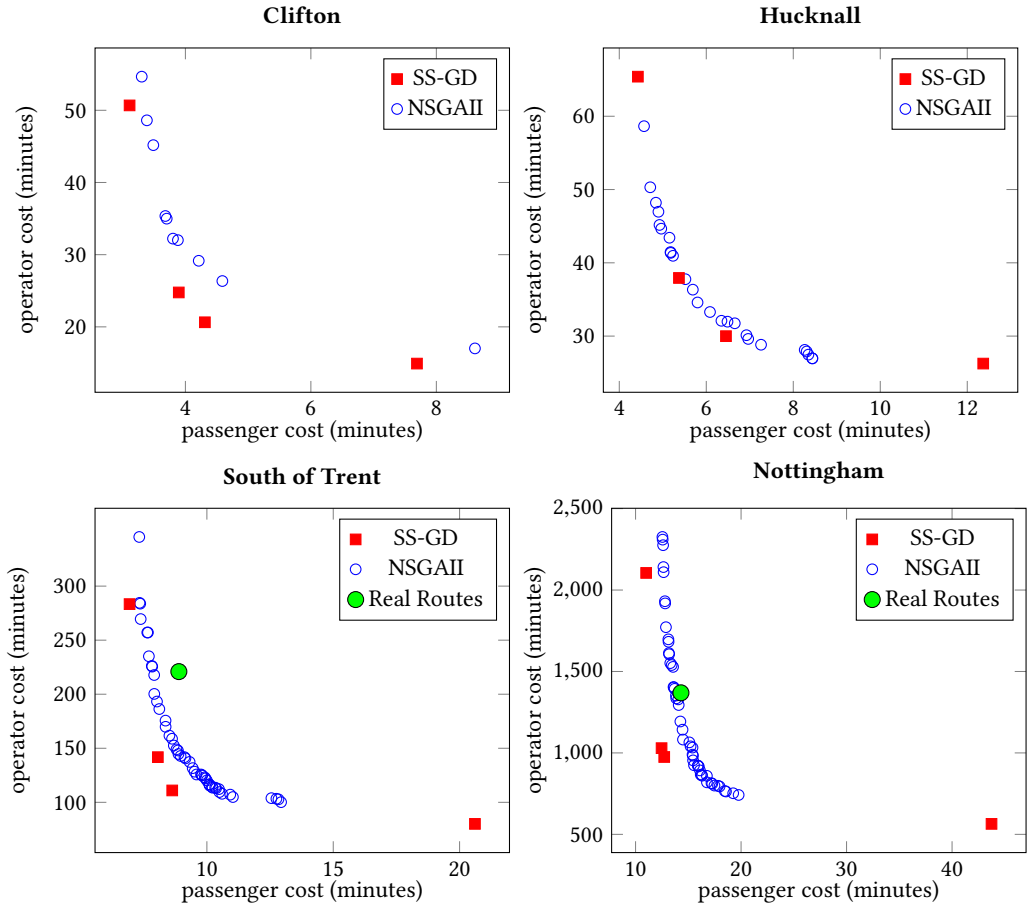


Figure 4: SS-GD results plotted against the evaluation results of the final population. The blue dots are the results of the population evaluation and the red dots are SS-GD results.

(11.00) decreased the average travel time by 3 minutes, while the average routes length increased by almost 50% (2105). On the other hand the most passenger friendly route set in the compromise route sets (12.46) improved the average travel time of the real routes by 2 minutes, and the routes length improved by almost 25% (1029). Also trip directness is enhanced by decreasing the percentage of passengers needing two transfers from 14% to 10% while increasing the percentage of direct travellers from 30% to 33%. More detailed results for this comparison with percentages of improvement over the real routes are found in the supplementary material [1].

6 CONCLUSIONS

In this paper we have described a new hyper-heuristic approach to solving the UTRP and presented a set of benchmark instances generated using a novel procedure that reduces and simplifies a real street network to be easily managed by optimisation algorithms, while at the same time maintaining the characteristics of the street network layout. Four new instances of varying sizes are introduced, and certain nodes in the network are designated as terminal nodes, which exclusively allow the start and end of bus journeys. The SS-GD hyper-heuristic is tested on the new data set with specific implementation tailored to the presence of terminal points and compared to the solutions generated by NSGAII genetic algorithm and also to the real bus routes used by local bus companies. Comparisons show the success of SS-GD in finding solutions better than NSGAII in all the instances from the perspective of passenger and operator. Also SS-GD was able to improve the existing routes service for both passengers and operators, and shows a great potential for handling complicated and real world versions of UTRP in very short run times compared to genetic algorithms. The data set is publicly accessible for free use by researchers.

In future work We plan to take the hyper-heuristics approach further to consider one-way streets, set bus arrival frequencies on the routes, and model passenger behaviours in a more realistic way.

REFERENCES

[1] Leena Ahmed, Philipp Heyken-Soares, Christine L Mumford, and Yong Mao. 2019. Data set download and Supplementary Material. (2019). https://www.nottingham.ac.uk/research/groups/lucas/ To download data, results, and other supplementary material.

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

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

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

[5] Joanne Suk Chun Chew, Lai Soon Lee, and Hsin Vonn Seow. 2013. Genetic algorithm for biobjective urban transit routing problem. *Journal of Applied Mathematics* 2013 (2013), 1–15.

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

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

[8] Gunter Dueck. 1993. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *J. Comput. Phys.* 104, 1 (1993), 86–92.

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

[10] Valérie Guihaire and Jin-Kao Hao. 2008. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice* 42, 10 (2008), 1251–1273.

[11] Philipp Heyken-Soares, Christine L Mumford, Kwabena Amponsah, and Yong Mao. 2019. An Adaptive Scaled Network for Public Transport Route Optimisation. (2019). In Review.

[12] OJ Ibarra-Rojas, F Delgado, R Giesen, and JC Muñoz. 2015. Planning, operation, and control of bus transport systems: A literature review. *Transportation Research Part B: Methodological* 77 (2015), 38–75.

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

[14] Ahmed Kheiri and Ed Keedwell. 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, 417–424.

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

[16] C.E. Mandl. 1979. *Applied network optimization.* Academic Press.

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

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

[19] SB Pattnaik, S Mohan, and VM Tom. 1998. Urban bus transit route network design using genetic algorithm. *Journal of transportation engineering* 124, 4 (1998), 368–375.

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