



## An improved solution methodology for the urban transit routing problem

G. Hüttelmann <sup>a,\*</sup>, J.H. van Vuuren <sup>a</sup>, S.J. Andersen <sup>b</sup>

<sup>a</sup> Department of Industrial Engineering, Stellenbosch University, Private Bag X1, Matieland, 7602, South Africa

<sup>b</sup> Department of Civil Engineering, Stellenbosch University, Private Bag X1, Matieland, 7602, South Africa



### ARTICLE INFO

Dataset link: [https://github.com/GHuss7/UTR\\_P-Improvements-Article-Results](https://github.com/GHuss7/UTR_P-Improvements-Article-Results)

#### Keywords:

Transit network design problem  
Urban transit routing problem  
Dominance-based multi-objective simulated annealing  
Non-dominated sorting genetic algorithm II  
Metaheuristic  
Hyperheuristic

### ABSTRACT

An improved solution methodology is proposed in this paper for the *urban transit routing problem* (UTRP). This methodology includes a procedure for the generation of improved initial solutions as well as improved metaheuristic search approaches, involving the use of hyperheuristics to manage search operators in both trajectory-based and population-based metaheuristics. The UTRP variant considered in this paper is that of deciding upon efficient bus transit routes. The design criteria embedded in our UTRP model are the simultaneous minimisation of the expected average passenger travel time and minimisation of the system operator's cost (measuring the latter as the sum total of all route lengths in the system). The model takes as input an origin–destination demand matrix for a pre-specified set of bus stops, along with an underlying road network structure, and returns as output a set of bus route trade-off solutions. The decision maker can then select one of these route sets subjectively, based on the desired degree of trade-off between the aforementioned transit system design criteria. This bi-objective minimisation problem is solved approximately in three distinct stages — a solution initialisation stage, an intermediate analysis stage, and an iterative metaheuristic search stage during which high-quality trade-off solutions are sought. A novel procedure is introduced for the solution initialisation stage, aimed at effectively generating high-quality initial feasible solutions. Two metaheuristics are implemented to solve instances of the problem, namely a dominance-based multi-objective simulated annealing algorithm and an improved non-dominated sorting genetic algorithm, each equipped with a hyperheuristic capable of managing the perturbation operators employed. Various novel operators are proposed for these metaheuristics, of which the most noteworthy take into account the demand of passengers.

### 1. Introduction

The problem of providing effective public transport in urban areas is increasingly becoming urgent. A growing population and rising levels of traffic congestion result in increased travel times for commuters (Mandl, 1979). The streamlining of public transport systems holds considerable potential in terms of alleviating this problem. In this paper we contribute towards such a streamlining effort in the context of designing high-quality bus transit systems.

The problem of designing an effective bus transit system involves three fundamental elements. The first involves dealing with the challenges associated with collecting passenger *origin–destination* (OD) travel data. The second, and more challenging, problem is making sense of such past OD pair data (under the assumption that past travel patterns are indicative of future transport demand) and finding a way to group or cluster these data in a logical and helpful manner so that they can be used to derive appropriate bus transit system requirements. Once future passenger OD travel demand is known or has been estimated,

the third element is that the bus service optimisation process becomes a vehicle routing problem — a notoriously difficult combinatorial optimisation problem in the operations research literature.

Traditional methodologies for collecting transport demand data are becoming obsolete in the digital age of big data (Dunlap et al., 2016). Whereas methodologies have historically relied on the manual collection of data by conducting surveys and inputting data manually into databases, these data are today widely available and easily accessible. Recently, there has been an increase in on-board mobile communication device sensing for transit data collection purposes. This approach towards data collection holds advantages over survey- and automated fare collection-based methods. The principle behind Wi-Fi and Bluetooth device sensors is that multiple sensors can be used to record the unique *media access control* (MAC) address of each wireless communication device at points along the travel routes of passengers. The re-identification of passengers over time along their travel routes by multiple sensors allows for the reconstruction of entire passenger trips and also for the

\* Corresponding author.

E-mail addresses: [ghussel94@gmail.com](mailto:ghussel94@gmail.com) (G. Hüttelmann), [vuuren@sun.ac.za](mailto:vuuren@sun.ac.za) (J.H. van Vuuren), [jandersen@sun.ac.za](mailto:jandersen@sun.ac.za) (S.J. Andersen).

computation of the corresponding travel speed (Dunlap et al., 2016). The availability of these data holds considerable exploitation potential when applying vehicle routing-based solution approaches to bus transit system design based on more accurate OD travel data. The challenge, however, is making sense of the resulting abundance of data and converting them into application-specific and actionable information.

Once passenger OD travel demand has been estimated, the subsequent combinatorial optimisation problem associated with the design of a system of bus routes tailored to service this demand is generally known as the *urban transit routing problem* (UTRP) in the literature (although various other naming conventions also exist). Our quest in this paper is to contribute towards the automation of a methodology in support of identifying high-quality decision recommendations with respect to transit route network design aimed at satisfying demand effectively. This automated methodology is based on the solution of an optimisation model which requires both a passenger OD travel demand matrix and the available road infrastructure as input. The model is bi-objective in nature and yields as output a set of high-quality bus routes achieving trade-offs between operator cost minimisation and passenger cost minimisation. A decision maker may then select a particular solution from this recommended trade-off set for implementation purposes, based on subjective preferences with respect to the importance of minimising operator cost relative to that of minimising passenger cost.

Various UTRP multi-objective optimisation models are available in the literature and these models are routinely solved approximately by invoking state-of-the-art trajectory-based and population-based metaheuristics. The objectives of these models are, however, usually scalarised into single objective functions instead of solving the models in a truly multiple objective trade-off fashion. Moreover, serious problems often arise during attempts at tailoring approximate model solution approaches to the intricacies of the UTRP, such as that the metaheuristic search operators employed are not all equally effective for all model instances, that these operators generally lack dynamic search flexibility, and that it is difficult to generate initial feasible solutions with which to seed the metaheuristics in a manner that spans the entire range attainable in objective function space.

The contributions of this paper are four-fold: (i) The proposal of a novel approach towards effectively generating initial feasible solutions to instances of the UTRP, (ii) the introduction of new metaheuristic search operators which may be employed to uncover high-quality transit networks, (iii) the suggestion of a hyperheuristic strategy for managing these search operators dynamically during solution search runs, and (iv) improved trade-off solutions for a number of popular academic UTRP benchmark instances documented in the literature.

Apart from the current introduction, this paper contains a further five sections. Section 2 is devoted to a brief literature review of the UTRP and its objectives, while Section 3 contains a derivation of the UTRP model adopted in this paper. Our initial candidate route set generation procedure is presented in Section 4, after which our model solution approaches are described in Section 5. Numerical results are finally reported and interpreted in Section 6, while a conclusion follows in Section 7, highlighting and contextualising the main contributions of the paper.

## 2. Literature review

The global transit planning process is based on certain prerequisite inputs to *urban transit network design problem* (UTNDP) instances. These are passenger demand, specifics related to the area underlying the transport network (including its topological characteristics), details about the available transport vehicles and the number of available drivers of these vehicles. The end goal is the establishment of a set of transport routes together with their corresponding timetables (Chakroborty, 2003; Guihaire and Hao, 2008).

The planning process for transit networks has been described as a sequence of five distinct steps by Ceder and Wilson (1986), namely network design, frequency setting, timetable development, bus scheduling

and driver scheduling. For each activity, different independent inputs are required and different outputs are produced, as summarised in Table 1. The output returned during each step, except the last, is taken as the input for the next step (in addition to the inputs listed under independent inputs). Such a clear and distinct order exists only in theory as each step is, in reality, intertwined and greatly influenced by the preceding steps, but it is nonetheless helpful to distinguish between the main components of the UTRP (Ceder and Wilson, 1986).

Transit network design (Level A in Table 1) may be thought of as a strategic planning phase, which includes long-term decision making, whereas the tactical planning phases include frequency setting (Level B) and transit network timetabling (Level C). Vehicle and driver scheduling are finally considered as the operational planning phase (Levels D and E) (Desaulniers and Hickman, 2007; Ibarra-Rojas et al., 2015). The first planning activity (in Level A), hereafter referred to as the UTRP, is the focus of the remainder of this review section, because of its relevance to the topic of this paper (Guihaire and Hao, 2008).

There is, however, an overarching problem, of which the UTRP described in Section 1 is, in fact, a special case. This overarching problem is the UTNDP, which comprises the first three levels (Levels A–C) of transit planning mentioned in Table 1. The smaller UTRP is a variation on the well-known vehicle routing problem (Toth and Vigo, 2002), being more complex in nature from a combinatorial perspective. It entails finding a specified number of closed travel circuits, in the context of a given transport network, which collectively connect origin and destination pairs, resulting in all network vertices being served without the demand levels at these vertices being known *a priori* (Mandl, 1979; Kepaptsoglou and Karlaftis, 2009). In some variations on the redesign of bus routes, the bus stop locations can also be changed, but in the case of the UTRP, these are considered as fixed.

The UTRP is inherently multi-objective in nature. The aim is to achieve suitable trade-offs between two or more conflicting objectives (Baaj and Mahmassani, 1991; Chakroborty, 2003). A popular measure of a good transit route set, when all the transit demand is satisfied, is that a large percentage of transit demand should be satisfied by direct routes and that the average time spent by passengers in transit should be as small as possible (Chakroborty, 2003).

Several UTRP benchmark instances have been established in the literature for the purpose of model verification and validation, among which the most popular by far is Mandl's Swiss network (Mandl, 1979), also being one of the smallest test instances. Four new benchmark instances of varying sizes were introduced by Mumford in 2013 in the hope that they may be used in the context of UTRP modelling by other researchers as more realistically sized instances. Since their introduction, these benchmark instances have indeed attracted the attention of various researchers.

A meta-data summary of the five aforementioned benchmark instances is provided in Table 2. These meta-data attributes for each test instance include the number of vertices contained in the instance, the number of edges present, the number of routes sought, the number of vertices allowed per route, and lower bounds on the two most common objective functions, namely minimising the *average travel time* (ATT) and the *total route time* (TRT).

### 2.1. Solution approaches

Various UTRP solution approaches have been proposed in the literature, which may broadly be partitioned into mathematical programming approaches, heuristic approaches, trajectory-based metaheuristic approaches, population-based metaheuristic approaches and hybrid approaches.

**Table 1**  
The overall transit network planning activities, as suggested by Ceder and Wilson (1986).

Level	Independent inputs	Planning activity	Output
A	Demand data Supply data Route performance indicators	Network design	Route changes New routes Operating strategies
B	Subsidy available Vehicles available Service policies Current patronage	Frequency setting	Service frequencies
C	Demand by time of day Times for first and last trips Running times	Timetable development	Trip departure times Trip arrival times
D	Deadhead times Recovery times Schedule constraints Cost structure	Vehicle scheduling	Vehicle schedules
E	Driver work rules Run cost structure	Driver scheduling	Driver schedules

**Table 2**  
Five popular UTRP benchmark instances available in the literature (Mandl, 1979; Mumford, 2013).

Instance	Number of vertices	Number of edges	Number of routes	Vertices per route	ATT lower bound [min]	TRT lower bound [min]
Mandl	15	21	4–8	2–8	10.0058	63
Mumford0	30	90	12	2–15	13.0121	94
Mumford1	70	210	15	10–30	19.2695	345
Mumford2	110	385	56	10–22	22.1689	864
Mumford3	127	425	60	12–25	24.7453	982

### 2.1.1. Mathematical programming approaches

Although cumbersome, the UTRP can be cast as a mathematical programming problem. The approaches available in the literature for solving mathematical programming UTRP formulations are based on classical, general purpose mixed-integer programming techniques (Guilhare and Hao, 2008).

In 2006, Guan et al. proposed a linear binary programming UTRP model for the simultaneous optimisation of routes and passenger transit assignment. A standard branch-and-bound method may be used to solve the model. Examples were given of the model applied to a few minimum spanning tree networks and a simplification of the mass transit railway network in Hong Kong. The authors acknowledged that their approach could only be applied to small networks, and that metaheuristic methods should instead be considered for larger networks.

Chakroborty (2003) had earlier identified certain limitations of mathematical programming UTRP model formulations. These limitations include not achieving an adequate representation of reality, the need to incorporate excessively many discrete decision variables, the requirement of having to accommodate complex logical conditions, and the need to linearise inherent non-linearities present in the objective function(s) and/or constraints. The benefit of mathematical programming UTRP model formulations, however, is that optimality of solutions can be proved, but these approaches are typically avoided due to the fact that they do not scale well in terms of computational complexity (Chakroborty, 2003). For this reason, heuristic and metaheuristic approaches have been applied to solve most instances of the UTRP in the literature (Guilhare and Hao, 2008; Kepaptsoglou and Karlaftis, 2009).

### 2.1.2. Heuristic approaches

Mandl applied a heuristic method in 1979 to a UTRP instance in which initial candidate routes were generated by finding shortest paths between the different OD pairs. Thereafter, the paths that served the largest demand portions were incorporated iteratively into the solution. These paths were then configured to meet the requirements of service coverage and route directness objectives. The routes were finally modified in an iterative fashion in pursuit of another objective, namely

to minimise the total travel time of passengers. Mandl's method was applied to a 15-vertex real transport network in Switzerland servicing 15 570 demand trips per day, on average.

Lee and Vuchic (2005) minimised passenger travel time by employing an iterative procedure under the assumption of variable demand. An initial network was generated by applying a shortest path algorithm between all pairs of vertices, and then eliminating undesirable paths such as those which were subpaths of other paths. Demand was assigned to routes by invoking a transit assignment procedure, and by concentrating the demand along certain routes, less efficient routes could be eliminated. Further refinement of the routes was facilitated by an improvement heuristic in which passenger travel time was decreased. Passenger travel mode choice was modelled by means of a logit formulation.

### 2.1.3. Trajectory-based metaheuristic approaches

Zhao and Gan (2003) presented an aggregated metaheuristic UTRP solution approach, incorporating three algorithms — a greedy search algorithm, a fast hill climbing algorithm and an integrated simulated annealing, tabu and greedy search algorithm. The two objectives pursued were minimisation of the number of transfers incurred by passengers and maximisation of service coverage by the routes. The authors applied their solution approach to a large transport network in Miami-Dade County, Florida. Zhao and Ubaka (2004) published further details of both the aforementioned basic greedy search algorithm and the fast hill climbing search algorithm for solving UTRP instances. Zhao et al. (2005) published another paper based on a combination of the aforementioned two works with similar approaches.

Fan and Machemehl (2004) wrote a report containing various detailed descriptions of the UTRP and its intricacies. The UTRP was cast as a multi-objective non-linear mixed-integer model, but weights were assigned to each objective. The sum of these weighted objectives was then minimised. The objectives represented the total passenger travel time, the total number of buses required, and the cost of unsatisfied demand. The proposed solution approach included three phases, namely an initial candidate route set generation procedure, a network analysis procedure for evaluating performance, and a metaheuristic solution

search procedure. Five metaheuristic search procedures were evaluated for use during the latter phase, including three trajectory-based metaheuristics (local search, simulated annealing, and tabu search) and two population-based metaheuristics (a genetic algorithm, and a random search). Sensitivity analyses were conducted in respect of each algorithm, and the relative performances of the algorithms were compared in the context of benchmark problem instances. Both fixed and variable transit demand were considered. The concepts of centroid vertices and distribution vertices applied to the demand of the UTRP were introduced by the authors, yielding a more accurate distribution of passenger demand across the network.

[Fan and Machemehl \(2006b\)](#) published a simulated annealing algorithm tailored for solving instances of the UTRP, based on their earlier report ([Fan and Machemehl, 2004](#)), claiming that the algorithm outperformed a genetic algorithm in most example network cases considered by them. Later, [Fan and Machemehl \(2008a\)](#) proposed three tabu search implementations for solving instances of the UTRP. Their overall approach closely conformed to the framework proposed in their previous report ([Fan and Machemehl, 2004](#)).

The relative quality of solution representation schemes, the initialisation procedures for algorithms and neighbourhood move suitability for a simulated annealing metaheuristic in the context of the UTRP were considered in some detail by [Fan and Mumford \(2010\)](#). For validation purposes, a simple hill climbing algorithm was employed. Their contribution was a basic metaheuristic framework for solving instances of the UTRP, including a solution representation scheme, an initialisation procedure and a set of neighbourhood moves. Their model included a single objective function in which weights were assigned to the objectives of minimising passenger travel time and minimising the number of transfers.

[Yan et al. \(2013\)](#) introduced a hybridised simulated annealing implementation in conjunction with Monte Carlo simulation to solve instances of the UTRP, considering stochastic travel times. The objective was to minimise operator cost, subject to service level constraints.

[Kiliç and Gök \(2014\)](#) introduced a new route generation procedure and tested it in the context of five benchmark instances. The procedure involved finding shortest paths in a network, and calculating the total passenger usage per edge. Discrete usage probabilities were determined by dividing the calculated total for each edge by the weight of each edge and normalising the quotients returned. The route generation procedure took into account these probabilities. Thereafter, the initial solutions were improved by utilising a hill climbing heuristic in conjunction with a tabu search procedure.

[Ahmed et al. \(2019\)](#) employed selection hyperheuristics to solve instances of the UTRP by exploring the space of low-level heuristics used to modify route sets. Each selection heuristic was empirically tested in respect of a number of benchmark instances made available by [Mumford \(2013\)](#). It was found that a sequence-based selection method in combination with a great deluge acceptance method performed the best overall, also achieving fast algorithmic run times. Objective functions previously considered by [John et al. \(2014\)](#) were utilised.

[Ahmed et al. \(2019\)](#) subsequently applied a hyperheuristic approach within the context of fixed terminal vertices for the UTRP. They required that buses be restricted to start and end their trips at specified terminal vertices. When results were compared with those returned by a *non-dominated sorting genetic algorithm II* (NSGA II) implementation, it was found that the hyperheuristic approach outperformed the NSGA II.

#### 2.1.4. Population-based metaheuristic approaches

Evolutionary algorithms are population-based search procedures based on the principles of natural selection in evolutionary biology. In essence, these searches are parallel searches through the solution space aimed at an iterative improvement of the mean fitness value of an evolving population of candidate solutions to an optimisation problem ([Fan and Machemehl, 2004; Guihaire and Hao, 2008](#)). The

papers by [Fan and Machemehl \(2006a, 2004\)](#), [Chakroborty \(2003\)](#), [Chakroborty and Wivedi \(2002\)](#), and [Mumford \(2013\)](#) contain examples of evolutionary algorithms being applied to instances of the UTRP.

Another class of population-based metaheuristics is based on the notion of swarm intelligence, where the decentralised flocking behaviour of animals foraging for food is simulated. The papers by [Nikolić and Teodorović \(2013\)](#), and by [Kechagiopoulos and Beligiannis \(2014\)](#) contain examples of swarm intelligence-based algorithms being applied to instances of the UTRP.

An innovative approach towards including additional routes in existing route networks was introduced by [Xiong and Schneider \(1992\)](#). They improved upon a standard multi-objective evolutionary algorithm that resembles a genetic algorithm by collecting non-dominated solutions during the entire search process, as opposed to only returning the last generation, and called this a cumulative genetic algorithm. Furthermore, a neural network was used to approximate one of the objective functions, namely passenger travel time, by predicting the output of a passenger trip assignment algorithm. Their approach not only delivered good results, but also achieved a substantial time gain over other transit assignment algorithms of the time.

Another genetic algorithm-based UTRP solution approach was proposed by [Chakroborty and Wivedi \(2002\)](#). A route set was initially constructed by invoking a heuristic. Five criteria were considered to determine the fitness value of a candidate route set, namely passenger travel time (including in-vehicle travel time and transfer time), the percentage of demand met with either zero, one, or two transfers, and the percentage of unsatisfied demand (requiring more than two transfers). For crossover, routes or parts thereof were exchanged with parts of other route sets. A string representation was adopted for route set encoding.

[Fan and Machemehl \(2006a\)](#) employed a genetic algorithm to examine the underlying characteristics of the UTRP with variable transit demand, based on their earlier report ([Fan and Machemehl, 2004](#)). The characteristics examined were the redesign of existing networks, the effect of demand aggregation, and the effect of various route set sizes.

[Nikolić and Teodorović \(2013\)](#) applied a bee colony optimisation metaheuristic for solving the UTRP, aiming to maximise the number of passengers serviced, minimise the total travel time of passengers and minimise the total number of transfers incurred.

A genetic algorithm was also employed by [Chew et al. \(2013\)](#) for solving instances of the UTRP. Their approach included a repair mechanism according to which vertices were inserted into routes contained in infeasible route sets in an attempt to restore feasibility during the search progression. The authors also introduced a new crossover procedure for incorporation into genetic algorithms by utilising a set of feasibility criteria in an attempt to reduce the probability of producing infeasible solutions.

[Kechagiopoulos and Beligiannis \(2014\)](#) solved instances of the UTRP by applying a particle swarm optimisation algorithm, considering both passenger and operator costs, and validated their approach in the context of [Mandl's](#) benchmark instance. The authors also developed a method for accommodating the discrete decision variables of the UTRP in their solution approach.

[John et al. \(2014\)](#) proposed an NSGA II framework for solving the UTRP, cast as a multi-objective optimisation problem, and used intuitive graph-theoretic principles to model the problem. Their results led to improvements on previously published results for the four benchmark instances published earlier by [Mumford \(2013\)](#), as well as for [Mandl's](#) Swiss network. The authors also proposed a new construction heuristic for seeding an initial population of routes. [John's](#) doctoral dissertation of 2016 contains more details on the UTRP and his approach towards solving it.

A differential evolution metaheuristic solution approach (a genetic algorithmic variant designed for continuous search spaces) was proposed by [Buba and Lee \(2016\)](#) for the UTRP. Their model incorporated

only the minimisation of the average travel time of passengers. The authors proposed a new repair mechanism that may be applied to infeasible route sets. Different repair mechanisms were compared, and a particular combination of multiple repair mechanisms was identified which returned superior results.

### 3. The UTRP model

This section is devoted to a derivation of the mathematical model for the UTRP adopted in this paper. The model closely follows the intuitive graph theoretic modelling approach taken by [John et al. \(2014\)](#) instead of a standard mathematical programming model formulation. In the model, a bi-objective optimisation paradigm is adopted in which the objectives are to minimise passenger cost and operator cost simultaneously (both measured in units of time), with the routes selected for inclusion in the transit network representing the decision variables. The road network structure is assumed to be an irregular grid, and the assumption is made that passenger demand is fixed over time and exhibits a many-to-many pattern. The four constraints included in the model are similar to those adopted by [Mumford \(2013\)](#) and [John et al. \(2014\)](#), and involve route shape, route length and vehicle stop specification, along with a requirement that the transit network should be connected. In line with the studies of [Mumford \(2013\)](#) and [John et al. \(2014\)](#), it is assumed that if a route transfer were to be made by a passenger, a transfer penalty of 5 min per transfer is applied to the travel time.

We model the road network on which the UTRP has to be solved as an edge-weighted graph  $G = (\mathcal{V}, \mathcal{E})$  in which the vertex set  $\mathcal{V} = \{v_1, \dots, v_n\}$  represents required bus stops and the edge set  $\mathcal{E} = \{e_1, \dots, e_m\}$  represents road links between these stops. Each edge represents a shortest direct travel route between a pair of vertices in  $G$ , containing no intermediate bus stops. The edge weights of  $G$  are captured in an  $n \times n$  weight matrix  $\mathbf{W}$  whose entry in row  $i$  and column  $j$  denotes the expected travel time between vertices  $v_i$  and  $v_j$ . If no edge is present between  $v_i$  and  $v_j$  in  $G$ , then the weight  $\infty$  is assigned instead ([Henning and Van Vuuren, 2022](#)). The graph  $G$  is also associated with an  $n \times n$  OD demand matrix  $\mathbf{D}$ , whose entry in row  $i$  and column  $j$  is the passenger demand from vertex  $v_i$  to vertex  $v_j$ .

A simple shortest path in  $G$  is used to represent each *transit route*, consisting of an ordered sequence of distinct vertices in  $\mathcal{V}$ . Each successive pair of vertices in such a transit route is, of course, adjacent in  $G$ . The subgraph of  $G$  induced by the vertices of a transit route set  $\mathcal{R}$  is denoted by  $G_{\mathcal{R}} = (\mathcal{V}_{\mathcal{R}}, \mathcal{E}_{\mathcal{R}})$ . A transit route set  $\mathcal{R}$  is considered infeasible in the context of the UTRP unless each vertex in  $\mathcal{V}$  is present in at least one route  $R \in \mathcal{R}$  ([Fan and Mumford, 2010; John et al., 2014](#)), expressed as

$$\bigcup_{R \in \mathcal{R}} \mathcal{V}_R = \mathcal{V}, \quad (1)$$

where  $\mathcal{V}_R$  is the vertex subset of  $\mathcal{V}$  contained in  $R$ . The number of bus stops is also limited by a minimum allowable number  $m_{\min}$  and a maximum allowable number  $m_{\max}$  for each route in the route set  $\mathcal{R}$  ([Fan and Mumford, 2010; John et al., 2014](#)). It is therefore required that

$$m_{\min} \leq |\mathcal{V}_R| \leq m_{\max}, \quad R \in \mathcal{R}. \quad (2)$$

These constraints are based on various considerations, including, for example, schedule adherence and driver fatigue ([Zhao and Gan, 2003](#)). Moreover, it is also required that the subgraph induced by the vertices of all the routes in a transit route set form a connected graph ([John et al., 2014](#)), expressed as

$$G_{\mathcal{R}} = \left( \bigcup_{R \in \mathcal{R}} \mathcal{V}_R, \bigcup_{R \in \mathcal{R}} \mathcal{E}_R \right) \text{ must be connected.} \quad (3)$$

Finally, the required number of transit routes in a route set  $\mathcal{R}$  is also specified. This requirement takes the form

$$|\mathcal{R}| = r, \quad (4)$$

for some natural number  $r$ .

Passengers generally prefer travelling to their destinations in the shortest time possible, while also attempting to avoid the inconvenience associated with making route transfers, if possible. The time duration of travelling along a shortest path between an OD pair  $v_i, v_j \in \mathcal{V}$  along a transit route set  $\mathcal{R}$  is denoted by  $\alpha_{v_i, v_j}(\mathcal{R})$ , measured in units of time. This may be computed by applying Dijkstra's shortest path finding algorithm to the subgraph  $G_{\mathcal{R}}$ . Both transport time and transfer time (if route transfers are required) are included in this shortest path calculation. The passenger (time) cost associated with a route set  $\mathcal{R}$  was calculated by [Mumford \(2013\)](#) as the ATT for all passengers, measured in units of time and expressed as

$$F_1(\mathcal{R}) = \frac{\sum_{i=1}^n \sum_{j=1}^n D_{v_i, v_j} \alpha_{v_i, v_j}(\mathcal{R})}{\sum_{i=1}^n \sum_{j=1}^n D_{v_i, v_j}}. \quad (5)$$

The numerator in (5) represents the total travel time of all passengers in the system, and the denominator represents the total number of passengers in the system.

A simple proxy was, however, proposed by [Mumford \(2013\)](#) for operator cost. This proxy is the sum of the total travel times associated with the transport vehicles along the various routes (in one direction). This *total route time* (TRT) may be expressed as

$$F_2(\mathcal{R}) = \sum_{R \in \mathcal{R}} \sum_{(v_i, v_j) \in R} W_{v_i, v_j}. \quad (6)$$

We minimise both objective functions (5)–(6) simultaneously in order to establish a Pareto front representing effective trade-offs between minimising passenger cost and minimising operator cost.

### 4. Initial candidate route set generation

In this section, we propose a novel *initial candidate route set generation procedure* (ICRSGP) for generating an initial transit route set  $\mathcal{R}$  that satisfies the constraints in (1)–(4), called the *K* shortest paths maximising missing vertices ICRSGP. Thereafter, a novel method is proposed for enhancing the quality of the routes generated by this ICRSGP, in terms of decreasing the ATT or the TRT.

#### 4.1. K shortest paths maximising missing vertices ICRSGP

The use of *K* shortest paths has widely been adopted in the literature in a variety of ways for generating initial feasible solutions to instances of the UTRP ([Fan and Machemehl, 2006b; John, 2016; Shih and Mahmassani, 1994](#)). One of the interesting features of the UTRP is that shortest paths alone are not always sufficient to form routes, as there is an underlying demand component which is not thus always adequately accounted for.

We propose a novel method for taking the demand component into account during initial route generation. This method is designed to promote both demand covered and achieving short travel times, and involves generating *K* shortest paths between each source-target pair of vertices  $i$  and  $j$  in  $G$ , upon which only the best  $\ell$  of these *K* shortest paths are selected based on sorting by route length in descending order first, and then resolving ties by sorting the demand serviced in ascending order. A pseudo-code description of this algorithm is presented in Algorithm 3 in the Appendix. This approach is aimed at minimising the average travel time of passengers not just simply by taking a single shortest path, but by determining other short paths which potentially cover more demand.

Our method of *K* shortest paths generation is based on Yen's well-known *K* shortest path finding algorithm. [Shih and Mahmassani \(1994\)](#) argued that ten shortest paths between each vertex pair is sufficient for route generation, as more paths are likely to lead to violating the in-vehicle travel time constraints. We found, however, that this is only true for smaller networks. In larger networks, on the other hand, many short paths typically achieve very similar TRT values. We therefore

computed  $K = 50$  shortest paths between vertex pairs for all of the numerical results reported in this paper.

During preliminary testing involving randomly selecting and inserting routes from a candidate route set  $C$  into a potentially feasible route set  $\mathcal{R}$ , we found that for larger instances of the UTRP it is often difficult to establish initial feasible route sets. This is because it is unlikely to include all the vertices of  $G$  by randomly selecting routes from a candidate list of short routes, thus violating constraint (1). We therefore propose a novel approach towards generating initial feasible solutions which is consistently able to find feasible solutions to instances of various sizes rather quickly. This approach was inspired by Mumford's method of crossover of 2013, and a pseudo-code description is presented in Algorithm 4 in the Appendix.

Mumford's approach takes as input two parent feasible route sets to the UTRP instance under consideration and returns a feasible offspring solution. Approximately half of the routes of each selected parent are used to produce one offspring solution  $\mathcal{R}'$ . In particular, one route is selected from each parent in an alternating fashion and included in the offspring route set  $\mathcal{R}'$ , until  $|\mathcal{R}'| = r$ . Routes are favoured if their inclusion in the offspring lead to a larger proportion of hitherto missing vertices being included. Measures are put in place to ensure that the routes included remain connected.

More specifically, the offspring  $\mathcal{R}'$  is seeded by randomly selecting one route from a parent solution (Mumford, 2013). Next, a route from the second parent is selected. Connectivity is ensured by only considering routes from the second parent which have at least one vertex in common with the current offspring solution  $\mathcal{R}'$ . This requirement can be expressed as  $\mathcal{V}_{R_i} \cap \mathcal{V}_{R'} \neq \emptyset$ , where  $\mathcal{V}_{R'}$  is defined as

$$\mathcal{V}_{R'} = \bigcup_{R_j \in R'} \mathcal{V}_{R_j} \quad (7)$$

and  $R_j$  is the  $j$ th route from the second parent that is eligible for inclusion (John, 2016).

The proportion of missing vertices from  $\mathcal{R}'$  contained in an eligible route  $R_i$  is

$$\frac{|\mathcal{V}_{R_i} \setminus \mathcal{V}_{R'}|}{|R_i|}, \quad (8)$$

Consider, for example, a current offspring solution  $\mathcal{R}' = \{(1, 2, 4, 5, 6)\}$  and a route  $R_i = \langle 3, 4, 7, 8, 9, 11 \rangle$  being considered for subsequent inclusion. The evaluation of  $\mathcal{V}_{R_i} \setminus \mathcal{V}_{R'}$  then yields  $\{3, 7, 8, 9, 11\}$ , and so the expression in (8) yields the value  $\frac{5}{6}$  (John, 2016). An eligible route  $R_i$  that achieves the maximum proportion of missing vertices is chosen to be inserted into the offspring solution. If more than one route achieves this maximum value, a uniform random choice is made between them for insertion (Mumford, 2013). Attention is then reverted back to the first parent, and a similar process is followed to augment the routes in  $\mathcal{R}'$ , until  $|\mathcal{R}'| = r$ .

It is possible that not all route sets constructed according to this procedure may be feasible, and in that case, a *repair strategy* is required to rectify the route set  $\mathcal{R}'$ . The repair strategy proposed by Mumford (2013) is adopted for this purpose, during which attempts are made to include the missing vertices from  $\mathcal{V}_{R'}$ . The missing vertices can be joined to an adjacent terminal vertex of a route in an infeasible route set, if such a relationship exists in the transit network (Mumford, 2013). More specifically, this can be achieved by first identifying all the missing vertices, and then selecting each one in a random order for evaluation and possible insertion. Thereupon, all the terminal vertices are identified and it is determined whether any adjacency relationship exists between a missing vertex and a terminal vertex. If such a relationship exists, the missing vertex is added to the corresponding terminal vertex in the relevant route. If more than one such relationship exists, a terminal vertex is randomly selected and joined to the missing vertex. If no such relationship exists, the repair strategy is unable to generate a feasible route set, the candidate route set is discarded and the process is repeated until a feasible route set has been generated.

We used this method of crossover (Algorithm 4 in the Appendix) proposed by Mumford (2013) to generate feasible route sets for the UTRP where, instead of taking two feasible parent route sets as input as was done by Mumford, two identical lists of all the candidate  $K$  shortest paths  $C$  (generated by invoking Algorithm 3) in the network were taken as the two input route sets.

#### 4.2. Initial candidate route set enhancement procedure

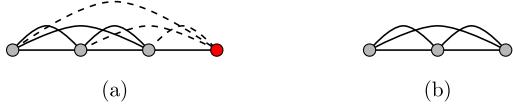
When comparing the quality of the route sets returned by the aforementioned ICRSGP with benchmark solutions, it was found that the objective function values were relatively more concentrated in certain regions in objective space than in others. It is, however, beneficial to obtain a good spread of solutions throughout the entire objective space so as to ensure the effectiveness of population-based search procedures. This may be achieved by enhancing the route sets returned by the ICRSGP according to a novel approach described in some detail in this section. An empirically observed phenomenon is that when vertices are added to routes, the ATT decreases while the TRT increases. Similarly, when vertices are removed, the ATT increases and the TRT decreases. Randomly adding vertices to or removing vertices from routes may, therefore, lead to sub-optimal choices. Smarter ways of altering routes are required to help guide an enhancement procedure towards finding higher-quality initial solutions which also achieve a good spread in objective space.

Two low-level modular operations are proposed for this purpose, called the *cost-based grow* and the *cost-based trim* procedures. These procedures operate on the marginal level of only one vertex by taking into account the demand per route length. The procedures may be applied a random number of times to route sets in order to improve their quality in either the ATT objective or the TRT objective. These procedures may, in fact, be applied to route sets until a model constraint is violated, giving more preference to a particular objective at the cost of the other.

##### 4.2.1. Cost-based trim

The term *cost-based trim* alludes to the fact that, for every vertex removed from a route, a certain demand per cost trade-off occurs. The reduction in TRT due to an edge no longer being present in a route, for instance, comes at the cost that certain direct routes no longer exist for passengers if no other alternatives are available, which increases the ATT. This cost trade-off is quantified by measuring the total direct demand satisfied by a route, and establishing a ratio in terms of the travel time cost, either by dividing demand by cost, or cost by demand. When a route has to be reduced in length, it would make sense to remove a vertex that exhibits the largest cost per unit of demand. When demand is measured in the case of adding a new vertex to a route, however, every combination of the current route's vertices and the new vertex should be considered as the additional vertex provides a direct route to all other vertices contained in the current route under consideration. Only direct routes are considered as they carry the most weight in the ATT calculation by avoiding transfer penalties, and in the end may be considered a proxy for identifying superior routes, relative to one another.

Fig. 1 illustrates this concept of the effect of removing vertices from a route, where the straight lines represent the road network or edges, and the curved lines represent the demand relationships exhibited by passengers. Fig. 1(a) contains an illustration of the situation where the red vertex is removed from the route, and the dashed edges represent the demand relationship lines that were removed as well. Fig. 1(b) contains three vertices joined by two edges, yielding six demand combinations when taking into account that a demand relationship occurs in both directions. Three additional demand relationship lines are removed, therefore bringing the total of demand combinations down from 12 to six. If only one route is considered, the calculation of the total direct demand satisfied by removing one vertex may be performed by stepping through each combination of vertices in a route  $R$  and



**Fig. 1.** The effect of meeting direct demand by removing a vertex from a route.

cumulatively adding the demand matrix entries corresponding to the vertex pair combinations to the demand satisfied, finally returning the total demand satisfied directly by the route  $R$  as output. This can also be done by considering an entire route set  $\mathcal{R}$ ; a pseudo-code description of this approach may be found in Algorithm 5 in the Appendix.

The cost-based trim operator involves stochastic removal of a terminal vertex based on a probability that is proportional to the cost per unit of demand of the pendent edge. This probability is calculated by dividing the value  $b_j$  by the sum of all the  $b_j$ -values, where  $b_j$  is calculated by dividing the edge weight of the edge removed by the demand no longer satisfied as a result of removal of the edge for each potential terminal vertex that can be removed, denoted by  $j$ . Further detail about the cost-based trim operation may be found in Hüttelmann (2022), whereas a pseudo-code description may be found in Algorithm 6 in the Appendix.

#### 4.2.2. Cost-based grow

Whereas cost-based trim involved removing vertices, cost-based grow involves adding vertices, and may be seen as the former operation's counterpart. Its implementation is, however, slightly more complex as not only the potential deletion of each terminal vertex is evaluated. Instead, the insertion of each possible feasible adjacent vertex is evaluated at both terminals of the route.

After having assigned probabilities to all candidate grows which are proportional to the respective demand per unit cost of the pendent edge, one grow is randomly selected and inserted into the relevant route. Further detail about the cost-based grow operation may again be found in Hüttelmann (2022), whereas Algorithm 7 in the Appendix contains a pseudo-code description of the algorithm.

#### 4.2.3. Random all cost-based trim

The two operations mentioned above each operates on only a single vertex at a time. In order to make significant improvements to route sets, various of these operations should ideally be performed successively on a given route set  $\mathcal{R}$ . This reasoning leads to the proposal of two further operations called *random all cost-based trim* and *random all cost-based grow*. The former operation involves randomly stepping through each route  $R_i$  in a given route set  $\mathcal{R}$ , and applying the cost-based trim operation of Section 4.2.1 a random number of times. This random number is determined by selecting a random integer uniformly between 1 and  $|R_i| - m_{\min}$ , inclusive, where  $|R_i|$  is the number of vertices already contained in route  $R_i$ , and  $m_{\min}$  is the minimum number of vertices allowed in a route of the UTRP instance under consideration.

#### 4.2.4. Random all cost-based grow

Random all cost-based grow, on the other hand, involves randomly stepping through each route in a given route set  $\mathcal{R}$ , and applying the cost-based grow operation of Section 4.2.2 a random number of times to each route. This random number of applications to route  $R_i$  is determined by selecting a random integer uniformly between 1 and  $m_{\max} - |R_i|$ , inclusive. Here,  $m_{\max}$  denotes the maximum number of vertices allowed in a route of the UTRP instance under consideration.

#### 4.2.5. Full cost-based trim

Preliminary testing revealed that the extremal values of each objective function associated with the non-dominated set returned when generating initial feasible solutions were not representative of the range of attainable values, and this led us to introduce an additional operation, called *full cost-based trim*. Full cost-based trim is when the cost-based trim operation of Section 4.2.1 is applied to all the routes in a given route set  $\mathcal{R}$ , in a random order. More specifically, the cost-based trim operation is applied  $|R_i| - m_{\min}$  times to route  $R_i$ , or until a constraint is violated, upon which the focus moves on to the next route in the random sequence of routes in  $\mathcal{R}$ . The reasoning behind the working of this operation is that the transport network graph is trimmed to a point where each route in a route set contains the minimum number of vertices required to maintain feasibility. Moreover, the highest cost per demand vertices are removed first, retaining the routes that are most beneficial in terms of both objectives. The aim of this operation is to find high-quality solutions to an UTRP instance that gives precedence to the TRT objective over the ATT objective.

The approach described above may be improved upon by evaluating all potential deletions of edges from a network, based on their cost per demand ratios, instead of only taking into account one route at a time. This is expected to lead to improved solution quality as a global view of the route set is taken, instead of focusing locally on one route at a time.

#### 4.2.6. Full cost-based grow

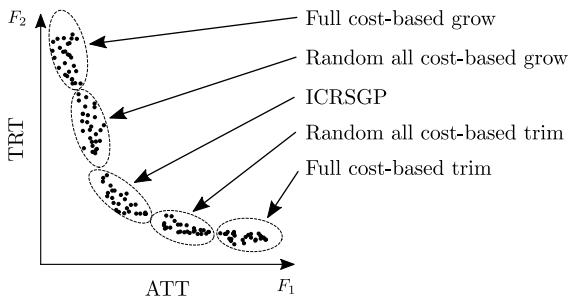
In order to achieve solutions that give precedence to the ATT objective over the TRT, a counterpart to the full cost-based trim operator is also proposed, called *full cost-based grow*. The operator involves applying the cost-based grow operation of Section 4.2.2 a maximum allowed number of times to a network or until a model constraint is violated. This is achieved by randomly stepping through each route  $R_i$  in a given route set  $\mathcal{R}$ , and applying the cost-based grow algorithm  $m_{\max} - |R_i|$  times to  $R_i$ , or until a model constraint is violated, upon which the next route is considered. The transit network graph will thus be densely packed with routes, giving precedence to adding vertices to terminals that would result in more demand being satisfied by direct routes, and hence lowering the ATT at the cost of the TRT.

This modification is computationally more expensive, but the quality of solutions returned is higher.

#### 4.2.7. An enhancement procedure

Our initial candidate route set enhancement procedure for generating initial feasible solutions is described in this section. The procedure is aimed at improving the spread of the initial solutions in objective space, as they typically tend to bunch together in the mid-portion of the attainable non-dominated set for larger UTRP instances. The procedure involves creating a pre-specified number of initial candidate solutions by invoking the ICRSGP described in Section 4.1. For each solution generated according to the ICRSGP, the application of the four enhancement operators (full cost-based grow, random all cost-based grow, random all cost-based trim, and full cost-based trim) yields four additional solutions that may be used as initial candidate route sets. When the full cost-based grow, random all cost-based grow, random all cost-based trim, or full cost-based trim operators were performed on an initial solution generated by the ICRSGP, the respective changes in the ATT and TRT objectives were observed to be somewhat correlated with those of the initial candidate solution used.

The full cost-based grow operator typically yields solutions with the smallest ATT objective function values, and the largest TRT objective function values, whereas the full cost-based trim operator typically yields solutions with the smallest TRT objective function values, and the largest ATT objective function values. Moreover, the solutions returned by the random all cost-based grow operator are typically between those returned by the full cost-based grow operator and those returned by the ICRSGP in objective space. Similarly, the solutions



**Fig. 2.** Results typically returned by the initial candidate route set enhancement procedure, where the initial candidate solutions are generated by invoking an ICRSGP, and then enhanced by applying the full cost-based grow, random all cost-based grow, random all cost-based trim, and full cost-based trim operators to each solution generated by the ICRSGP. The horizontal axis measures the ATT objective function, and the vertical axis the TRT objective function. The figure illustrates the areas of the Pareto front typically covered by each procedure.

returned by the random all cost-based trim operator are typically between those returned by the ICRSGP and those returned by the full cost-based trim operator in objective space. When generating numerous initial candidate route sets according to this approach, it was noted that the solution qualities and spread one might expect from this approach is as shown in Fig. 2. For all numerical results reported later in this paper, the number of initial feasible candidate route sets to be produced by the ICRSGP was set to 2000. An additional 8000 candidate solutions were therefore generated by the enhancement procedure described in this section, yielding a total of 10 000 solutions for each UTRP instance considered. A non-dominated set containing a pre-specified number of solutions could then be extracted from these 10 000 initial feasible solutions based on the *fast non-dominated sorting algorithm* (FNSA) (Srinivas and Deb, 1994) and by applying the crowding distance operator described in Deb et al. (2002).

#### 4.3. ICRSGP results

The ICRSGP was applied to all five UTRP benchmark instances mentioned in Section 2, generating 2000 initial solutions in each case. The ICRSGP enhancement procedure of Section 4.2.7 was then applied to these 2000 solutions for each benchmark instance, and so 8000 new solutions were obtained in each case.

The FNSA (Srinivas and Deb, 1994) was applied to the 10 000 initial solutions, the crowding distance operator (Deb et al., 2002) was subsequently applied, and the best required number of solutions was selected thereafter, based on crowding distance. For visualisation purposes of the quality of these initial solutions, 200 were selected. These initial solutions, along with the results reported by other researchers, are presented in Fig. 3 for the five benchmark instances mentioned in Section 2. The blue circles and the green triangles represent the initial solutions returned by the ICRSGP and the enhanced ICRSGP, respectively, in objective space. It is interesting to note how the initial solutions returned by the ICRSGP become all the more concentrated in a smaller area in the objective space as the problem instance size increases. The subfigures of Fig. 3 are conveniently presented in order of increasing instance size. It is also evident how the enhanced ICRSGP spreads out the concentrated initial solutions returned by the ICRSGP over wider regions of the attainment fronts reported by other researchers, indicating that a good solution spread has been achieved. The high quality of the initial solutions returned by the enhanced ICRSGP is also clear as the attainment fronts of the initial solutions are in close proximity to the final attainment fronts reported by John (2016), which were, at the time of writing, the best reported bi-objective non-dominated attainment fronts reported in the literature for the five benchmark instances.

## 5. Model solution methodology

As mentioned, the UTRP solution methodology proposed by Fan and Machemehl (2004, 2006b) consists of three main components, namely an ICRSGP, a *network analysis procedure* (NAP), and a metaheuristic solution search procedure which combines the ICRSGP and NAP. This is also the solution methodology adopted in the current paper, with the ICRSGP already discussed in Section 4 and the NAP being the same as that of Mumford (2013), John et al. (2014) and Ahmed et al. (2019). We employ algorithms in the two main classes of metaheuristics, (population-based) evolutionary algorithms and (trajectory-based) neighbourhood searches, in this paper due to the complexity of the UTRP. Zhao and Gan (2003), Zhao and Zeng (2006), and Fan and Machemehl (2006b) have claimed that simulated annealing is the best-suited solution approach for the UTRP. They found that simulated annealing outperformed a genetic algorithm with respect to the quality of route sets achieved. Simulated annealing was selected as the solution method within the class of trajectory-based metaheuristics implemented in this paper due to its simplicity and the high-quality results that it reportedly achieves, providing near-optimal results within very reasonable time frames (Fan and Machemehl, 2004).

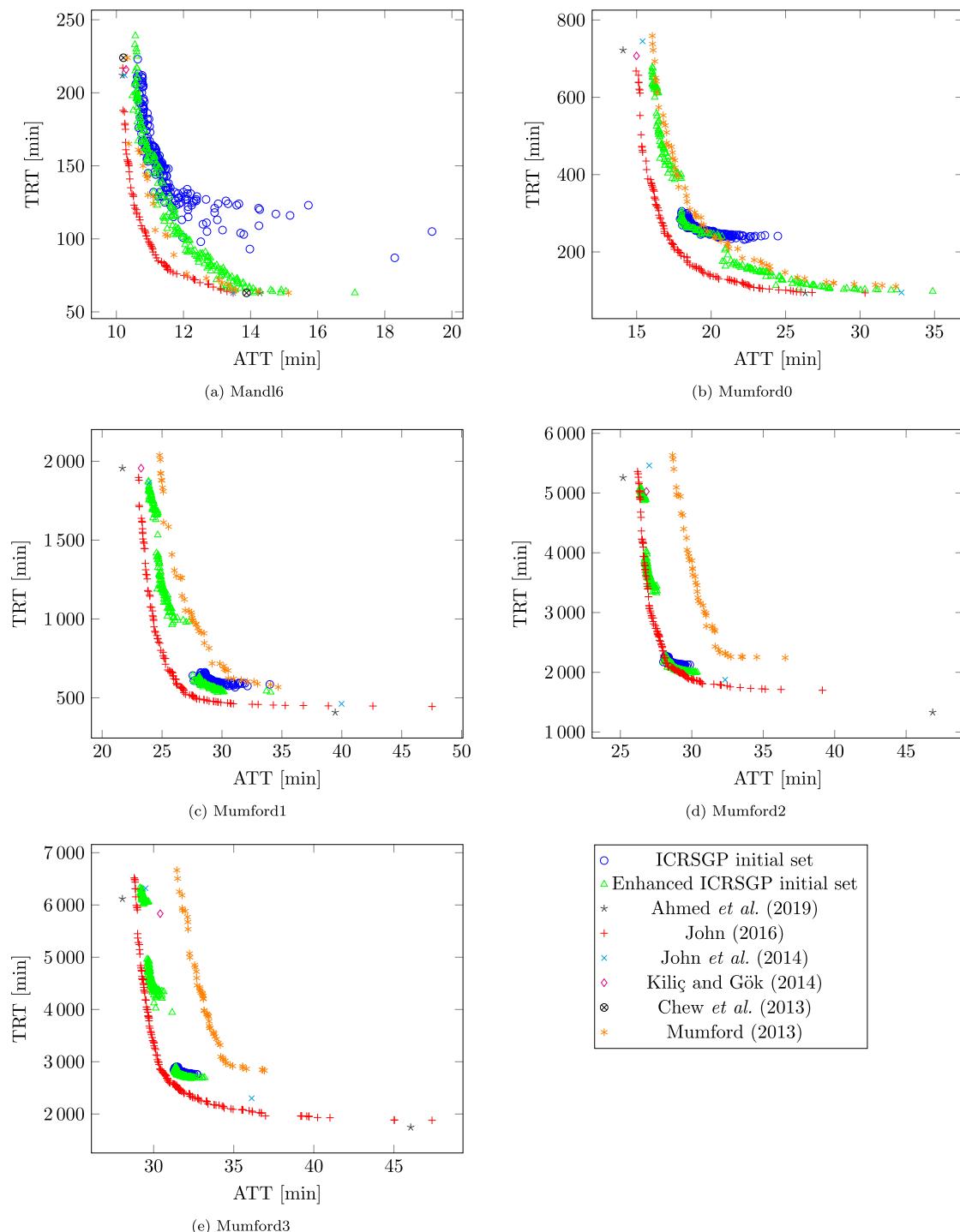
Population-based approaches are, however, more commonly adopted for solving instances of the UTRP than trajectory-based approaches, with genetic algorithms most frequently applied as solution methodology for the UTRP, often leading to very high-quality solutions (Ibarra-Rojas et al., 2015; Mumford, 2013; John, 2016). As a result, a genetic algorithm was chosen as solution approach for the UTRP within the class of population-based metaheuristics in this paper, thus paving the way for a comparison of the results returned by the method of simulated annealing and a genetic algorithm. Furthermore, the combination of such a trajectory-based approach and population-based approach may potentially lead to higher-quality solutions than are achievable when applying only one of these metaheuristics individually (Zhao and Zeng, 2006; Fan et al., 2008b).

The specific variant of the method of simulated annealing adopted in this paper is the *dominance-based multi-objective simulated annealing* (DBMOSA) (Smith et al., 2008), while that for the genetic algorithm is the NSGA II (Deb et al., 2002). Fig. 4 outlines the overall experimental approach followed in this paper, consisting of four main steps. The first step, being the ICRSGP, was discussed in Section 4, whereas Steps 2 to 4 were discussed in Section 6. The current section, Section 5, is devoted to a discussion on the details of the UTRP model solution methodologies that were adopted in Steps 2 to 4 of Fig. 4. Each methodology is described in detail in this section in the context of the UTRP. The section opens in Section 5.1 with a brief discussion on the *lower level heuristics* (LLHs) adopted during our solution approach. The hyper-heuristics employed to manage these LLHs within the aforementioned metaheuristic frameworks are discussed in Section 5.2, after which the section closes with an in-depth discussion in Sections 5.3 and 5.4 on how the entire model solution methodology was implemented in the cases of DBMOSA and the NSGA II, respectively.

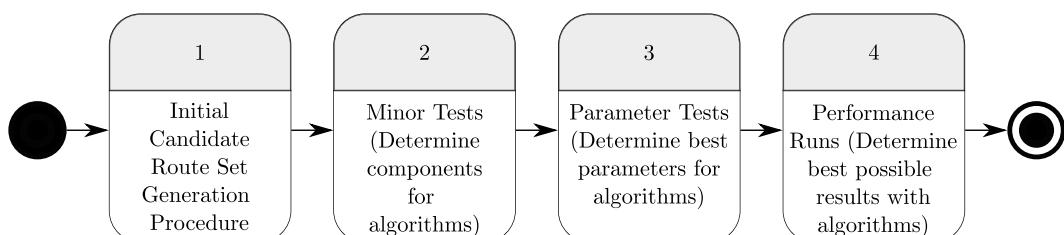
#### 5.1. Lower-level heuristics as move operators

We incorporated ten LLHs into our DBMOSA and NSGA II solution implementations when solving instances of the UTRP. Eight of these LLHs were taken from the literature, and were chosen based on their reported success and ease of implementation. The other two LLHs are novel contributions of this paper. This section is devoted to a description of the various LLHs applied, as summarised in Table 3. Fig. 5 contains illustrations of the working of these LLHs. The term LLH is used interchangeably with the terms perturbation and mutation in the descriptions of this section, depending on the context.

The first four operators represent the simplest form of adding or deleting operator types. The *add vertex* (a) and *delete vertex* (b) operators were originally introduced by Fan and Mumford (2010). They



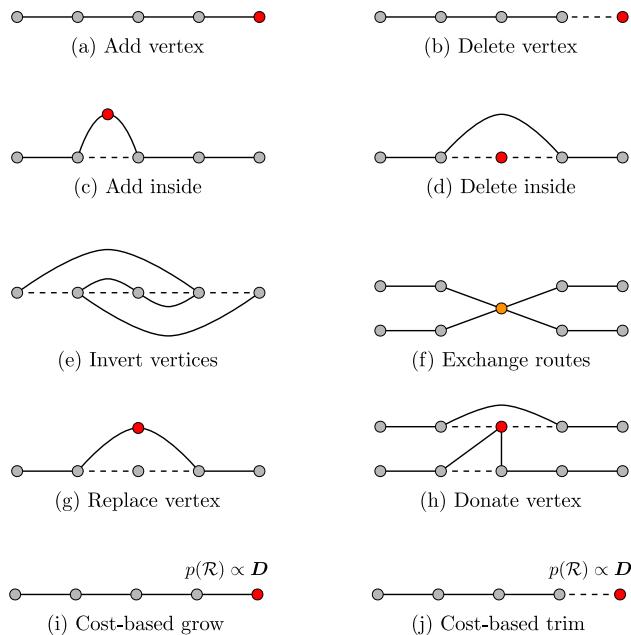
**Fig. 3.** Initial pre-optimisation solution sets returned by the ICRSGP and the enhanced ICRSGP, together with the final post-optimisation results reported by various researchers, all pertaining to the UTRP benchmark instances mentioned in Section 2.



**Fig. 4.** The overall experimental approach followed in this paper.

**Table 3**  
The LLHs considered in this paper.

Letter	LLH name	Source	Type
a	Add vertex	Fan and Mumford (2010)	Add
b	Delete vertex	Fan and Mumford (2010)	Delete
c	Add inside vertex	Ahmed et al. (2019)	Add
d	Delete inside vertex	Ahmed et al. (2019)	Delete
e	Invert vertices	Ahmed et al. (2019)	Rearrange
f	Exchange routes	Mandl (1979)	Rearrange
g	Replace vertex	Ahmed et al. (2019)	Rearrange
h	Donate vertex	Ahmed et al. (2019)	Inter route rearrange
i	Cost-based grow	This paper	Add
j	Cost-based trim	This paper	Delete



**Fig. 5.** LLH operators (a)–(h) proposed in the literature and LLH operators (i)–(j) introduced in this paper for solving instances of the UTRP. The solid lines represent the resulting edges in a route, while the dashed lines represent edges that have been deleted after having applied the LLH to the route set. The grey vertices represent stops in a route that have not been altered, red vertices represent either stops that have been included or deleted, based on the context of the adjacent edges, and the orange vertices represent special operations performed, such as exchanging or merging route segments. The curves represent changes in edges that have occurred after having applied the LLH operator. The notation  $p(R) \propto D$  represents an operator in which a probability is associated with considering an edge. This probability is proportional to travel demand.

entail randomly adding an adjacent vertex to a terminal vertex of a randomly selected route in a route set, and randomly deleting a terminal vertex from a randomly selected route, respectively. The *add inside vertex* (c) and *delete inside vertex* (d) operators, proposed by Ahmed et al. (2019), involve randomly adding or deleting a non-terminal vertex from a randomly selected route in a route set.

The next four operators involve rearranging vertices within routes in some way. The first (e), third (g) and fourth (h) of these operators were proposed by Ahmed et al. (2019), while the third (f) was introduced by Mandl (1979). The *invert vertices* (e) operator inverts the vertices of a randomly selected route portion in a chosen route set. The *exchange operator* (f) identifies a random common vertex in two randomly chosen routes, and then splits each of these routes at the common vertex, recombining a segment of the first route with a segment of the second route. This recombination is performed randomly. There are two possible outcomes for this recombination procedure. If, however, both changes lead to infeasible routes, an attempt is made to repair the route set, and if unsuccessful, the exchange procedure is re-applied to

the same offspring until all route combinations have been exhausted and repairs attempted. If no feasible change is found, the operator is abandoned and no change is performed on the original offspring. The *replace vertex* (g) operator selects one random vertex from a randomly selected route of a route set, and replaces it with another vertex. The *donate vertex* (h) operator randomly selects a vertex from a randomly selected route, removes it from its current route, and then inserts it into a random location in another randomly selected route.

For all of the above operators, additional checks are implemented so that each operator takes as input a route set, and efficiently searches through all possibilities of changing the route set so that only feasible routes are returned. Ahmed et al. (2019) allowed the operators to return infeasible solutions and penalised those solutions, but the approach adopted here is aimed at maintaining feasibility.

The final two operators are new operators proposed for use within the context of the UTRP. The *cost-based grow* (i) and *cost-based trim* (j) operators were described in some detail in Section 4.2, and are illustrated graphically in Fig. 5(i) and (j).

## 5.2. Hyperheuristic approaches towards managing LLHs

When many LLHs are available for use, it is often difficult to determine which LLHs are anticipated to be most beneficial and contribute most significantly towards finding high-quality solutions, as well as the extent to which these LLHs aid in quality improvement. The need therefore arises to design methods for managing these LLHs effectively during the UTRP solution procedure. The method we adopt was inspired by a hyperheuristic proposed by Vrugt and Robinson (2007). To the best of our knowledge, only one other hyperheuristic approach has been adopted in the literature within the context of the UTRP, namely by Ahmed et al. (2019), as also confirmed by Soares (2020). This section is devoted to discussions on the methods incorporated in this paper to manage LLHs, first for trajectory-based searches, and then for population-based searches.

### 5.2.1. Trajectory-based perturbation management

The idea is that a set of LLHs is provided as perturbation mechanisms during the trajectory-based search, and that an overarching hyperheuristic dynamically updates the probabilities according to which LLHs are applied with a view to guide the search towards finding high-quality solutions. The selection probabilities are updated based on the success ratios achieved by the LLHs during the previous search iterations (a temporal record is kept of each LLH's performance). This equips the trajectory-based search with a robust search mechanism, prioritising perturbations that have previously led to the inclusion of solutions in the non-dominated set. Each LLH starts off with an equal probability of being chosen.

As mentioned by Burke et al. (2013), all score-based heuristic selection techniques comprise five main component implementations, namely initial scoring, memory length adjustment, a strategy for heuristic selection based on the scores, a score update rule accounting aimed at accommodating improvements, and a score update rule for worsening solutions.

In our implementation, a roulette-wheel selection strategy associates a probability of selection with each LLH, whereas a max strategy involves deterministically selecting the LLH that performs the best. The initial LLH scores are set to  $\lceil (1/I) \times 100 \rceil$ , where  $I$  denotes the total number of LLHs considered during the search, and the memory length is set to encompass the entire search history. The score update rule for improvement entails increasing the score of the relevant LLH by one if the LLH applied, in fact, leads to the solution's inclusion in the non-dominated set (and if the total score before the perturbation is less than 100). The score update rule for worsening solution quality involves decreasing the score of the relevant LLH by one if the solution is rejected (and if the score before the perturbation is larger than 1). The probabilities for the roulette-wheel selection are calculated by dividing the score of LLH  $i$  by the total score of all the LLHs in the set.

### 5.2.2. Population-based mutation management

Multiple candidate solutions are maintained simultaneously during a population-based search. Excessive random application of a variety of LLH mutations during a population-based search may lead to a random search during which little exploitation is achieved. Since population-based algorithms are intrinsically dynamic and adaptive processes, the adoption of search parameters that are static stand in stark contrast to such a dynamic search paradigm. It has, in fact, been demonstrated both empirically and theoretically that, at different stages of an evolutionary process, different parameter values might be optimal (Eiben et al., 1999).

As a consequence, a mutation management strategy for population-based searches is proposed for use in the context of the UTRP based on the AMALGAM method (Vrugt and Robinson, 2007). The approach closely follows that of the AMALGAM strategy for trajectory-based searches, with the exception that now an entire population's LLH success scores and LLH application totals are maintained, for each LLH  $i \in \{1, \dots, I\}$ , one generation at a time. In terms of initial scoring, a selection probability of  $1/I$  is assigned to each LLH. The memory length is taken as one generation, and the strategy for LLH selection is based on selecting an LLH randomly according to updated selection probabilities  $p_i$  for each LLH  $i \in \{1, \dots, I\}$ .

The AMALGAM mutation strategy for population-based searches is described in pseudo-code form in Algorithm 8 in the Appendix. The selection probability  $p_i$  for each LLH  $i \in \{1, \dots, I\}$ , along with a selection minimum probability threshold, is provided as input. Moreover, the success score  $q_i$  achieved and the total number of applications  $t_i$  implemented during the previous generation are also provided as input to the algorithm for each LLH  $i \in \{1, \dots, I\}$ . The success score  $q_i$  is the total number of solutions perturbed according to LLH  $i$  that were included in the non-dominated set. The algorithm starts out by testing whether the sum of all the success scores is non-zero. If this is the case, a success proportion  $s_i$  is calculated and the selection probability  $p_i$  is updated accordingly for each LLH  $i \in \{1, \dots, I\}$  in Lines 2–4. The AMALGAM expression is formulated in the first term on the right-hand side of the expression in Line 3. The second term on the right-hand side of the expression in Line 3 takes into account a selection minimum probability threshold. The selection probabilities are updated in Line 4, where the success proportion  $s_i$  is added to the selection probability threshold.

### 5.3. Simulated annealing implementation

An overall process diagram for our DBMOSA solution implementation is provided in Fig. 6 as a guideline to assist the reader during the discussion in this subsection. A pseudo-code description of this procedure may also be found in Algorithm 1. Our simulated annealing implementation for solving instances of the UTRP incorporates the ICRSGP and the NAP described above, and guides a candidate solution generation procedure embedded in an iterated search framework (Fan and Machemehl, 2006b).

Although the literature contains various suggestions for values of the simulated annealing parameters, Dréo et al. (2006) noted that in the absence of general theoretical results, empirical adjustments are typically required by the analyst in respect of these parameter values. These adjustments become all the more time consuming as the problem complexity increases, and may further be complicated by increased sensitivity of the results. Therefore, the choice of parameter values was first based on suggestions in the literature, and thereafter adjusted empirically for improved results. The method of determining these values is described in more detail in the following section.

Algorithm 1 takes as inputs the UTRP instance parameters  $W$ ,  $D$ ,  $m_{\min}$ ,  $m_{\max}$  and  $r$  described in Section 3. In addition, the required number  $L_{\max}$  of iterations per epoch is an input parameter to the algorithm, initially taken as  $L_{\max} = 100$ , as suggested by Fan and Mumford (2010), but later adjusted to 250 upon empirical experimentation. Throughout the simulated annealing search process, the temperature is progressively lowered according to a pre-specified cooling schedule. Occasional reheating is also incorporated so as to avoid search entrapment at local optima. The well-known geometric schedule is employed for both cooling and reheating. The initial temperature calculation suggested by Fan and Mumford (2010) was originally adopted, but after some experimentation seemed to deliver initial temperatures that performed poorly. The approach was therefore adopted of determining an acceptable range of initial temperatures, and then conducting experimental tests in which the average *hypervolumes* (HVs) of the non-dominated sets returned were compared while varying the initial temperature.

The cooling parameter  $\beta$  of the geometric cooling schedule, specifying that the temperature during search epoch  $c$  should be  $T_c = \beta^c T_0$ , was computed by means of the method suggested by Fan and Machemehl (2006b), but also did not deliver sufficiently high-quality results. It was therefore again decided to determine the appropriate cooling rate  $\beta$  by means of an empirical analysis carried out on a test instance-specific basis. After experimentation, the reheating parameter  $\zeta$  of the geometric reheating schedule was chosen as 1.05.

A number of other parameters which control the simulated annealing search process are also taken as algorithmic input. These include the minimum number  $B_{\min}$  of accepted moves allowed per epoch, the maximum number  $H_{\max}$  of reheatings allowed, the maximum number  $A_{\max}$  of attempts allowed per epoch, the maximum number  $C_{\max}$  of epochs that may pass without inserting any new solutions into the archive, and the maximum number  $K_{\max}$  of epochs allowed during the entire search.

A number of initialisation steps are performed in Lines 1–4 of Algorithm 1. More specifically, an initial solution is generated in Line 1 (Process 1 in Fig. 6, as described in Section 4). Lines 2–3 correspond with Process 2 in Fig. 6, where the archive  $\mathcal{A}$  to be maintained throughout the search is initialised to contain this initial solution, the epoch counter  $c$  is initialised to zero, and so is the number  $\epsilon$  of epochs that have elapsed without having accepted a candidate solution. Further initialisations involve setting values for the selection probability  $p_i$  and the corresponding success score  $q_i$  for each LLH  $i \in \{1, \dots, I\}$ . The selection probabilities and success scores are maintained and stored in database D2 in Fig. 6 during the execution of the algorithm, and all of the LLHs chosen to be employed are stored for further reference in database D3. In Line 6, counters are initialised which are used to keep track of certain parameters during the search, per epoch. The number  $a$  of attempts, the number  $b$  of acceptances and the number  $c$  of iterations per epoch are each initialised to zero at the start of every epoch. Furthermore, a boolean flag is included in Line 7, called *poor\_epoch\_flag*. This flag is set to 1 in order to indicate that no solutions have been inserted into the archive during the current epoch, and is only lowered to zero once a solution has been inserted into the archive. Later, in Lines 32–33 of Algorithm 1, this mechanism is used to determine whether an epoch should be classified as a poor epoch, in which case the poor epoch counter  $\epsilon$  is incremented accordingly.

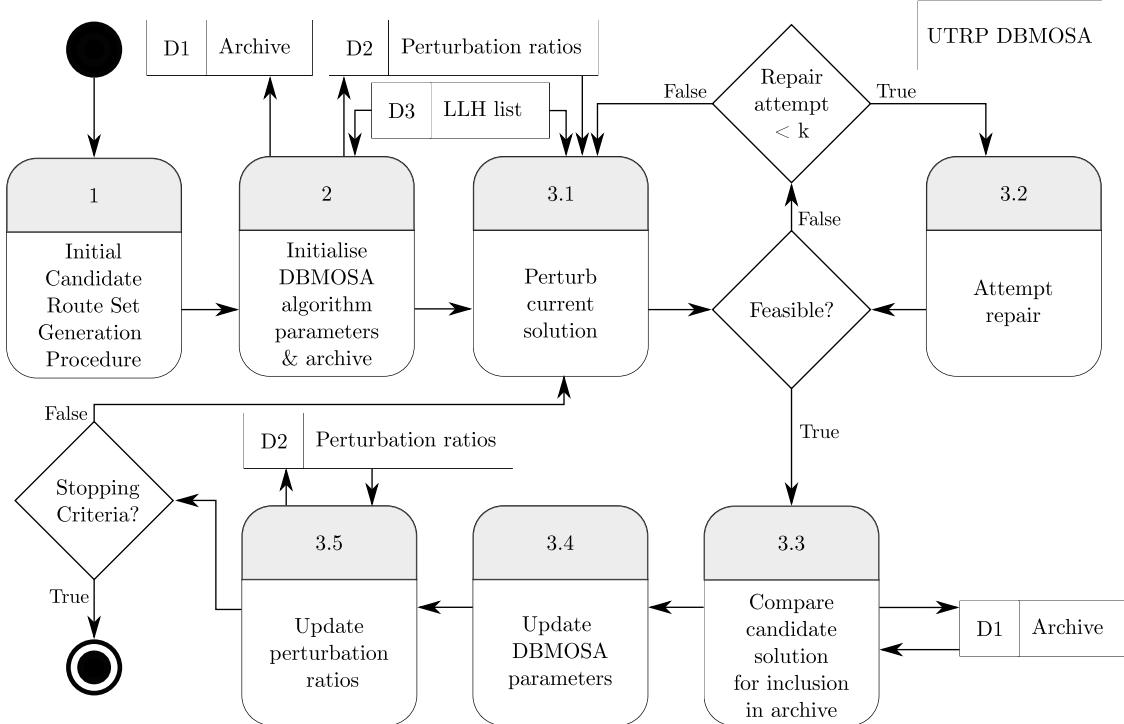


Fig. 6. The overall process diagram for the DBMOSA UTRP solution implementation employed in this paper.

Processes 3.1 to 3.5 in Fig. 6 represent the five main processes included in the overall while loop (Lines 5–33) of the DBMOSA algorithm (Algorithm 1).

The generation of a neighbouring solution takes place in Line 9 of Algorithm 1, represented by Process 3.1 in Fig. 6. The method of generating this solution is governed by applying one of the LLHs (stored in database D3 of Fig. 6) provided as input to the algorithm, randomly chosen based on the selection probabilities  $p_1, \dots, p_I$  (stored in database D2 of Fig. 6). Checks are put in place to ensure that the solutions generated remain feasible, and when an infeasible solution is encountered, a repair attempt is made by adding vertices to terminal vertices of routes, if possible (Process 3.2 in Fig. 6). If no feasible solution is generated,  $k$  attempts are allowed to repair the perturbed solution; otherwise, the incumbent solution is perturbed again (Process 3.1 in Fig. 6) and the entire process repeated until a feasible neighbouring solution has been found. After extensive empirical experimentation, we found that the value of  $k = 10$  provides a good trade-off between generating successful repairs and incurring additional computation time.

The probability of acceptance of a worsening neighbouring solution  $\mathcal{R}$  is governed by Line 11 of Algorithm 1 according to the well-known Metropolis rule

$$P(\mathcal{R}') = \min \left\{ 1, \exp \left( -\frac{\Delta_E(\mathcal{R}', \mathcal{R})}{T_c} \right) \right\}, \quad (9)$$

where  $\Delta_E(\mathcal{R}', \mathcal{R})$  denotes the energy function which evaluates the energy difference between the current and neighbouring solutions  $\mathcal{R}$  and  $\mathcal{R}'$ , and where  $T_c$  denotes the temperature of the search during epoch  $c$  (Kirkpatrick et al., 1983). The random number generated in Line 10 is compared with the probability of acceptance function value, and if the random number is smaller than the probability of acceptance, the neighbouring solution is accepted as the new solution. A neighbouring solution achieves a lower energy value than the current solution when it is dominated by fewer elements of the estimated Pareto front than the current solution. In this case, the neighbouring solution is accepted with probability 1 as the new current solution since it is interpreted as representing an improving move. When, however, the energy difference between the two solutions is large and positive, and the temperature is

low, the probability of accepting the neighbouring solution is small. If a move is accepted, the neighbouring solution becomes the new current solution during the next iteration of the search (Smith et al., 2008), and the number  $b$  of accepted moves per epoch is incremented by 1.

Lines 14–21 of Algorithm 1 govern the handling of a non-dominated solution, where Line 14 specifically represents the process of counting the number of solutions dominating the current solution. If  $|\mathcal{A}_R| = 0$ , then the current solution is non-dominated and should be included in the archive  $\mathcal{A}$  (Process 3.3 in Fig. 6). When a new solution is inserted into the archive, the number of accepted moves  $b$  is decremented by 1 in order to compensate for the fact that the current move is not accepted as a non-improving move, but instead as an improving move. Furthermore, the *poor\_epoch\_flag* is lowered by setting it equal to zero and the poor epoch counter  $e$  is also reset to zero. Lines 19–21 affect the removal of those solutions from the archive that are dominated by the current solution  $\mathcal{R}$ .

If, however, the current solution is not accepted in Line 11, the else if statement is invoked in Line 22, testing whether the maximum number of reheatings  $H_{\max}$  has been reached. If it has not been reached, the attempts per epoch counter  $a$  is incremented, and if  $a$  is larger than  $A_{\max}$ , then the system temperature is updated by reheating the temperature by a factor  $\zeta$ , upon which the total number of reheatings  $h$  is incremented (all the DBMOSA algorithmic parameter updates are encapsulated in Process 3.4 in Fig. 6). Moreover, the Break statement breaks out of the current while loop in Line 27, thereby entering a new epoch. The selection probabilities  $p_1, \dots, p_I$  and success scores  $q_1, \dots, q_I$  are updated during each iteration according to the LLH management strategy update rules (Process 3.5 in Fig. 6). Any one of the three LLH management strategies defined in Section 5.2.1 may be employed for this purpose.

The stopping criteria for the inner while-loop are found in Line 8 of Algorithm 1. This loop controls the temperature decrease per epoch and the criteria state that as long as the number of accepted moves per epoch  $b$  is smaller than the minimum allowed number of accepted moves per epoch  $B_{\min}$  and the iteration counter  $t$  is smaller than the allowable number  $L_{\max}$  of iterations per epoch, new neighbouring solutions should continue to be generated and tested for inclusion in

**Algorithm 1:** DBMOSA for the UTRP

---

**Input :** An  $n \times n$  weight matrix  $\mathbf{W}$  representing the graph  $G$ , an  $n \times n$  OD demand matrix  $\mathbf{D}$ , the minimum allowed number of vertices in a route  $m_{\min}$ , the maximum allowed number of vertices in a route  $m_{\max}$ , the maximum allowable number of iterations per temperature  $L_{\max}$ , the initial temperature  $T_0$ , the minimum accepted number of moves per epoch  $B_{\min}$ , the maximum number of reheatings allowed  $H_{\max}$ , the maximum number of attempts allowed  $A_{\max}$  per epoch, the maximum number of epochs that may pass without inserting any new solutions into the archive  $C_{\max}$ , the maximum number of epochs  $K_{\max}$ , and a set of LLHs.

**Output:** A set of non-dominated solutions  $\mathcal{P}_S$  in solution space and each solution's objective function values as a vector  $P_O$  in objective space.

```

1 Generate an initial feasible route set  $\mathcal{R}$  containing  $r$  routes
2 Initialise the archive  $\mathcal{A} \leftarrow \{\mathcal{R}\}$ 
3 Initialise the counters  $c \leftarrow 0$  and  $\epsilon \leftarrow 0$ 
4 Initialise  $p_i$  and  $q_i$  according to the LLH management strategy for each LLH  $i$ 
5 while  $\epsilon \leq C_{\max}$  and  $c \leq K_{\max}$  do
6    $a, b, t \leftarrow 0$ 
7    $poor\_epoch\_flag \leftarrow 1$ 
8   while  $b < B_{\min}$  and  $t < L_{\max}$  do
9     Generate a neighbour  $\mathcal{R}'$  from the current solution  $\mathcal{R}$  using LLH  $i$  based on the probability  $p_i$ 
10    Generate a random number  $p \in (0, 1)$ 
11    if  $p < \min\{1, \exp(-\Delta_E(\mathcal{R}', \mathcal{R})/T_c)\}$  then
12       $\mathcal{R} \leftarrow \mathcal{R}'$ 
13       $b \leftarrow b + 1$ 
14      if  $|\mathcal{A}_{\mathcal{R}}| = 0$  then
15         $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathcal{R}\}$ 
16         $b \leftarrow b - 1$ 
17         $poor\_epoch\_flag \leftarrow 0$ 
18         $\epsilon \leftarrow 0$ 
19        for  $Q \in \mathcal{A}$  do
20          if  $\mathcal{R} < Q$  then
21             $\mathcal{A} \leftarrow \mathcal{A} \setminus \{Q\}$ 
22    else if  $h < H_{\max}$  then
23       $a \leftarrow a + 1$ 
24      if  $a > A_{\max}$  then
25        Update system temperature  $T_c \leftarrow \zeta T_c$  by reheating
26         $h \leftarrow h + 1$ 
27        Break
28    Update  $p_i$  and  $q_i$  values based on LLH management strategy update rules
29     $t \leftarrow t + 1$ 
30   $c \leftarrow c + 1$ 
31  Update system temperature  $T_c \leftarrow \beta T_{c-1}$  by cooling
32  if  $poor\_epoch\_flag = 1$  then
33     $\epsilon \leftarrow \epsilon + 1$ 
34  $\mathcal{P}_S \leftarrow \mathcal{A}$ 
35  $P_O \leftarrow$  evaluate the objective function for each solution  $\mathcal{R}$  in  $\mathcal{P}_S$ 
36 return  $[\mathcal{P}_S, P_O]$ 
```

---

the archive. The value of  $B_{\min}$  is chosen as 25 in this paper, based on empirical experimentation.

The overall stopping criteria are found in Line 5 of Algorithm 1 and are met when the maximum number of epochs  $C_{\max}$  that may elapse without inserting any solution into the archive is reached, or when the overall maximum number of epochs  $K_{\max}$  is reached. A value of  $C_{\max} = 3$ , as proposed by Dréo et al. (2006), was initially considered, and then later adjusted to the value 400 upon empirical experimentation. The latter value is subsequently adopted in this paper. The maximum number of epochs is introduced to avoid a situation where the solutions are non-improving and computational resources are wasted. The value of  $K_{\max}$  is taken as 1 000 in this paper, as suggested by Fan and Mumford (2010).

When the set of non-dominated solutions  $\mathcal{P}_S$  is returned by Algorithm 1, further exploration is performed of the search space by restarting the simulated annealing algorithm, but instead of generating another initial feasible solution as described above, taking each solution

in turn from the current non-dominated set  $\mathcal{P}_S$  as the initial solution for the trajectory-based search (Smith et al., 2008).

#### 5.4. NSGA II implementation

The genetic algorithm employed in this paper to solve instances of the UTRP also incorporates the ICRSGP and NAP described above, and guides the population of solutions to converge towards a set of high-quality non-dominated solutions. Fig. 7 contains an overarching guideline accompanying the discussion on our NSGA II implementation in this section. A pseudo-code description of this procedure may also be found in Algorithm 2.

Algorithm 2 takes as inputs the UTRP instance parameters  $\mathbf{W}$ ,  $\mathbf{D}$ ,  $m_{\min}$ ,  $m_{\max}$  and  $r$  described in Section 3. In addition, the required number  $L_{\max}$  of generations is an input parameter to the algorithm,

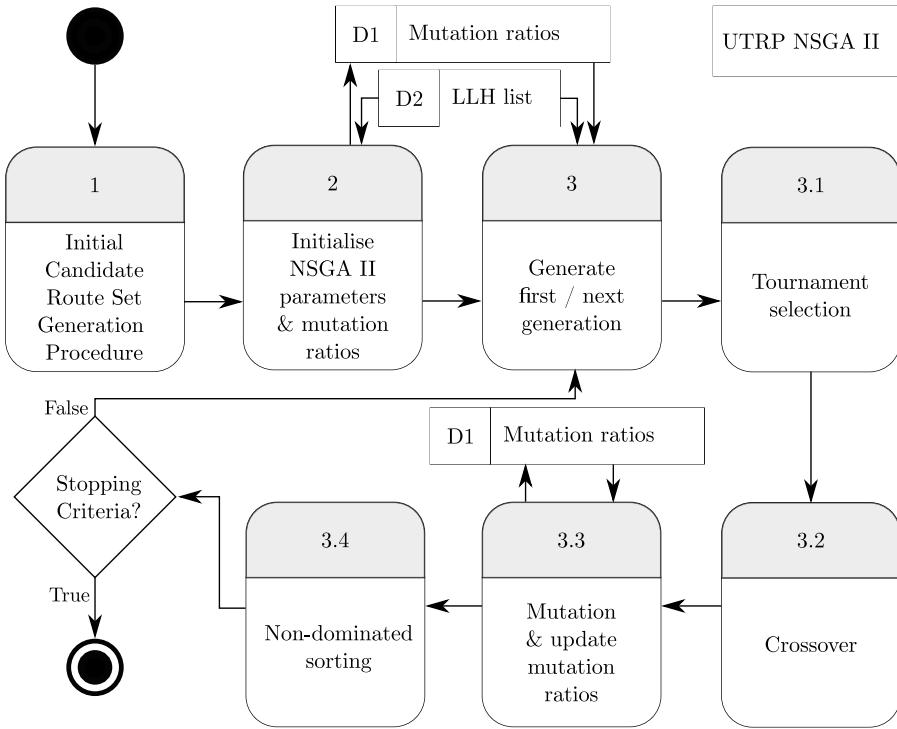


Fig. 7. The overall process diagram for the NSGA II solution implementation employed in this paper.

taken here as  $L_{\max} = 200$ , as suggested by John et al. (2014) and Mumford (2013). An initial population size  $N$  is also required as an input parameter, chosen here as  $N = 200$  (Mumford, 2013; John et al., 2014).

The algorithm is initialised by an initial population  $P_0$  of  $N = 200$  random feasible route sets  $\mathcal{R}$ , generated according to the ICRSGP, described in Section 4, and depicted as Process 1 in Fig. 7 and Line 1 of Algorithm 2. The algorithmic parameters of the NSGA II are initialised along with the mutation ratios (Database D1 in Fig. 7) based on the list of LLHs chosen for the specific NSGA II run (stored in database D2 in Fig. 7). Next, members of the initial population  $P_0$  are ranked and sorted by invoking the FNSA (Srinivas and Deb, 1994). The crowding distance of each solution is then determined by applying a crowding distance assignment algorithm (Deb et al., 2002) to  $P_0$ . A child population  $Q_0$  is generated (according to Process 3 in Fig. 7, containing four sub-processes numbered 3.1 to 3.4) and incorporating the aforementioned crowding distance values of  $P_0$ , for the selection of parents during binary tournament selection (Process 3.1 in Fig. 7). Child solutions are formed by applying crossover (Process 3.2 in Fig. 7) and mutation operators (Process 3.3 in Fig. 7) specific to the UTRP (John et al., 2014; Mumford, 2013).

The crossover operator adopted (Process 3.2 in Fig. 7) in this paper for the UTRP is that of Mumford (2013), but with the extension that after the procedure has been performed, a check takes place to test whether any subroutes of other routes exist in the route set, as this is never beneficial in the UTRP (because a subroute does not contribute towards improving the ATT objective function, yet it deteriorates the TRT objective function). Such routes are replaced by routes contributing towards the ATT objective function. The approach involves maximising the proportion of missing vertices included when crossover occurs between two parent routes. Any subroute of another route is replaced by a route in one of the parents, maintaining the qualities of the parents. The repair strategy mentioned in Section 4 is repeated for all missing vertices. If a route set cannot be repaired, the offspring is abandoned and the next two parents are selected for crossover. Offspring are created and added to  $Q_0$  in this fashion until  $|Q_0| = N$ .

The LLHs described in Section 5.1 are employed as mutation operators (Process 3.3 in Fig. 7). The probability that an offspring solution is mutated is specified by the decision maker, and remains constant throughout the search. For each offspring solution, a random number is generated in the unit interval, and if this number is smaller than the pre-specified mutation probability, a mutation operator is selected and applied to the respective offspring solution. This is achieved by generating another random number in the unit interval, upon which LLH  $i$  is selected if the random number falls within the cumulative selection probability interval of the respective LLH, based on the selection probability  $p_i$ , as described in Section 5.2.2. These selection probabilities (stored in Database D1 in Fig. 7) are updated after each generation according to the LLH management strategy update rule mentioned in Section 5.2.2, as may be seen in Line 24 of Algorithm 2 (also part of Process 3.3 in Fig. 7).

Next, the initial population  $P_0$  and the offspring population  $Q_0$  are merged, and subsequently ranked and sorted into non-dominated fronts (Process 3.4 in Fig. 7) according to the FNSA (Srinivas and Deb, 1994), pertaining to the objective functions (5) and (6). The next population  $P_1$  is then selected, based first on the non-dominated rank, and then on the crowding distance to break ties. This procedure is repeated iteratively (the repetition of Process 3 and sub-processes 3.1–3.4 in Fig. 7), until  $L_{\max}$  generations have elapsed, as described in Lines 8–25 of Algorithm 2. Finally, the non-dominated attainment set  $\mathcal{P}_S$  is extracted from all the populations  $P_0, \dots, P_{L_{\max}}$ , along with the objective function values  $P_O$  of this set, and returned as output at termination of the algorithm.

## 6. Results

In order to validate our implementations of the DBMOSA and NSGA II solution processes for the UTRP described in the previous section, the algorithms were applied to the five benchmark problems reviewed in Section 2. The overall fine-tuning of these algorithms consisted of three phases, as indicated in the last three steps of Fig. 4, namely minor tests (for determining appropriate subalgorithms and methods to include in the solution methodologies), parameter tests

**Algorithm 2:** NSGA II for the UTRP.

---

**Input :** An  $n \times n$  weight matrix  $\mathbf{W}$  representing the graph  $G$ , an  $n \times n$  demand matrix  $\mathbf{D}$ , the minimum allowed number of vertices in a route  $m_{\min}$ , the maximum allowed number of vertices in a route  $m_{\max}$ , the number of required routes  $r$ , the maximum number of generations  $L_{\max}$ , the population size  $N$ , and a set of LLHs.

**Output:** A set of non-dominated solutions  $\mathcal{P}_S$  in the solution space together with each solution's objective function values captured in a vector  $P_O$  in objective space.

- 1 Generate an initial population  $\mathbf{P}_0$  comprising  $N$  random route sets  $\mathcal{R}$ , each containing  $r$  routes, by means of the enhanced ICRSGP of Section 4.2
- 2 Initialise the selection probabilities  $p_1, \dots, p_I$  for the LLHs
- 3 Use the FNSA (Srinivas and Deb, 1994) to rank and sort  $\mathbf{P}_0$
- 4 Use a crowding distance assignment algorithm (Deb et al., 2002) to determine the crowding distance of each solution in  $\mathbf{P}_0$
- 5 Generate a child population  $\mathbf{Q}_0$  of size  $N$  from  $\mathbf{P}_0$ , by crossover and mutation, using binary tournament selection, based on the crowding distance operator
- 6 Repair any infeasible offspring; if irreparable, generate further offspring from the parents until  $|\mathbf{Q}_0| = N$
- 7  $t \leftarrow 0$
- 8 **while**  $t < L_{\max}$  **do**
- 9      $\mathbf{R}_t \leftarrow \mathbf{P}_t \cup \mathbf{Q}_t$
- 10    Rank and sort  $\mathbf{R}_t$  by means of the FNSA into non-dominated fronts  $\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_k$
- 11     $\mathbf{P}_{t+1} \leftarrow \emptyset$
- 12     $k \leftarrow 1$
- 13    **while**  $|\mathbf{P}_{t+1}| < N$  **do**
- 14      **if**  $|\mathcal{F}_k| + |\mathbf{P}_{t+1}| \leq N$  **then**
- 15         $\mathbf{P}_{t+1} \leftarrow \mathbf{P}_{t+1} \cup \mathcal{F}_k$
- 16      **else**
- 17        Determine the crowding distance for each solution in  $\mathcal{F}_k$
- 18        Sort the solutions in  $\mathcal{F}_k$  in descending order of crowding distance
- 19         $\mathbf{P}_{t+1} \leftarrow \mathbf{P}_{t+1} \cup [\text{top } (N - |\mathbf{P}_{t+1}|) \text{ solutions in the sorted set } \mathcal{F}_k]$
- 20       $k \leftarrow k + 1$
- 21    Determine the crowding distance for each solution in  $\mathbf{P}_{t+1}$
- 22    Generate a child population  $\mathbf{Q}_{t+1}$  from  $\mathbf{P}_{t+1}$  by crossover and mutation, using binary tournament selection based on the  $\prec_c$  crowding distance operator
- 23    Repair any infeasible offspring; if irreparable, generate further offspring from the parents until  $|\mathbf{Q}_{t+1}| = N$
- 24    Update the selection probabilities  $p_1, \dots, p_I$  according to the AMALGAM LLH management strategy for each LLH
- 25     $t \leftarrow t + 1$
- 26  $\mathcal{P}_S \leftarrow \text{get non-dominated set obtained over all populations } \mathbf{P}_0, \dots, \mathbf{P}_{L_{\max}}$
- 27  $P_O \leftarrow \text{return objective function values for each solution } \mathcal{R} \text{ in } \mathcal{P}_S$
- 28 **return**  $[\mathcal{P}_S, P_O]$

---

(for tuning the various metaheuristic hyper-parameters to appropriate values), and finally, performance runs (determining the best results that are attainable by the relevant solution methodology). All results reported here are available online.<sup>1</sup>

The minor tests (Step 2 of Fig. 4) were a set of smaller tests aimed at determining whether to choose or disregard the new components in the various algorithms, and were by no means comprehensive. Instead they provided a method for determining whether the different avenues are worth exploring further due to so many possibilities existing. After having completed the minor tests, decisions were made as to which algorithmic components to include, and then only did the parameter fine-tuning process start (Step 3 of Fig. 4). After having fine-tuned the parameters, performance runs were conducted to determine the best possible performance of the new algorithms (Step 4 of Fig. 4).

The performance metric chosen to evaluate the quality of solution sets in this section is HV, due to its incorporation of the four main Pareto front quality aspects, namely convergence, spread, uniformity and cardinality (Li and Yao, 2019). When adopting the HV performance indicator, it is important to choose the reference point appropriately. The reference points chosen for minimum and maximum ATT, as well as the minimum and maximum TRT, are shown in Table 4 for each benchmark UTRP instance.

**Table 4**  
UTRP benchmark instance HV reference points adopted, all measured in minutes.

Instance	Minimum ATT	Maximum ATT	Minimum TRT	Maximum TRT
Mandl6	10	15	63	224
Mumford0	13	32	94	700
Mumford1	19	50	345	2 000
Mumford2	22	45	864	6 000
Mumford3	24	50	982	6 600

The values in Table 4 were used to measure the normalised HV for a given set of non-dominated solutions. The lower bounds on the TRT in Table 4 were calculated by determining minimum spanning trees for the transport networks, since the weights of these trees represent theoretical lower bounds on the TRT that operators would be able to achieve. These lower bounds on the TRT were calculated by Mumford (2013), as tabulated in Table 2. The lower bounds on the ATT in the table were determined for the ideal case where all passengers have direct routes towards their destinations (Mumford, 2013). Mumford (2013) also calculated these values. The minimum reference points were therefore taken as the respective lower bounds on the ATT and the TRT. Furthermore, a comparison of the published results of Mumford (2013) and those of John (2016) was performed, observing the range achieved by their results. These bounds led to the upper bounds in Table 4 (as well as a confirmation of the lower bounds). These reference

<sup>1</sup> <https://github.com/GHuss7/UTRP-Improvements-Article-Results>.

points were also used to normalise the objective function values for each instance during execution of the NSGA II, as is appropriate for genetic algorithms. The results obtained for these instances are compared in this section with the non-dominated sets published by Mumford (2013) and by John (2016), as well as with the single-point results reported by various other researchers. Details of the various tests may be found in Hüttelmann (2022).

### 6.1. Minor tests

The idea behind the minor tests (Step 2 of Fig. 4) was to ascertain how to constitute the DBMOSA algorithm and the NSGA II in terms of components included or excluded so that additional parameter tests could be conducted (Step 3 of Fig. 4) with a view to fine-tune the algorithms. The three smallest test instances in Table 2, namely Mandl6, Mumford0 and Mumford1 were utilised for these tests due to the long runtimes of these problem instances, and the Mumford2 and Mumford3 instances requiring significantly longer runtimes.

When testing the potential components to add, fixed basic structures were assumed for the DBMOSA and NSGA II, and these were taken as baselines. A single variable being tested was varied at a time, so that its effect could be measured. The values of the test parameters were changed incrementally, and the differences in HV obtained were observed and compared. If an improvement in HV was observed for the three instances within a minor test, the component tested was subsequently included in the final DBMOSA and NSGA II implementations described in Sections 5.3 and 5.4, respectively. For the details of these minor tests, the reader is referred to the PhD dissertation of Hüttelmann (2022).

### 6.2. Parameter tests

This section is devoted to a discussion on the parameter tests (the third step of Fig. 4) carried out for the DBMOSA and the NSGA II solution implementations, individually. The DBMOSA algorithm presented in Section 5.3 requires specification of various input parameters that affect the algorithmic performance. These parameter values were identified when designing the algorithm for best performance. The literature recommends various values for these parameters which were adopted as a starting point. These values produced acceptable performance. When subsequently testing the suitability of a parameter value, the test parameter values were varied one at a time, while all other parameters were kept at their baseline values. The test parameters thus considered were the initial temperature, the maximum number of reheatings, the cooling rate, the reheating rate, the maximum number of iterations, the minimum number of accepts, the maximum number of accepts, and finally the maximum number of poor epochs. The four values, other than the baseline value, were chosen in such a way that they reflect acceptable minimum and maximum parameter values, upon which two parameter values were selected midway between the extremes and the baseline value.

The initial temperature, the maximum number of reheatings that may occur, the cooling rate and the reheating rate control the temperature of the DBMOSA. Each test utilised ten identical starting solutions generated by the enhanced ICRSGP to ensure consistent testing. The remaining parameters collectively achieve epoch control in the DBMOSA algorithm, namely the maximum number of iterations allowed per epoch, the maximum number of poor epochs allowed, the minimum number of acceptances required per epoch and the maximum number of attempts allowed per epoch.

A similar approach was followed in order to select parameter values for the NSGA II as that described for the DBMOSA. For the NSGA II, the relevant parameters are the crossover probability, the mutation probability, the population size, and the number of generations. Both Mumford (2013) and John (2016) adopted a crossover probability of 1 and employed a crossover operator proposed by Mumford (2013). After

**Table 5**

Average run times for the performance runs on an Intel® Core™ i7-4770 processor operating at 3.4 GHz, equipped with 16 GB of RAM and running in a Windows 10 operating system. The format of the times are hh:mm:ss.

Instance	DBMOSA (long run)	NSGA II (long run)	NSGA II (uncapped by number of iterations run)
Mandl6	00:03:55	00:12:02	00:14:04
Mumford0	00:06:07	00:48:02	00:44:12
Mumford1	00:35:26	03:12:58	04:52:04
Mumford2	05:26:59	33:30:10	55:16:11
Mumford3	07:52:56	54:53:04	76:50:30

extensive empirical experimentation, it was determined that 1 is not an appropriate crossover probability for the NSGA II of Section 5.4, but that a value of 0.6 provides better performance. Moreover, John allowed  $r$  mutation attempts for each offspring solution produced, with a mutation probability of  $\frac{1}{r}$  for each trial. He also incorporated eight different mutation operators, while the NSGA II of Section 5.4 incorporates one of ten mutation operators for each trial. The mutation probability for the NSGA II of Section 5.4 was set to 0.9 after having determined this value to be a good baseline value after initial testing over all the benchmark instances. The other parameter test values were chosen based on the same reasoning as for the DBMOSA parameter test values. It was decided to conduct these parameter tests on all five benchmark instances of Section 2 so that appropriate genetic algorithm parameters could be identified for each of the instances. The parameter tuning analysis involved performing ten algorithmic runs for each parameter test value combination, (the test values were varied for each parameter in turn, while keeping all the other parameters at their baseline values). Thereafter, box-plots were constructed to evaluate the algorithmic performance in terms of the HV values achieved by the ten runs.

### 6.3. Performance runs

The last step of Fig. 4 is elaborated upon in this section, which also contains a summary and discussion of the results obtained. In particular, Section 6.3.1 contains the HV measurements and graphical plots of the objective function values returned by the algorithms, Section 6.3.2 contains a comparison of our results with other benchmark results found in the literature, and Section 6.3.3 contains an analysis of the LLH selection probabilities over time.

#### 6.3.1. HV and graphical results

After having conducted the parameter tests and having evaluated their outcomes, the parameter values were selected in preparation for longer runs of the DBMOSA algorithm so that its performance could be measured against the results reported by other researchers. The same simulated annealing parameters identified to be best suited for the Mumford1 instance were also adopted for the Mumford2 and Mumford3 instances, as mentioned. A summary of all the final parameter values adopted in the DBMOSA algorithm are presented in Table 6. For the longer runs, thirty individual DBMOSA algorithmic runs were initialised, starting with thirty different initial solutions returned by the enhanced ICRSGP. The main differences between the shorter and longer runs were that the maximum number of iterations was set to 400 000 during the longer runs, and that the number of iterations for HV improvement comparison was set to 10 000 during these runs. This means that the algorithm took longer to terminate, thereby in effect allotting it more computation time. The average run times are presented for each instance, solution implementation and run type in Table 5.

After having selected the NSGA II parameters as described in Section 6.2 and tabulated in Table 7, 20 NSGA II runs of 2 000 generations each were conducted for each of the benchmark instances of Section 2, and the attainment sets over all these runs were computed (called the

**Table 6**

The parameter values adopted during the performance tests involving the DBMOSA algorithm when solving the five UTRP benchmark instances of Section 2, based on a sensitivity analysis.

Parameter	Mandl6	Mumford0	Mumford1/2/3
Initial solutions	Enhanced KSP-MMV-ICRSGP		
Perturbation management strategy	Stochastic roulette-wheel strategy		
Perturbation LLHs	{a, b, c, d, e, f}	{a, b, c, d, e, f, g, h}	{c, d, e, f, g, h, i, j}
Number of runs	30	30	30
Termination criterion	First to breach	First to breach	First to breach
Maximum total iterations	400 000	400 000	400 000
Iterations for HV improvement comparison	10 000	10 000	10 000
HV improvement threshold	0.005%	0.005%	0.005%
Initial temperature	0.1	0.001	0.0001
Maximum reheatings	10	5	3
Cooling rate	0.97	0.95	0.97
Reheating rate	1.5	1.1	1.05
Maximum iterations per epoch	500	50	50
Maximum poor epochs	600	800	600
Minimum accepts	10	10	10
Maximum attempts	50	50	50

**Table 7**

The parameters considered during the performance tests involving the NSGA II when solving the five UTRP benchmark instances of Section 2, based on a sensitivity analysis.

Parameter	Mandl6	Mumford0	Mumford1	Mumford2	Mumford3
Initial solutions	Enhanced KSP-MMV-ICRSGP				
Crossover strategy	RSD crossover				
Mutation management strategy	AMALGAM mutation management strategy				
Mutation LLHs	{a, b, c, d, e, f, g, h, i, j}				
Number of runs	20	20	20	20	20
Mutation selection threshold	3%	3%	3%	3%	3%
Termination criterion	First to breach				
Maximum total iterations	800 000	800 000	800 000	800 000	800 000
Generations for HV improvement comparison	40	40	40	40	40
HV improvement threshold	0.01%	0.01%	0.01%	0.01%	0.01%
Generations	2 000	2 000	2 000	2 000	2 000
Population size	400	400	400	400	400
Crossover probability	0.5	0.4	0.4	0.5	0.6
Mutation probability	1	1	1	1	1

long runs). The number of runs was set to 20 in order to allow for sufficient testing, as was also done by Mumford (2013) and by John et al. (2014). The uncapped runs mentioned in Table 5 refer to runs during which the termination criterion was removed which limited the runs to a certain number of iterations. The total number of iterations limit was initially imposed due to the long runtimes of the NSGA II induced by the largest instances, but it was observed that the performance of the NSGA II was limited due to reaching the maximum number of iterations termination criterion while it was still improving solutions. These uncapped runs were only applicable to the NSGA II, because the DBMOSA usually terminated within reasonable run times (without reaching the maximum number of iterations termination criterion during the longer runs). As a result, the NSGA II was allowed to run without a cap on the number of iterations over the remainder of the time available. The performance of the NSGA II was thus pushed to its limits, only allowing it to terminate once the HV stopped improving over a pre-defined number of iterations. In this way, the run time of the largest instance (Mumford3) increased by about 22 h per run, on average.

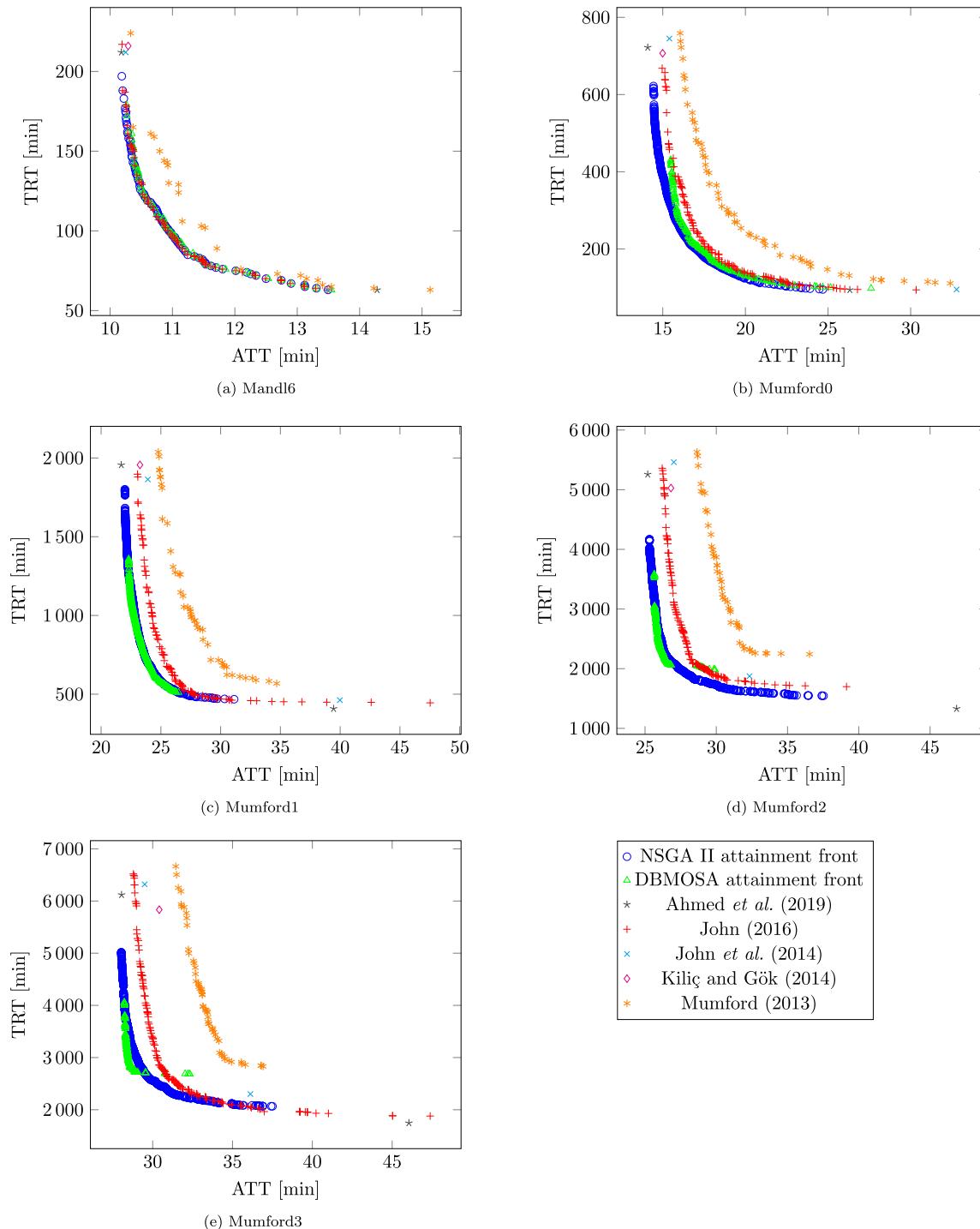
Mumford (2013) published results for the five benchmark instances of Section 2 in 2013, obtained by an evolutionary algorithm, and later (John, 2016) also published results for the same benchmark instances in 2016, demonstrating that his NSGA II algorithm outperformed the approach of Mumford. The approximate Pareto sets of Mumford and John are presented graphically in Fig. 8 together with the attainment fronts obtained by thirty different initialisations of our DBMOSA search process of Section 5.3, adopting the best parameter values as identified during the aforementioned design analysis. As may be seen in Fig. 8, the DBMOSA algorithm outperformed the evolutionary algorithm of Mumford, and outperformed almost all of the results of John's population-based search, dominating most of John's results

in terms of the ATT objective and uncovering solutions in areas of the Pareto front not represented in the results of John.

The DBMOSA algorithm could, however, not outperform the results reported by John for the Mandl6 benchmark instance, which is the smallest of the five test instances. It is interesting to note the difference in the qualities of the non-dominated sets returned by the DBMOSA algorithm in Fig. 8. It would seem that the DBMOSA algorithm performs better in respect of the ATT objective, but less so for the TRT objective. This may be because the DBMOSA algorithm is not equipped with mechanisms whereby feasibility may be lost and then regained later — a required mechanism when extremely small TRT objective function values are to be obtained, like in the case of Ahmed et al. (2019).

The HV values for the solutions returned by the DBMOSA algorithm of Section 5.3 are compared in Table 8 with the HV values obtained by John (2016) for each of the five benchmark instances.

When observing the HV over time for the NSGA II, it was clear that the HV gradients computed for the Mumford2 and Mumford3 instances were still growing steadily at termination, indicating that the NSGA II perhaps terminated prematurely (*i.e.* that 2 000 generations were not enough). A number of additional uncapped runs were therefore performed in order to push the NSGA II to its limits in terms of uncovering the highest possible quality solutions. The number of generations for these runs were therefore set to 10 000 and the stopping criterion was set to first-to-breach, specifying that the NSGA II should terminate when either 10 000 generations had been completed, or when the HV no longer improved by at least 0.01% over a moving window of 40 generations. The uncapped run results returned by the NSGA II of the previous section may be found in Fig. 8, plotted alongside the results obtained by John (2016) and those of Mumford (2013), as well as extreme point solutions reported by various other researchers. The average run times for the long and uncapped runs are summarised in Table 5.



**Fig. 8.** Attainment fronts of solutions produced by the DBMOSA algorithm and the NSGA II during the performance runs, together with results reported by various researchers, all pertaining to the UTRP benchmark instances described in Section 2.

**Table 8**

Normalised HV values obtained during the performance tests of the DBMOSA algorithm when solving the five UTRP benchmark instances of Section 2, along with the HV values obtained by John (2016). The HV values typeset in boldface are the best among the two approaches.

Solution approach	Mandl6	Mumford0	Mumford1	Mumford2	Mumford3
The DBMOSA of Section 5.3	85.3846	81.6712	78.7682	65.3447	58.2235
The NSGA II of Section 5.4	<b>86.0690</b>	<b>85.5074</b>	<b>81.1267</b>	<b>70.4683</b>	<b>65.977</b>
John (2016)	85.9769	80.8722	77.5761	64.9003	63.8547

**Table 9**

Objective function and transfer percentage evaluations for the ATT extremal solutions in Fig. 8 uncovered by the DBMOSA algorithm in respect of the five benchmark instances of Section 2, with the best overall performance for each instance highlighted in boldface.

Instance	Performance metric	Mumford (2013)	John et al. (2014)	Kılıç and Gök (2014)	Ahmed et al. (2019)	DBMOSA of Section 5.3	NSGA II of Section 5.4
Mandl6	Passenger	10.27	10.25	10.29	<b>10.18</b>	10.27	10.19
	Operator	221	212	216	212	179	197
	$d_0$	95.38	—	95.50	<b>97.17</b>	95.94	97.36
	$d_1$	4.56	—	4.50	<b>2.82</b>	3.93	2.64
	$d_2$	0.06	—	0.00	<b>0.00</b>	0.13	0.00
	$d_u$	0.00	—	0.00	<b>0.00</b>	0.00	0.00
Mumford0	Passenger	16.05	15.40	14.99	<b>14.09</b>	15.48	14.34
	Operator	759	745	707	722	431	635
	$d_0$	63.20	—	69.73	<b>88.74</b>	65.50	86.94
	$d_1$	35.82	—	30.03	<b>11.25</b>	34.50	13.06
	$d_2$	0.98	—	0.24	<b>0.00</b>	0.00	0.00
	$d_u$	0.00	—	0.00	<b>0.00</b>	0.00	0.00
Mumford1	Passenger	24.79	23.91	23.25	<b>21.69</b>	22.31	21.94
	Operator	2 038	1 861	1 956	1 956	1 359	1 851
	$d_0$	36.60	—	45.10	<b>65.75</b>	57.14	62.11
	$d_1$	52.42	—	49.08	<b>34.18</b>	42.63	37.84
	$d_2$	10.71	—	5.76	<b>0.07</b>	0.23	0.05
	$d_u$	0.26	—	0.06	<b>0.00</b>	0.00	0.00
Mumford2	Passenger	28.65	27.02	26.82	<b>25.19</b>	25.65	25.31
	Operator	5 632	5 461	5 027	5 257	3 583	4 171
	$d_0$	30.92	—	33.88	<b>56.68</b>	48.07	52.56
	$d_1$	51.29	—	57.18	<b>43.26</b>	51.29	47.33
	$d_2$	16.36	—	8.77	<b>0.05</b>	0.64	0.11
	$d_u$	1.44	—	0.17	<b>0.00</b>	0.00	0.00
Mumford3	Passenger	31.44	29.50	30.41	28.05	28.22	<b>28.03</b>
	Operator	6 665	6 320	5 834	6 119	4 060	5 018
	$d_0$	27.46	—	27.56	50.41	45.07	<b>48.71</b>
	$d_1$	50.97	—	53.25	48.81	54.37	<b>51.10</b>
	$d_2$	18.79	—	17.51	0.77	0.56	<b>0.19</b>
	$d_u$	2.81	—	1.68	0.00	0.00	<b>0.00</b>

The NSGA II clearly performed favourably compared with published results, dominating most of John's solutions, and all of Mumford's solutions. The HV achieved by the NSGA II for each benchmark instance may also be found in Table 8, compared with the HV values obtained by John (2016).

### 6.3.2. Benchmark results comparison

Apart from the two objective functions, the TRT and the ATT, represented by the two axes of Fig. 8, another measure of the quality of a route set is the percentage of demand satisfied without any transfer (denoted by  $d_0$ ), with one transfer (denoted by  $d_1$ ), with two transfers (denoted by  $d_2$ ), and with more than two transfers (denoted by  $d_u$ ). Chakroborty (2003) claimed that a good route set is one that serves the entire transit demand, with a large percentage of the demand being satisfied by direct connections and with the ATT per passenger as small as possible.

Tables 9 and 10 contain the results of the best extremal solutions (solutions attaining the best value for one of the objective functions of a multi-objective optimisation problem) uncovered by the DBMOSA over all the runs conducted, compared with the best extremal solutions found in the literature. The results obtained by our DBMOSA algorithm do not outperform any of the best extremal solutions reported in the literature, but its power lies in the high-quality intermediate trade-off routes returned along the non-dominated front. In a practical sense, these solutions would be preferred over those at the extremes of the Pareto front, due to the one extreme corresponding to the lowest ATT requiring large costs from the operator's perspective (TRT) which might not be practically affordable. The extreme corresponding to the smallest TRT, on the other hand, would mean that, for passengers, the travel time (ATT) would be longer than they are willing to endure (due to the bus networks containing too few bus routes), and therefore requiring many transfers and long detours. A balance somewhere between the extreme points would cater better for the needs of both passengers and operators. It is clear that all the sets of results in the table meet the

aforementioned criteria of Chakroborty (2003) and that the DBMOSA algorithm returned very favourable results overall.

Finally, a last set of runs was performed in which a static perturbation management strategy was adopted (*i.e.* adopting the same selection probabilities for all LLH perturbations). The results indicate that the stochastic roulette-wheel perturbation management strategy of Section 5.2.1 performed significantly better than when a static strategy was adopted, for all the benchmark instances.

When considering the best results returned overall by the NSGA II in Table 9 for the best passenger cost and in Table 10 for the best operator cost, some interesting insights arise. When specifically considering the best passenger costs returned in Table 9 in conjunction with Fig. 8, it is clear that at the extremal points, the ATT times are a few seconds longer than the best times reported in the literature, but in terms of the second objective, represent a considerable benefit over the TRT incurred when achieving these results. This is especially evident in the larger instances (the Mumford2 and Mumford3 instances), consisting of network sizes of 110 and 127 vertices, respectively.

For example, the best ATT achieved for the Mumford2 instance is 25.31 min with a TRT of 4 171 min, while the best recorded results in the literature are due to Ahmed et al. (2019), who reported an ATT of 25.19 min (which equates to a difference of 7.2 s, on average), but with a TRT of 5 257 min, which is a difference of 1 086 min of additional routes that have to be maintained for a relatively small benefit in ATT. The same is true for the Mumford3 instance. Here, however, the best ATT returned by the NSGA II of Section 5.4 outperformed the best ATT reported in the literature, also due to Ahmed et al. (2019). Our solution achieved an ATT of 28.03 min, with a TRT of 5 018 min, whereas the ATT of Ahmed et al. (2019) achieved 28.05 min, but with a TRT of 6 119 min, which represents an additional 1 101 min of routes that have to be maintained. This results in about one sixth of unnecessary bus routes according to Ahmed et al. (2019), compared with the bus routes returned for the large instances by the NSGA II of Section 5.4. These improvements elucidate the power of adopting a bi-objective modelling

**Table 10**

Objective function and transfer percentage evaluations for the TRT extremal solutions in Fig. 8 uncovered by the DBMOSA algorithm in respect of the five benchmark instances of Section 2, with the best overall performance for each instance highlighted in boldface.

Instance	Performance metric	Mumford (2013)	John et al. (2014)	Ahmed et al. (2019)	DBMOSA of Section 5.3	NSGA II of Section 5.4
Mandl6	Operator	<b>63</b>	<b>63</b>	<b>63</b>	<b>63</b>	<b>63</b>
	Passenger	<b>13.48</b>	<b>13.48</b>	14.28	13.55	13.49
	$d_0$	70.91	—	62.23	70.99	71.18
	$d_1$	25.5	—	27.16	24.44	25.21
	$d_2$	2.95	—	9.57	4.00	2.97
	$d_u$	0.64	—	1.03	0.58	0.64
Mumford0	Operator	111	95	<b>94</b>	98	<b>94</b>
	Passenger	32.4	32.78	26.32	27.61	27.17
	$d_0$	18.42	—	14.61	22.39	24.71
	$d_1$	23.4	—	31.59	31.27	38.31
	$d_2$	20.78	—	36.41	18.82	26.77
	$d_u$	37.4	—	17.37	27.51	10.20
Mumford1	Operator	568	462	<b>408</b>	511	465
	Passenger	34.69	39.98	39.45	26.48	31.26
	$d_0$	16.53	—	18.02	25.17	19.70
	$d_1$	29.06	—	29.88	59.33	42.09
	$d_2$	29.93	—	31.9	14.54	33.87
	$d_u$	24.66	—	20.19	0.96	4.33
Mumford2	Operator	2244	1875	<b>1330</b>	1979	1545
	Passenger	36.54	32.33	46.86	29.91	37.52
	$d_0$	13.76	—	13.63	22.77	13.48
	$d_1$	27.69	—	23.58	58.65	36.79
	$d_2$	29.53	—	23.94	18.01	34.33
	$d_u$	29.02	—	38.82	0.57	15.39
Mumford3	Operator	2830	2301	<b>1746</b>	2682	2043
	Passenger	36.92	36.12	46.05	32.33	35.97
	$d_0$	16.71	—	16.28	23.55	15.02
	$d_1$	33.69	—	24.87	58.05	48.66
	$d_2$	33.69	—	26.34	17.18	31.83
	$d_u$	20.42	—	32.44	1.23	4.49

approach, where both objectives are minimised simultaneously. There is considerable benefit in such an approach, as it obtains excellent results in terms of route set quality, especially for the larger network instances, which are more representative of real-world scenarios.

In terms of operator cost, the best TRT objective function values returned by the NSGA II are summarised in Table 10. Here, the NSGA II actually achieved the lower bounds for the Mandl6 and Mumford0 instances, although it could not achieve the corresponding smallest ATT reported by other researchers. The results are nevertheless of high quality when compared with other benchmark results. The NSGA II seems to exhibit a better capability of exploiting the ATT objective than exploiting the TRT objective in a single-objective paradigm, but when keeping these two objectives in conflict with one another, it would seem that the algorithm performs especially well in the domain of the ATT objective function. It can be argued that of the two objective functions, the ATT may well be the most important as it deals with passenger satisfaction.

### 6.3.3. LLH selection probabilities results

Some interesting insights may be gained when stacked area charts are plotted for the perturbation selection probabilities over the total number of iterations for each of the long runs of the DBMOSA algorithmic implementation. Such plots may be found in Fig. 9 for each of the benchmark instances. The data for these plots were generated by keeping track of the selection probabilities of each LLH during each run of the DBMOSA, and were updated by means of the stochastic roulette-wheel selection probability management strategy. The averages of all the runs were taken, up to the points of the minimum termination iterations over all the runs, so that insight might be gained into trends that arise during the progression of each search. Exponential smoothing was applied on the raw data points in order to achieve a more pleasurable viewing experience when plotting LLH selection probabilities.

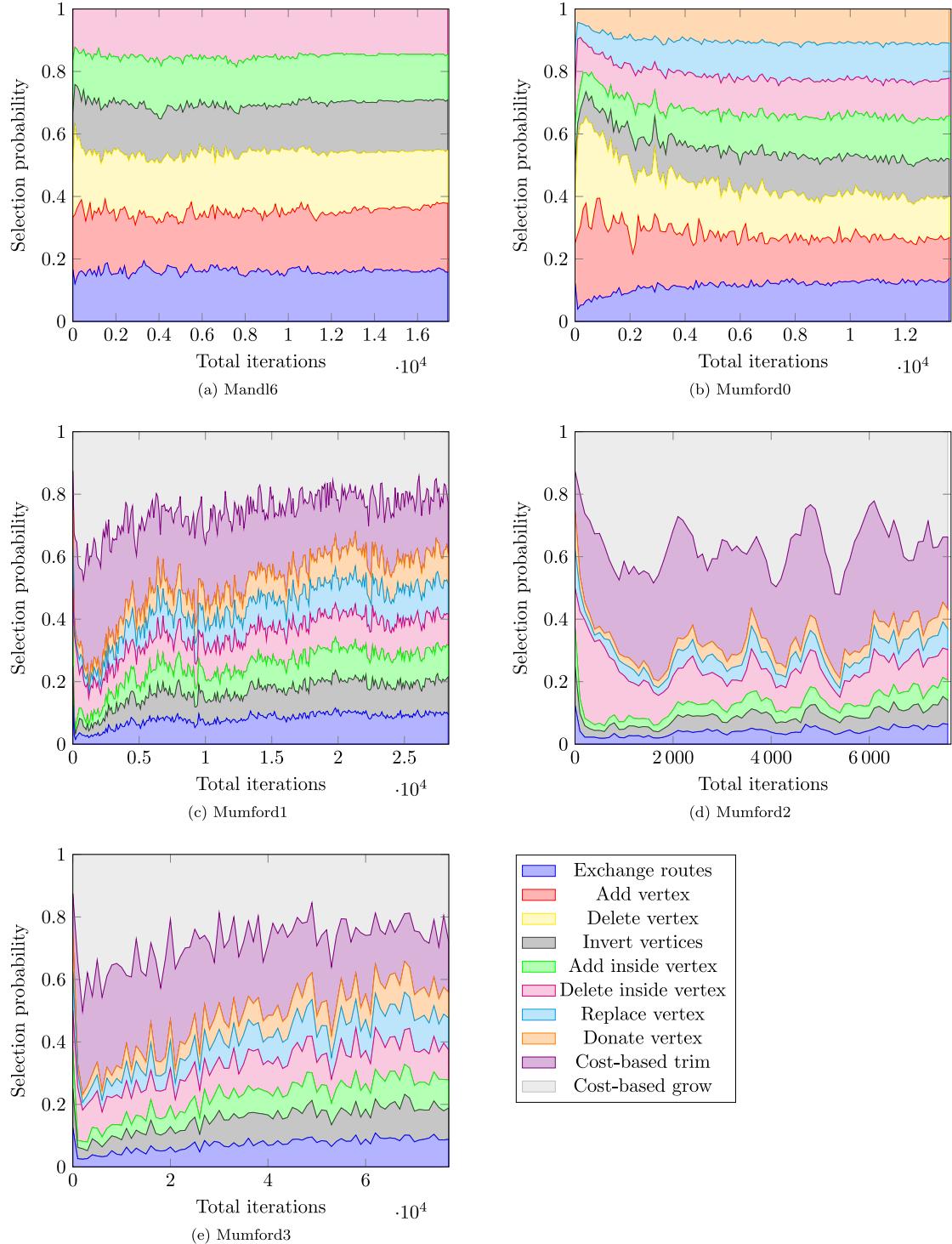
These graphs should be read from left to right as if combed by a vertical line from iteration 0 to the end, with the vertical line per

iteration representing the average probabilities ascribed to the LLHs at that given iteration, and with these probabilities summing up to 1. For example, iteration 0 represents the starting LLH probabilities which are all equal, and as the search progresses, the probabilities change. The change in vertical line proportion of each colour represents the change in probability for each LLH as a function of algorithmic iterations. Therefore, the more area covered by the colour of an LLH, the more it is preferred, and vice versa.

The selection probabilities for the Mandl6 instance in Fig. 9(a) exhibit rapid variations in the selection probabilities closer to the start of the search, but then even out towards the end of the search. It would seem that the add vertex operator was the most favoured among the LLHs, while the add inside vertex was the least favoured. When considering the selection probabilities for the Mumford0 instance, shown in Fig. 9(b), it is clear that even more rapid variations are present at the start of the search, where considerable computational resources were afforded to the add and delete vertex operators, and less so for the add inside vertex, replace vertex, and donate vertex operators. As the search progressed, however, the selection probabilities tended to even out once again.

The observations for the larger instances are more interesting. Here Fig. 9(c)–(e) indicate that the novel cost-based trim and cost-based grow operators were greatly favoured throughout the searches for the Mumford1, Mumford2, and Mumford3 instances. The exchange routes and invert vertices operators also enjoyed some favour, but not to the degree of the cost-based trim and cost-based grow operators. These changes in selection probabilities are intuitive in the sense that throughout a DBMOSA search run, different operators may be required in different regions of the search space in order to exploit or escape from local optima.

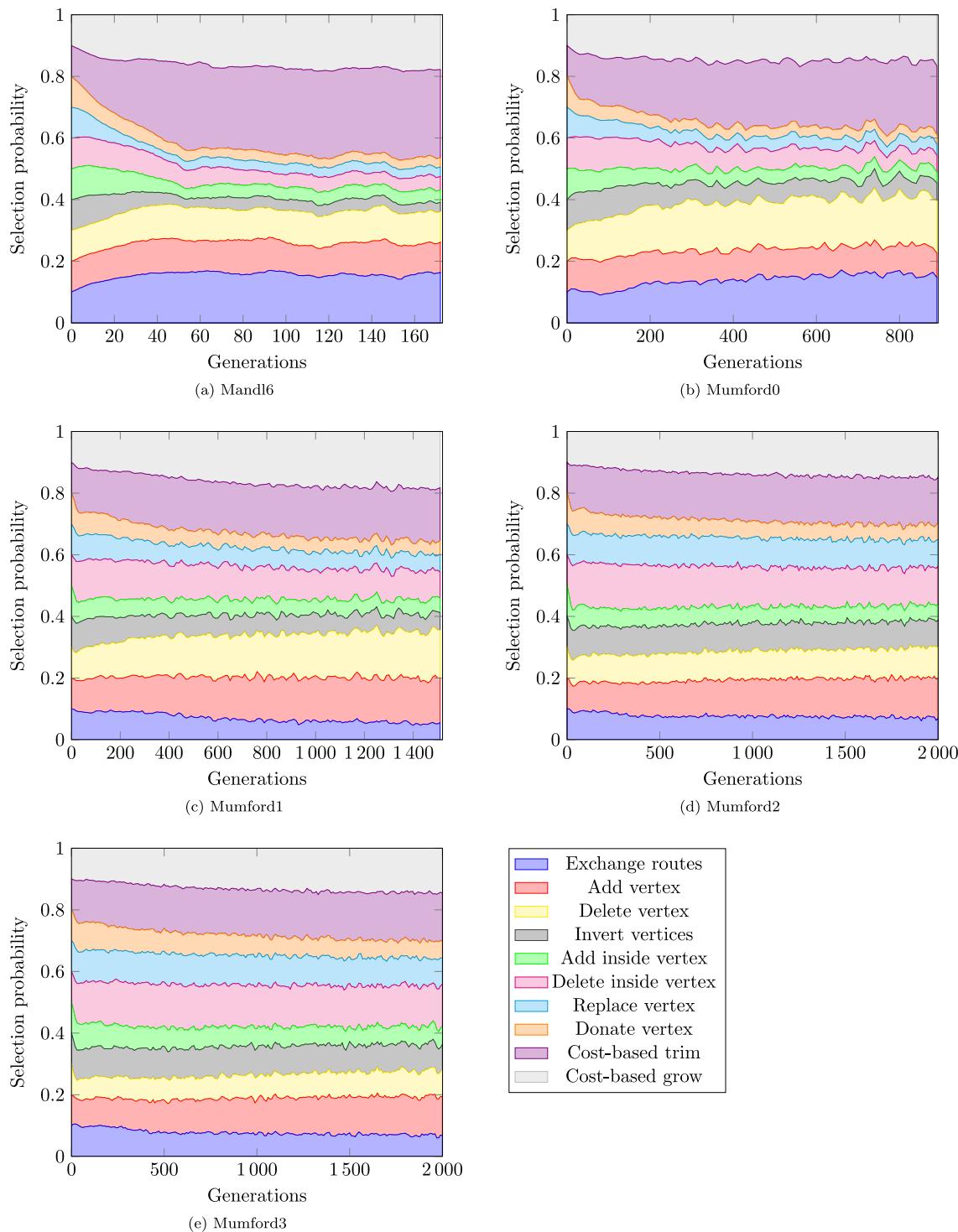
The mutation selection probabilities over the number of generations produced by the NSGA II are also plotted graphically as stacked area charts in Fig. 10. Fig. 10(a) indicates that the cost-based trim LLH operator, on average, received greater computational resources as the



**Fig. 9.** Average perturbation selection probability results over 30 separate longer runs of the DBMOSA algorithm. Stacked area charts illustrate the changes in LLH selection probabilities when read from left to right, with the distribution of the selection probabilities for each LLH on the vertical axes over the total number of iterations on the horizontal axes.

search progressed for the Mandl6 instance. A similar trend was observed for the Mumford0 instance, but to a lesser extent. In general, it would seem from Fig. 10 that the cost-based grow and cost-based trim operators were favoured in all the instances. It should be noted that for different instances, it appears that different mutation operators are prioritised. For example, the delete inside vertex operator received the greatest prioritisation in the case of the Mumford3 instance, and very little in the Mandl6 and Mumford0 instances. When taking a step

back and considering these selection probabilities, a few interesting observations may be made. In the past, most studies employed a few simple LLHs, and what has become evident based on the results of this paper and those in the paper of Ahmed et al. (2019), is that in order to obtain better solutions for larger problem instances, smarter management systems are required. More specifically, by designing some of the LLHs smarter, such as in the cost-based grow and cost-based trim operators, they are preferred in most of the cases because they



**Fig. 10.** Average mutation selection probability results over 20 separate longer runs of the NSGA II. Stacked area charts illustrate the changes in LLH selection probabilities when read from left to right, with the distribution of the selection probabilities for each LLH on the vertical axes over the number of generations on the horizontal axes.

have the largest impact in terms of improving solution quality based on HV.

## 7. Conclusion

A novel ICRSGP was proposed in this paper that allows for consistently generating high-quality feasible initial route sets, even for large networks, as all vertices of the network are included each time. Some researchers have resorted to applying repair strategies in order to

generate initial feasible solutions (Ahmed et al., 2019; Mumford, 2013), which is not required in this procedure.

None of the LLH operators proposed in the literature thus far, except for one, takes into consideration the underlying demand or cost components inherent to the UTRP. This gap in the literature was bridged by introducing two novel LLH operators in Section 5.1, which contributed most significantly to the high-quality solutions reported in this paper.

**Algorithm 3:** Generate all candidate  $K$  shortest paths

**Input :** An  $n \times n$  weight matrix  $\mathbf{W}$  representing the graph  $G$ , the minimum allowed number of vertices in a route  $m_{\min}$ , the maximum allowed number of vertices in a route  $m_{\max}$ , the number of shortest paths  $K$  to generate and the number of source-target pairs to retain  $\ell$ .

**Output:** A  $K$  shortest paths candidate route set  $C$ .

```

1  $S \leftarrow$  Generate a list of  $K$  shortest paths between all pairs of
   vertices in  $G$  using Yen's  $K$  shortest path algorithm
2 for  $i \leftarrow |S|$  to 1 do
3   if  $|V_{S_i}| < m_{\min}$  or  $|V_{S_i}| > m_{\max}$  then
4      $S_i \leftarrow \emptyset$ 
5 foreach source-target pair  $i-j$  do
6   Sort the route length in descending order first, and then
      resolve ties by sorting the demand serviced in ascending
      order
7   Select the top  $\ell$  number of  $i-j$  source-target pairs and
      insert into  $C$ 
8 return  $[C]$ 
```

**Algorithm 4:** Mumford's crossover procedure (Mumford, 2013).

**Input :** Two sets of routes,  $\mathcal{R}_1$  and  $\mathcal{R}_2$ , and the number of routes allowed in a route set  $r$ .

**Output:** A feasible set of routes  $\mathcal{R}'$ .

```

1  $\mathcal{P} \leftarrow \text{list}(\mathcal{R}_1, \mathcal{R}_2)$ 
2  $b \leftarrow \text{random\_boolean}()$ 
3  $\mathcal{R}' \leftarrow$  random route from  $\mathcal{P}[b]$  and remove route from  $\mathcal{P}[b]$ 
4  $b \leftarrow \text{switch}(b)$ 
5 while  $|\mathcal{R}'| \leq r$  do
6   forall  $R_i$  in  $\mathcal{P}[b]$  if  $\mathcal{V}_{R_i} \cap \mathcal{V}_{R'} \neq \emptyset$  do
7      $p_i \leftarrow |\mathcal{V}_{R_i} \setminus \mathcal{V}_{R'}| / |R_i|$ 
8      $R_t \leftarrow R_i$  with maximum  $p_i$  from  $\mathcal{P}[b]$ 
9      $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{R_t\}$  and remove  $R_t$  from  $\mathcal{P}[b]$ 
10     $b \leftarrow \text{switch}(b)$ 
11 if  $\mathcal{R}'$  is feasible then
12   return  $\mathcal{R}'$ 
13 else
14   return false
```

**Algorithm 5:** Calculate cumulative direct demand for a route set

**Input :** A route set  $\mathcal{R}$ , and an  $n \times n$  OD demand matrix  $\mathbf{D}$ .

**Output:** The direct demand satisfied  $d$  by the route set  $\mathcal{R}$ .

```

1  $d \leftarrow 0$ 
2  $c \leftarrow \text{list}()$ 
3 for  $R$  in  $\mathcal{R}$  do
4    $\quad$  Insert all  $(i, j)$  vertex pair combinations of  $\mathcal{V}(R)$  into  $c$ 
5 Remove any duplicate combinations from  $c$ 
6 foreach  $(i, j)$ -pair in  $c$  do
7    $\quad$   $d \leftarrow d + \mathbf{D}[i, j]$ 
8 return  $d$ 
```

In the minor tests, various LLHs were tested individually and also in various combinations, for both the DBMOSA and the NSGA II, and for the three smallest test instances (Mandl6, Mumford0 and Mumford1). The details of all these tests could not be included in this paper due to space constraints, but are included in the dissertation associated with this paper by Hüttelmann (2022).

One of the limitations of this study was the limited time available for the minor tests described in Section 6.1. Further studies may well build on this study by conducting more simulation runs and also adding additional tests aimed at improving even further on the solution implementations presented in this paper.

Various types of metaheuristics have been applied to this problem (Iliopoulou et al., 2019). It is, however, only very recently that instances of the UTRP have been solved in multi-objective optimisation contexts (as is appropriate for this type of problem) — earlier solution methodologies mainly scalarised a number of objective functions (Iliopoulou et al., 2019; Possel et al., 2018).

When the results returned by the DBMOSA algorithm of Section 5.3 and the NSGA II of Section 5.4 were compared with the best results reported in the literature, as summarised in Table 8, it became clear that the quality of the non-dominated solutions sets returned by the NSGA II of Section 5.4 outperformed the DBMOSA algorithm of Section 5.3, and also the results reported by John (2016) and Mumford (2013) in terms of the HV obtained by their non-dominated sets. To the best of our knowledge, the non-dominated sets of John (2016) were the best results available in the literature in terms of HV performance at the time of writing this paper.

It should be noted that the methods in this paper are outperformed by single-objective methods in the literature at the extreme Pareto points. The results in this paper are, however, superior in terms of the Pareto trade-off front even though they do not match the best extreme values of ATT and TRT for most test instances.

For the Mumford3 UTRP benchmark instance, the largest of the benchmark instances, an improved extremal solution in terms of the ATT objective was contributed in this paper, as indicated in Table 9, along with a one sixth improvement in terms of TRT objective function value (similar TRT improvements were observed for the Mumford2 instance). This indicates that for larger UTRP instances, the NSGA II of Section 5.4 has the ability to beat state-of-the-art algorithms, improving significantly on especially the TRT objective.

Our aim in this paper was to present improvements to the current state of the art when attempting to solve large, realistically sized instances of the UTRP. Our results may prove valuable to researchers aiming to improve their results when solving instances of the UTRP by taking into consideration the ICRSGP, the LLHs introduced and the hyperheuristic management strategies that may accompany metaheuristics, and finally the insight gained through the various tests conducted.

**CRediT authorship contribution statement**

**G. Hüttelmann:** Conceptualization, Methodology, Software, Validation, Formal analysis, Writing – original draft, Investigation, Visualization, Project administration. **J.H. van Vuuren:** Conceptualization, Methodology, Validation, Resources, Writing – review & editing, Supervision, Funding acquisition. **S.J. Andersen:** Conceptualization, Resources, Writing – review, Supervision, Funding acquisition.

**Data availability**

Results: <https://github.com/GHuss7/UTRP-Improvements-Article-Results>.

**Acknowledgements**

The authors thank the South African National (Department of Transport, 2021) for providing funding to the first author, as well as the Stellenbosch Unit for Operations Research in Engineering (2018) for providing computational resources, various opportunities and financial support. The Department of Transport did not play any active role in this research, and the SUNORE research group was the incubator for the research.

**Algorithm 6:** Cost-based trim

---

**Input :** A route set  $\mathcal{R}$ , the index  $i$  of a route in  $\mathcal{R}$  to be evaluated, an  $n \times n$  weight matrix  $\mathbf{W}$ , and an  $n \times n$  demand matrix  $\mathbf{D}$ .

**Output:** A trimmed route set  $\mathcal{R}'$ .

```

1  $d_0 \leftarrow \text{calc\_cum\_demand\_route\_set}(\mathcal{R}, \mathbf{D})$  [Algorithm 5]
2  $j \leftarrow 1$ 
3  $\mathcal{R}_t \leftarrow \mathcal{R}$  after removing the first vertex from route  $\mathcal{R}[i]$ 
4 if  $\text{test\_feasibility}(\mathcal{R}_t)$  then
5    $d_j \leftarrow d_0 - \text{calc\_cum\_demand\_route\_set}(\mathcal{R}_t, \mathbf{D})$ 
6    $c_j \leftarrow \mathbf{W}[\mathcal{R}[i][0], \mathcal{R}_t[i][0]]$ 
7   if  $d_j = 0$  then
8      $d_j \leftarrow 0.001$ 
9    $b_j \leftarrow c_j/d_j$ 
10   $a_j \leftarrow \text{true}$ 
11   $j \leftarrow j + 1$ 
12  $\mathcal{R}_t \leftarrow \mathcal{R}$  after removing the last vertex from route  $\mathcal{R}[i]$ 
13 if  $\text{test\_feasibility}(\mathcal{R}_t)$  then
14    $d_j \leftarrow d_0 - \text{calc\_cum\_demand\_route\_set}(\mathcal{R}_t, \mathbf{D})$ 
15    $c_j \leftarrow \mathbf{W}[\mathcal{R}[i][-1], \mathcal{R}_t[i][-1]]$ 
16   if  $d_j = 0$  then
17      $d_j \leftarrow 0.001$ 
18    $b_j \leftarrow c_j/d_j$ 
19    $a_j \leftarrow \text{false}$ 
20    $j \leftarrow j + 1$ 
21  $\ell \leftarrow j - 1$ 
22 if  $\ell = 0$  then
23   return  $\mathcal{R}$ 
24 for  $j = 1$  to  $\ell$  do
25    $p_j = b_j / \sum_{k=1}^{\ell} b_k$ 
26 Randomly choose a trim candidate  $j$  based on the probabilities  $p_j$ 
27 if  $a_j$  then
28    $\mathcal{R}' \leftarrow \mathcal{R}$  after removing the first vertex from route  $\mathcal{R}[i]$ 
29 else
30    $\mathcal{R}' \leftarrow \mathcal{R}$  after removing the last vertex from route  $\mathcal{R}[i]$ 
31 return  $\mathcal{R}'$ 
```

---

**Funding sources**

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

**Appendix**

See Algorithms 3–8.

**Algorithm 7:** Cost-based grow

---

**Input :** A route set  $\mathcal{R}$ , the index  $i$  of a route in  $\mathcal{R}$  to be evaluated, an  $n \times n$  weight matrix  $\mathbf{W}$ , an  $n \times n$  demand matrix  $\mathbf{D}$ , and the maximum number of allowed vertices  $m_{\max}$ .

**Output:** A grown route set  $\mathcal{R}'$ .

```

1  $d_0 \leftarrow \text{calc\_cum\_demand\_route\_set}(\mathcal{R}, \mathbf{D})$  [Algorithm 5]
2  $j \leftarrow 1$ 
3  $\mathcal{R}_i \leftarrow \mathcal{R}$ 
4 if  $|\mathcal{R}_i[i]| < m_{\max}$  then
5   foreach vertex  $h$  adjacent to the first vertex in route  $\mathcal{R}_i[i]$  not in route  $\mathcal{R}_i[i]$  do
6      $\mathcal{R}_i \leftarrow \mathcal{R}_i$  with vertex  $h$  inserted into the front of route  $\mathcal{R}_i[i]$ 
7      $d_j \leftarrow \text{calc\_cum\_demand\_route\_set}(\mathcal{R}_i, \mathbf{D}) - d_0$ 
8      $c_j \leftarrow \mathbf{W}[\mathcal{R}[i][0], \mathcal{R}_i[i][0]]$ 
9      $b_j, a_j, v_j, j \leftarrow d_j/c_j, \text{true}, h, j + 1$ 
10  $\mathcal{R}_i \leftarrow \mathcal{R}$ 
11 if  $|\mathcal{R}_i[i]| < m_{\max}$  then
12   foreach vertex  $h$  adjacent to the last vertex in route  $\mathcal{R}_i[i]$  not in route  $\mathcal{R}_i[i]$  do
13      $\mathcal{R}_i \leftarrow \mathcal{R}_i$  with vertex  $h$  inserted into the back of route  $\mathcal{R}_i[i]$ 
14      $d_j \leftarrow \text{calc\_cum\_demand\_route\_set}(\mathcal{R}_i, \mathbf{D}) - d_0$ 
15      $c_j \leftarrow \mathbf{W}[\mathcal{R}[i][-1], \mathcal{R}_i[i][-1]]$ 
16      $b_j, a_j, v_j, j \leftarrow d_j/c_j, \text{false}, h, j + 1$ 
17  $\ell \leftarrow j - 1$ 
18 if  $\ell = 0$  then
19   return  $\mathcal{R}$ 
20 if  $\sum_{k=1}^{\ell} b_k \neq 0$  then
21   for  $j = 1$  to  $\ell$  do
22      $p_j = b_j / \sum_{k=1}^{\ell} b_k$ 
23 else
24   for  $j = 1$  to  $\ell$  do
25      $p_j = 1/\ell$ 
26 Randomly choose a grow candidate  $j$  based on the probabilities  $p_j$ 
27 if  $a_j$  then
28    $\mathcal{R}' \leftarrow \mathcal{R}$  after inserting vertex  $v_j$  into the first vertex position of route  $\mathcal{R}[i]$ 
29 else
30    $\mathcal{R}' \leftarrow \mathcal{R}$  after inserting vertex  $v_j$  into the last vertex position of route  $\mathcal{R}[i]$ 
31 return  $\mathcal{R}'$ 
```

---

**Algorithm 8:** AMALGAM mutation strategy for population-based searches

---

**Input :** The selection probabilities  $p_i$ , a selection probability threshold  $p_t$ , and the LLH success scores  $q_i$  and total LLH applications  $t_i$  for each LLH  $i \in \{1, \dots, I\}$  during the previous generation.

**Output:** Updated selection probabilities  $p_i$ .

```

1 if  $\sum_{i=1}^I q_i \neq 0$  then
2   foreach  $i \in \{1, \dots, I\}$  do
3      $s_i \leftarrow \left( \frac{q_i}{t_i} \middle/ \sum_{j=1}^I \frac{q_j}{t_j} \right) \times (1 - p_t)$ 
4      $p_i \leftarrow s_i + p_t$ 
5 return  $p_i$ 
```

---

## References

- Ahmed, L., Mumford, C., Kheiri, A., 2019. Solving urban transit route design problem using selection hyper-heuristics. European J. Oper. Res. 274 (2), 545–559. <http://dx.doi.org/10.1016/j.ejor.2018.10.022>.
- Baaj, M.H., Mahmassani, H.S., 1991. An AI-based approach for transit route system planning and design. J. Adv. Transp. 25 (2), 187–209. <http://dx.doi.org/10.1002/atr.5670250205>, URL <http://onlinelibrary.wiley.com/doi/10.1002/atr.5670250205/abstract>.
- Buba, A.T., Lee, L.S., 2016. Differential evolution for urban transit routing problem. J. Comput. Commun. 4 (14), 11–25.
- 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. J. Oper. Res. Soc. 64 (12), 1695–1724.
- Ceder, A., Wilson, N.H., 1986. Bus network design. Transp. Res. B 20 (4), 331–344. [http://dx.doi.org/10.1016/0191-2615\(86\)90047-0](http://dx.doi.org/10.1016/0191-2615(86)90047-0).
- Chakroborty, P., 2003. Genetic algorithms for optimal urban transit network design. Comput.-Aided Civ. Infrastruct. Eng. 18 (3), 184–200.
- Chakroborty, P., Wivedi, T., 2002. Optimal route network design for transit systems using genetic algorithms. Eng. Optim. 34 (1), 83–100. <http://dx.doi.org/10.1080/03052150210909>.
- Chew, J.S.C., Lee, L.S., Seow, H.V., 2013. Genetic algorithm for bi-objective urban transit routing problem. J. Appl. Math. 2013, 1–15.
- Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197. <http://dx.doi.org/10.1109/4235.996017>.
- Department of Transport, 2021. Department of transport: Home. [Online], [Cited September 9th, 2021], Available from <https://www.transport.gov.za/>.
- Desaulniers, G., Hickman, M.D., 2007. Public transit. In: Barnhart, C., Laporte, G. (Eds.), Handbooks in Operations Research and Management Science. vol. 14, Elsevier, Amsterdam, pp. 69–120.
- Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271. <http://dx.doi.org/10.1007/BF01386390>.
- Dréo, J., Pétrowski, A., Siarry, P., Taillard, E., 2006. Metaheuristics for hard optimization: Methods and case studies, first ed. Springer, Berlin.
- Dunlap, M., Li, Z., Henrickson, K., Wang, Y., 2016. Estimation of origin and destination information from bluetooth and wi-fi sensing for transit. Transp. Res. Record: J. Transp. Res. Board 2595, 11–17. <http://dx.doi.org/10.3141/2595-02>, URL <http://trjournalonline.trb.org/doi/10.3141/2595-02>.
- Eiben, Á.E., Hinterding, R., Michalewicz, Z., 1999. Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. 3 (2), 124–141.
- Fan, W., Machemehl, R.B., 2004. Optimal transit route network design problem: Algorithms, implementations, and numerical results. Technical Report, Center for Transportation Research, University of Texas, Austin (TX), p. 267, URL <https://static.tti.tamu.edu/swutc.tamu.edu/publications/technicalreports/167244-1.pdf>.
- Fan, W., Machemehl, R.B., 2006a. Optimal transit route network design problem with variable transit demand: Genetic algorithm approach. J. Transp. Eng. 132 (1), 40–51. [http://dx.doi.org/10.1061/\(ASCE\)0733-947X\(2006\)132:1\(40\),](http://dx.doi.org/10.1061/(ASCE)0733-947X(2006)132:1(40),) URL [http://ascelibrary.org/doi/10.1061/\(%28ASCE%290733-947X%282006%29132%3A1%2840%29](http://ascelibrary.org/doi/10.1061/(%28ASCE%290733-947X%282006%29132%3A1%2840%29).
- Fan, W., Machemehl, R.B., 2006b. Using a simulated annealing algorithm to solve the transit route network design problem. J. Transp. Eng. 132 (2), 122–132. [http://dx.doi.org/10.1061/\(ASCE\)0733-947X\(2006\)132:2\(122\),](http://dx.doi.org/10.1061/(ASCE)0733-947X(2006)132:2(122),) URL [http://ascelibrary.org/doi/10.1061/\(%28ASCE%290733-947X%282006%29132%3A2%28122%29](http://ascelibrary.org/doi/10.1061/(%28ASCE%290733-947X%282006%29132%3A2%28122%29).
- Fan, W., Machemehl, R.B., 2008a. Tabu search strategies for the public transportation network optimizations with variable transit demand. Comput.-Aided Civ. Infrastruct. Eng. 23 (7), 502–520.
- Fan, W., Machemehl, R.B., Lownes, N.E., 2008b. Some computational insights on the optimal bus transit route network design problem. J. Transp. Res. Forum 47 (3), 60–75.
- Fan, L., Mumford, C.L., 2010. A metaheuristic approach to the urban transit routing problem. J. Heuristics 16 (3), 353–372. <http://dx.doi.org/10.1007/s10732-008-9089-8>.
- Guan, J., Yang, H., Wirasinghe, S.C., 2006. Simultaneous optimization of transit line configuration and passenger line assignment. Transp. Res. B 40 (10), 885–902.
- Guiaire, V., Hao, J.K., 2008. Transit network design and scheduling: A global review. Transp. Res. A 42 (10), 1251–1273. <http://dx.doi.org/10.1016/j.tra.2008.03.011>, URL <http://linkinghub.elsevier.com/retrieve/pii/S0965856408000888>.
- Henning, M.A., Van Vuuren, J.H., 2022. *Optimal Paths*, Pp. 87–113 in Graph and Network Theory — an Applied Approach using Mathematica. Springer, Cham.
- Hüttelmann, G., 2022. Bus route design and frequency setting for public transit systems (Ph.D. thesis). Stellenbosch University, Stellenbosch, South Africa, URL <http://hdl.handle.net/10019.1/124533>.
- Ibarra-Rojas, O.J., Delgado, F., Giesen, R., Munoz, J.C., 2015. Planning, operation, and control of bus transport systems: A literature review. Transp. Res. B 77, 38–75. <http://dx.doi.org/10.1016/j.trb.2015.03.002>.
- Iliopoulos, C., Kepaptsoglou, K., Vlahogianni, E., 2019. Metaheuristics for the transit route network design problem: A review and comparative analysis. Public Transp. 11 (3), 487–521.
- John, M.P., 2016. Metaheuristics for Designing Efficient Routes & Schedules for Urban Transportation Networks. Ph.D. thesis. Cardiff University, URL <http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.709603>.
- John, M.P., Mumford, C.L., Lewis, R., 2014. An improved multi-objective algorithm for the urban transit routing problem. In: Blum, C., Ochoa, G. (Eds.), Evolutionary Computation in Combinatorial Optimisation. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 49–60.
- Kechagiopoulos, P.N., Beligiannis, G.N., 2014. Solving the urban transit routing problem using a particle swarm optimization based algorithm. Appl. Soft Comput. 21, 654–676.
- Kepaptsoglou, K., Karlaftis, M., 2009. Transit route network design problem: Review. J. Transp. Eng. 135 (8), 491–505. [http://dx.doi.org/10.1061/\(ASCE\)0733-947X\(2009\)135:8\(491\),](http://dx.doi.org/10.1061/(ASCE)0733-947X(2009)135:8(491),) URL [http://ascelibrary.org/doi/10.1061\(%28ASCE%290733-947X%282009%29135%3A8%28491%29](http://ascelibrary.org/doi/10.1061(%28ASCE%290733-947X%282009%29135%3A8%28491%29).
- Kılıç, F., Gök, M., 2014. A demand based route generation algorithm for public transit network design. Comput. Oper. Res. 51, 21–29. <http://dx.doi.org/10.1016/j.cor.2014.05.001>.
- Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680. <http://dx.doi.org/10.1126/science.220.4598.671>, URL <http://science.sciencemag.org/content/220/4598/671>.
- Lee, Y.J., Vuchic, V.R., 2005. Transit network design with variable demand. J. Transp. Eng. 131 (1), 1–10.
- Li, M., Yao, X., 2019. Quality evaluation of solution sets in multiobjective optimisation: A survey. Assoc. Comput. Mach. Comput. Surv. 52 (2), 1–38. <http://dx.doi.org/10.1145/3300148>.
- Mandl, C.E., 1979. *Applied Network Optimization*. Academic Press, London.
- Mumford, C.L., 2013. New heuristic and evolutionary operators for the multi-objective urban transit routing problem. In: 2013 IEEE Congress on Evolutionary Computation. Cardiff, pp. 939–946. <http://dx.doi.org/10.1109/CEC.2013.6557668>.
- Nikolić, M., Teodorović, D., 2013. Transit network design by bee colony optimization. Expert Syst. Appl. 40 (15), 5945–5955.
- Possel, B., Wismans, L.J., Van Berkum, E.C., Bliemer, M.C., 2018. The multi-objective network design problem using minimizing externalities as objectives: Comparison of a genetic algorithm and simulated annealing framework. Transportation 45 (2), 545–572.
- Shih, M.C., Mahmassani, H.S., 1994. A design methodology for bus transit route networks with coordinated operations. Technical Report 1, Center for Transportation Research, University of Texas at Austin, Austin (TX), [http://dx.doi.org/10.1016/0965-8564\(96\)81137-2](http://dx.doi.org/10.1016/0965-8564(96)81137-2), URL <http://linkinghub.elsevier.com/retrieve/pii/0965856496811372>.
- Smith, K.I., Everson, R.M., Fieldsend, J.E., Murphy, C., Misra, R., 2008. Dominance-based multiobjective simulated annealing. IEEE Trans. Evol. Comput. 12 (3), 323–342. <http://dx.doi.org/10.1109/TEVC.2007.904345>.
- Soares, H.P., 2020. Three steps towards practical application of public transport route optimisation in urban areas (Ph.D. thesis). University of Nottingham, Nottingham.
- Srinivas, N., Deb, K., 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. Evol. Comput. 2 (3), 221–248. <http://dx.doi.org/10.1162/evco.1994.2.3.221>.
- Stellenbosch Unit for Operations Research in Engineering, 2018. Sunore home. [Online], [Cited September 30th, 2018], Available from <http://sunore.co.za/>.
- Toth, P., Vigo, D., 2002. *The Vehicle Routing Problem*. In: SIAM monographs on discrete mathematics and applications, Society for Industrial and Applied Mathematics, Philadelphia (PA).
- Vrugt, J.A., Robinson, B.A., 2007. Improved evolutionary optimization from genetically adaptive multimethod search. Proc. Natl. Acad. Sci. 104 (3), 708–711.
- Xiong, Y., Schneider, J.B., 1992. Transportation network design using a cumulative genetic algorithm and neural network. Transp. Res. Rec. 1364, 37–44.
- Yan, Y., Liu, Z., Meng, Q., Jiang, Y., 2013. Robust optimization model of bus transit network design with stochastic travel time. J. Transp. Eng. 139 (6), 625–634.
- Yen, J.Y., 1971. Finding the K shortest loopless paths in a network. Manage. Sci. 17 (11), 712–716.
- Zhao, F., Gan, A., 2003. Optimization of transit network to minimize transfers. Technical Report, Lehman Center for Transportation Research, Department of Civil & Environmental Engineering, Florida International University, Miami (FL).
- Zhao, F., Ubaka, I., 2004. Transit network optimization — Minimizing transfers and optimizing route directness. J. Public Transp. 7 (1), 63–82.
- Zhao, F., Ubaka, I., Gan, A., 2005. Transit network optimization: Minimizing transfers and maximizing service coverage with an integrated simulated annealing and tabu search method. Transp. Res. Rec. 1923 (1), 180–188.
- Zhao, F., Zeng, X., 2006. Optimization of transit network layout and headway with a combined genetic algorithm and simulated annealing method. Eng. Optim. 38 (6), 701–722. <http://dx.doi.org/10.1080/03052150600608917>.