Three Steps Towards Practical Application of Public Transport Route Optimisation in Urban Areas

Philipp Heyken Soares

Laboratory of Urban Complexity and Sustainability (LUCAS)

University of Nottingham

A thesis submitted for the degree of

Doctor of Philosophy

2020

Abstract

The research on computer-based optimisation of public transport networks has seen huge advances in recent years. However, there is almost no application of its results in real-world planning processes. A review of the literature identifies several possible reasons for this: a) many developed algorithms do not consider important details for urban applications, b) methods to generate the required input datasets struggle when applied to irregular urban networks, or c) the absence of an interface to use the developed algorithms with transport modelling software. The three publications presented in this thesis address these three issues.

The first publication introduces a method to scale down an available street network to a level where optimisation methods can be applied while preserving its characteristics. All travel times, demand data, and information regarding the permitted route endpoints are derived from openly available data. The methodology is applied to the urban area of Nottingham, UK, to generate new benchmark datasets for bus route optimisation. An optimisation procedure centred on a genetic algorithm and adapted for the use of restricted route endpoints is applied to the generated dataset.

In the second publication, these methodologies are extended to use a more realistic zone-based representation of passenger journeys. Zone-based trip representations are rarely used in academic studies as the conventional node-based approach is considered simpler to implement and many input datasets are publicly available. The publication presents a new hybrid-approach to calculate zone-based journey times using established node-based concepts, and introduces a first publicly available, zone-based input dataset.

The third publication introduces an interface between route optimisation algorithms and the professional transport modelling software PTV Visum. The interface manages the differences in data requirements between the two modelling approaches, allowing users to directly optimise the public transport network in a given Visum network model. It is successfully combined with Selection Hyper Heuristics to optimise passenger travel time and operator costs in network models based on real-world planning processes. An additional optimisation experiment exemplary reduces the number of private vehicles on a selected street, demonstrating the capabilities of the interface.

Acknowledgements

First of all, I would like to thank Dr. Yong Mao, my supervisor over the past four years. Without his advice, empathy, and patience this research would not have come to completion. Further thanks goes to Prof. Terry Moore for taking me on as a student at a late stage of my project and for all the help and guidance he provided me throughout its last phase. Additionally, I want to express my gratitude to Dr. Andrew Allen, my supervisor during my first year, my internal assessor Prof. Markus Owen, and the postgraduate student advisor Prof. Stuart Marsh for their advice and support during various stages of my research. Special thanks go to Dr. Christine Mumford who, although never filling an official supervisor role, was still willing to provide me with continuous advice that proved invaluable for this research.

I further want to thank my co-authors Kwabena and Leena for our productive cooperation. Special thanks also go to all my fellow PhD students in the Laboratory for Urban Complexity and Sustainability (LUCAS) (Ana, Ashley, Ben, Gustavo, Hannah, James, Kunpeng, Kwabena, Nguyen, Parag, Phil, Renoy, and Tim) for the amazing atmosphere created in our office and the many great moments we had both in and outside of it. Also, I want to extend my gratitude to the LUCAS research fellows, in particular, Gavin, Sameh, Julian, and Jenni, for helping me with their experience in various situations. Further, I have to thank the Leverhulme Trust for funding the LUCAS and allowing us the academic freedom which made this research possible.

Last but not least, I want to express my deepest gratitude to my family and all my friends who were always there for me and providing much-needed support and encouragement throughout the past years. Special thanks go to my parents Petra and Tönjes for all the moral, financial, and technical support they provided me over the years. Também eu expresso meu agradecimento para minha sogra Magnólia por ter me fornecido comida e um teto durante o tempo que escrevi essa tese[1]. Most of all I want to thank my amazing Mônicat for always being by my side and helping me through my darkest hours. Without her, the past years would not have been worth struggling through.

[1]English translation: I also express my thank to my mother-in-law Magnolia for providing me with food and roof during the write-up of this thesis.

Contents

1 Introduction

1.1 Background

Around the world, the number of people living in urban areas is increasing. In 2018 the UN estimated the world's urban population to have reached 4.2 billion, a number which is expected to grow by over 50% over the next thirty years [1]. These numbers indicate the enormous challenges urban planners are facing in the upcoming decades. One of these challenges is the adaptation of urban transportation systems.

The transportation system is of enormous importance for the functioning of an urban area. Its design influences many factors ranging from travel times between destinations, pollution levels, to aspects of social cohesion. In the past, many urban transport systems were designed with a focus on suitability for private cars. However, this resulted in excessive congestion and pollution in many cities [2]. In recent years many reports have emphasised the importance of efficient public transport (PuT) [3]. In 2013 a UN report [4] even found that adequate public transport systems improve the quality of life for both users and non-users and have a positive influence on all aspects of urban prosperity. The design of efficient public transport systems is therefore of great importance.

To date, the planning of public transport networks relies on local knowledge, planning experience and published guidelines [5, 6]. Further, most of these networks have evolved over time rather than being designed as a whole, a process which inherently leads to inefficiencies [7, 8]. With cities around the world struggling to adapt to growing populations and technological changes, the near future will likely require significant revisions of many urban public transport networks. Multiple reports have highlighted the insufficiencies in the manual planning processes and the need for optimisation algorithms to support planners in this task [6, 9].

Researchers have been working on such tools for decades; however, there appear to be high barriers for their practical application, especially when it comes to optimising the public transport routes themselves. This thesis presents steps to bring the research for this vital aspect of public transport network optimisation closer to practical applications.

1.2 Problem formulation

This Section will give a brief overview of the problems addressed in this thesis. Please note that all topics will be discussed in more detail in Chapters 2 and 3.

The task to generate an efficient public transport network is described in [10] as a process of five phases[1]: 1st) Route Design, 2nd) Vehicle Frequency Setting, 3rd) Timetable Development, 4th) Vehicle Scheduling, and 5th) Driver Scheduling.

These phases are interconnected, and a procedure which solves all five phases simultaneously would be optimal. However, due to the tasks' high complexity, researchers simplify the problem by focusing on a subset of phases. One such simplification is to separate route design phase from the remaining four by assuming

[1]The five phases of PuT network design introduced in [10] are the most commonly used formulation, however differing formulations also exist. E.g. in [8] the phases Vehicle Frequency Setting and Timetable Development are combined into one phase, and in [11] the Driver Scheduling phase is split up into separated phases for short-term and long-term planning.

a unified standard waiting time for all interchanges. This formulation in the following is referred to as the "Urban Transit Routing Problem" (UTRP) and is the focus of this thesis.

For many decades, researchers have been working on methods to automatically generate solutions to the UTRP. The proposed methods range from mathematical programming to a vast array of heuristic and meta-heuristic algorithms [12–14]. However, despite the numerous studies existing in this field, only in very few cases have their results found a practical application in planning processes [5, 6]. The reasons why the results of these studies are not utilised have so far not been analysed conclusively[2]. However, several possible reasons are related to the used problem instances, i.e., the input data sets required to run the UTRP algorithms.

In its simplest form, UTRP instances consist of a graph structure with nodes and links representing the available transport infrastructure and information regarding the travel demand in the respective area. Other information can be added to increase the realism of the instance, especially for the representation of urban areas. Unfortunately, most researchers base their algorithms on relatively simple instances which bear little resemblance to the complexity and size of real-world urban areas. Although algorithms developed in this way are compelling, they can also easily be too far removed from applications used in real-world planning processes.

Among urban specific aspects often missing from UTRP instances is information on potential terminal nodes. The most common definition of routes in UTRP studies requires vehicles to perform a U-turn when reaching the end of a route to start the journey in the opposite direction (more on this in Section 3.2.1). Given the fact that the possibility for vehicles to turns is often restricted, it can not be assumed

[2]To the best of the author's knowledge, the only work on this question is a series of interviews with the planning agencies of major cities in Europe and the Middle East, which included questions on the use of UTRP algorithms. It was conducted by Walter in 2010 [6]. The responses showed that none of the agencies in question included such algorithms in their planning process; however, they could not give a conclusive answer on why this was the case.

that such manoeuvres are possible at all points of an urban street or rail network. Therefore, an urban instance should include information on where U-turns can be realistically performed so the route construction can be restricted accordingly (more details in Section 3.2.2). A further aspect is the representation of the travel demand. Most UTRP instances provide the travel demand as the number of trips between PuT access points. This is more abstract than a representation of trips based on a zonal division of the study area typically used in real-world planning processes (more details in Section 3.3.2). Using zone-based demand requires an extension of the instances to include zones and the connectors between them and the PuT network. If a UTRP algorithm takes these aspects into account it is more likely that it can be applied in real-world planning processes. However, the development of such algorithms requires the availability of instances featuring the respective information.

As generating instances is very time consuming, many researchers prefer to develop and test their algorithm on publicly available instances. This practice also allows using these instances as benchmarks to compare results from different studies. Unfortunately, the instances which are currently available publicly are relatively small and only rarely include the above mentioned urban specific aspects (more details in Section 3.2.4). These problems are well known, and many researchers have in the past called for larger and more realistic instances to be published [12].

Another problem related to instances is that there are very few published procedures on the generation of instance data sets. Another problem related to instances is that there are very few published procedures on the generation of instance data sets. Although several studies feature instances generated by the researchers, the procedures used for this task are usually not sufficiently described to reproduce them. Those generation procedures which are described in more detail cannot be reliably applied to all urban areas. They are often best suitable for areas with a repetitive, preferentially grid-like, layout. However, for the application on urban areas with an irregular layout, like most European cities, the available methods are not sufficient. More on this in Section 3.2.3. As a result, planners who want to use UTRP algorithms in their work have no guidelines on how to prepare the required input data.

There are two possible approaches to solve these problems. The first is to develop a new methodology to generate instances. It needs to include procedures to scale down an irregular urban street (or rail) network to a size manageable for optimisation algorithms while preserving its characteristics. Preferably, this will include information on zones and connectors required for using zone-based demand representation. Additionally, procedures are required to both prepare the travel demand, and to extract information on potential terminal nodes. In the optimal case, these procedures should be based on freely available data to boost its usability to researchers and smaller planning agencies. Such a procedure would also allow the generation and publication of larger and more realistic instances to be used as benchmarks for all researchers.

The alternative approach is to interface UTRP algorithms directly with transport the modelling software packages used in real-world planning processes. Planners use these software packages to build models allowing them to simulate a wide array of transport-related phenomena. The implementation and calibration of these network models is based on detailed information on the street layout, transport infrastructure, and real-world travel behaviour. Completed, these models include all the necessary information to run UTRP algorithms. If this information can be exchanged with the algorithm via an interface, there would be no need to generate a separate UTRP instance. Such a tool would significantly reduce the barriers for planners to use UTRP algorithms. Additionally, researchers would profit from such an interface as they would gain access to a vast array of evaluation tools embedded in these software packages. The main problem in developing such an interface is to translate route information between the undirected graph representation usually used in the UTRP and the network models which are generally based on directed connections. More on this in Section 3.3.3.

1.3 About this Thesis

1.3.1 Aims and Objectives

The publications presented in this thesis aim to move UTRP research closer to real-world applications by following both approaches outlined above:

A) Introduction of a new generation procedure for urban UTRP instances applicable to irregular urban areas. It needs to include both information on potential terminal nodes and should allow for a zone-based representation of passenger trips.

B) Development of an interface between UTRP algorithms and a professional transport modelling software.

The secondary aim is to facilitate the research on UTRP algorithms which better take urban characteristics into account. Here the objectives are:

- publishing new, and more realistic instances based on data from real-world urban areas.
- demonstrate how existing optimisation procedures can be adapted to include urban specific aspects (restricted terminal nodes and zone-based trip representation).

1.3.2 Thesis Structure and Cohesion

The presented thesis is a Published Works Thesis. At its core are three journal papers which make up the Chapters 4, 5, and 6. In addition to these publications themself, these chapters further feature information on the contribution of each author to the respective work[3] as well as a summary of the content. The latter also contains explanations on why certain research decisions were taken. Preceding these core publications, the Chapters 2 and 3 provide a more general context. Chapter 2 introduces basic concepts around optimisation and graph theory, while Chapter 3 reviews the available literature on public transport route optimisation. This review focuses especially on the above-mentioned obstacles for practical application detailing the gaps in research which are addressed in the later Chapters. In this, Chapter 3 goes both broader and deeper into the relevant subjects than the respective introduction and background Sections of the later presented publications.

The first two publications follow approach A. The publication presented in Chapter 4 introduces a generation procedure for instances usable for urban areas with an irregular layout. The methodology includes the determination of node positions and travel times between these, the identification of potential terminal nodes, and the construction of a node-based demand matrix. Additionally, the publication presents an optimisation procedure centring on a genetic algorithm adapted for the use with restricted terminal nodes. Experimental results show that the optimised route networks are superior to those pre-existing in the study area.

Chapter 5 extends the methodologies presented in Chapter 4 to work with a zone-based representation of trips and travel demand. This includes adding zones and connectors to the instance generation procedure. Additionally, a new concept to calculate zone-based journey times is introduced and utilised to adapt the optimisation procedure for it to work with zone-based travel demand. Computer experiments show that the presented procedure can to generate efficient solutions for a variety of parameter settings.

Approach B is explored in Chapter 6 with the development of an interface between a UTRP algorithms and the professional transport modelling software

[3]The formulations of the Authors' Contribution sections was agreed to by the respective co-authors.

PTV Visum. This includes interface procedures to extract a graph structure from a given Visum network model and to implemented changed routes into Visum for evaluation. These evaluations are by default using a zone-based trip representation. The presented optimisation algorithm is based on selection hyper heuristics and takes terminal nodes into account. The optimisation results show considerable improvements in an example network taken from a real-world city.

Both branches of research are connected as the adaptations to the optimisation procedure introduced in Chapter 5 includes elements vital to implement such a procedure with the Visum interface presented in Chapter 6. These interconnections will be outlined in more detail in a concluding discussion in Section 7.2. In addition, Chapter 7 further provides a summary and outlines avenues for further research.

Additional to these Chapters, there is a short appendix to this thesis. The conference paper in appendix A introduces three smaller instances generated with the procedure introduced in Chapter 4. It further provides an additional link between the core publications by providing a comparison between the genetic algorithm used in Chapter 4 and the sequence-based selection hyper heuristics used in Chapter 6. Further, Appendix B presents additional results not used in any of the presented publications, and Appendix C outlines two potential research projects which are supported by the work presented in this thesis but branch out of the definitions of the UTRP.

1.3.3 Main Contributions

The main contributions of the publications assembled in this thesis are the introduction of procedures to generate both node-based and zone-based instances, as well as the development of an interface between UTRP algorithms and a professional transport modelling software. Further, the adaptation of optimisation algorithms for urban specific aspects, such as restricted terminal nodes is demonstrated and new more realistic instances are published for free use by other researchers. More details are listed below with indications of the Chapters containing them.

Instance generation procedure

Chp. 4: Introduction of a procedure to generate UTRP instance dataset. It is applicable to urban areas with irregular layouts, and includes a data-driven process to identify potential terminal nodes.

Chp. 4: An additional methodology for mapping the pre-existing public transport routes onto the generated graph to allow comparisons.

Chp. 5: Extension of the procedure described in Chapter 4 to generate dataset necessary to use a zone based representation of passenger trips.

Chp. 5: Extension of the demand matrix generation to include cross-boundary flow.

Adaptation of optimisation algorithms for urban specific aspects.

Chp. 4: Adaptation of a multi-objective genetic algorithm for restricted terminal nodes.

Chp. 5: Improvement in the generation of routes with restricted terminal nodes to better connect non-terminal nodes.

Chp. 5: Introduction of a new concept for the calculation of zone-base journey times.

Chp. 5: Adaptation of construction heuristic and genetic algorithm described in Chapter 4 for using zone-based trip representations.

Publication of instances including urban specifications.

multiple: The instances generated in the Chapters 4, 5, and Appendix A were published online for free use for all researchers.

Interface between a UTRP algorithm and transport modelling software.

Chp. 6: Development of an interface to facilitate transferring and translating data between UTRP algorithms and PTV the professional transport modelling software PTV Visum.

Chp. 6: Use of the interface to optimise the PuT routes in two Visum network models representing real-world urban areas with an optimisation procedure based on Selection Hyper-Heuristics.

Chp. 6: Examples for how to use Visum functions in the evaluation of altered route networks.

2 Theoretical Background

This chapter introduces basic concepts which are of relevance throughout the following literature review and the later chapters. This includes the mathematical definition of optimisation, brief introductions into graph theory and computational complexity and short descriptions of different types of solution methods.

2.1 Basics of Optimisation

Optimisation is the task of identifying the best solution for a given problem from all possible solutions. Mathematically speaking, a minimisation[1] optimisation process can be described as:

$$\text{minimise } f(x) \text{ subject to } x \in X \tag{2.1}$$

Where $f(x)$ is the *objective function*, defined by the problem formulation, and X the *solution space*. The optimisation process aims to reach a globally optimal solution $\bar{x}$ which fulfils

$$f(\bar{x}) \leq f(x) \, \forall \, x \in X \tag{2.2}$$

[1]Depending on the problem formulation the optimisation is either a maximisation or a minimisation process. In the following only minimisation is considered as all problem formulations used through this thesis are minimisation problems. However, all concepts can also be be applied to maximisation problems.

The time required to do so depends on the *problem instance*, the employed *solution methods* as well as the used computer infrastructure. In cases where the time to reach $\bar{x}$ is considered infeasibly long the aim is to approximate $\bar{x}$ as best as possible.

2.1.1 Solution Space and Constraints

X is the space of all possible solutions. For so-called *combinatorial problems*, which includes the UTRP, the solutions are combinations of a finite set of solution components. Although, in theory, this allows finding the optimal solution via an exhaustive search, the number of possible solutions makes this infeasible in most cases. The number of available solutions for a concrete application is defined by the respective *problem instance*, i.e. the set of input data for the concrete application.

The solution space defined by the instance can be reduced by *constraints* additional conditions a solution x has to fulfil to be considered feasible. Constraints are often introduced to exclude solutions which are considered unrealistic. Doing so speeds up the optimisation process, as less time is required to explore the reduced search space. However, constraints can also be forced by the construction of the optimisation algorithm, or the objective function. Such constraints are problematic, as they might exclude realistic solutions which are potentially superior solutions from the search.

2.1.2 Multi objective optimisation

Multi-objective optimisations are used for problems with several, often conflicting, objectives. This modifies equation 2.1 to

$$\text{minimise } f(g_1(x), g_2(x), \ldots, g_k(x)) \text{ subject to } x \in X \tag{2.3}$$

For multi-objective problems it is common that no single globally optimal solution can be found and instead solutions offering a good compromise need to be selected. In this regard two concepts are important: a) a solution $x \in X$ is considered

Pareto[2] *optimal* if it is impossible to change it without worsening at least one objectives. b) A solution $x_s \in X$ is called *dominating* another solution $x_p \in X$ if $g_i(x_s) \leq g_i(x_p) \, \forall i \in \{1, ..., k\}$. Some optimisation algorithms return a set of solutions which are non-dominant to one another and approximate Pareto optimality as close as possible[15].

Therefore, it is common that multiple objectives have to be combined into one function. Therefore, it is common that multiple objectives have to be combined into one function. This can be done in different ways. In the ϵ- constraint method maximal values ϵ_p are defined for all but one objective function. This objective function g_s is then minimised under the condition that none of the other objectives g_p exceeds its assigned ϵ_p. When using this method the function f is defined as

$$f(g_1(x), g_2(x), \ldots, g_k(x)) = \begin{cases} g_s(x) & \text{if } g_p(x) \leq \epsilon_p \, \forall \, p \neq s \\ \infty & \text{otherwise.} \end{cases} \tag{2.4}$$

This method is only used if one objective is clearly preferred over others. The alternative is the weighted sum method which assigns different weighting factors $w_s > 0$ to individual objective functions $g_s(x)$ to define the function f as

$$f(g_1(x), g_2(x), ..., g_k(x)) = \sum_{s}^{k} w_s g_s(x) \tag{2.5}$$

This method is very sensitive to the chosen w_s, however, if properly calibrated it can generate well-balanced solutions.

2.2 Graph Theory

Graphs are mathematical structures to model the pairwise interactions between objects. In urban planning, they are commonly used for the representation of transport networks. A *graph* $G = (N, E)$ is defined by a set of *nodes*[3] N and a set

[2]The term Pareto optimum and related terms (Pareto efficient, Pareto set, Pareto front, etc.) are named after the Italian engineer, and economist Vilfredo Federico Damaso Pareto (1848 to 1923) who first brought forward these concepts.

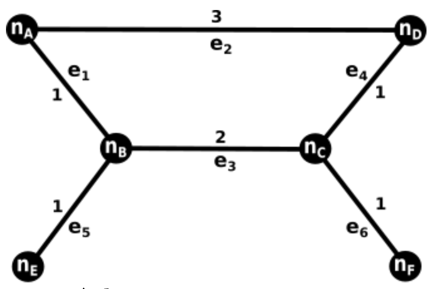
[3]Different terms for nodes and links are used throughout the literature. Alternative terms for nodes include *vertex*, *hubs*, and *points*. Alternative terms for links include *edges*, *arcs*, and *lines*.

of *links*[3] E. Both sets are finite, non-empty, and disjointed. Each link connects two nodes associated to it the incident function ψ: if $\psi(e_k) = (x_i, x_j)$ the link e_k connects the node x_i with the node x_j. This connection can be either *directed*, i.e. go only from x_i to x_j, or *undirected*, i.e. be equal form both directions. If at least on link $e_i \in E$ is directed, G is considered an directed graph. Otherwise G is called an undirected graph. Each link $e_i \in E$ can be associated with a *weight*. In graphs representing transport network, an often-used weight is the travel time along the links.

The number of links connected to a node are called its *degree*, denoted as $d_G(n_i)$. The set of nodes which is directly connected to a node n_i via at least one link is called the *neighbourhood* of n_i. In directed graphs, the degree of a node is separated in its in- and out-going degree. In the same way, the neighbourhood is separated into in- and out-neighbourhood.

In so-called multigraphs, there can be more than one link between the same pair of nodes. However, in all the graphs used within this thesis a more simple structure where two nodes are connected by maximally one link is used. The connectivity of such graphs can be described by its *adjacency matrix* A. This matrix is of size $|N| \times |N|$, where $|N|$ denotes the number of nodes. Its entries are binary. If $a_{i,j} = 1$, the nodes n_i are n_j adjacent, i.e. directly connected. A special form of the adjacency matrix is the weight matrix. Its entries give the weight of the respective link with the non-existing links being represented by a default value (e.g. ∞ in the case of the travel times). In undirected graphs, both the adjacency matrix and the weight matrix are symmetric.

A *path* through the graph is a sequence of nodes and connecting links. The length of a path is sometimes given in *steps*, i.e. the number of used links. However, if the weights of the links represent length measures (e.g. travel times) the length of a path is more commonly defined as the sum of the weights of the links it uses. A



$$N = \{n_A, n_B, n_C, n_D, n_E, n_F\}$$
$$E = \{\, e_1\, ,\, e_2\, ,\, e_3\, ,\, e_4\, ,\, e_5\, ,\, e_6\, \}$$

$$\psi(e_1) = (n_A, n_B)\, ,\ \psi(e_2) = (n_A, n_D)$$
$$\psi(e_3) = (n_B, n_C)\, ,\ \psi(e_4) = (n_B, n_C)$$
$$\psi(e_5) = (n_B, n_E)\, ,\ \psi(e_6) = (n_C, n_F)$$

Adjacency matrix

0	1	0	1	0	0
1	0	1	0	1	0
0	1	0	1	0	1
1	0	1	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0

Travel time matrix

-	1	∞	3	∞	∞
1	-	2	∞	1	∞
∞	2	-	1	∞	1
3	∞	1	-	∞	∞
∞	1	∞	∞	-	∞
∞	∞	1	∞	∞	-

Figure 2.1: Example for an undirected graph with 6 nodes and 6 links.

graph allowing trips between all pairs of its nodes is called a *connected graph.* If this is not the case the graph is considered separated into several *components* (or *islands*).

One of the most well-studied graph-based optimisation problems is that of finding the shortest path between pairs of nodes. Several standard algorithms for this task exist. One of these is Floyd's algorithm[4], outlined in algorithm 1 (on page 16). It determines the shortest path between all pairs of nodes in one run, making it useful to, for example, evaluate the travel times in graph representations of PuT networks. Another often used algorithm is Dijkstra's algorithm[19], which determines the shortest path from one origin node to all other nodes. It can be used as an all-pairs shortest path algorithm by repeating the operation for every node.

2.3 Computational time complexity

One important aspect for the practicability of an optimisation algorithm is its run time. Comparing the run times in computer experiments is in general not a sufficient approach, as the results can differ drastically between different problem instances and the used computer infrastructure.

[4]Floyd's algorithm is also known as Floyd-Warshall algorithm, Roy–Warshall algorithm, or Roy–Floyd algorithm. The reason these different names is that the algorithm was published three times in almost identical form by Roy in 1959 [16], Warshall in 1962 [17] and Floyd in 1962 [18].

Algorithm 1: Floyd's all-pairs shortest path algorithm.

```
Data: graph $G = (N, E)$ (includes travel times along edges)
Result: distance matrix $K$ (size $|N| \times |N|$)
begin
    // A) Initialisation
    $k_{i,j} \leftarrow \infty \forall \{n_i, n_j\} \in N$
    foreach $e_k$ in $E$ do
        $\Psi_G(e_k) \rightarrow (n_i, n_j)$
        $k_{i,j} \leftarrow t_k$ // travel time along $e_k$ written into entry $k_{i,j}$
    // B) Loop over all nodes
    foreach $n_k \in N$ do
        foreach $n_i \in N$ do
            foreach $n_j \in N$ do
                if $k_{i,j} > (k_{i,k} + k_{k,j})$ then
                    $k_{i,j} \leftarrow k_{i,k} + k_{k,j}$
    return $K$
```

A commonly used measure for the general estimation of run time is its time complexity. The (worst case) time complexity[5]$O(m)$ of an Algorithm gives the number of elementary operations which are maximally required to complete the Algorithm for an input of size m. For example, due to its triple loop over all nodes Floyd's Algorithm (see Algorithm 1) has a time complexity of $O(N^3)$, with n being the number of nodes. Such an Algorithm, which has a time complexity expressible by a polynomial function, is called a polynomial-time Algorithm.

The time complexity of Algorithms also allows classifying the problems they can solve. If a problem A can be solved by a (deterministic) polynomial-time Algorithm it is considered being part of complexity class $\mathcal{P}$. If, instead, a non-deterministic[6] polynomial-time Algorithm is required to solve A, it is considered being part of class $\mathcal{NP}$[7]. Other classes exist for problems requiring non-polynomial

[5]Time complexity is the most often considered type of computational complexity. Other types also exist, e.g. space complexity which estimates the required memory space.

[6]A non-deterministic polynomial-time Algorithm is a hypothetical polynomial-time Algorithm which can explore an unlimited number of solutions in parallel.

[7]It is at present not proven if $\mathcal{P} = \mathcal{NP}$ or $\mathcal{P} \neq \mathcal{NP}$.

time Algorithms[20].

Within the class $\mathcal{NP}$ is the subclass $\mathcal{NP}$-complete. Problems of this class can be transformed into any other problems in $\mathcal{NP}$. As a result, an Algorithm solving a $\mathcal{NP}$-complete problem can be adapted to solve any problem in $\mathcal{NP}$ [20]. The UTRP is one such $\mathcal{NP}$-complete problem, as proven in [21, 22].

2.4 Solution Methods

This Section introduces basic concepts on different types of algorithms, while their applications in the existing UTRP literature will be discussed in Section 3.1.

2.4.1 Mathematical Programming

Mathematical programming methods relay on strict mathematical formulations of both the objective functions and constraints. The field is subdivided into numerous classes. An example is linear programming where objective functions and constraints are given as linear functions. If some of the decision variables of the solution can be restricted to integers, the problem belongs to the subclass of mixed-integer linear programming. Other examples for classes include quadratic programming, for problems which can be described using quadratic functions, and stochastic programming, where objectives or constraints include stochastic variables.

For some classes, standardised algorithms exist which can be applied to all problems included in the class. One example is the simplex algorithm for finding solutions to linear programming problems [23].

2.4.2 Heuristics

Heuristics do not rely on strict mathematical problem formulation and the objective functions are only used for the evaluation of competing candidate solutions. As a result, heuristics are more flexible in the way they explore the solution space. In contrast to mathematical programming, heuristics seek less to find provably optimal solutions but rather a sufficiently optimal solution for the benefit of shorter search time [24].

Heuristics can be classified by way they explore the search space. While construction heuristics aim to build optimal solutions from given components, local search heuristics start with initial solutions and iteratively search through the neighbouring solution space to find more optimal solutions [25]. With the exceptions of meta-heuristics, heuristics are specific to the problem they are trying to solve.

2.4.3 Meta-heuristics

Meta-heuristics are heuristic frameworks which can be applied to different problems, following the additions of problem-specific operators. Due to the large number of meta-heuristics, this Section reviews only the concepts most often applied to solve the UTRP.

Improved Neighbourhood Search: The most basic local search algorithms always selects the neighbouring solution which most dominates the current solution. This leads to a fast search, however, it often stops in local optima. *Simulated Annealing (SA)* overcomes this disadvantage by accepting worsening solutions with a certain probability. Another modification is the concept of *Tabu Search (TS)* which always moves forward to the neighbouring solution with the lowest objective value. To prevent repetitions, TS is blocked from selecting various solutions, including those which have already been visited [25].

Genetic Algorithm: The basic concept of genetic algorithms (GA) is inspired by the concept of biological evolution: a set of solutions, the parent population, is used to generate new solutions by exchanging solution components. Afterwards these children solutions undergo mutations. Both parent and children populations are then combined and evaluated to select the best solutions as a new parent population for the next generation. This process continues for a fixed number of generations. A concrete implementation of a genetic algorithm is described in Chapters 4 and 5.

Swarm intelligence methods: Several types of meta-heuristics are inspired by self-organisation in groups of social animals. One such approach is *ant-colony optimisation (ACO)*, based on the principle of pheromone traces that ants use to find the best path from their nest to a food source. Individual ants are assigned to the different possible paths. The ant which completes the journey first will leave behind the strongest pheromone trace, attracting other ants which strengthens the pheromone trace further. Other swarm intelligence methods are *particle swarm optimisation (PSO)*, inspired by the behaviour of bird flocks, and *bee colony optimisation (BCO)* based on the way bee colonies determine the distribution of workers over available food sources [26].

2.4.4 Hyper-Heuristics

Hyper-heuristics are a relatively new category of solution methods. While regular heuristics manipulate the solutions directly, hyper-heuristics work on a higher level determining the lower-level heuristics which perform the manipulations. The different operational level allows the hyper-heuristic to function without direct knowledge of the concrete optimisation problem and allows simple adaptations to different problem domains. Hyper-heuristics can be classified into two main categories: while construction hyper-heuristics generate the necessary low-level heuristics from a set of components, selection hyper-heuristics select existing low-level heuristics from a given set [27]. Concrete implementations of the latter are described in Chapter 6 and Appendix A.

3
Review of Literature

This Chapter reviews the available literature on the UTRP. In total 132 publications were analysed and are summarised in Table 3.1. This review is designed to go both wider and deeper than respective Sections in the later Chapters. The reader is therefore invited to skip Section 1 in the Chapters 4 and 5, and Sections 1 and 2 in Chapter 6.

The review includes publications on the UTRP[1] as well as publications on the combined optimisation of routes and other phases of PuT network design, most often the optimisation of frequencies. In the latter case, only the parts for the optimisation of routes are discussed. Also included are variations such as the E-TNDP, which adds the optimisation of the battery location [34], or the “Rapid transit network design problem" (RTNDP), which includes constructions costs for rail infrastructure into its objectives [35, 36]). Not included are publications on related yet distinct problems such as the "Feeder Bus Network Design Problem" (e.g. [37, 38]), or the "School Bus Routing Problem" [39].

[1] The literature knows the same problem formulation also under other name and acronyms. One often-used name is the “Transit Network Design Problem" (TNDP) (used e.g. in [28–30]). Other, less-common names include “Bus Network Design Problem" [31], “Bus Transit Route Network Design Problem" [32], and “Urban Transit Network Design Problem"[33].

Most of the listed publications are journal and conference papers. However, technical reports and thesis were included if no other publications of the same author could be found covering the work in question. Publications were only considered when all the required information could be identified. Therefore, publications were excluded if they could not be found in full-text or did not include the required information. The review does not claim to be complete.

3.1 A brief history of applied solution methods

The first recognised work on the UTRP was published in 1925 by Patz [40]. This first approach uses linear programming to select the best combination of candidate routes for a tram network represented by a simple graph with 10 nodes and 9 links [22]. Since then, several more mathematical programming approaches have been published with most of them using mixed-integer linear programming. Nevertheless, with just 20 studies (15%), such approaches remain relatively few in number, compared to heuristic and meta-heuristic approaches. In the past, one hindering factor the limited applicability on larger instances due to long run-times[41]. However, in recent years some approaches have been applied to at least medium size instances based on real-world examples (see e.g [35, 42, 43]).

Probably the first heuristic approach to the UTRP was published in 1961 by Nebelung et al., [44]. Over subsequent decades many more were added as can be seen in figure 3.1. In total, Table 3.1 lists 26 publications (20%) which only use heuristics (excluding meta-heuristics). Most of these are construction heuristics. Usually, these first generate a palette of candidate routes based on shortest path criteria and then assemble and/or recombine these routes in an iterative process (examples for such studies are [45–47]). Approaches relying only on construction heuristics have declined since the 1990s. However, such algorithms are still used to generate the required initial solutions as part of optimisation procedures centred on meta-heuristic local search approaches.

Table 3.1: Summary of reviewed publications. Description of columns:
• "Method" - solution methods (see 3.1): MP - Mathematical Programming, Heu - Heuristic, GA - Genetic Algorithm, SA - Simulated Annealing, TS - Tabu Search, ACO - Ant Colony Optimisation, BCO - Bee Colony Optimisation, ACO - Ant Colony Optimisation, PSO - Particle Swarm Optimisation, oHM - other Meta-Heuristics, HH - Hyper Heuristics; "+" - combination , "/" - comparison.
• "Instance" - instance type (see 3.2.2): fi - fictional instance, ui - urban instance, ri - regional instance, [reference] if instance was introduced in other publications, if "()" instance is not public.
• "Terminal" - if publication uses restricted terminal nodes (see 3.2.2).
• "Objectives" - category of objectives used (see 3.3.1): PT - Passenger Travel Time, OC - Operator Cost, Tr - Passenger Transfers, Cv - PuT network coverage, oth - other.
• "TR" - used trip representation (see 3.3.2): N - node-based, Zn - zone-based using standard shortest path algorithms, Za - zone-based using other assignment algorithms, Zp - zone-based interfacing with transport modelling software, Z* - zone-based using other method.

Ref.	Year	Authors	Method[2,3]	Instance	Term.	Objective[2]	TR
[40][4]	1925	Patz et al.	MP	fi		OC	N
[44][5]	1961	Nebelung	Heu	ui	✓	PT	N
[51]	1967	Lampkin & Saalmans	Heu	ui		PT, Tr	N
[52][6]	1967	Mueller	MP	ui-[44]	✓	PT	N
[45]	1974	Silman et al.	Heu	ui	✓	PT, OC	N
[46]	1979	Sonntag	Heu	ui-[44]	✓	PT, Tr	N
[54]	1979	Dubois et al.	Heu	ui		PT	N
[53]	1979	Mandl	Heu	ri		PT, Tr	N
[55]	1984	Marwah et al.	Heu	ui	✓	PT	N
[10]	1986	Ceder & Wilson	Heu	fi		PT, Tr	N

[2]For many publications the information on solution methods and objectives were taken from of the review papers [12, 14, 48, 49]

[3]Not listed in this column is the use of constructions heuristics for the generation of initial solutions are not listed as combinations. Such initialisations are required for most meta-heuristic approaches, most especially GA approaches.

[4]Original not found, information taken from [22]

[5]Original not found, information taken from [46, 50]

[6]Original not found, information taken from [46]

[56]	1987	Oudheusden et al.	MP	fi		OC	N
[57]	1988	vanNes et al.	Heu	ui		Tr	Z*
[28]	1991	Baaj & Mahmassani	Heu	ri-[53]	√	PT, OC, Tr	N
[58]	1992	Xiong & Schneider	GA	fi		PT	N
[47]	1995	Baaj & Mahmassani	Heu	ui	√	PT, Tr	N
[59]	1995	Israeli & Ceder	MP	fi	√	PT, OC	N
[50]	1995	Pape et al.	Heu	ui-[44]		Tr, oth	N
[60]	1997	Chien & Schonfeld	MP	fi		PT, OC	Zn
[66]	1998	Ceder & Israeli	Heu	ui		PT, OC	N
[64]	1998	Bussieck	MP	ri		OC, Tr	N
[61]	1998	Pattnaik et al.	GA	ri	√	PT	N
[62]	1998	Bruno et al.	Heu	fi-[63]		OC, Cv	Zn
[65]	1998	Shih et al.	Heu	ui	√	PT, Tr	N
[67]	1998	Bielli et al.	GA	ui		PT, OC	N
[68]	1999	Soehodo & Koshi	Heu	ui-[69]	√	PT, OC, oth	N
[72]	2002	Carrese & Gori	Heu	ui		PT, OC	N
[71]	2002	Fusco et al.	Heu	ri-[53]		PT, OC	N
[74]	2002	Chakroborty & Dwivedi	GA	ri-[53]		PT, Tr	N
[73]	2002	Bielli et al.	GA	ri-[53], ui		PT, OC	Za
[70]	2002	Chien & Spasovic	MP	fi		PT, OC	Zn
[76]	2003	Tom & Mohan	GA	ui		PT, OC	N
[75]	2003	Chakroborty	GA	ri-[53]		PT, Tr	N
[77]	2003	Ngamchai & Lovell	GA	fi		PT, OC	N
[22]	2003	Quak et al.	Heu	fi, ui	√	PT, OC	N
[78]	2003	Wan & Hong	MP	fi		OC	N
[80]	2004	Gao et al.	MP	fi-[81]		PT, OC	N
[82]	2004	Agrawal & Mathew	GA	ui		PT, OC	N
[83]	2004	Zhao & Ubaka	SA+TS	ri-[53], ui		Tr	N
[79]	2004	Petrelli et al.	GA	ui		PT, OC	Zp

[90]	2005	Bachelet et al.	MP	ui		PT	N
[85]	2005	Cipriani et al.	GA	ui		PT, OC	Zp
[84]	2005	Hu et al.	ACO/GA+SA	ui	✓	PT, OC	N
[87]	2005	Zhao et al.	SA+TS	ri-[53], ui(-[83])		Tr	N
[86]	2005	Yu et al.	ACO	ui	✓	Tr	N
[88]	2005	Lee & Vuchic	Heu	fi-[89]		PT, OC	N
[95]	2006	Cipriani et al.	GA	ui	✓	PT, OC	Zp
[98]	2006	Guan et al.	MP	ui	✓	PT, OC, Tr	N
[92]	2006	Fan & Machemehl	GA	fi		PT, OC, Cv	Zn
[94]	2006	Fan & Machemehl	SA	fi(-[92])		PT, OC, Cv	Zn
[96]	2006	Zhao & Zeng	GA+SA	ri-[53], ui(-[83])		Tr	N
[97]	2006	Yu & Yang	ACO	ui(-[86])	✓	Tr	N
[91]	2006	Zhao & Zeng	GA+SA	ri-[53], ui(-[83])		PT	N
[93]	2006	Zhao	SA	ri-[53], ui		PT	N
[100]	2007	Zhao & Zeng	SA+TS	ri-[53]		PT, OC	N
[101]	2007	Yang et al.	ACO	fi, ui(-[86])	✓	Tr	N
[99]	2007	Barra et al.	MP	ri-[53]		PT, OC	N
[104]	2008	Zhao & Zeng	SA+TS	ri-[53]		PT	N
[103]	2008	Fernandez et al.	Heu	ui		PT	Zn
[102]	2008	Borndorfer et al.	Heu	ui	✓	PT, OC	N
[105]	2008	Fan & Machemehl	TS	fi(-[92])		PT, OC, Tr	Zn
[106]	2009	Wan & Hong	MP	fi		Tr	N
[107]	2009	Marin & Garcia-Rodenas	MP	fi-[108]		Cv, oth	Zn
[108]	2009	Marin & Jaramillo	MP	fi, fi, ui		PT, OC, Cv, oth	Zn
[111]	2009	Mauttone & Urquhart	Heu	ui		PT, OC	N
[112]	2009	Fan et al.	oMH	fi(-[92]), ri-[53]		PT, OC	N
[110]	2009	Mauttone & Urquhart	oMH	ui(-[111])		PT, OC	N
[109]	2009	Beltran et al.	GA	fi, ri-[53]		PT, OC	N
[33]	2009	Pacheco et al.	TS	ui	✓	PT	N

[113]	2010	Fan & Mumford	SA	ri-[53]		PT, OC	N
[6]	2010	Walter	Heu	ui		oth	Zp
[114]	2010	Blum & Mathew	ACO	ui	√	PT, OC	N
[117]	2011	Szeto & Wu	GA	ui		PT, Tr	N
[116]	2011	Bagloee & Ceder	GA+ACO	ui, ui		PT, Tr	Zp
[31]	2011	Poorzahedy & Safari	ACO	ui-[69], ui(-[31])	√	PT	Zp
[115]	2011	Fan & Machemehl	GA	fi(-[92])		PT, OC, Cv	Zn
[119]	2011	Marauli	Heu	ui		oth	Zp
[118]	2011	Alt & Weidmann	ACO	ri-[53], ui	√	PT, OC	Zp
[120]	2012	Shimamoto et al.	GA	fi	√	PT, OC	N
[169]	2012	Sadrsadat et al.	GA	ui	√	PT	Zp
[125]	2012	Yu et al.	ACO	ui(-[86])	√	Tr	N
[124]	2012	Roca-Riu et al.	TS	ui	√	PT, OC	Zn
[123]	2012	Cipriani et al.	GA	ui		PT, Tr	Zp
[122]	2012	Szeto & Jiang	BCO	ui	√	PT, Tr	Za
[121]	2012	Chew & Lee	GA	ri-[53]		PT, Tr	N
[131]	2013	Chew et al.	GA	ri-[53]		PT, OC	N
[43]	2013	GutierrezJarpa et al.	MP	ui		PT, OC	N
[126]	2013	Afandizadeh et al.	GA	ri-[53], ui		PT, OC, Tr	N
[29]	2013	Jiang et al.	BCO	ui	√	PT, Tr	N
[128]	2013	Nikolic & Teodorovic	BCO	fi, ri-[53]		PT, Tr	N
[7]	2013	Mumford et al.	oMH	fi, ui, ui, ui		PT, OC	N
[129]	2013	Yan et al.	SA	ri-[53]		OC	N
[130]	2013	Walteros et al.	GA	ui		PT, OC	N
[141]	2014	Kechagiopoulos & Beligiannis	PSO	ri-[53]		PT, Tr	N
[138]	2014	Yao et al.	TS	fi		Tr	N
[136]	2014	Cooper et al.	GA	fi-[7], ui		PT, OC	N
[137]	2014	John et al.	GA	fi-[7]		PT, OC	N
[132]	2014	Owais et al.	GA	ri-[53]		PT	N

[142]	2014	Nikolic & Teodorovic	BCO	ri-[53]		PT, OC, Tr	N
[143]	2014	Nayeem et al.	GA	fi-[7], ri-[53]		PT, OC, Tr	N
[140]	2014	Xu et al.	SA	ui		PT, OC	N
[144]	2014	Amiripour et al.	GA	ui(-[135])	✓	PT, OC	N
[135]	2014	Amiripour et al.	GA	ui	✓	PT, OC	N
[134]	2014	Szeto et al.	BCO	ui, ui	✓	Tr	Za
[133]	2014	Kilic & Gok	Heu	fi-[7], ri-[53]		PT, OC, Tr	N
[139]	2014	Amiripour et al.	GA	ri-[53], ui		PT, OC	N
[147]	2015	Rahman et al.	GA	fi-[7], ri-[53]		PT, Tr	N
[41]	2015	Cancela et al.	MP	fi, ri-[53]		PT	N
[30]	2015	Arbex & da Cunha.	GA	ri-[53]		PT, OC, Tr	N
[146]	2015	Zhao et al.	oMH	ri-[53]		PT, Tr	N
[145]	2015	Liu et al.	oMH	ui		PT	N
[148]	2015	Pternea et al.	GA	ui		PT, OC, Tr, oth	N
[149]	2016	Wu & Wang	oMH	ri-[53]		PT, Tr	N
[151]	2016	Buba & Lee	oMH	ri-[53]		PT	N
[150]	2016	Owais et al.	GA	ri-[53]		PT, OC	N
[21]	2016	John	GA	fi-[7], ui, ui		PT, OC	N
[153]	2017	Khakbaz et al.	GA	ui		Tr	N
[36]	2017	LopezRamos et al.	GA	ui, ui		PT, OC	N
[152]	2017	Cadarso et al.	MP	fi-[108]		OC, Cv	Zn
[35]	2017	GutierrezJarpa et al.	MP	ui(-[43])		PT, OC	N
[154]	2018	Chu et al.	MP	ri-[53]		PT, OC, Tr, Cv	N
[156]	2018	Huang et al.	BCO	ri-[53], ui	✓	PT, OC	N
[157]	2018	Buba et al.	oMH	ri-[53]		PT, Tr	N
[155]	2018	Owais & Osman	GA	ui(-[111])		PT, OC	N
[34]	2019	Iliopoulou et al.	PSO	fi-[7]		PT, OC	N
[159]	2019	Islam et al.	oMH	fi-[7], ri-[53], ui		PT, Tr, Cv	N
[158]	2019	Jha et al.	GA/PSO	ri-[53]		PT, Tr	N

[166]	2019	Ahmed et al.	HH	fi-[7], ri-[53]		PT, OC	N
[161]	2019	Feng et al.	GA	ui		PT	N
[163]	2019	Bourbonnais et al.	GA	ui, ui, ui	$\surd$	PT, OC	Za
[160]	2019	Kim et al.	GA	ri-[53]		PT, OC	N
[164]	2019	Mahdavi Moghaddam et al.	GA	ri-[53]		PT, OC	N
[165]	2019	Duran et al.	GA	fi-[41], fi-[78], ri-[53]		PT, Tr, oth	N

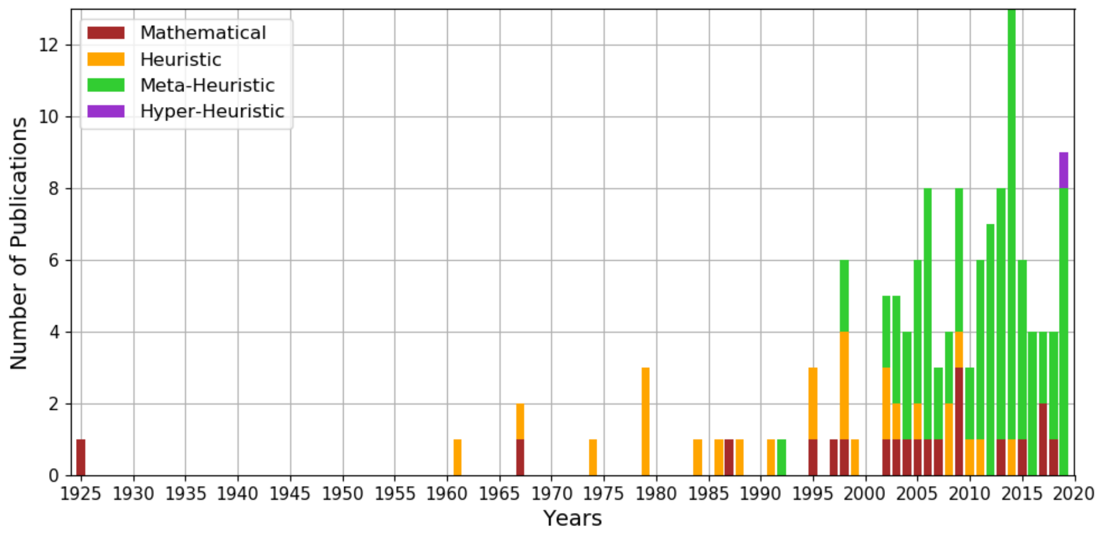


Figure 3.1: Number of publications in Table 3.1 for every year summarised by solution method. Publications which combine several solution methods in one study are counted multiple times.

Meta-heuristics were first used in studies on the UTRP in the 1990s. They quickly grew in importance and have dominated the field since the 2000s (see figure 3.1). From the publications listed in Table 3.1, 85 (64%) use some kind of meta-heuristic. By far the most commonly used type of meta-heuristics is GA (48 publications). This includes the combination of GAs with other techniques, e.g. as additional improvements through neuronal networks in [58], or combinations with other meta-heuristic types, e.g. the combinations GA and ACO in [37, 116]. Other commonly applied meta-heuristic concepts are SA (12 publications), ACO (9 publications), TS (8 publications), BCO (6 publications), and PSO (3 publications). Further, nine publications use other meta-heuristics. These include, for example, a memetic algorithm in [146] and discrete wolf pack search in [149].

Only a single publication ([166]) could be identified which uses a hyper-heuristic approach. However, this excludes the publications on two hyper-heuristic approaches which are presented in Chapter 6 and Appendix A in this thesis.

3.2 On UTRP Instances

3.2.1 Basic components of UTRP Instances

The problem instances for the UTRP generally have at least two components: a graph which represents the available transport infrastructure and the information of the transport demand.

In a UTRP graph, also termed instance graph, the nodes are treated as PuT access and interchange points while the links represent connecting infrastructure (e.g. streets for bus travel, or tram rails). With such a graph, the public transport PuT routes can be represented as a series of directly connected nodes. Usually, these routes are considered undirected by assuming that the vehicles turn around after reaching the final node and start the journey again in the opposite direction. The alternative, using directed half route for the different directions of travel, would effectively double the number of routes considered and therefore drastically increase the complexity of the problem. Further, it would likely lead unwelcome divergence between the course of two directions of the same route. For these reasons, routes and UTRP graph are usually undirected. The possible solution for the UTRP is given by a set of routes [12, 14].

Most[7] publications listed in Table 3.1 use the above described graph concept as basis for their work. In some studies the graph structure is extended by, for example, adding information on other modes of transport (e.g. in [106, 152]), information on potential terminal nodes (discussed in Section 3.2.2), or the addition of zones and connectors (discussed in Section 3.3.2).

The travel demand is provided in the form of origin-destination matrices giving the number of trips between pairs of nodes, or pairs of zones, in a given time. Some

[7]Exceptions are as follows: the route proposal algorithm used in [6, 119] only determines the end points of routes and leave their design to a transport modelling software. Further, the approach in [33] does split routes into directed half routes. These approaches will be discussed in more detail in the Sections 3.3.3 and 3.4 respectively.

instances include additional information such as a limited number of terminal nodes, i.e. nodes which can be used as beginning or end nodes of routes.

3.2.2 Fictional and Real-World Instances

In Table 3.1 the instances used in the listed studies are separated into three categories: fictional instances (fi), which are purely theoretical, regional instances (ri) which are based on real-world scenarios of inter-urban travel, and urban instances (ui), which are based on real-world inner-urban travel.

Fictional instances are usually comparatively rather small, with the largest ones having 30 nodes [7, 138]. In total, the publications listed in Table 3.1 introduced 21 different fictional instances. These are used especially often in studies employing mathematical programming approaches. However, since this area of research is increasingly shifting towards the use of larger urban instances, the number of studies solely relying on fictional instances has drastically declined.

Urban instances form the largest of the three groups. Over the past decades, urban areas from around the world have served a basis of such instances. Examples include Cardiff (UK) [136], Ahmedabad (India) [167], Nanjing (China) [156], Austin (USA) [47], Bogota (Colombia) [130], and many more. These instances hugely vary in their size and level of detail, with the simplest ones not being substantially different from larger fictional instances. However, to show that an algorithm can be used in urban planning processes, they need to be applied to instances resembling real-world urban areas in sufficient possible detail. This also includes urban specific constraints and conditions.

One such urban specific condition is the use of zone-based trip representation, which will be discussed in Section 3.3.2. Another is the restriction of route endpoints. As mentioned in Section 3.2.1, it is assumed that vehicles, after finishing a journey along a route, perform a U-turn and start the journey in the opposite direction[8]. In urban areas, it is a common situation that the possibilities for vehicles to perform U-turns is heavily restricted. Consequently, it can not be assumed that such a manoeuvre are possible at every node of the UTRP graph. Therefore, instances representing urban areas should specify which of their nodes are terminal nodes, i.e. nodes in the vicinity of which vehicles can perform U-turns [46, 168]. However, from the 66 urban instances found in the reviewed publications only 22 offer information on terminal nodes. Of these, only one is publicly available (see Section 3.2.4).

In regional instances, where the nodes usually represent whole cities, such details are not necessary. With only three identified instances, this group is by far the smallest. Nevertheless, one of these instances, introduced by Mandl in [53] and based on the national road network of Switzerland, is the most often used public instance (see Section 3.2.4). Even studies which specifically aim to develop algorithms for use in urban scenarios often only use this instance (e.g. [71, 154, 160]).

3.2.3 Generation of Instances

As mentioned in Section 1.2, one of the hurdles for the practical application of UTRP algorithms is the absence of generation procedures for urban instances. Although a large number of urban instances exists, there are only very few publications describing the generation of their instances in detail. These approaches can be separated into three categories.

Some instance generation procedures are based on the spatial layout of the urban street (or rail) network. In [111], the generation of the instance is based on the street network of Rivera (Uruguay). Each of the 84 nodes represents a 400m×400m square of housing blocks. They were placed on street junctions closest to the block centre and the shortest path links connect the nodes of adjacent blocks. Unfortunately, this procedure is only applicable urban areas built in a strict grid

[8]Only few studies (e.g. [21, 36, 159]) mention this assumption explicitly. However, it is implicit for all studies which use undirected routes (see Section 3.2.1).

pattern, it could not be applied to cities built with an irregular layout, such as most European cities. Another method is outlined in [31, 169] for the generation of an instance representing Mashhad (Iran). It places nodes on street junctions 700m to 1km apart and connects those on adjacent junctions with shortest path links. Terminal nodes are selected based on optimal population coverage. This method is described in less detailed, but it seems it requires a very regular street layout to determine adjacency relations between nodes.

Other procedures define node location based on the demand structure. In [127][9] both demand and node locations are derived from taxi GPS data from Hangzhou (China). Pick-up and drop-off points are aggregated in 500m grid cells which are themselves aggregated in clusters based on their proximity and usage. In each cluster, a cell is selected as node-location. The demand between clusters and travel times between the nodes are also determined by taxi travel data. In the generation procedure described in [170] the position of nodes is optimised with a GA considering on the demand structure in the study area. Node adjacency is determined based on the angle between direct connections and travel times between adjacent nodes are calculated as Euclidean distances. Terminal nodes have to be selected manually. An instance generation with this method was used [118].

Finally, there are procedures which determine node position based on those of real-world PuT stop points. Due to the large number of stop points in most study areas, it is usually infeasible to use all stop points as a nodes. Therefore, different methods have been proposed to select a subset of the available stop points as graph nodes. In [21] this selection is made at random while ensuring a minimal distance between the selected stop points. The final number of nodes is set by the user. Also the connectivity and travel demand between the nodes were determined via randomised processes. The node selection in [116] is based on stop points who

[9][127] is not listed in Table 3.1 as it deals the design of a single bus route, rather than a route network. However, it is discussed here, as the described instance generation procedure could also be used to generate instances for UTRP algorithms.

are ranked by their expected travel demand. All stop points which are not within 300 meters around another stop point with higher expected demand are selected. Both procedures can easily be applied to different urban areas. However, as the location of the selected stops within the transport network is not fully taken into account the resulting graph is likely to insufficiently reflect the study areas spatial layout [21]. Further, both methods do not include a procedure to determine realistic adjacency relations. The procedure most recently introduced in [163] bases the stop point selection on both demand structure and the layout of major streets. Unfortunately, its parameters are not described in great detail. In this model, all nodes are considered adjacent and travel times between them are generated based on shortest path searches.

It can be concluded that the existing instance generation procedures are not well suited to applications on irregular urban street or rail networks. While layout-based methods rely on regular layouts and optimally grids to work accurately, methods based on stop point selections or demand patterns lack procedures to determine the correct adjacency relationships between the selected node positions. Also, none of the procedures includes an infrastructure-based process to determine terminal nodes. Such an instance generation is in Chapter 4 of this thesis.

3.2.4 Publicly Available Instances

Most researchers understandably prefer to focus their research efforts on solutions methods rather than the generation of instances. Therefore, many researchers use instances which have been fully published in earlier publications. The additional advantage of this is that it allows comparing the results of different studies using the same instance. However, the problem with this approach is the limited variety of instances publicly available.

Table 3.2: Publicly available instances used in the publications listed in table 3.1. Description of columns:
"Ref" - reference of publication first publishing the instance;
"Name" - term most commonly used term when referring to this instance;
"Represents" - real world area the instance is based on ("-" for fictional instances);
"$|N|$", "$|E|$","$|U|$" - Number of nodes, links, and potential terminal nodes (subset of N);
"num" - Number of reviewed publications using the instance;
"Ref" - reference of publication first publishing the instance.

Ref.	Year	Name	Represents	num	$\|N\|$	$\|E\|$	$\|U\|$
[53]	1979	Mandl Instance	Swiss road network	45	15	21	-
[7]	2013	Mumford3	Cardiff (UK)	9	127	425	-
[7]	2013	Mumford1	Yubei (China)	9	70	210	-
[7]	2013	Mumford2	Brighton (UK)	8	110	385	-
[7]	2013	Mumford0	-	6	30	90	-
[44][13]	1961	Nebelung Network	Düsseldorf (Germany)	4	32	33	19
[69]	1975	Sioux Falls Network	Sioux Falls (USA)	2	23	38	-
[78]	2003	Example Network	-	2	10	19	-
[108]	2009	R2	-	2	8	15	-
[41]	2015	Small Instance	-	2	8	10	-
[108]	2009	R1	-	2	6	9	-
[138]	2014	Medium Size Network	-	1	30	59	-
[77]	2003	Testing Model	-	1	25	39	-
[63]	1987	Sample Network	-	1	21	39	-
[101]	2007	Example Network	-	1	17	29	3
[89][11]	1972	Rea's Network	-	1	16	34	-
[106]	2009	Multi-modal Network	-	1	13	26	-
[10]	1986	Example Network	-	1	5	7	-
[81]	1993	Transit Network	-	1	4	6	-

Table 3.2 lists all publicly available instances[10] used in the reviewed publications. The collection shows mostly small fictional instances which were rarely reused. The fictional instances from [63, 81, 89][11] were originally designed for other routing problems. The same is true for the Sioux Falls instance from [69].

The instance used most often is the one published by Mandl in [53]. About one-third of the publications listed in Table 3.1 use this instance. Its popularity allowed the authors of [14] to compare the performance of meta-heuristic approaches.

[10]Not included are instances which were your multiple times by the same researchers or were passed on through institutional links(see "()" notation in Table 3.1).

[11][89] could not be found online, but instance was reprinted in [88].

Unfortunately, it is not an urban instance but based on the national road network of Switzerland. Further, it is relatively small. In 2013 Mumford published four more instances in [7], three of which are urban instances[12]. These are the largest instances which have been published so far.

Terminal nodes only feature in two of the available instances. One is a small fictional instance introduced in [101], the other is an instance already published in 1961 by Nebelung based on the tram network of Düsseldorf (Germany) [44][13]. It was mainly used in German publications. Today it would no longer be considered a large instance.

In the past, several researchers have called for the publication of larger and more realistic instances [7, 99]. For this reason, all instances introduced throughout the following Chapters will be made public.

3.3 Evaluating route networks

3.3.1 Objectives

The result of an optimisation process is largely influenced by the chosen objectives. Over the years, many different objectives for the UTRP have been explored in various studies. In Table 3.1 these have been grouped into three main types: operator costs (OC), passenger travel times (PT), passenger transfers (Tr) and PuT network coverage (Cv) as outlined below. Additionally, there are a few other objectives used in more specialised problem definitions, such as the reduction of air pollution in [148, 165], or travel demand taken up by newly added routes [6, 119].

[12]For the generation of the instances introduced in [7], only the number of nodes and their level of connectivity were extracted from the urban areas they represent. Node positions, demand matrix, and links were generated via randomised processes. The instances therefore do not capture the spatial layout of these areas in any detail.

[13][44] could not be found online, but instance was reprinted in [46].

Calculating the real cost of operating a PuT network is difficult and requires a large number of techno-economic variables, such as fleet composition and vehicle crowding [171, 172]. Correctly determining such detailed information is beyond the scope of most UTPR studies. Instead, the 74 publications considering the operator costs use very different measures for their estimations. Some studies used the combined length of all the routes as a simple approximation (e.g. [7, 21, 166]). In studies which combine the optimisations of routes with that of vehicle frequencies, operator costs are typically represented by the required fleet size (e.g. [73, 139, 154]). Further, in studies focusing on rail-based systems, infrastructure costs are considered (e.g. [35, 43, 62]). Also included in this category are estimations of unused vehicle capacity (e.g. used in [40, 51, 68]).

Passenger travel time is used in the evaluation process of 109 publications. Most common is the average travel time, but also the total travel time is used [14]. The exact approaches to determine the passenger path, and thereby their travel times, differ. One differing factor is the way transfer times are considered. Few publications (e.g. [106]) only consider connections without transfers. Other studies ignore transfer times when assigning passenger paths and either calculate the total transfer times separately (e.g. [98]) or evaluate transfers in other ways as discussed below. Such approaches result in fast run times for the travel time calculation; however, they are less accurate as transfer times can have a large impact on the path a passenger chooses. Other studies (e.g. [7, 53, 135]) include transfers by duplicating nodes which are used in several routes and inserting links representing possible transfers between them. This approach will be discussed in more detail in Chapter 5 - Section 2.2. Another important factor for the calculation of travel times is the representation of trips which will be discussed in Section 3.3.2.

Passenger transfers as an objective separate from the travel times are used in 47 of the reviewed publications. This includes minimising the total number of transfers required by all passengers (e.g. [142, 143, 167]), or to maximise the number of passengers which can reach their destination without transfers (e.g. [83, 91, 96]). Another common transfer-based objective is to minimise unsatisfied demand defined as number or percentage of passengers which require ore than a defined limit of transfers t complete their journeys. Usually, three or more transfers are assumed unsatisfactory [14]; however, a two-transfer limit is used in some cases (e.g. [135, 139, 144]).

Finally, nine studies allow the designed PuT network to cover only a part of the available trips but use this coverage as an objective. In five of these, e.g. [92, 115, 159], the number of travellers with no access to PuT is to be minimised. Further, four studies ([62, 107, 108, 152]) provide travellers with the option to travel with alternative modes such as walking or private cars. In these studies, one objective is to maximise the number PuT passengers.

3.3.2 Node-based and zone-based trip representation

The way trips and travel demand are represented has a big impact on determined passenger path and, therefore, travel times and number of transfers. Figure 3.2 illustrates this by depicting the two approaches used in the reviewed publications.

In 105 of the 132 reviewed publications, the travel demand is given as trips between pairs of nodes, as shown on the right side of Figure 3.2. This node-based approach assumes that travellers begin and end their PuT trips on the same pair of nodes independent from the available routes. One possible reason for the popularity of this concept is the availability of instances, as all publicly available instance identified during this review (see Table 3.2) are node based. Another likely factor is that node-based approaches are more straightforward to implement and allows to calculate the passenger travel times efficiently with standard shortest path algorithms (see Section 2.2). However, the fact that the nodes are also the demand sources results in parts of the travel demand not being covered when the designed PuT network does not include all of the nodes. An uncovered demand complicates

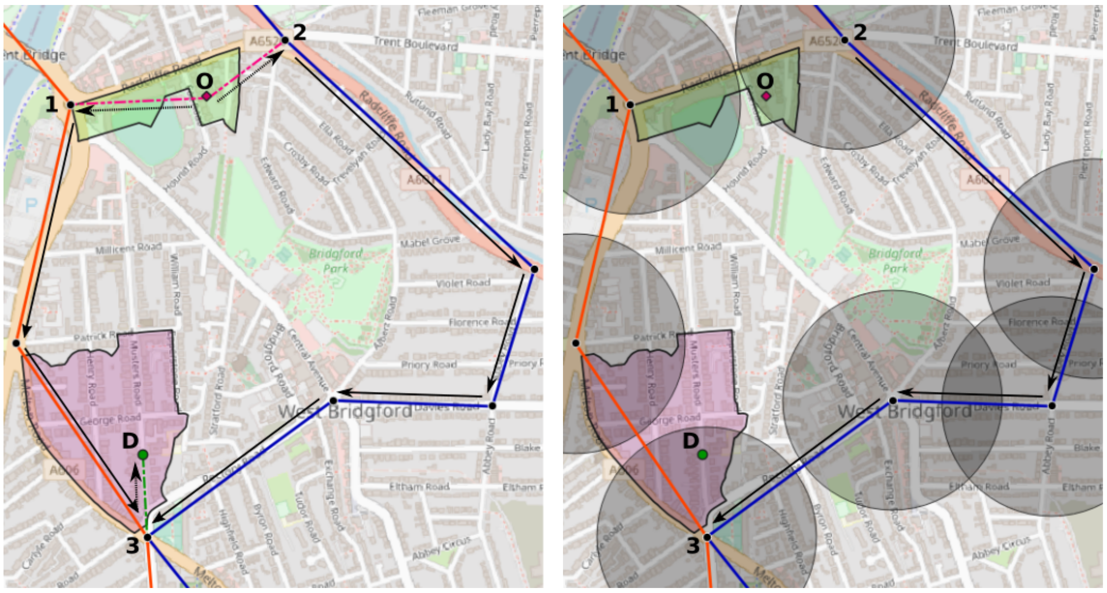


Figure 3.2: Example of the impact of different trip representations on the trip options. Assumed is a trip between two zones centroids O and D. Two PuT routes can be used: "orange" over nodes 1 and 3, and "blue" over nodes 2 and 3. (Figure and caption are slightly variations of figure 1 in chapter 5)
In a zone-based approach (left) travellers can choose from two options: a) walk from O to 1, ride on route "orange" to 3, walk on destination connector from 3 to D. b) walk from O to 2, ride on route "blue" to 3, walk on from 3 to D.
In a node-based approach (right) the trip is assigned to nodes based on their catchment areas (grey circles). The travellers can only travel with route "blue" from 2 to 3.

.

the evaluation of route sets, as it has to be excluded from the calculation of the travel time- or transfer-based objectives and instead needs to be represented separately via coverage objectives. Of the 105 reviewed node-based studies, only two ([154, 159]) allowe the exclusion of nodes. All of the other studies instead constrain the route sets to include all the nodes. Unfortunately, this constraint can exclude route sets which do not include all the nodes even though they might be favourable in reality. Further, it restricts comparisons between the optimised and the existing PuT networks, as will be discussed in Chapter 4.

The alternative approach is to divide the study area into zones, each of them represented by a centroid. These centroids are connected with the nodes of the UTRP graph via connectors, allowing them to represent a trip between two zones as a path between the respective centroids. This approach allows passengers to choose the nodes where they start and end their PuT journeys on the basis of the available routes.

The concept of zone-based trip representations offers several advantages. In most cases, it drastically simplifies the generation of demand matrices[14]. Zones can further serve as the basis for trips of both public and private modes. This is essential for other modelling procedures such as trip distribution and mode choice models. These are an integral part for many more complex transport modelling processes, e.g. the standard four-step model [173–175]. Consequently, zone-based approaches are the most common concept in macroscopic transport modelling and, by extension, the basis of many real-world planning processes. It is, therefore, important that more UTRP algorithms be developed using a zone-based trip representation.

At least if the zones are sufficiently small in size, a zone-based approach is a more realistic concept of trip representation. This is particularly true in an urban setting where passengers often have access to several stop points (see Figure 3.2). It also allows for a more straightforward generation of demand matrices[15].

From the 132 studies listed in Table 3.1, only 27 use a zone-based trip representation. The models in question are often described as a bi-level concept. The upper level, represented by the UTRP graph, is where the PuT network is designed. The lower level is represented by a travel graph made of centroids and connectors, as well as nodes and the PuT connections given by the current route set. This

[14]Most travel demand data sources, such as surveys or mobile phone data, produce origin-destination data based on zones. For a node-based demand matrix, these have to be aggregated on node-level using a node catchment area (as illustrated in Figure 3.2). Even when node-based demand data can be directly generated (e.g. by accessing boarding data from smart card ticketing systems), using this data for network optimisation is not ideal as it comes with an undesirable bias towards the existing PuT network.

[15]Most travel demand data sources, such as surveys or mobile phone data, produce origin–destination data based on zones. For a node-based demand matrix, these have to be aggregated at the node level by using a node catchment area (as illustrated in Figure 3.2). Even when node-based demand data can be directly generated (e.g. by accessing boarding data from smart card ticketing systems), the use of these data for network optimisation is not ideal, as they come with an undesirable bias toward the existing PuT network.

travel graph is used to assign passengers travelling between specific zones to a path through the PuT network. A variety of methods can be used for this task. The once used in zone-based studies listed in Table 3.1 can be classified in three categories[16].

There are ten studies which use interfaces to professional macroscopic transport modelling software to execute the assignment. These approaches are denoted as "Zp" in Table 3.1 and will be discussed in Section 3.3.3. Unfortunately, not all researchers and planners have access to such software packages. It is also possible to use the specialised assignment algorithms employed in these software packages separately (e.g. [177–179]). However, it seems few researchers have the resources to recreate them, as only four publications, denoted as "Za", could be found using such algorithms. Finally, there are 12 publications that use regular shortest path algorithms (see Section 2.2) on the travel graph. These are denoted with "Zn". Such a set-up is more straightforward to implement than the other approaches; however, it is still more complex than using a node-based concept.

The fact that the number of zone-based UTRP studies is comparatively low is one possible reason why only few UTRP algorithms are applied in real world planning processes. Transport planning processes are usually based on macroscopic transport models using zone-based approaches [175]. In this context, differences in data structures between existing zone based datasets and node based algorithms can lead to high application barriers. Therefore, it is important to demonstrate how node-based UTRP algorithms can be converted to work with zone-based trip representations. This task is taken on in Chapter 5. The Chapter will further introduce a publicly available zone-based instance to make zone-based UTRP research more accessible.

[16]There is one study which does not fit this concept: In the study from van Nes et al. in 1998 [57] travel times between nodes only. However these where then used to distribute the passengers of a zone to the nodes available to them based on the trip distribution model from [176]. This study is denoted "Z*" in Table 3.1.

3.3.3 Interfacing with transport modelling software

Macroscopic transport modelling is used to support the decision making in planning processes. Among the most commonly used software packages for this task are PTV's Visum [180] and INRO's Emme [181], which have been used in UTRP studies in the past. The input for these simulations are so called network modes which are based on directed graph structures[17] representing the relevant infrastructure of the application area. Generating and calibrating these so-called network models requires detailed information about the street layout, transport infrastructure, and travel demand and is a complex and time-consuming task. The resulting models, however, are very powerful and allows planners to simulate and study a variety of transport-related phenomena.

Interfacing between these software packages and UTRP algorithms would drastically reduce the barriers for planners to use the latter. Additionally, it would allow researchers easy access for a vast pool of powerful evaluation tools. The challenge for a generally usable interface process is to handle the differences between the directed connections in the network models and the undirected graph which forms the basis of almost[18] all UTRP studies. Although there have been several studies utilising transport modelling software packages for their evaluation procedures, they usually used network models based on undirected graphs which were likely not designed for general planning purposes.

Two of these studies were already mentioned in Section 3.2.3, as both include procedures to generate the required graph. The first is the 2011 study of Bagloee and Ceder which implements an undirected graph in Emme for evaluation [116]. The second is a study by Alt and Weidmann which describes an algorithm interfacing with Visum on several points during the optimisation process [118]. In the latter case, the authors mention that the procedure can be used on existing network

[17]The structure of Visum network models is discussed in more detail in Chapter 6 Section 3.2.
[18]The only study working with directed connections is [33] (discussed in Section 3.4).

models. However, problems with directed connections do not appear to be discussed and all applications presented are based on undirected graphs.

Further, there is a series of studies interfacing a GA with Emme conducted by researchers at the Università di Roma Tre, Rome (Italy) [79, 85, 95, 123], the last of which in cooperation with the mobility agency of Rome. Unfortunately, it is unclear if the undirected graph representations used were generated specifically for these studies or existed before. Further, none of these publications describes the interface procedures that were used. Both are also true for [31, 169] which, likewise interfaced their algorithms with Emme and used an undirected graph representing the city of Mashhad (Iran).

So far, the line proposal algorithm for Visum, developed by PTV itself in 2006 [182], seems to be the only algorithm applicable to network models with directed connections. It is designed to add new routes to an existing Visum network model. It first generates a palette of candidate routes between predefined terminals which are not directly connected with existing routes. In a second step, the candidate routes are individually evaluated in Visum and the route taking up the most demand is then permanently added to the network. The second step is repeated as long as required. The user can directly interfere in parts of the process (e.g. select a different route to be added in the system). This set-up does not require a UTRP graph as the algorithm only determines the beginning and endpoints of routes, leaving the design to Visum's inbuilt shortest path procedure. It also should be noted, that this algorithm in no way alters the existing PuT network. Although this algorithm has been successfully used in at least two academic studies [6, 119], it is only a prototype and was never fully developed into an official Visum extension.

All these studies underline the potential that interfacing UTRP algorithms and transport modelling software can bring. Unfortunately, the available approaches seem to only work for either very special algorithms or when the used network models exclude common complications like one-way streets. To transfer data between more general network models and more standard UTRP algorithms, more sophisticated interface procedures are required. Such processes will be presented in Chapter 6.

3.4 Applications in planning processes

From the 133 reviewed publications in this Chapter only two describe studies which were part of real-world planning processes.

The first was published in 2009 by Pacheco et al., and describes the optimisation of the bus network in Burgos (Spain) [33]. This study is the only one using directed connections by subdividing each route into two directed half-routes beginning and ending on the same node. Waiting- and passenger travel time were optimised under a fixed total number of buses and drivers. All 382 bus stops of the study area were used as nodes in the instance graph provided by the Burgos transport authority together with a node-based demand matrix and a list of terminal nodes. The solutions obtained improved significantly over those created manually by planners.

The second is the work from Cipriani et al., conducted in 2012 in cooperation with the mobility agency of Rome (Italy) [123], as mentioned in Section 3.3.3. It describes the application of a parallel genetic algorithm on one of the largest recorded instances composed of over 1300 nodes, 7000 undirected links, and 450 zones. The results show improvements over the existing bus route network in terms of waiting times, operator costs, and unsatisfied demand. According to the authors, the mobility agency of Rome began implementing these results in 2012.

Combined, the two studies listed here include many features which are missing in most other studies and were identified throughout the past Sections as possible reasons for the general lack of practical applications. For example it can be noted that [33] is using restricted terminal nodes and [123] a zone-based trip representation. Both aspects are only in a minority of publications (26% and 20%, respectively). Their use in studies on actual practical applications is an indication of their importance for such applications. Also the fact that [33] is the only study using directed connections, and [123] one of the few publications interfacing their algorithm with transport modelling software, highlights the usefulness of both for the practical application.

3.5 Conclusion of literature review

The past Sections reviewed different aspects of the publications listed in Table 3.1, with a special focus on potential obstacles to applications in real-world planning processes.

Section 3.2.2 classified the used instances into fictional-, regional-, and urban instances. Although most studies use urban instances, only few of these instances include urban specific aspects. For example, only one-third of the used urban instances offer information on which nodes offer sufficient infrastructure for vehicles to perform U-turns. As it is usually assumed that vehicles perform U-turns at the end of each route to start the next journey in the opposite direction such information is vital to incorporate realistic constraints.

Section 3.2.4 highlighted that the absence of information on such terminal nodes from the publicly available instances identifies can be one reason why terminal nodes rarely feature in UTRP studies. Such instances are used by many researchers who otherwise would not have access to the required input datasets. In line with other reviews [99, 141], it was concluded that there is a need for more publicly available instances which better reflect size and complexity in real-world urban areas.

One problem for both the practical application of UTRP algorithm and the publication of more realistic instances is the generation of such instances. Section 3.2.3 reviews publications which previously described instance generation processes. It was concluded that the generation procedures presented in the past struggle with determining suitable node positions and adjacency relations, especially when applied to urban areas which are not built in a repetitive pattern. Further, none of the described processes include a infrastructure base procedures to determine potential terminal nodes.

These issues will be addressed in Chapter 4 of this thesis which will introduce an instance generation procedure which scales down the available street network while maintaining its characteristics and extracts information about potential terminal nodes from the existing PuT services. This procedure will be used to construct several instances which have been made publicly available.

Section 3.3.2 discussed the differences between zone-based and node-based trip representation and their use in UTRP studies. Being used in 80% of publications, the node-based approach is more prominent among UTRP researchers. Among the reasons for this dominance is a more straightforward implementation and the absence of publicly available zone-based instances However, most planning process use zone-based macroscopic transport modelling processes as basis for their work. Consequently, the differences in data requirements between both approaches, create barriers for the application of UTRP algorithms in planning processes.

To facilitate the development of zone-based UTRP algorithms, Chapter 5 will extend the instance generation procedure introduced in Chapter 4 and generate a publicly available zone-based instance. Further it will introduce a concept to calculate zone-based travel times, which is straightforward to implement and can be utilised in the adaptation operations used in the optimisation procedures in Chapter 4 and other node-based publications to a zone-based set-up.

Finally, Section 3.3.3 reviewed publications which interface their algorithms with transport modelling software. This approach has high potential to effectively reduce barriers for applications as a generally usable interface would allow planners to use

UTRP algorithms with already existing datasets and in a familiar environment. Unfortunately, all reviewed approaches only work for either very limited algorithms or when the used input datasets only include undirected connections. For this reason, Chapter 6 of this thesis will present interface procedures to translate between directed and undirected network- and route structures, and demonstrate their effectiveness in an application on datasets based on real-world urban areas.

4 An Adaptive Scaled Network for Public Transport Route Optimisation

List of Authors

Philipp Heyken Soares, Christine L. Mumford, Kwabena Amponsah, Yong Mao

Status by thesis submission

Published in 2019 in Springer Nature journal "Public Transport", issue 11(2).

Authors' contributions

The principle idea for this paper is the generation of a new instance and the modification of optimisation procedure to work with restricted terminal nodes. This initial concept was developed through group discussion between Philipp Heyken Soares, Christine Mumford, and Yong Mao.

Regarding the research work, Philipp Heyken Soares developed the procedures to generate instances (Section 2 and Appendix A) and map real-world routes on the designed graph (Section 4). This included the identification of suitable data sources, the data preparation, as well as the required programming and post-processing. The initialisation procedure (Section 3.2) was designed and programmed by Christine

Mumford. She also provided the script containing the evaluation programme taken from earlier research. Kwabena Amponsah programmed the core elements of the NSGAII algorithm (Section 3.3) based on the descriptions in [21]. Adaptations of the genetic operations (Appendix B) were designed and implemented by Philipp Heyken Soares. He further designed and analysed the experiments presented in Section 5. Christine Mumford and Yong Mao provided guidance throughout the entire project.

For most sections of this paper, the initial draft was written by Philipp Heyken Soares. The exceptions are sections 3.1 and 3.2, which were initially written by Christine Mumford. After the initial draft, all authors contribute in proofreading and editing of all parts of the manuscript. All figures were generated by Philipp Heyken Soares.

Summary of content

This paper introduces a new procedure to generate input datasets for UTRP algorithms (Section 2). The aim is to scale down the street network of a mid-size urban area with an irregular layout to generate a graph to represent local bus networks with less than 500 nodes. This boundary was set after a run time estimation judged it to be the uppermost limit for practical work with the available computing infrastructure.

As the first step, the available streets are selected on the basis of official classifications and the position of the existing stop points (Section 2.3.1). Graph nodes are then placed on the basis of the position of street junctions and demand sources (Section 2.3.2). Certain placement rules were set to maintain the characteristics of the street network as best as possible, while covering all of the available demand sources and keeping the number of nodes as low as possible.

The adjacency relations and travel times between the nodes are determined on the basis of the shortest paths between nodes (Appendix 1). This process used the graphical information system ArcGIS, because of its python interface and the ability to calculate the routing information on spatial networks.

A separate procedure maps the course of the existing bus routes on the generated graph to determine their endpoints as the potential terminal nodes (Section 2.5 and Section 4). With this, the detailed route data available for the study area allowed us to identify the possible terminal locations significantly better than alternative manual methods.

The procedures are implemented on in an application of the extended urban area of Nottingham, UK. This area was chosen because its size and layout fit well with the aims of the project. The street and the demand data were obtained from official sources. A second, reduced version of this instance is generated to allow comparisons between the optimised route sets and a representations of the pre-existing services (Section 4). For this, all nodes which are not used by the pre-existing bus services operating within a given time window are excluded. Both instances are published online alongside the paper. The datasets contain the following files: a) a list of nodes giving the respective positions and stating whether a node is a potential terminal, b) a matrix giving the travel times between the directly connected nodes, and c) a node-based matrix.

Further, an optimization procedure centres on an NSGAII-type genetic algorithm (previously used e.g. in [137]), which is modified to work with the restricted terminal nodes (Section 3.3) is presented. The main adaptations are in two genetic operations (Appendix 2): The operation deleting nodes from a route is only allowed to stop when the reduced route ends again on a terminal node. The operation adding nodes to routes uses a guided random walk to connect the current endpoint to a new available terminal node. This prevents the extension from meandering before reaching a terminal node and, therefore, balances the delete and add-nodes operations. The same principle is also used in the adapted version of the add-missing-nodes repair operation.

Public Transport (2019) 11:379–412
https://doi.org/10.1007/s12469-019-00208-x

ORIGINAL PAPER

An adaptive scaled network for public transport route optimisation

Philipp Heyken Soares[1] · Christine L. Mumford[2] · Kwabena Amponsah[1] · Yong Mao[1]

Accepted: 3 June 2019 / Published online: 29 July 2019

Abstract
We introduce an adaptive network for public transport route optimisation by scaling down the available street network to a level where optimisation methods such as genetic algorithms can be applied. Our scaling is adapted to preserve the characteristics of the street network. The methodology is applied to the urban area of Nottingham, UK, to generate a new benchmark dataset for bus route optimisation studies. All travel time and demand data as well as information of permitted start and end points of routes, are derived from openly available data. The scaled network is tested with the application of a genetic algorithm adapted for restricted route start and end points. The results are compared with the real-world bus routes.

Keywords Public transport · Route optimisation · Network design · Benchmark instance · Genetic algorithm

1 Introduction

In the majority of cities around the world, public transport networks have been developed using a combination of local knowledge, planning experience and published guidelines. Most often these networks have evolved over time rather than being designed as a whole (Mumford 2013). Multiple reports have pointed out

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s12469-019-00208-x) contains supplementary material, which is available to authorized users.

✉ Philipp Heyken Soares
philipp.heyken@nottingham.ac.uk

Christine L. Mumford
MumfordCL@cardiff.ac.uk

1 Laboratory of Urban Complexity and Sustainability, University of Nottingham, Nottingham, UK

2 School of Computer Science and Informatics, Cardiff University, Cardiff, UK

the insufficiencies of this process and the need for a systematic computer-based approach (see e.g. Zhao and Gan 2003; Nielsen et al. 2005).

The task to generate efficient public transport networks can be treated as five interconnected phases (Ceder and Wilson 1986): (1) route design, (2) vehicle frequency setting, (3) timetabling, (4) vehicle scheduling and (5) crew scheduling. Solving all five phases simultaneously would be optimal. However, due to the high complexity of the task, most researchers focus their efforts on simplified versions of the problem. One common simplification is to focus on the route design phase under the assumption of a fixed transfer time between different routes. This approach will also be used in this paper.

Research for public transport route optimisation requires information on the available transport network and the travel demand. These combined datasets are usually referred to as instances. Many researchers have tested their algorithms on the few fully published instances. A prominent example is the instance published by Mandl (1979) [used e.g. in Ahmed et al. (2019), Arbex and da Cunha (2015) Baaj and Mahmassani (1991), Fan and Mumford (2010) and Nayeem et al. (2014)]. Another often used test instance [e.g. used in Poorzahedy and Safari (2011) and Soehodo and Koshi (1999)], a 24-node network published by Leblanc (1975), is based on the city of Sioux Falls, USA. As these two instances are rather small (15 and 24 nodes), Mumford published four larger instances ranging from 30 to 127 nodes, based on the Chinese city Yubei and the two UK cities of Brighton and Cardiff (Mumford 2013). These have been used by several studies since (e.g. John et al. 2014; Nayeem et al. 2014). In addition to these studies, other researchers built their own test instances based on data from urban areas around the world. Among these are Silman et al. (Haifa, Israel, 1974) (Silman et al. 1974), van Nes et al. (Groningen, Netherlands, 1988), Pattnaik et al. (Madras, India, 1998) (Pattnaik et al. 1998), Feng et al. (Taoyuan-County, Taiwan, 2011) (Feng et al. 2010), Cipriani et al. (Rome, Italy, 2012) (Cipriani et al. 2012), or Gutiérrez-Jarpa et al. (Concepcíon, Chile, 2017) (Gutierrez-Jarpa et al. 2017).

Few publications describe the rules of instance generation in detail. One exception is the work by Mauttone and Urquhart (2009) who generated a network with 84 nodes to represent the city of Rivera, Uruguay. The nodes were placed on street junctions close to the centres of housing blocks. This method is only suited to cities built in a strict grid pattern. Another node selection algorithm was proposed by Bagloee and Ceder (2011) to generate instances for Winnipeg, Canada, and Chicago, USA. They select every stop point as a node provided it is further than 300 m from another with a higher expected travel demand. A similar method was also used by John (2016) to generate networks for the UK cities of Nottingham and Edinburgh. Here a fixed number of stops was selected randomly while a minimal distance between selected stops was ensured.

The methods from Bagloee and Ceder (2011) and John (2016) are applicable to most urban areas, but both share the same disadvantage: as the locations of the selected stops within the street network are not fully taken into account, the chance that the resulting network does not reflect the real spatial layout of the city is high (John 2016), especially if the number of selected stops is only a fraction of the total number available.

As the layout of the street network is essential for the design of bus routes, it is important that an instance network sufficiently reflects the characteristics of the street network. We therefore propose a network design procedure which scales down the network to a size manageable for meta-heuristic-based optimisations, while at the same time preserving the characteristics of the urban street network. Scaling down an urban street network is desirable principally to restrict the computation times needed for the passenger objective, which is usually the main bottleneck and leads to an increase of the run time with $f(N^3)$, N being the number of nodes in the network [see Mumford (2013) for a full explanation]. For our modest desktop set up,[1] we determined 500 nodes, which would result in a runtime of about 600 h, to be the upper-most limit for practical work.

The down-scaling of the street network is achieved by devising simple and robust rules applicable to all urban layouts. The procedure further includes the identification of potential terminal nodes, an aspect vital for route design in an urban context.

We will limit this work to instance generation and route network optimisation with restricted start and end points, as it is part of an incremental approach to more realistic public transport network optimisation. Our generation procedure is used to produce an instance for bus route optimisation in the extended urban area of Nottingham, UK. Further, a route initialisation procedure and a modified heuristic route optimisation algorithm, both specialised for work with restricted route start and end points, are applied to the generated instance. The optimisation results are compared to the real-world bus routes.[2]

The main contributions of this paper are as follows:

1. A novel methodology for generating an instance dataset, by systematically scaling down a street network and utilising census data (see Sect. 2). The generated instance will be published online for free use for all researchers.
2. An additional methodology for transforming pre-existing public transport routes to fit the scaled-down street network (see Sect. 4).
3. A multi-objective genetic algorithm modified for restricted route start and end points, to allow direct comparison with the pre-existing public transport routes, showing potential to improve performance (methodology in Sect. 3 and results in Sect. 5).

[1] We used a desktop PC with an Intel i5-6500 3.20GHz Quadcore CPU and 8GB RAM.

[2] It should be noted that this work focuses on mono-modal transport, and the Nottingham tram network, with only two lines in 2011, was not taken into account. For the instance generation, the interoperation of the tram could be included in principle with the rules described in Sect. 2. However, the optimisation algorithm (see Sect. 3) would require modifications beyond the scope of this paper to deal with multi-modal optimisation.

2 Generating a scaled instance network for bus route optimisation

This section outlines a systematic approach to generate a street network, consisting of node placement, link generation, and the production of the travel time and demand matrices. The methodology is applied to the extended urban area of Nottingham, UK (see Fig. 1), using UK-specific street and census data. However, the same can be applied to areas outside the UK using other data sources. Examples for potential sources are given in the footnotes 4, 5, and 6.

2.1 Problem description

An available transport network can be represented as an undirected graph $G = \{N, E\}$, with nodes $N = \{n_1, n_2, \dots, n_{|N|}\}$ representing access and interchange points (see Sect. 2.3.2) and links $E = \{e_1, e_2, \dots, e_{|E|}\}$ representing the edges (e.g. streets) connecting the nodes (see Sect. 2.4). Given such a graph, a public transport route can be represented as a list of directly connected nodes $r = [n_f, \dots, n_l]$ and the public transport network as a set of routes $R = \{r_1, r_2, \dots, r_{|R|}\}$. It is necessary to ensure that the first and the last node on each route is a designated terminal node $V = \{v_1, v_2, \dots, v_{|V|}\} \in N$ which allows u-turns (see Sect. 2.5). Travel times T and travel demand D are symmetrical matrices as defined below.

In order to allow an optimisation of the route set R on the graph G, the travel time and travel demand between the nodes N need to be given. We do this in form of two symmetrical matrices:

T:	Gives the travel times $t_{i,j}$ along the connection between the nodes n_i and n_j. Travel times between nodes that are not directly connected are set to $t_{i,j} = \infty$ and self connections are set to $t_{i,i} = 0$.
D:	Gives the number of passengers $d_{i,j}$ travelling from source n_i to destination n_j. Travellers staying at one node are not considered ($d_{i,i} = 0$).

2.2 Definitions

We first define some parameters:

- Catchment radius c:
  Defines circular catchment areas around the nodes used to assign travel demand (see Sect. 2.6). For the Nottingham instance we used $c = 400$ m, a value widely considered as an acceptable walking distance to a bus stop (Ammons 2001).
- Snapping Distance s:
  Distance within which two or more junctions are snapped together to be represented by one single node in between (see Sect. 2.3.2). The value for

this snapping distance is not critical and has been chosen to be $s = c \cdot \sin(\frac{\pi}{4})$, resulting in $s = 283\,\text{m}$.

2.3 Constructing the network

The first step is to select which streets will be taken into consideration for the network. This is followed by a process to determine the positions of the nodes on the street map.

2.3.1 Defining the street network

For the Nottingham application we base our selection of streets on the integrated network layer from 2011[3] generated by the UK Ordnance Survey.[4] We selected all streets classified as “A-”, “B-” or “Minor Road”. Streets classified as “Local Street” were only selected if they fulfil two conditions: (a) bus stop points exist alongside them[5]; (b) they are not parallel to any street classified as “A-”, “B-” or “Minor Road” within a distance s. One-way streets are only selected if travel in the other direction is possible on streets within s.

2.3.2 Placing nodes

With the streets selected, the positions of the nodes N need to be determined. In contrast to the instances generated by Bagloee and Ceder (2011) and John (2016), the nodes in our network do not directly represent bus stop points. Instead they usually represent street junctions. The interpretation is not that buses stop at each node but rather that they travel from node to node and stop at the stop points they encounter on the way. The nodes simply allow us to identify the path a bus will take.

We determine the position of our nodes in three steps:

1. Initial nodes are placed on all points where two or more of the previously selected streets meet. This includes junctions, (t-)intersections, roundabouts, etc.

[3] The same year of census data used to assign the demand (see Sect. 2.6).

[4] Researchers with UK institutional access can download ITN data from http://digimap.edina.ac.uk/. The procedure can also be applied to data from other sources. The only constraints are a sufficient classification to select streets available for bus travel and the ability to convert the data to a network dataset for use in ArcGIS to generating the travel time matrix (see Appendix 1). Such data should be available from most national transport authorities, or local authorities. Alternatively, street data from OpenStreetMap (https://www.openstreetmap.org) can be used as long as it is sufficiently accurate for the study area.

[5] The location of bus stops can be extracted from the National Public Transport Access Nodes (NaPTAN) which is included in the National Public Transport Data Repository (NPTDR) downloadable from https://data.gov.uk/dataset/nptdr. Outside the UK similar datasets should be available from national transport authorities, local authorities or public transport operators. For operators which use General Transit Feed Specification (GTFS) these datasets can be downloaded from https://transitfeeds.com/. Also OpenStreetMap (https://www.openstreetmap.org) usually provides bus stop locations with a sufficient accuracy.

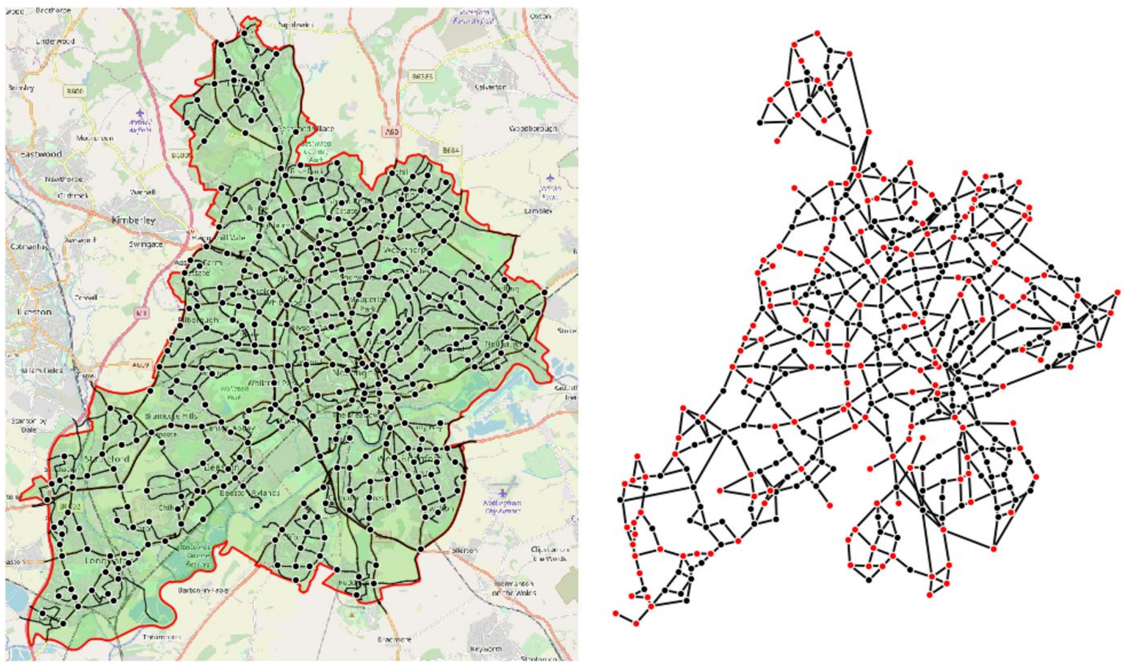


Fig. 1 Left: map of the study area together with the street network and placed nodes (source: UK Ordnance Survey, map source: https://www.openstreetmap.org, street data source: UK Ordnance Survey; see footnote 4). Right: graph network showing the connections between the nodes with terminal nodes highlighted in red (colour figure online)

2. Nodes within a distance s of each other form clusters, and each cluster of nodes is snapped together to form a new node. This leads to a simplification error of up to $\frac{s}{2}$ for the distance between two nodes, but removes the clustering nodes (see Figs. 2, 3).
3. It can happen that two directly connected nodes are too far apart to assign all the demand from zones along the connection to one of the nodes (see Sect. 2.6). In these cases additional nodes need to be placed in-between the regular nodes to properly capture the demand for a bus route along this route (see Fig. 4). The same situation might occur in case of dead-end streets which are longer than c.

Snapping clusters of initial nodes together has several implications. It further simplifies the representation of transfers between bus stops located at the same node, and it increases the effective length between some snapped nodes compared to a single pair of the snapped nodes (see Fig. 3). However, it drastically reduces the total number of nodes and thereby the processing time. In our case snapping reduces the number of 497 initial nodes by about one third to 324. The mean distance between an initial node and the snapped node placed instead is 73.8 m. The accumulated simplification error for the travel time estimation is about 1.6% of the total travel time along all network edges. As 104 additional street nodes have to be added to ensure all demand can be assigned, giving a total of 324 + 104 = 428 (see Fig. 2), the snapping process is important to keep the total number of nodes sufficiently low. Using an estimation of run time of $f(N^3)$, the snapping process reduces the run time for the network with 428 nodes by a factor of 3 when compared to a network with about 600 nodes (initial nodes plus street nodes).

2.4 Generating links

After placing the nodes, a travel times matrix T is required for the optimisation stage. Thus the travel times $t_{i,j}$ between connected nodes need to be determined. We do this by calculating along the shortest path between two directly connected nodes using ArcGIS. The process takes turning restrictions and traffic calming zones into account. The details are given in Appendix 1. Travel times between nodes that are not directly connected are set to $t_{i,j} = \infty$ and self connections are set to $t_{i,i} = 0$.

2.5 Determining terminal nodes

One often overlooked aspect in route optimisation studies is that public transport routes cannot terminate everywhere, as this implies that the vehicles are able to turn around and traverse the route in the opposite direction. Therefore, possible terminal nodes require sufficient infrastructure to perform u-turns. It is vital to take this constraint into account when modelling real-world data in an urban context (see e.g. Amiripour et al. 2014).

While it is possible to check the surrounding area of every node manually for turning possibilities, this process would be very time-consuming. Instead we identified possible terminal nodes based on the journey patterns of real-world buses. These journey patterns can be extracted in the form of lists of traversed stop points from the UK's 2011 National Public Transport Data Repository (NPTDR).[6] The stop point lists are then projected on the generated instance network (this process is described in Sect. 4.2). This allows us to determine the nodes at which real-world journey patterns end, either by leaving the study area or by using an existing turn possibility to start the journey in the reverse direction. There may be further nodes which offer such possibilities and are not used by the existing routes. However, as this method already identified 168 terminal nodes in our Nottingham instance (39% of all nodes), we are confident that this process is sufficient. Four additional nodes had to be added to the list of terminal nodes. These are dead-end nodes and in the algorithm used in this paper can only be included at the beginning or end of a route. For them, manual examination ensured the turning possibilities.

2.6 Assigning travel demand

The final step is to generate a node-to-node travel demand matrix D for the network, which gives the demand $d_{i,j}$ between the nodes n_i and n_j, with $(d_{i,i} = 0)$.

Generating a fully realistic travel demand dataset is a very complex task and not within the scope of this work. What we will show here is not a process for extracting

[6] The 2011 NPTDR contains a snapshot of every public transport journey in Great Britain in a selected week in October 2011 (UK Department for Transport 2015). It can be downloaded from https://data.gov.uk/dataset/nptdr. For outside the UK, datasets containing lists of traversed stop points as well as the location of stops should be available from national transport authorities, local authorities or public transport operators. For operators which use GTFS, the datasets can be downloaded from https://transitfeeds.com/.

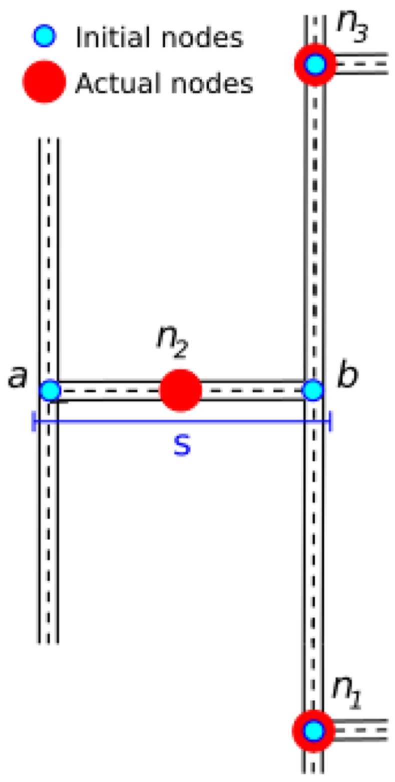


Fig. 2 Example for snapping nodes in one part of the Nottingham street network: the nodes n_1 and n_3 emerge directly from initial nodes placed on junctions, while the node n_2 results from the snapping together of initial nodes a and b (which are closer than distance s). It should be noted that node n_2 remains representative of both the junctions it emerged from (i.e. a and b). Thus it is not possible to travel directly between nodes n_1 and n_3 without passing through node n_2. This generates a maximum simplification error of s for a travel between node n_1 and node n_3. In our case the accumulated error is estimated to be 1.6% of the total travel time

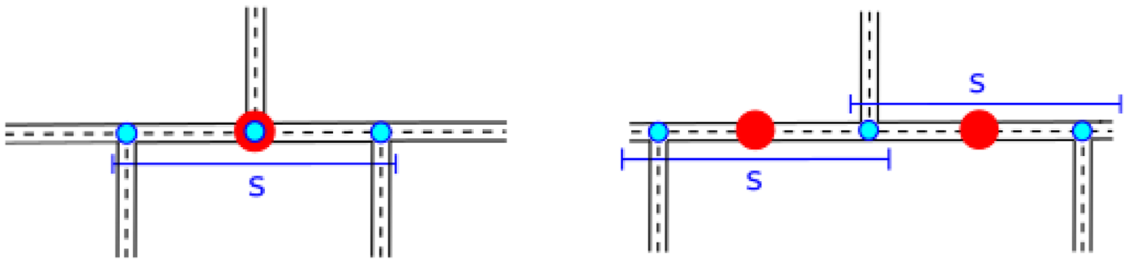


Fig. 3 Left: three initial nodes which are all within snapping distance and which are snapped together to a single node. Right: three initial nodes of which two pairs are within snapping distance. They are snapped into two nodes and the junction between them is represented by either node, depending on the best option for a particular route

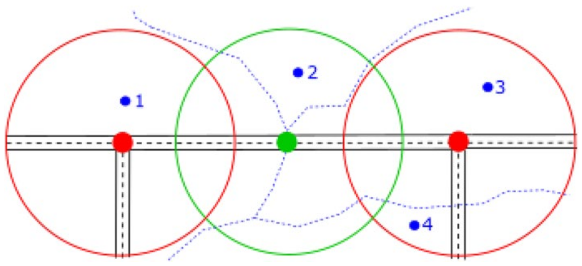


Fig. 4 Example for placing a street node: two regular nodes (red) with catchment area are displayed. The blue dots mark centroids of census zones used to assign the transport demand (see Sect. 2.6). Zone 2 is completely outside the catchment of the two regular nodes; therefore, an additional street node (green) is placed in between them to capture the travel demand for that zone (colour figure online)

travel patterns from raw data sources, but rather a method to transform given travel data between specified zones to a node-to-node demand matrix. Zonal demand data can be generated in different ways. One option is to estimate the number of trips between zones based on the number of origins and destinations in each zone and other parameters [see e.g. Wills (1986) or Wilson (1969)]. This methodology, however, depends much on the quality of the land use data and requires trip survey data or trip counts for calibration. Another methodology well studied in recent years is to extract the travel demand from mobile phone data [see e.g. Golding (2018) or Zhang et al. (2010)], which requires the cooperation of one or several telecommunication companies. A third option is to use survey data. Producing trip surveys is usually time consuming and expensive, and it is preferable to try to access existing survey data if available. In our case we used travel-to-work flow data from the 2011 UK census. It lists the number of commuters from all output areas (in the following origin zones) to all workplace zones (in the following destination zones).[7]

To generate a node-to-node origin-destination matrix (OD-matrix) we take the population weighted centroids of the origin and destination zones[8] and assign them to nodes based on the catchment areas. The catchment area of a node is of circular shape with a radius c but can be modified under the following circumstances:

- Natural or man-made barriers (rivers, rail tracks, etc.) prevent the commuters from having access to the part of the street network the node represents.
- If a centroid is outside of the catchment area of any node, even when its associated zone overlaps with one or more catchment areas, then these catchments are extended to include this population. (This is to reduce the number of additionally placed street nodes as described in step 3 in Sect. 2.3.2).
- A street node (see step 3 in Sect. 2.3.2) represents the whole street and not just the point where it is placed. (This removes the need to place several nodes along longer streets.)

After this assignment, trips between origin zones and destination zones can easily be converted to trips between the nodes to which the respective zones are assigned. For zones which are within two or more catchment areas, their trips are divided equally between the nodes.

There are of course problems with using travel-to-work data to generate the demand matrix. It represents only a subset of all trips, e.g. trips for shopping or

[7] Output areas and workplace zones are low-level census geographical types. They are redefined for every census. Every output area includes between 40 and 250 households (for the 2011 census on average 309 residents). Workplace zones are created using similar criteria to output areas, to capture employment statistics. It should be noted that output areas and workplace zones overlap (UK Office for National Statistics 2016a). The geometries and flow data can be downloaded from https://census.ukdataservice.ac.uk/.

[8] Population weighted centroids are generated by the UK data service, UK Office for National Statistics, together with the respective areas (UK Office for National Statistics 2016b). They can be downloaded from https://census.ukdataservice.ac.uk/. If using other datasets where the exact distribution of the population within the zones is not known, the geometric center of a zone can be used instead. These can be calculated with GIS software.

leisure purposes are not included. We will therefore in the following limit our comparisons with the real-world bus routes to the morning rush hour when trips to work dominate the overall travel pattern. This represents a sufficient estimate for this step of our incremental approach. It is worth noting that the demand matrix does not have to be an accurate count of trips but only a weighting for the trip demand within the generated instance network.

3 Optimisation procedure

The main goal of this paper is to present a new approach to bus route network design. Thus, we use a genetic algorithm (GA) here simply to produce some viable results to demonstrate the potential of the new network design method. Improvements to the optimisation procedure will follow as future work. For our present purpose, we have implemented an NSGAII algorithm based on the one in John et al. (2014), making some necessary changes to the initialisation process and the genetic operators to accommodate the specified terminal nodes in our Nottingham instances [in the paper by John et al. (2014), any node in the network could act as terminal node for a route].

3.1 Initial definitions

3.1.1 Passenger and operator objectives

Optimisation will attempt to minimise the travel times for individual passengers, while at the same time ensuring the cost to the bus company is reasonable. Our objectives are the average travel time as cost for the passengers, and the total drive time of the route sets as cost for the operator. These are the same objective functions which have been used in John et al. (2014) and other studies (e.g. Mumford 2013). We are using them as it is our aim to only adapt the algorithm described in John et al. (2014) for the use of terminal nodes. Changes to these objectives are possible but beyond the scope of this paper.

The minimum journey time for a given passenger travelling between their origin a and destination b, can be given by $\alpha_{a,b}(R)$, representing the shortest path in terms of travel time. However, this path is made up of two components: in vehicle travel time and transfer time. In this paper (in line with other recent studies) we will assume that the transfer time is a constant, which is imposed each time a passenger changes vehicle. We set this time constant, representing the average waiting time, to be 5 min.[9] Furthermore, we will assume that each passenger will choose to travel on their shortest-time paths. We define the *passenger objective* for a route set R to be the mean journey time over all passengers,

[9] This is in line with the definition for "frequent services" given by the UK Department for Transport, which is to have at maximum 10 min between buses (Turfitt 2018). For Nottingham the majority services in the morning rush hour fall into this category.

$$C_P(R) = \frac{\sum_{i,j=1}^{|N|} d_{ij}\alpha_{ij}(R)}{\sum_{i,j=1}^{|N|} d_{ij}} \tag{1}$$

where α_{ij} is the shortest journey time from i to j using route set R. The operator objective depends on many factors; however, we shall use a simple proxy for operator costs: the sum of the costs (in time) for traversing all the routes in one direction,

$$C_O(R) = \sum_{i=1}^{|R|} \sum_{i=1}^{|r|-1} t_{i,i+1}(r) \tag{2}$$

where r is a typical route in R, and $|R|$ is the number of routes in the route set. $t_{i,i+1}(r)$ represents the travel time between adjacent nodes in route r. The passenger cost C_P (average travel time) and the operator cost C_O (total route length) will be traded off as conflicting objectives by our multi-objective optimisation algorithm.

Changes in route frequency would effect both costs for the operator and the average travel time, which would in turn require a re-optimisation of the route design. For the present we will set a standard headway of 10 min, in line with UK Government definitions for frequent services (Turfitt 2018).

3.1.2 Optimisation constraints

While optimising sets of routes for the two objectives, the optimiser has to follow the specifications of the instance generated in Sect. 2, including the recognition of terminal nodes. These are in addition to the constraints applied in John et al. (2014). The full list of constraints is given below:

1. Each route set R consists of a predefined number of routes, $|R|$
2. Each route r in a route set R will consist of a number of nodes greater than or equal to l_{min}, and less than or equal to l_{max}: $l_{min} \leq |r| \leq l_{max}$.
3. No route fully overlaps with another route in the same set.
4. The route set is connected—it is possible to travel between each pair of nodes in the transport network.
5. Each node n_i is present in at least one route in a route set.
6. No loops or cycles are present in a route.
7. Buses travel back and forth along each route, for which the inbound journey is the reverse of the outbound journey.
8. There is a fixed waiting/transfer time for passengers transferring from one route to another.
9. Each route r begins and ends at a terminal node $v \in \{v_1, v_2, \dots, v_{|V|}\}$, where buses can turn around.

3.2 Heuristic construction of route sets

Our initialisation procedure attempts to produce a population of high quality, legal route sets, obeying all of the constraints listed above. We will tackle the construction of route sets in three separate stages:

1. production of a *shortest path usage map* and a *transformed shortest path usage map*
2. generation of candidate routes to form a palette of routes
3. selection of candidate routes to form route sets

3.2.1 First step: constructing a shortest path usage map and transforming it

We use the technique described in Kiliç and Gök (2014) to create a demand map based on edge usage. The process begins by evaluating shortest travel time paths between each pair of nodes for which there is a source to destination travel demand. Then, by recording the total usage of each edge (assuming that each passenger is able to travel along their shortest path) it is an easy matter to create a 'shortest path usage map'.

Next we perform a simple transformation on the usage map, to reverse the ranks of the labels on the edges. This is where our technique diverges from that of Kiliç and Gök (2014), who convert the usage values directly into edge selection probabilities. In our approach we transform usages into distances, so that the highest usage value becomes the shortest distance and vice versa. We use these transformed values in a deterministic way to create routes, based on shortest path distances through the transformed usage network. The transformation is achieved simply by subtracting the usage on each edge from the total demand for the network as a whole.

3.2.2 Second step: generating candidate routes

This second step produces a palette of routes from which selections can be made to construct the initial population of route sets for our GA. It makes sense to ensure that edges with the highest usage occur in one or more routes within the palette. On the other hand, the inclusion of less busy links will almost inevitably be required to satisfy some of the demand. Thus, although our algorithm begins by selecting the busier edges, the weight on each edge will be very slightly increased every time it is selected, so that the more times an edge is chosen, the less likely it is to be chosen again. We use an arbitrary factor of 1.1 (chosen after some sensitivity tests), making the updated weights equal to 1.1 times the previous weight.

In our initialization we ensure that all routes generated for the route palette begin and end at a terminal node. To this end, our algorithm iterates through all pairs of terminal nodes in turn, choosing the source-destination pair with the highest total demand first, and then the pair with the second-highest total demand, and so on. The algorithm then selects the source-destination node pairs from a pre-sorted list (on total demand for each node pair), created at the start of this second step. For each pair of terminal nodes, the shortest paths from source to destination are then computed according to the transformed demand usage map. (Note: the edge weights for the shortest paths here are entirely different from those used in stage 1. The stage 1 weights are travel times, but the stage 2 weights are transformed demand usages).

Each shortest path through the transformed usage map forms a candidate route which can be added to our route palette. Following the creation of each new route, the transformed demand usage map is updated by applying the factor 1.1 to the edge weight of each link selected for the route. The iterations cease once we have included each of the nodes in the instance in at least one of the routes included in our palette. It is likely that the algorithm iterates more than once through the list of terminal node pairs, to ensure that all the nodes are included somewhere in the palette. One further point is that in the presence of problem constraints, such as limitations to the lengths of the routes, our procedure will discard any routes that do not obey all the constraints. However, the transformed usage map is updated, whatever the outcome. It is worth noting that, if the imposed constraints are too severe, it may not be possible to generate a route set that obeys all of the constraints, so care is needed when setting the parameters for an instance.

The question now arises whether it is possible to construct legal route sets by making selections from the routes generated by our heuristic construction method. To begin with, we delete duplicate routes from our palette of routes. In the next paragraph we describe a simple method to select and aggregate subsets of routes from the above palette of routes to form route sets that obey all the constraints.

3.2.3 Third step: forming route sets by combining routes from the palette of candidate routes

We select routes from the palette one at a time until we have generated an initial population, of route sets P_0 (50 for our experiments), each containing exactly $|R|$ (i.e. 69) routes. For the first route set, the procedure begins by selecting the first route in the palette. We then add further routes in such a way that: (1) the chosen route has at least one node in common with a route already in the selected subset (beginning with just the first route), and (2) the addition of the selected route maximises the proportion of new nodes included, related to the total number of nodes in the candidate routes. At each iteration, the current subset of routes is tested for inclusion of all the nodes present in the instance. Once all the nodes are present, unused routes from the palette are added at random until the first route set contains $|R|$ routes. To generate the second route set for the initial population, the second route in the palette is the first route to be selected, and the third route set is seeded with the third route in the palette, and so on.

3.3 Genetic algorithm

We adopt an NSGAII genetic algorithm to evolve the generated route sets further. NSGAII is an elitist non-dominated sorting genetic algorithm for the optimisation with multiple objectives (Deb et al. 2002). We constructed our implementation of the NSGAII after the one in John et al. (2014) but made some changes to adapt to the use of terminal nodes. A flow diagram of the algorithm can be seen in Fig. 5.

3.3.1 Outline of our implementation of NSGAII

To save space we do not give full details of the genetic algorithm here, but refer the interested reader to Deb et al. (2002) and John et al. (2014). In summary, let P_0 be an initial population with $|P|$ route sets. All these route sets are evaluated and sorted into a series of Pareto fronts (f_1,f_2, …) based on their level of domination. Each solution is assigned a fitness value according to its front membership, with f_1 being the fittest, f_2 the second fittest, and so on. An offspring population Q_0, also of size $|P|$, is then generated from P_0 through binary tournament selection, crossover and mutation (see Fig. 5).

Next, the combined (mating) population $M_0 = P_0 \cup Q_0$ is used to select $|P|$ route sets as a new parent population P_1. This selection is primarily based on domination, but crowding distance is also taken into account (see Fig. 5). Crowding distance is an additional fitness measure used to obtain a wide spread of solutions that adequately covers the full extent of the Pareto front (Deb et al. 2002). P_1 is then used to generate Q_1 via binary tournaments, crossover and mutation as previously. The stages of NSGAII repeat for a predetermined number of generations.

3.3.2 The genetic operators

Crossover: In the crossover step, route sets of the current parent population P_k are selected in binary tournaments and used to generate in total $|P|$ offspring route sets. For each parent route set it is decided probabilistically[10] if it is either directly inserted in the offspring population, or if it performs a crossover operation with one other parent to generate an offspring route set. In the crossover operation, routes from both parent route sets are selected in alternation to construct the new route set. The route selection prefers routes which visit nodes not already included in the offspring route set. Before the new route set is allowed to enter the offspring population, a feasibility test is applied (see below). If it fails the test, the crossover process is restarted.

Mutations: In the mutation stage, all route sets of the offspring population undergo changes through mutation operations. The number of mutations in each route set is defined by a binomial distribution $B(|R|, \frac{1}{|R|})$, with $|R|$ being the number of routes in each route set. For every mutation one of the following mutation operations[11] is selected at random:

[10] The probability to perform a crossover with another route set is in our case set to $\rho_{cross} = 0.9$.

[11] We use in principle the same mutation operators as John et al. (2014). The operators Delete Node and Add Node, both originally from Mumford (2013), had to be modified to work with terminal nodes. The operator Exchange is used as it was originally proposed by Mandl (1979). The operators Merge and Replace, both from John et al. (2014), stay as they were, apart from the changes in the route generation procedure. Only the operator Remove Overlapping (John et al. 2014) is deleted from the set of mutations as it is part of the feasibility test.

- "Delete nodes": selects a route at random, ensures that it includes more than two terminals and starts to delete nodes from one of its ends until it reaches another terminal. If less than Z nodes[12] are deleted in this way, another route is chosen and the process is repeated until at least Z nodes are deleted.
- "Add nodes": selects a route at random and adds nodes at one of its ends. The new nodes are selected by a guided random walk to ensure that the route ends at the next possible terminal. If fewer than Z nodes (see footnote 12) are added in this way, another route is chosen, and the process is repeated until at least Z nodes are added.
- "Exchange": extracts two intersecting routes at random, splits them at one common vertex and forms two new routes from the split paths.
- "Replace": calculates which route satisfies the least demand and deletes it. The route is replaced by a newly generated route[13].
- "Merge": randomly selects two routes which share one terminal node and do not overlap elsewhere and merge them into one route. Afterwards a new route is generated[13].

After every mutation operation, the mutated route set is subject to a feasibility test[14] (see below). If it fails the test, all mutations are undone and a new mutation operation is selected at random.

Further details of add nodes and delete nodes are given in the appendix Sects. 1 and 2, respectively. The other operators are changed very little from those in John (2016).

Feasibility test: Every offspring route set generated by a crossover as well as every route set changed by mutation is subject to a feasibility test to ensure that all route sets obey the constraints listed in Sect. 3.1.2. However, to avoid rejecting invalid solutions that are easily corrected, we implemented two repair operations to fix two common constraint violations:

- "Add missing nodes": In cases where nodes are missing, we adopt a repair procedure based on that used in Mumford (2013). However, this method required some adaptation to work effectively with pre-defined terminal nodes and is explained in detail in Appendix 3.
- "Replace overlapping": In cases where one route fully overlaps with another route in the same set, we replace the shorter route with a newly generated route[13]. Details are given in John (2016).

[12] $Z \in [0, \frac{n_{max}}{2}]$ is randomly selected at the beginning of the operation [as in Mumford (2013)].

[13] The new route is generated by a version of the generation procedure of Shih and Mahmassani (1994) which is modified for the use of terminal nodes. The original procedure selects the node pair with the highest travel demand that is not yet directly connected and generates a shortest path route between them. For the use with terminal nodes it has to be ensured that only terminal node-pairs can be picked.

[14] It is important to note that for the operation "Delete Nodes" the repair function is disabled as it could undo the mutation.

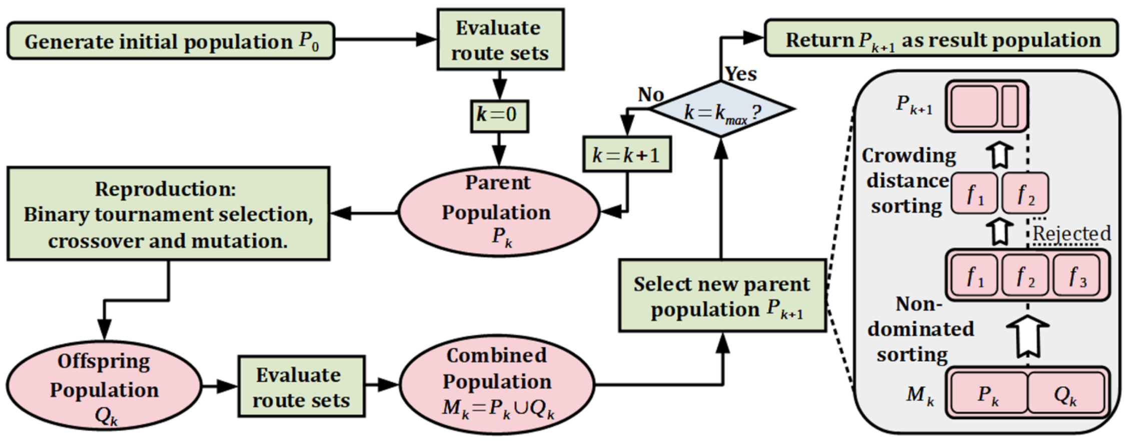

Generate initial population P_0
Evaluate route sets
$k=0$
Parent Population P_k
Reproduction: Binary tournament selection, crossover and mutation.
Offspring Population Q_k
Evaluate route sets
Combined Population $M_k=P_k \cup Q_k$
Select new parent population P_{k+1}
$k=k+1$
$k=k_{max}$?
No
Yes
Return P_{k+1} as result population
P_{k+1}
Crowding distance sorting
f_1
f_2
Rejected
f_3
Non-dominated sorting
M_k
P_k
Q_k

Fig. 5 Flow diagram for NSGAII. In every generation k the parent population P_k is used to generate an offspring population Q_k. The parent population of the next generation P_{k+1} is then selected from the combined population M_k based on domination and crowding distance

These repair algorithms are automatically run when the corresponding constraint validation is detected. If the repair is successful, the feasibility test continues for the remaining constraints.

4 Comparison of optimisation result and real bus routes

One way to evaluate the effectiveness of our bus routing optimisation is by comparison with real-world bus routes. Such comparisons are relatively rare in the available literature. One reason for this could be that the often used published instances, such as the Mandl instance (Mandl 1979), do not come with a given set of real-world bus routes. From the researchers who create their instances, a few have made comparisons with the real-world routes within their study area [e.g. Bagloee and Ceder (2011), Bielli et al. (2002), or Cipriani et al. (2012)].

4.1 Necessary network reduction

For our Nottingham instance generated with the method in Sect. 2, a comparison between the real-world bus routes and the routes generated by the algorithm described in Sect. 3 is not straightforward. As the evaluation procedure for route sets is based on the average travel time between the different nodes, it consequently requires that all nodes of the network are included in the route sets. The real-world route set, however, does not include all the nodes generated by the rules in Sect. 2.3, as bus operators drop those parts of the network they consider too unprofitable. To make a comparison with the real-world routes possible, we first need to generate a reduced version of the Nottingham instance in which all the nodes not visited by the real-world routes are excluded (see Fig. 6).

Excluding the non-visited nodes essentially generates a second instance network. Travel times between the remaining nodes have to be recalculated and the travel demand has to be reassigned, as described in Sects. 2.4 and 2.6, respectively.

Demand from zones which are not within the catchment area of any of the remaining nodes will not be taken into account.

The final reduced instance has 376 nodes, 52 less then the full instance. The total demand is reduced by 14%. In the following we will run experiments with both instances to point out the differences necessary to serve different network sizes.

4.2 Extracting real-world routes

As mentioned in Sect. 2.5, it is possible to extract information about the existing bus routes from the UK's 2011 National Public Transport Data Repository (NPTDR)[6]. The NPTDR provides this information in the form of 990 Journey Patterns (JP). A JP consists of a list of stop points the buses traverse and the starting times of the bus journeys. To convert this information into a route set comparable with the results of our optimisation requires a three-step process: filtering JPs by starting time, converting stop points to nodes, and filtering out overlaps.

4.2.1 Filtering journey patterns by starting time

As not all Journey Patterns are in use at a given time, we need to filter out those routes which fit our demand data. As we generated our demand data from travel to work data, we aim to include only JPs active during the morning rush hour into our real-world route set. We therefore select only the 210 JPs which, according to the journey starting times, cover the entire time window of 7:30 am to 10:30 am.

4.2.2 Converting stop point lists to node lists

The NPTDR comes with a 2011 version of the National Public Transport Access Nodes (NaPTAN) containing the location of all bus stop points. We use this to link every stop point with the node closest to it, up to a distance $\rho = \frac{s}{2} + 2s_p$. Where $\frac{s}{2}$ is the maximal distance between a node and a junction it represents. s_p is the usual distance between a junction and a bus stop. For the Nottingham street network, we found $s_p = 30\,\text{m}$ which results in $\rho = 202\,\text{m}$.[15] Mapping the stop point lists of the JP to node lists allows the real-world routes to be compared with the routes generated by the optimisation algorithm.

4.2.3 Filtering out overlaps

Journey Patterns always contain only one journey in one direction. In contrast, the routes in our optimisation are undirected and represent travel in both directions. Also, there are several bus routes which consist of similar JPs which are used in alternation. To filter out these overlaps, we add the node lists of the JPs in randomised order to the real-world route set. Every time a new node list j is added it

[15] It should be noted that some post-processing is always needed to make sure that all stop points are correctly allocated.

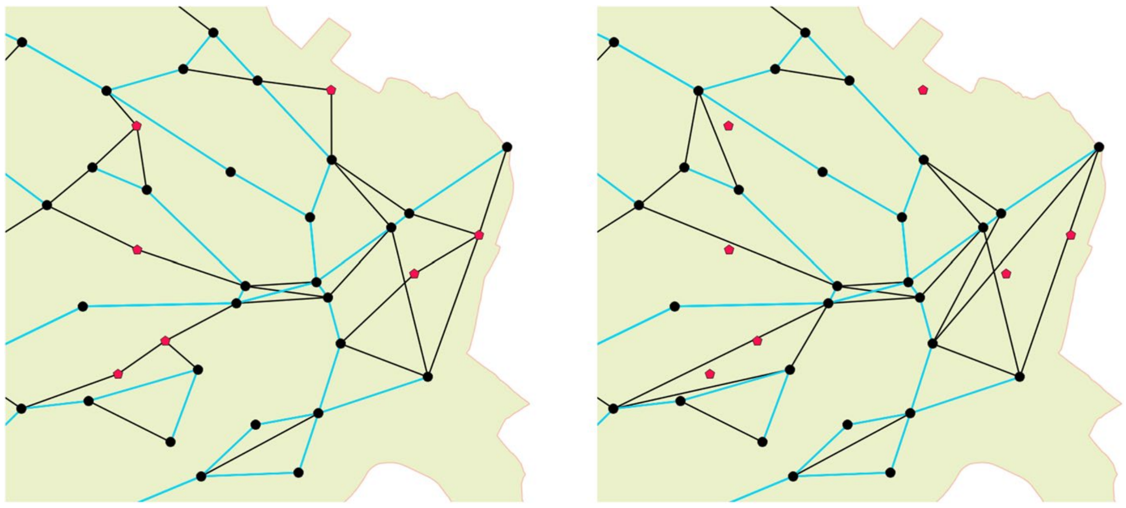


Fig. 6 Illustration of the differences between the original and the reduced instance network in the area of Gedling at the east end of the study area. Left: the real-world bus routes in projection on the original instance network (blue). Nodes of the network are represented as black circles and connections unused by the real-world routes as black lines. Red pentagons mark the nodes which are not served by the real-world routes. Right: real-world bus routes in projection on the reduced network: All nodes not served are removed from the network and the remaining nodes are connected accordingly (colour figure online)

is compared to all routes i already part of the real-world route set. This is done by calculating $\omega_{j,i}$ the overlap of j and i as well as the length λ of ω, j and i :

- If $\lambda(\omega_{j,i}) = \lambda(j)$: j is fully overlapped by i and is not inserted.
- If $\lambda(\omega_{j,i}) = \lambda(i)$: i is fully overlapped by j and j replaces i.
- If $|\lambda(\omega_{j,i}) - \lambda(j)| \leq m_i$ and $|\lambda(\omega_{j,i}) - \lambda(i)| \leq m_i$ and $|\lambda(i)) - \lambda(j)| \leq \frac{m_i+m_j}{2}$ with $m_x = \min(2, \frac{\lambda(x)}{10})$:
  There are only very small variations between j and i. If j's first journey starts earlier than i's first journey, j replaces i. Otherwise j is not inserted.

After this final filter process, 69 JPs remain as routes in our real-world route set.

5 Experimental results

We ran two computer experiments. Experiment 1 used the reduced instance (following Sect. 4) and its results can be directly compared with the real-world route set. Experiment 2 used the full instance (following Sect. 2). As the full instance network includes about 14% more nodes than served by the real-world routes the results of experiment 2 cannot be directly compared. However, optimising the route set on the full instance network is, nevertheless, a useful exercise in demonstrating the conditions under which a public transport service for the entire area can be realised.

In each case, we generated $|P| = 50$ initial route sets and optimised them with the described GA over 200 generations. Each of the route sets has $|R| = 69$ routes, the same number as the real-world route set.[16] For experiment 1 the maximal number of nodes per route was set to $l_{max} = 45$, which is around 10% more than the longest real-world route.[17] For experiment 2 we used $l_{max} = 52$, to reflect the approximately 14% larger size of the instance network. The minimal number of nodes per route was set to $l_{min} = 3$ for both experiments, one less than the shortest real-world route.

The results of both experiments are presented in Fig. 7. In both cases the evaluation results of the final route sets (black dots) form a clear Pareto front. In experiment 1 five route sets surpass the performance of real-world route sets (black x) in both objectives. Even for experiment 2 the results show that route sets serving the entire study area can be optimised to achieve evaluation results close to those of the real-world route set in the reduced instance.

To ease the discussion, the results at four key positions in the Pareto fronts for both experiments are highlighted and compared. At the extremes, the most passenger-friendly route sets (red) and the most operator-friendly route set (blue) are identified in the figure. Further, in yellow we can see the most passenger-friendly route set with shorter total route length than the real-world route set, while the most operator-friendly route set with shorter average travel time than the real-world route set is identified in green.

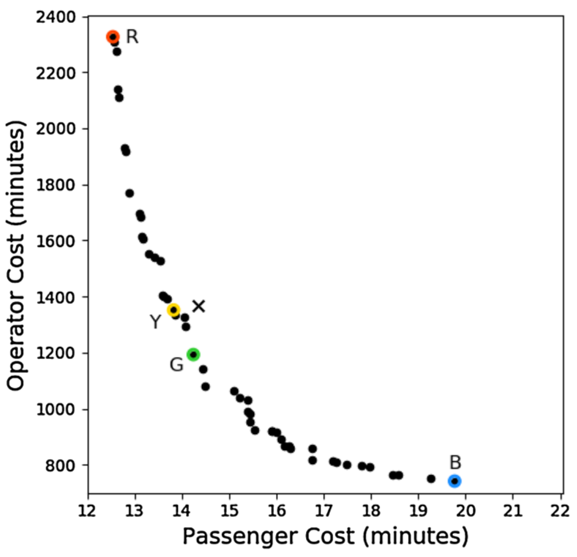
For experiment 1, the highlighted results are shown in more detail in Table 1. From the table we can see that the route set marked in yellow has slightly cheaper operator costs (− 1.24%) than the real-world route set, shortening the average travel time by half a minute. On the other hand, the route set marked in green reduces the operator cost by 12.9%, yet still achieves a better passenger cost (− 0.7%) than the passenger cost for the real route set. The general trend towards higher optimisation potential on the operator side is also present at the extreme ends of the Pareto fronts. We can observe that the most passenger-friendly route set (red), reduces the average travel time by 12.6 % (to 12.5 min) but increases the operator-cost by 69.8%. On the other hand, the most operator-friendly route set (blue) reduces the operator cost by 46% by driving up the average travel time by 38.5% (to almost 20 min).

While the yellow and green marked route sets of experiment 1 indicate that the real-world routes can be improved upon with our optimisation procedure, yellow and green marked route sets of experiment 2 show that it is possible to construct route sets serving the entire study area producing very similar operator and passenger costs to the real-world route set. The green route set offers approximately the same travel time as the real-world routes (14.2 min), although this comes at an increase in operator cost of about 22.7%. The yellow route set serves the larger network for approximately the same operator cost (− 1.7%) by prolonging the average travel time to 15 min.

It should be noted that all these comparisons are based on route optimisation only, and thus assume a fixed frequency of 10 min for all routes compared. Our

[16] We have chosen this after a sensitivity analysis showed that changing the number of routes does not lead to a general improvement of results.

[17] The higher limit reflects the possibilities for planners to construct slightly longer routes than currently exist.



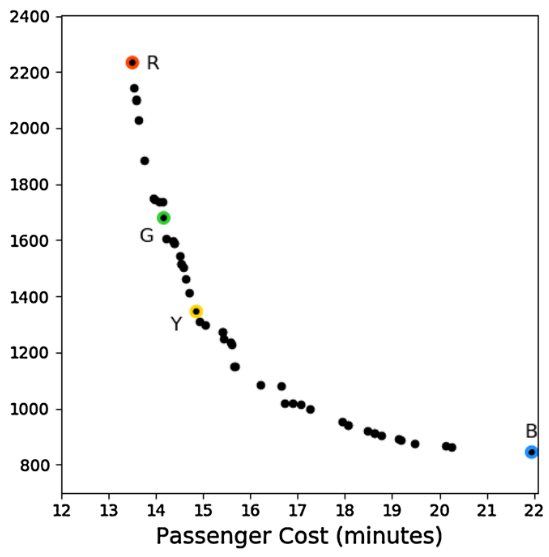


Fig. 7 Evaluations of the final population of 50 route sets from the GA (black dots) for both experiments, the reduced instance (left), and the full instance (right). In each plot four evaluation results are highlighted: the most passenger-friendly route set (red-R), the most operator-friendly route set (blue-B), the most passenger-friendly route set with shorter total route length than the real-world route set (yellow-Y), and the most operator-friendly route set with shorter average travel time than the real-world route set (green-G). Further, passenger and operator cost for the real route set is marked on the reduced instance (left) with an X (colour figure online)

Table 1 Comparison between optimisation results (as highlighted in Fig. 7) and the real world for optimisation criteria and transfer statistics

	Real routes	Red		Blue	
Average travel time	14.3 min	12.5 min	− 12.6%	19.8 min	+ 38.5%
Total route length	1369 min	2325 min	+ 69.8%	741 min	− 45.9%
% direct trips	30.6%	39.5%	+ 8.9%	16.9%	− 13.7%
% of 1 transfer trips	54.1%	48.5%	− 5.6%	34.4%	− 19.7%
% of 2 transfer trips	13.9%	11.1%	− 2.8%	28.0%	+ 14.1%
% of 3 + transfer trips	1.4%	0.9%	− 0.5%	20.7%	+ 19.3%
	Real routes	Green		Yellow	
Average travel time	14.3 min	14.2 min	− 0.7%	13.8 min	− 3.5%
Total route length	1369 min	1192 min	− 12.9%	1352 min	− 1.24%
% direct trips	30.6%	29.2%	− 1.4%	30.5%	− 0.1%
% of 1 transfer trips	54.1%	47.7%	− 6.4%	48.5%	− 5.6%
% of 2 transfer trips	13.9%	20.0%	+ 6.1%	18.8%	+ 4.9%
% of 3+ transfer trips	1.4%	3.1%	+ 1.7%	2.27%	+ 0.9%

future work looks to improve the realism of these comparisons by using more accurate demand data as well as including aspects such as frequency setting and multimodal interactions.

Table 1 further shows transfer statistics of the route sets, another performance measure used in the literature (see, e.g. Feng et al. 2010). The transfer statistics show the percentage of travellers reaching their destination, with none, one, two, three or more transfers. For the four marked route sets we see an evident increase in transfers for route sets with shorter total route length and longer average travel times. This observation is confirmed by the transfer statistics for all route sets shown in Fig. 8. This graphic shows a clear correlation between the passenger cost and proportion of passengers needing to make a specific number of transfers, with the number of passengers making zero or one transfer decreasing and the number making two or more transfers increasing, with increasing passenger cost. This behaviour is present in the results of both experiments indicating that it is independent from the specific network or network size. The reason for it lies in the differences in route coverage density in the network as shown in Fig. 9 for the two extreme Pareto route sets for the full instance. By the route coverage density, we simply mean the number of routes covering each link of the network.

Figure 8 further shows that the real-world route set (displayed by 'X' markers) has a comparatively low number of transfers, minimising transfers at the expense of operator costs. One potential explanation for this low number of transfers is the fragmentation of the Nottingham bus market. In the absence of tickets valid for all companies, direct travel is monetarily attractive for passengers. Companies optimise their networks individually to attract passengers with direct travel. This leads to a network with higher operator costs overall, due to unutilised transfer potential. However, more research would be required to confirm this.

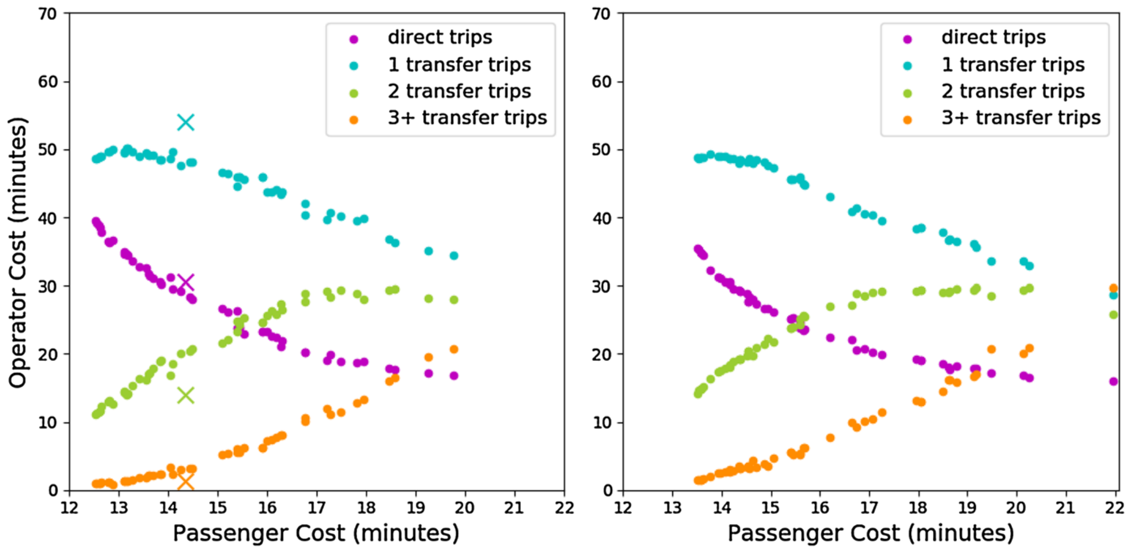


Fig. 8 Transfer statistics for the reduced instance experiment (left) and full instance (right). The coloured dots show the percentage of travellers reaching their destination with direct trips (purple), with one transfer (blue), two transfers (green), or three or more transfers (orange). Further, 'X' markers display the transfer statistics for the real-world route set (colour figure online)

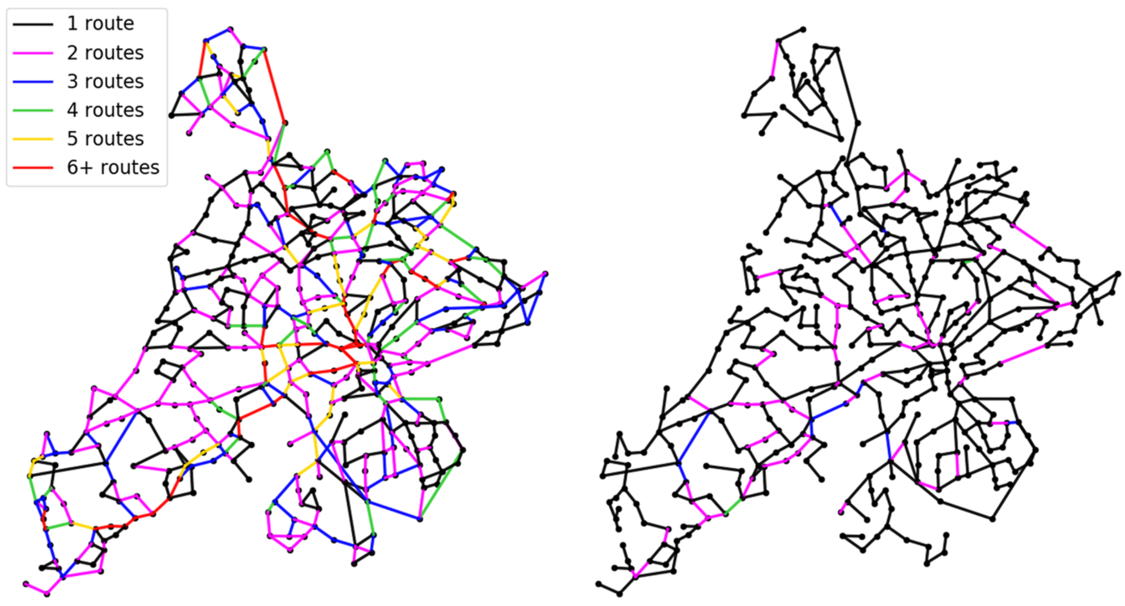


Fig. 9 Network density for the extreme Pareto points in the full instance. Left: the most passenger-friendly route set (i.e. the red route set in Fig. 7). Right: the most operator-friendly route set (i.e. the blue route set in Fig. 7). The colour coding shows the number of routes sets serving a specific connection within the network. The most operator-friendly route set serves most connections with only one route causing many transfers. The most passenger-friendly route set has many connections served multiple times increasing the chance that passengers can travel directly to their destination

6 Conclusion

In this paper we introduced a new methodology for modelling and scaling down a street network to facilitate optimisation of public transport networks in a realistic way. Furthermore, we built our model using only data that is freely available from public sources. The process involves a systematic placement of nodes on junctions of a street network suitable for bus travel. The travel times between the nodes are generated from street data, the information about potential terminal nodes is derived form existing bus routes, and the travel demand was extracted from UK census data. In the initial study, we applied our techniques to the bus network of greater Nottingham.

We showed that the pre-existing public transport routes can be projected onto the generated instance network, allowing the direct comparison between optimisation results and pre-existing public routes in a reduced version of the generated instance. Additionally, we adapted an evolutionary multi-objective optimisation algorithm (NSGAII) to operate effectively on our Nottingham instance characterised by restricted terminal nodes (i.e., where buses can turn around). The comparison between our results and the actual 2011 bus routes of the study area indicate that it is possible to reduce both the average passenger travel time and the operator cost simultaneously. Our results further suggest that a bus network optimised for direct travel (i.e. for a fragmented service with many different operators) is not the most effective network for passengers or operators. Given that this work is only the first step in an incremental approach for general public transport optimisation, these results are very promising.

There are several possible directions for future work. One is to improve the instance generation, e.g. by using better resolved demand data, or by generating asymmetric travel time matrices to better capture the impact of turn restrictions in a similar way to what was done in Perugia et al. (2011). Extending the optimisation to one or more of the four remaining phases mentioned in Sect. 1 (Vehicle Frequency Setting, Timetabling, Vehicle Scheduling and Crew Scheduling), is another. Allowing variable numbers of routes in our route sets, is also an important investigation to be carried out, as well as introducing multiple modes of public transport. In addition, many improvements could be made by changes in the objective functions. This may include techno-economic aspects such as vehicle crowding or required fleet size for more realism [see e.g. Jara-Diaz and Gschwender (2003) or Moccia et al. (2018)]. Further, changes to the objective function can also alleviate some of the less realistic constraints for valid route sets. Most notably is the constraint that all nodes have to be part of every route set, which forced us to generate a reduced instance in order to compare our optimisation results against real-world routes.

We provide the instance data as well as the real-world route set, our results and a python program for route set evaluation alongside this paper to allow other researchers to employ their own optimisation algorithm. The material is also available online under https://data.mendeley.com/datasets/kbr5g3xmvk.

Acknowledgements We are thankful for the very helpful comments and suggestions of three anonymous reviewers. This work is partly funded by the Leverhulme Programme Grant RP2013-SL-015.

Appendix 1: Script to generate travel time matrix

This section describes the travel time matrix generation, as mentioned in Sect. 2.4. The process is done in three steps

1. Extracting travel time information from route data.
2. Feasibility check and auto-correction.
3. Final check and output.

We had all three steps executed by a single python script. Its structure can be seen in the flow diagram in Fig. 10.

Extracting travel time information

Auto-generate travel times from node positions

The basis for the first step is an ArcGIS network dataset of the available street network. We generated such a dataset from the UK Ordnance Survey integrated network layer with the procedure described in ESRI (UK) Ltd (2007). The described procedure can also be applied to data from other sources, provided these can be converted to network datasets. Such data should be available from most national transport authorities or from local authorities. Alternatively, street data from OpenStreetMap (https://www.openstreetmap.org) can be used as long as it is sufficiently accurate for the study area.

A network dataset allows us to generate travel time data with ArcGIS using the Network Analyst function "Closest Facility". This function generates routes[18] between the elements of one set of points ("Incidents") and the elements of another set of points ("Facilities").[19] The attribute table of the generated routes contains information about start and end points as well as the required travel time. By selecting the bus routing nodes (see Sect. 2.3.2) as both "Incidents" and "Facilities" we are able to generate routes from every node to every other node.

We only wish to use the travel times between directly connected nodes. Therefore, we need to determine which of the above generated routes are valid (connect two directly connected nodes) and which are invalid (connect two nodes which are not directly connected).

A route f_{ij} between two nodes n_i and n_j would be considered valid if it does not pass through any other node. It is therefore possible to check if f_{ij} is valid by counting the number of nodes alongside it, defined as A_{ij}. A node n_x is considered as alongside f_{ij}, if the path of f_{ij} is either directly going over n_x, or, in case n_x has been generated by snapping, over a junction represented by n_x (see Fig. 2). If only the starting node n_i and its end node n_j are alongside f_{ij} ($A_{ij} = 2$), f_{ij} is valid. If there are more nodes alongside f_{ij} ($A_{ij} \geq 3$), f_{ij} is not valid. Nodes are considered alongside f_{ij} if the distance between them and f_{ij} is $s/2$ or less. This is because a node can represent junction up to $s/2$ away from it (see Sect. 2.3.2).

The values of A_{ij} can be obtained with the ArcGIS function "Locate Features Along Routes" which gives out the nodes within a certain distance of a route.[20]

However, the fact there may be junctions which are represented by more than one node leads to situations where $A_{ij} \geq 3$ although n_i and n_j are directly connected (see

[18] These routes are not related to the public transport routes we talked about in other sections of this paper. The reason why we use the term again here is to be consistent with the ArcGIS terminology.

[19] Parameters used for "Closest Facility" function:

- Analysis settings:
  - Impedance: drive (min)
  - Facilities to find: *as many as possible*
  - U-Turns at junctions: not allowed
  - Output shape type: true shape with measures
  - Use hierarchy: yes
  - Ignore invalid locations: yes
  - Restrictions: MandatoryTurnRestrictions, OneWay, TurnRestrictions
- Accumulation attributes: drive (min)
- Network locations—finding network locations:
  - Search tolerance: snapping distance s
  - Snap to: closest street-network (shape).

[20] Parameters used for locating features along routes function:

- Search radius: half snapping distance $\frac{s}{2}$
- Keep only the closest route location: no

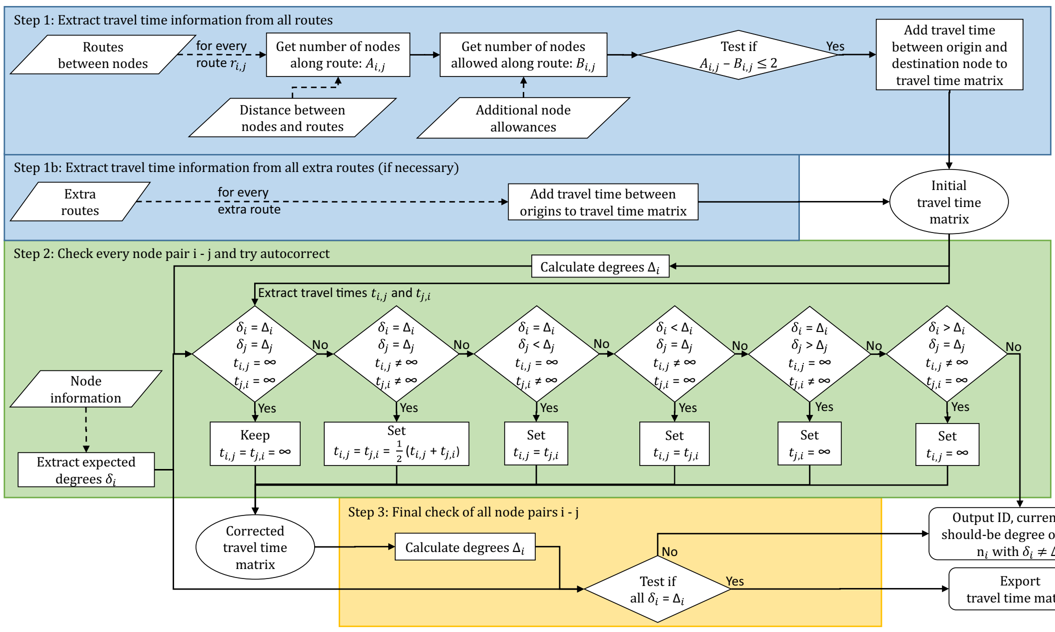

Step 1: Extract travel time information from all routes
Routes between nodes
for every route $r_{i,j}$
Get number of nodes along route: $A_{i,j}$
Get number of nodes allowed along route: $B_{i,j}$
Test if $A_{i,j} - B_{i,j} \leq 2$
Yes
Add travel time between origin and destination node to travel time matrix
Distance between nodes and routes
Additional node allowances
Step 1b: Extract travel time information from all extra routes (if necessary)
Extra routes
for every extra route
Add travel time between origins to travel time matrix
Initial travel time matrix
Step 2: Check every node pair i - j and try autocorrect
Calculate degrees Δ_i
Extract travel times $t_{i,j}$ and $t_{j,i}$
Node information
Extract expected degrees δ_i
Keep $t_{i,j} = t_{j,i} = \infty$
Set $t_{i,j} = t_{j,i} = \frac{1}{2}(t_{i,j} + t_{j,i})$
Set $t_{i,j} = t_{j,i}$
Set $t_{i,j} = t_{j,i}$
Set $t_{j,i} = \infty$
Set $t_{i,j} = \infty$
Corrected travel time matrix
Step 3: Final check of all node pairs i - j
Calculate degrees Δ_i
Test if all $\delta_i = \Delta_i$
No
Yes
Output ID, current und should-be degree of nodes n_i with $\delta_i \neq \Delta_i$
Export travel time matrix

Fig. 10 Algorithm to generate travel time matrix. The process consists of three steps from the input files generated manually and via ArcGIS. In step 1, the initial travel time matrix is generated from ArcGIS route information (see Sect. 1). Step 2 is an auto-correction procedure to fix simple mistakes in the route data based on comparisons with the expected degree of each node. This comparison is repeated in step 3 with the corrected matrix as a final check before outputting the travel-time matrix

Fig. 11). To cope with this situation, a third input file is necessary, which states how many more nodes B_{ij} are allowed alongside a route f_{ij} before f_{ij} is considered invalid. This additional input can be generated manually.

With both A_{ij} and B_{ij} known, it is finally possible to classify routes into valid and invalid. A route f_{ij} is called valid if $(A_{ij} - B_{ij}) \leq 2$. If this is the case, the route's travel time is inserted in the initial OD matrix. If f_{ij} turns out to be invalid, the travel time between n_i and n_j is considered as infinite.

Placing extra nodes

The "Closest Facilities" function generates the shortest path between two points. This, however, creates problems if the shortest path is not the most direct path (see Fig. 12). In these cases, the travel times for the direct connection has to be generated separately. In order to do so, we placed a so-called extra node along the direct connection and used the "Closest Facility" function again to generate routes from this extra node to the two nodes to be connected.[21] By summing up the travel time of both routes the travel time between the two nodes is generated and stored in the initial travel time matrix.

[21] For the generation of extra routes the same settings for the "Closest Facility" function are used as before. The only differences are that the extra nodes are used as "Incidents" and the "number of facilities to find" is set to 2.

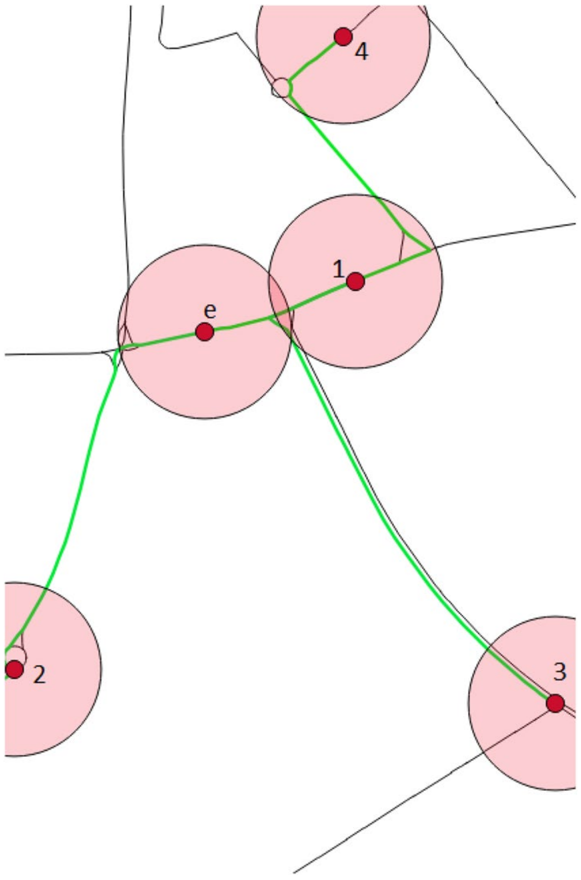


Fig. 11 ArcGIS generated routes (green) leading from four different starting nodes 1, 2, 3 to node e. The distance of $\frac{s}{2}$ around the nodes displayed (red circle). Note that the junction between node e and node 1 is represented by both nodes. The valid routes $f_{1,e}$ and $f_{2,e}$ have only two nodes alongside them. The invalid route $f_{4,e}$ has three nodes alongside it (e, 1, and 4). The route $f_{3,e}$ has to be valid; however, it has three nodes alongside it (e, 1, and 3). In this case, an additional node allowance $B_{3,e} = 1$ has to be used to allow $f_{3,e}$ to be considered a valid route (colour figure online)

Feasibility check and auto-correction

The "Closest Facility" function sometimes produces nonsensical routes (see Fig. 13) and it is therefore important to check if the entries in the initial travel time matrix are correct. In order to do this, we calculate the degree δ_i, the sum of all direct neighbours, of node n_i, and compare to its expected degree Δ_i.[22]

It is then possible to check for every node pair n_i, n_j if the calculated and expected degrees of both nodes match up. The result of this comparison is used for an auto-correction procedure:

1. If $\delta_i = \Delta_i$ and $\delta_j = \Delta_j$ and $t_{ij} \neq \infty$ and $t_{ji} \neq \infty$:
   Everything correct: Both travel times are averaged to create a symmetrical matrix $t_{ij} = t_{ji} = \frac{1}{2}\left(t_{ij} + t_{ji}\right)$
2. If $\delta_i = \Delta_i$ and $\delta_j = \Delta_j$ and $t_{ij} = t_{ji} = \infty$:
   Everything correct.
3. If $\delta_i = \Delta_i$ and $\delta_j < \Delta_j$ and $t_{ij} \neq \infty$ and $t_{ji} = \infty$:
   f_{ji} seem to have been wrongly classified as invalid: set $t_{ji} = t_{ij}$
4. If $\delta_i < \Delta_i$ and $\delta_j = \Delta_j$ and $t_{ij} = \infty$ and $t_{ji} \neq \infty$:
   f_{ij} seem to have been wrongly classified as invalid: set $t_{ij} = t_{ji}$
5. If $\delta_i = \Delta_i$ and $\delta_j > \Delta_j$ and $t_{ij} = \infty$ and $t_{ji} \neq \infty$:

[22] The expected degree Δ_i of a node n_i has to be generated manually.

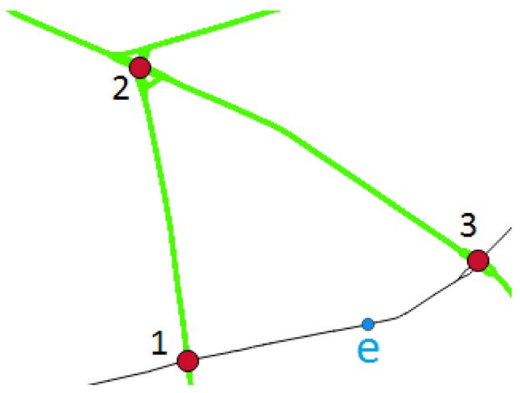


Fig. 12 ArcGIS generated routes (green) between several nodes. The shortest path both between the nodes 1 and 3 goes over node 2. The street connecting node 1 and node 3 is not used by the "Closest Facility" function. In order to extract the travel times along this connection an extra node e is inserted. This allows to generate the routes $f_{e,1}$ and $f_{e,3}$ and to sum up their travel time to $t_{1,3} = t_{e,1} + t_{e,3}$ (colour figure online)

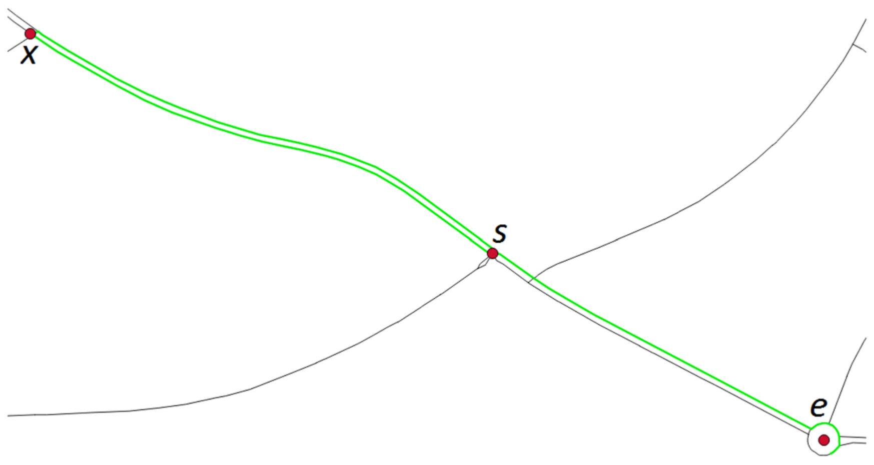


Fig. 13 ArcGIS generated route (green) from node s to node e leading over node x for no identifiable reason. (Node s is clipped to a to center of a junction allowing turns in both directions.) (colour figure online)

f_{ji} seem to have been wrongly classified as valid: set $t_{ji} = \infty$

6. If $\delta_i > \Delta_i$ and $\delta_j = \Delta_j$ and $t_{ij} \neq \infty$ and $t_{ji} = \infty$:
   f_{ij} seem to have been wrongly classified as valid: set $t_{ij} = \infty$
7. If none of the above:
   An issue that has not been caused by ArcGIS but by mistakes in one of the input files. The script outputs the ID, calculated degree and expected degree of the nodes where $\delta_i \neq \Delta_i$ to help identify the source of the problem.

Final check and output

Finally the degrees of all nodes are calculated and compared against the expected degrees. If $\delta_i = \Delta_i$ for all nodes i, the final travel time OD matrix is generated. Otherwise, the script outputs the list of nodes with $\delta_i \neq \Delta_i$ which serves as a basis for adjustments on the inputs for the next run of the script. Further, a lines shape file for all valid connections is generated to allow a separate visual check if all connections were generated correctly. (These shape files were used in the generation of Fig. 6 and the right side of Fig. 1.)

Appendix 2: Mutation operations

The following sections outline the mutation operations “Delete nodes” and “Add nodes” as well as the “Add missing nodes” repair operation, all used as part of the GA. The other mutation operations “Exchange”,“Replace”, and “Merge” as well as the repair operation “Replace Overlapping” are not described here as our implementation of these operations does not differ significantly to earlier descriptions of these algorithms e.g. in John (2016) and Mumford (2013).

As the operators described here are quite complex, we advise following the provided flow diagrams while reading the explanatory text.

Terminology

Before starting the description of the algorithms we need to introduce some terms we use during the description:

- “List”: Described is a randomised list of all possible elements (e.g. a route list is a list of all routes in a route set in a randomised order.)
- “Select from list”: takes the first entry from a list and thereby returns the entries in a pseudo-random order. The selected entries are removed from the list so the list is empty once all elements have been selected.
- “Reset list”: reinserts all previously selected elements back into the list and reshuffles it.
- “Reverse route”: Routes are lists of nodes which can be changed to reverse order (e.g. $[n_1, n_2, n_3]$ to $[n_3, n_2, n_1]$). As it is assumed that routes are travelled in both directions reversing a route does not change the connectivity in the route set.
- “Reaching a terminal in X steps”: “Add nodes”- and “Add missing nodes”-operation require information about how many steps a node n_i is away from the next terminal node. (Each step means passing another node.) This information can be calculated from the adjacency matrix in advance.
- The maximal number of steps possible X_{max}: Determines the maximal number of steps allowed in “Add nodes“- and “Add missing nodes”-operation. X_{max} is set as the largest number of steps between any node in and the nearest terminal node in the instance network.

Delete nodes

The “Delete nodes” operator was first described in Mumford (2013) and used to delete nodes from the end of randomly selected routes. However, this process needs to be more complex if a route has to end on a terminal node. Figure 14 shows the flow diagram of the “Delete nodes” mutation operation adapted for the use with terminal nodes.

At the beginning of the operation $Z \in [0, \frac{l_{max}}{2}]$ is determined at random. Z is the minimal number of nodes to be deleted in the entire route set.

After Z is set, the routes in the selected route set are sorted into a random order, and the first route of this list is selected as r. It is first checked if r includes more then two terminal nodes, as otherwise deleting one node would make r invalid. If r includes enough terminals, a copy r_{orig} is made and one node after another is deleted until r again ends on a terminal node. If $len(r) \geq l_{min}$ is still true, the shortened route is accepted and reinserted into the route set. If r becomes too short, the original route r_{orig} is restored. If the route has not yet been reversed, it is reversed now and a new attempt to delete nodes is started with the reversed route. If deleting nodes in the reversed route again leads to a too short route, the next route in the route list is selected.

This process is repeated until at least Z nodes have been deleted from the total route set or all routes have been tried out.

Add nodes

The "Add nodes" operator was first described in Mumford (2013) and added adjacent nodes at the ends of randomly selected routes. However, this process needs to be more complex if a route has to end on a terminal node. This new version uses a guided random walk to connect a given route to the next possible terminal node. This process ensures that the "Add nodes" operation is balanced with the "Delete nodes" operation (which deletes nodes at least until it reaches the next terminal). Figure 15 shows the flow diagram of the "Add nodes" mutation operation.

At the beginning of the operation $Z \in [0, \frac{l_{max}}{2}]$ is determined at random. Z is the minimal number of nodes to be added in the entire route set.

After Z is set, the routes in the route set are arranged in random order, and the first route of this list is selected as r. A copy r_{orig} made from it, and a step counter X set to $X = 1$. It is then tested if X nodes can be added to r (if $len(r) + X \leq l_{max}$). If this is true, the algorithm checks if there are terminal nodes V_t (at least one), which are not yet part of r and which can be reached in X steps from $r's$ current last node n_l. If no V_t can be found, the step counter is increased to $X = X + 1$ and, it is again checked if terminal nodes V_t, fulfilling the above mentioned conditions, do exist. This process is repeated as long as $X \leq X_{max}$ and $len(r) + X \leq l_{max}$ is true.

If nodes V_t do exist, but there is more than one step needed to reach them ($X > 1$), it is tried to close the gap by appending other nodes to r. As the success of this process is not guaranteed, a copy X_{orig} is made from X to be able to restore it later.

The nodes to close the gap are selected via a guided random walk: It is tested if there are nodes N_k adjacent to r's last node n_i, which are not yet part of r and can reach a node in V_t in $X - 1$ steps. If node(s) N_k exist, one of them is selected at random and is appended to r, thereby becoming the new last node n_l and X is reduced by one. This process is repeated until $X = 1$ is reached. When this happens, one of the terminal nodes in V_t which is adjacent to r's current last node n_l is selected at random and appended to r. Route r is now again valid (it ends again on a terminal node).

If node(s) N_k do not exist (because all potential nodes are already used within r), the original route r_{orig} is restored, the step counter is set to $X = X_{\text{orig}} + 1$).

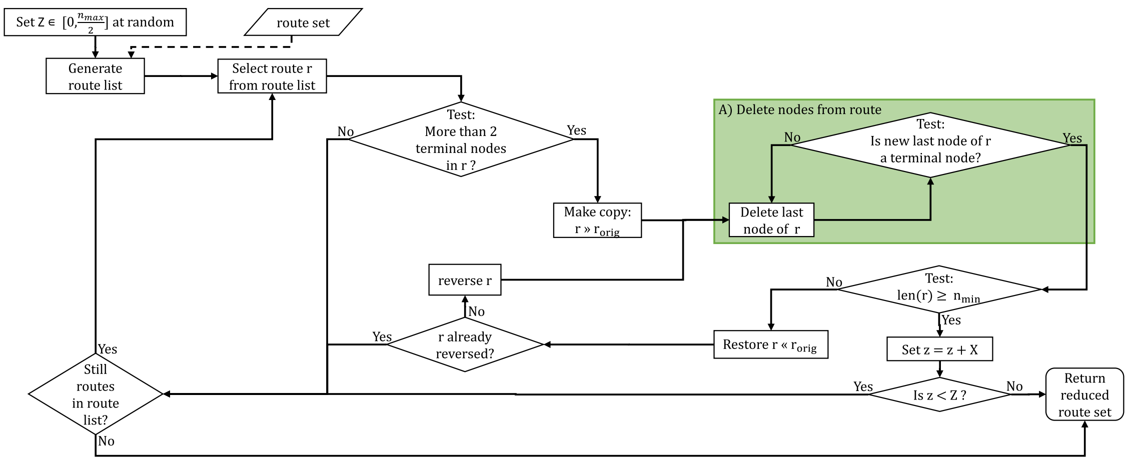


Fig. 14 Flow diagram of "Delete nodes" mutation operation: the operation begins with the selection of a route from a randomised route list. If the routes include more then two terminal nodes, the operation starts to delete nodes until either the route becomes too short or it ends again on a terminal node (box A). If the resulting route is too short, the original route is restored and reversed and the process started again. If the attempt fails, a new route is selected

If either the step counter reaches the maximal level ($X > X_{max}$) or r would get too long if X nodes would be added to it, the attempts to append nodes to this route are stopped. If the route has not yet been reversed, it is now reversed and the whole process starts again with a new last node n_l. However, if the reversed route can not be extended, a new route is selected.

The entire process is repeated until either at least Z nodes have been added to the route set or all routes have been tried out.

Add missing nodes

A repair operation which reinserts missing nodes back into the route set was first introduced in Mumford (2013). However, the original version only added the missing nodes to randomly selected routes. This process is not sufficient if a route has to end on a terminal node.

To take the constraint of terminal nodes into account, this version has two phases: The first phase ensures that all terminal nodes are included in at least one route. It also tries to reconnect as many other missing nodes as possible in the process. The flow diagram of the first phase is shown in Fig. 16. The second phase tries to insert remaining missing non-terminal nodes in randomly selected routes. Its flow diagram is shown in Fig. 17.

Add missing terminal nodes

The first phase starts with a step counter X set to $X = 1$, the generation of a randomised route list and the selection of a route r from that list. It is tested if r can be extended by X nodes. If yes, it is further tested if there are missing terminal nodes V_t

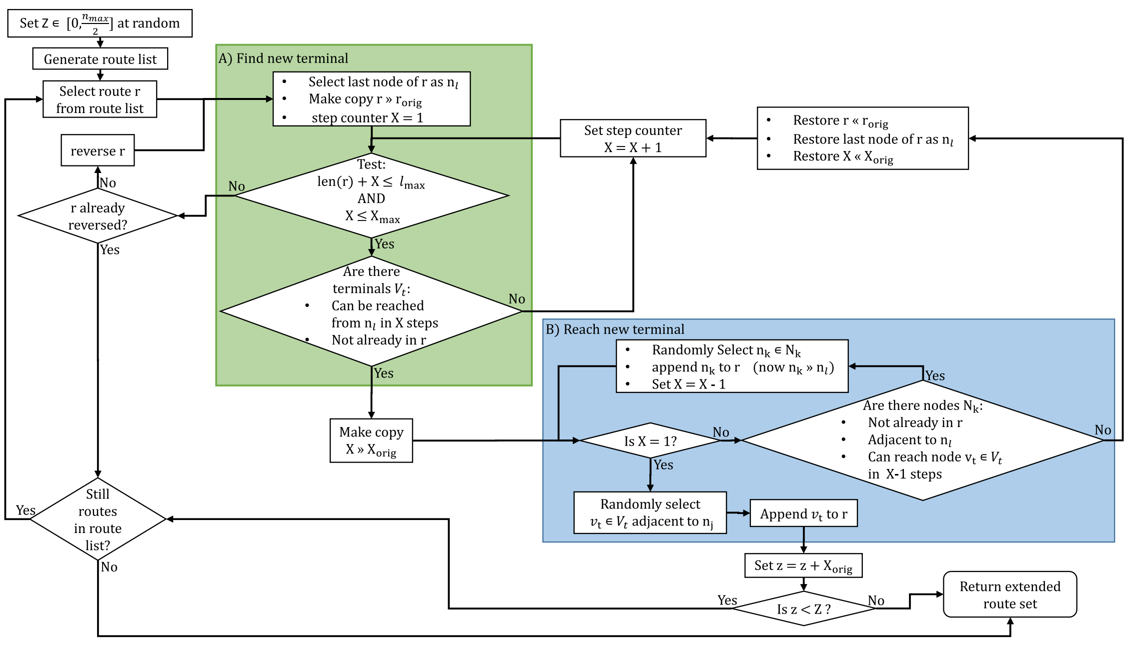

Set Z ∈ [0, n_max/2] at random
Generate route list
Select route r from route list
reverse r
No
r already reversed?
Yes
A) Find new terminal
Select last node of r as n_l
Make copy r » r_orig
step counter X = 1
Test: len(r) + X ≤ l_max AND X ≤ X_max
No
Yes
Are there terminals V_t: Can be reached from n_l in X steps; Not already in r
No
Yes
Set step counter X = X + 1
Restore r « r_orig
Restore last node of r as n_l
Restore X « X_orig
B) Reach new terminal
Randomly Select n_k ∈ N_k
append n_k to r (now n_k » n_l)
Set X = X - 1
Yes
Are there nodes N_k: Not already in r; Adjacent to n_l; Can reach node v_t ∈ V_t in X-1 steps
No
Make copy X » X_orig
Is X = 1?
No
Yes
Randomly select v_t ∈ V_t adjacent to n_l
Append v_t to r
Still routes in route list?
Yes
No
Set z = z + X_orig
Is z < Z ?
Yes
No
Return extended route set

Fig. 15 Flow diagram of the “Add nodes” mutation operation: the operation begins with the selection of a route r from a randomised route list and then testing if X nodes can be added to r and if there are possible new terminal nodes V_t within X steps (box A). If node(s) V_t are found but cannot be reached directly, the additional steps are filled with other nodes n_k, each one step closer to the new terminal (box B). If no suitable nodes can be found, the original route is restored and the step counter is increased. If too many steps would be required, r is first reversed and the process started again. If this attempt also fails, a new route is selected

which can be reached in X steps from either r’s last node n_l or r’s first node n_f of r. If no V_t exists, or if r would become too long, a new route is selected.

If V_t exists but there is more than one step needed to reach a node in V_t, other nodes are appended to r to close the gap. At this point a copy X_{orig} is made from X to be able to restore it later. (If V_t can only be reached from n_f r is reversed so that n_f becomes n_l.)

It is then tested if there are nodes N_k which are adjacent to r’s current last node n_l and can reach a node in V_t in $X - 1$ steps. Furthermore, these adjacent nodes cannot be already part of r. If suitable adjacent nodes exist, one such node n_k is selected at random and appended to r, thereby becoming the now n_l, and X is reduced by one. Missing nodes are preferred to speed up the repair process. This process is repeated until $X = 1$. Now V_t can be reached directly and one node $v_t \in V_t$ which is adjacent to n_l is selected at random and appended to r.

After a missing terminal node v_t has been successfully connected to a route, it is checked to see if further terminal nodes are missing. If yes, the next route from the route list is selected and the process starts again. If all routes have been tried, the step counter X is increased by one and the route list is reset. The process then starts again, checking for missing terminals one step further away from the end nodes of r.

If X reaches X_{max} before all terminal nodes could be connected, the process stops and the route set is returned as not repairable. If all missing terminal nodes can be connected to routes, it is tested if there are other (non-terminal) nodes

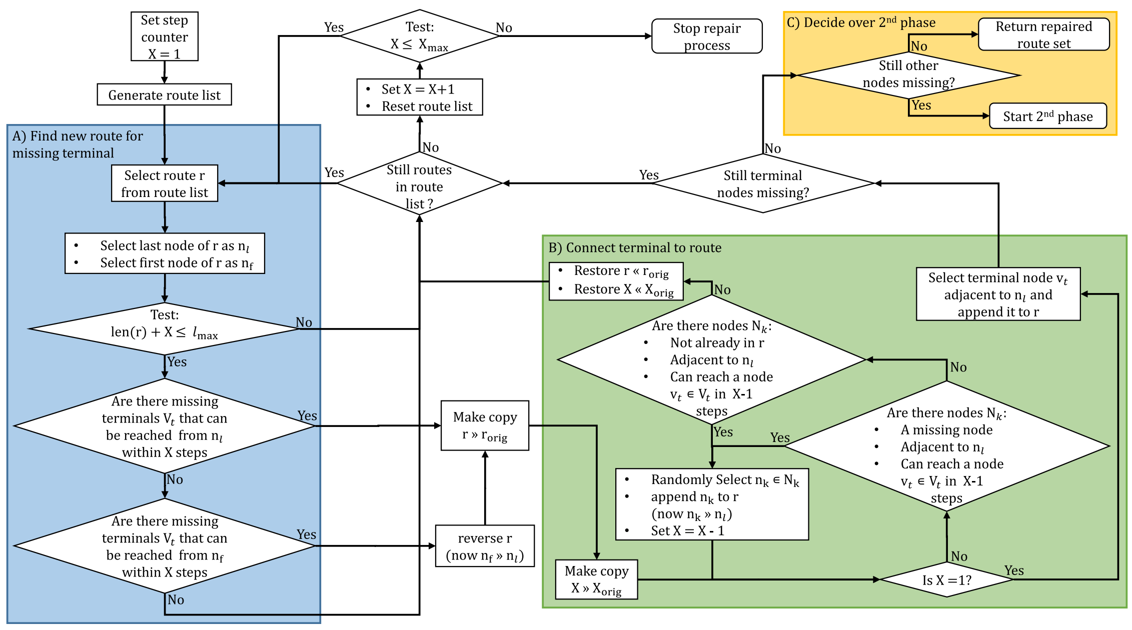

Set step counter X = 1
Generate route list
A) Find new route for missing terminal
Select route r from route list
Select last node of r as n_l
Select first node of r as n_f
Test: len(r) + X ≤ l_{max}
No
Yes
Are there missing terminals V_t that can be reached from n_l within X steps
Yes
No
Are there missing terminals V_t that can be reached from n_f within X steps
Yes
No
Make copy r » r_{orig}
reverse r (now n_f » n_l)
Yes
Test: X ≤ X_{max}
No
Stop repair process
Set X = X+1
Reset route list
No
Still routes in route list ?
Yes
Yes
Still terminal nodes missing?
No
C) Decide over 2nd phase
Return repaired route set
No
Still other nodes missing?
Yes
Start 2nd phase
B) Connect terminal to route
Restore r « r_{orig}
Restore X « X_{orig}
No
Select terminal node v_t adjacent to n_l and append it to r
Are there nodes N_k: Not already in r; Adjacent to n_l; Can reach a node v_t ∈ V_t in X-1 steps
No
Yes
Yes
Are there nodes N_k: A missing node; Adjacent to n_l; Can reach a node v_t ∈ V_t in X-1 steps
Randomly Select n_k ∈ N_k
append n_k to r (now n_k » n_l)
Set X = X - 1
Make copy X » X_{orig}
No
Is X =1?
Yes

Fig. 16 Flow diagram of the first phase of "Add missing nodes": the process starts a step counter set to $X = 1$ and the selection of a route r from a route list. It is tested if there are missing terminal nodes V_t within X steps from either the first or last node of r (box A). If at least one such node is found, it is either inserted directly, or, in case of $X > 1$, the additional steps are filled with other nodes, each one step closer to the missing terminal node. Other missing nodes are prioritised in this process (box B). If at least one missing terminal node cannot be connected to any route within X steps, X is increased by one and the routes are tried again. Once there are no missing terminal nodes the second phase starts, or, if there are no other missing nodes left, the repair process ends (box C)

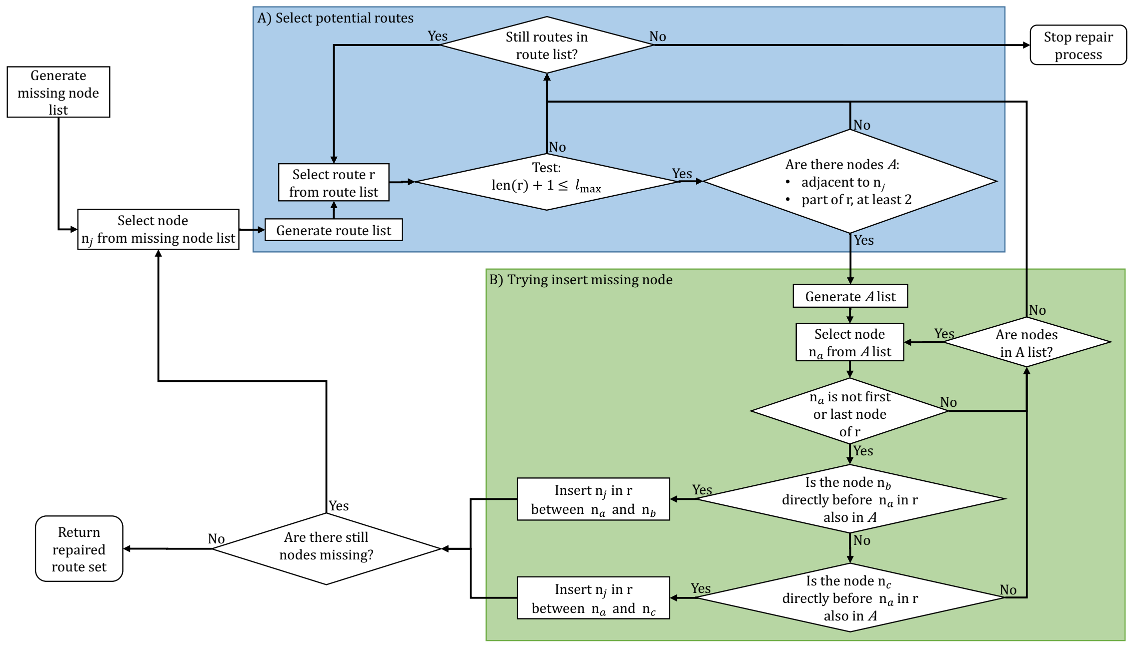

Generate missing node list
Select node n_j from missing node list
A) Select potential routes
Yes
Still routes in route list?
No
Stop repair process
Select route r from route list
Generate route list
No
Test: len(r) + 1 ≤ l_{max}
Yes
No
Are there nodes A: adjacent to n_j; part of r, at least 2
Yes
B) Trying insert missing node
Generate A list
Select node n_a from A list
Yes
Are nodes in A list?
No
n_a is not first or last node of r
No
Yes
Insert n_j in r between n_a and n_b
Yes
Is the node n_b directly before n_a in r also in A
No
Insert n_j in r between n_a and n_c
Yes
Is the node n_c directly before n_a in r also in A
No
Yes
Are there still nodes missing?
No
Return repaired route set

Fig. 17 Flow diagram of the second phase of "Repair nodes": the process starts with selecting a missing node n_j and a route r from randomised lists. It is tested if the route could take another node and if there is an overlap of at least two nodes between the nodes currently in r and the once adjacent to n_j (box A). If yes, it is tested if two of these nodes are consecutive nodes in r so n_j can be inserted in between them (box B). If such a combination cannot be found, a new route is selected until either all missing nodes are inserted in routes or there are no routes that were not tried

missing. If yes, the second phase of the repair procedure is started. If not, the repaired route set is returned to the GA.

Add remaining missing nodes

The second phase starts with the generation of a randomised route list and the selection of one node n_j at random from the missing nodes. Next, a route r is selected from the route list and is tested if it got too long if one node was inserted. If inserting a node is possible, it is tested if there is a group A of at least two nodes which is part of r and also adjacent to n_j. If no overlap can be found or r would get too long, a new route is selected.

If A can be found, one node $n_a \in A$ is selected at random. If n_a is neither the first nor the last node of r, it is tested if either the node n_b which is in r directly before n_a, or the node n_c which is in r directly after n_a is also in A. If one of these is the case, the node n_j is inserted into r either in between n_a and n_b or between n_a and n_c. If there are further nodes missing, a new node n_j is selected and the route list is reset. Otherwise the route set is returned as repaired.

If there is one node n_j that cannot be inserted into any route, the route set is returned as not repairable.

References

Ahmed L, Mumford CL, Kheiri A (2019) Solving urban transit route design problem using selection hyper-heuristics. Eur J Oper Res 274(2):545–559

Amiripour SM, Ceder AA, Mohaymany AS (2014) Designing large-scale bus network with seasonal variations of demand. Transp Res Part C Emerg Technol 48:322–338

Ammons DN (2001) Municipal benchmarks: assessing local perfomance and establishing community standards, 2nd edn. SAGE Publications, Thousand Oaks

Arbex RO, da Cunha CB (2015) Efficient transit network design and frequencies setting multi-objective optimization by alternating objective genetic algorithm. Transp Res Part B Methodol 81:355–376

Baaj MH, Mahmassani HS (1991) An ai-based approach for tansit route system planning and design. J Adv Transp 25:187–209

Bagloee SA, Ceder AA (2011) Transit-network design methodology for actual-size road networks. Transp Res Part B Methodol 45(10):1787–1804

Bielli M, Caramia M, Carotenuto P (2002) Genetic algorithms in bus network optimization. Transp Res Part C Emerg Technol 10(1):19–34

Ceder A, Wilson NHM (1986) Bus network design. Transp Res B 20B(4):331–344

Cipriani E, Gori S, Petrelli M (2012) Transit network design: a procedure and an application to a large urban area. Transp Res Part C Emerg Technol 20(1):3–14

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

ESRI (UK) Ltd (2007) Using OS MasterMap Integrated Transport Network (ITN)^{TM} Layer with ArcGIS . ESRI (UK) White Paper

Fan L, Mumford CL (2010) A metaheuristic approach to the urban transit routing problem. J Heuristics 16(3):353–372

Feng C, Hsieh C, Peng S (2010) Optimization of urban bus routes based on principles of sustainable transportation. J East Asia Soc Transp Stud 7:1137–1149

Golding J (2018) Best practices and methodology for OD-matrix creation from CDR-data. Technical report, University of Nottingham, Business School, N-LAB

Gutierrez-Jarpa G, Laporte G, Marianov V, Moccia L (2017) Multi-objective rapid transit network design with modal competition: the case of Concepción, Chile. Comput Oper Res 78:27–43

Jara-Diaz SR, Gschwender A (2003) Towards a general microeconomic model for the operation of public transport. Transp Rev 23(4):453–469

John MP (2016) Metaheuristics for designing efficient routes & schedules for urban transportation networks. Ph.D. thesis, University of Cardiff

John MP, Mumford CL, 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, pp 49–60

Kiliç F, Gök M (2014) A demand based route generation algorithm for public transit network design. Comput Oper Res 51:21–29

Leblanc LJ (1975) An algorithm for the discrete network design problem. Transp Sci 9(3):183–199

Mandl CE (1979) Applied network optimization. Academic Press, Cambridge

Mauttone A, Urquhart ME (2009) A route set construction algorithm for the transit network design problem. Comput Oper Res 36(8):2440–2449

Moccia L, Allen DW, Bruun EC (2018) A technology selection and design model of a semi-rapid transit line. Public Transp 10(3):455–497

Mumford CL (2013) New heuristic and evolutionary operators for the multi-objective urban transit routing problem. In: 2013 IEEE congress on evolutionary computation. IEEE, pp 939–946. https://ieeexplore.ieee.org/abstract/document/6557668

Nayeem MA, Rahman MK, Rahman MS (2014) Transit network design by genetic algorithm with elitism. Transp Res Part C Emerg Technol 46:30–45

Nielsen G, Nelson J, Mulley C, Tegner G, Lind G, Lange T (2005) HiTrans Best Practice Guide 2: Public transport – planning the networks. HiTrans. https://www.crow.nl/downloads/documents/13359

Pattnaik S, Mohan S, Tom V (1998) Urban bus route network design using genetic algorithm. J Transport Eng 124(4):368–375

Perugia A, Moccia L, Cordeau JF, Laporte G (2011) Designing a home-to-work bus service in a metropolitan area. Transp Res Part B Methodol 45(10):1710–1726

Poorzahedy H, Safari F (2011) An ant system application to the bus network design problem: an algorithm and a case study. Public Transp 3(2):165–187

Shih M, Mahmassani H (1994) A design methodology for bus transit networks with coordinated operations. Technical report, University of Texas, Center for Transportation Research

Silman LA, Barzily Z, Passy U (1974) Planning the route system for urban busses. Comput Oper Res 1(2):201–211

Soehodo S, Koshi M (1999) Design of public transit network in urban area with elastic demand. J Adv Transp 33(3):335–369

Turfitt R (2018) Statutory Document No. 14 Local bus services in England (outside London) and Wales. Technical report, Senior Traffic Commissioner (UK)

UK Department for Transport (2015) National public transport data repository. https://data.gov.uk/dataset/nptdr. Accessed 6 June 2019

UK Office for National Statistics (2016a) Census geography. http://www.ons.gov.uk/ons/guide-method/geography/beginner-s-guide/census/index.html. Accessed 6 June 2019

UK Office for National Statistics (2016b) Census output areas population weighted centroids 2011. https://geoportal.statistics.gov.uk/datasets/ba64f679c85f4563bfff7fad79ae57b1_0. Accessed 6 June 2019

Wills M (1986) Gravity-opportunities trip distribution model. Transp Res Part B Methodol 208(2):89–111

Wilson AG (1969) The use of entropy maximising models, in the theory of trip distribution, mode split and route split. J Transp Econ Policy 3(1):108–126

Zhang Y, Qin X, Dong S, Ran B (2010) Daily O-D matrix estimation using cellular probe data. In: TRB 89th annual meeting compendium of papers DVD, Transport Research Board. http://odd.topslab.wisc.edu/publications/2010/Daily%20O-D%20Matrix%20Estimation%20using%20Cellular%20Probe%20Data%20(10-2472).pdf

Zhao F, Gan A (2003) Optimization of transit network to minimize transfers. Technical report, Florida International University, Department of Civil and Environmental Engineering, Transport Research Board. https://trid.trb.org/view/697978

5 Zone-based public transport route optimisation in an urban network

List of Authors

Philipp Heyken Soares

Status by thesis submission

Accepted by the Springer Nature journal "Public Transport" since the 6th of July 2020 and in the final stage of production. Despite layout and formatting, the version presented here is identical to the one which will be published in "Public Transport".

Authors' contributions

The principle idea for this paper, the hybrid-procedure to determine zone-to-zone transit times (Section 2), was conceptualised by Philipp Heyken Soares.

The implementation of the hybrid-procedure, as well as the other required adaptations (Section 3), were implemented by modifying the programs written during the work on Chapter 4. Additional programming, as well as data selection and preparation, was necessary to prepare connector- and demand data (Section 4).

All this work was carried out by Philipp Heyken Soares.

All sections of this paper were written by Philipp Heyken Soares. At various stages of this process, the manuscript was shown to others (listed in the acknowledgement) for feedback on structure and understandability.

Summary of content

This paper expands the concepts and methods used in Chapter 4 from a node-based to a zone-based approach (see Section 3.3.2). As zone-based approaches are the standard approach for macroscopic transport modelling processes, this paper serves as an intermediate step between Chapter 4 and the Visum interface presented in Chapter 6.

The core element of this work is a hybrid approach for calculating zone-to-zone journey times on the basis of a previously calculated node-to-node travel time matrix (Section 2). This concept can further be used to determine the end nodes of a routes which optimally connects a pair of specified zones (Section 3.3). The fact that the process to determine the optimal route end nodes is not depending on the way the node-to-node travel times are calculated allows this method to be easily transferred to other models[1].

Additionally presented are a few further adaptations necessary to use the optimisation procedure introduced in Chapter 4 with the zone-based approach. In addition to the changes in route generation mentioned above, modifications are required in the selection of routes in the initialisation (Section 3.4.3) and the crossover operations (Section 3.5.1). In general, changes are kept to a minimum to change the working principles of the optimisation procedure as little as possible.

[1]One possibility would be to use it to generate routes for the optimisation of a Visum network model (more on this in Chapter 7)

Finally, the extension of the instance generation includes determining the connector travel times between zone centroids and nodes, based on official data on available walking connections (Section 4.3). The study area chosen for this work is a sub-area of the one used in Chapter 4. The smaller size of the instance was chosen, as the larger number of experimental runs for this work required a short runtime. The specific study area was selected for two reasons: a) The graph data were already available as a by-product of the work described in Appendix A. b) The small number of crossing points allowed us to include the movement of travelers across the boundary of the study area. For this, a method is introduced, which simplifies trips between zones inside and outside the primary study area to trips between zones and virtual zones at the crossing points (Section 4.4.2).

As in Chapter 4, the instance is published in the form of several files. In addition to node information, travel times, and travel demand, the zone-based instance presented here contains files with the locations of all the zone centroids, a file with direct walking connections between the centroids, and two files with connector times (one each for the origin and destination connectors). Further published is an additional node-based matrix that can be used with the above-mentioned travel time file to form an additional node-based instance.

Disclaimer

The publication strongly builds on the one presented in Chapter 4, which is referenced under the number 59. However, what was name "travel time" in Chapter 4 is renamed in this Chapter to "transit time" to better differentiate it from the full-time length of a PuT journey which here also includes walking times.

In a slight deviation of the description of the zone-based concept in Section 3.3.2, the zonal division used in this chapter is done in two separate layers for origin zones Z^O and destination zones Z^D. Further, to include a simple form of mode choice modelling, direct walking connections W are added between origin-

and destination centroids. Despite these unusual features, the presented methods can be applied to models with just one zone type by setting $Z^O = Z^D = Z$. The notation of all instance objects are summarised in info box 1.

Public Transport manuscript No.
(will be inserted by the editor)

Zone-based public transport route optimisation in an urban network

Philipp Heyken Soares

Received: date / Accepted: date

Abstract The majority of academic studies on the optimisation of public transport routes consider passenger trips to be fixed between pairs of stop points. This can lead to barriers in the use of the developed algorithms in real-world planning processes, as these usually utilise a zone-based trip representation. This study demonstrates the adaptation of a node-based optimisation procedure to work with zone-to-zone trips. A core element of this process is a hybrid approach to calculate zone-to-zone journey times through the use of node-based concepts. The resulting algorithm is applied to a input dataset generated from real-world data, with results showing significant improvements over the existing route network. The dataset is made publicly available to serve as a potential benchmark dataset for future research.

Keywords Public Transport · Route Optimisation · Network Design · Benchmark Instance · Genetic Algorithm

1 Introduction

1.1 Opening

The efficiency of public transport (PuT) is of vital importance for urban areas worldwide to decrease car dependency and the accompanying pollution and congestion. In general, the task to design efficient PuT networks can be described as five interconnected phases: 1) Route design, 2) Vehicle frequency setting, 3) Timetable development, 4) Vehicle scheduling, and 5) Crew scheduling [26]. Due to the interconnections, the combined task has a very high complexity and researchers typically work with simplifications. One such simplification is the Urban Transit Routing Problem (UTRP). It focuses on optimising the layout of routes while assuming a fixed time

Philipp Heyken Soares
Laboratory of Urban Complexity and Sustainability
University of Nottingham
E-mail: philipp.heyken@nottingham.ac.uk

penalty for all transfers (instead of varying transfer times resulting from different frequencies and starting times). The work presented in this paper is based on this approach.

Researchers have been working for many decades on automated procedures with which to solve the UTRP. Thus far, however, no results of this research have found widespread real-world application, and most planning processes are still based on experience and published guidelines [96, 126]. The reasons for this gap have not yet been researched in detail [126]. However, one possible explanation is that the concepts used in many studies are based on instances (i.e. sets of required input data) which are far removed from real-world planning processes [75].

This study is part of an incremental approach for better applicable UTRP research. The previous publication [60] focused on the generation of more realistic instances. The present paper builds on this work by adapting and extending the concepts used in [60] to a zone-based representation of journeys and travel demand. Compared with the node-based concept utilised in the vast majority of research studies (including [60]), the zone-based concept reduces restrictions for comparing the optimisation results to existing PuT networks, and allows including the effects of mode choice between PuT and other modes more easily in evaluation and optimisation. More importantly, however, it is the concept more commonly used in macroscopic transport modelling processes and, by extension, transport planning. Differences in data requirements between node-based and zone-based approaches can therefore create barriers to the practical application of UTRP algorithms.

The primary aim of this study is to adapt the methods used previously in [60] to work with zone-based travel demand. This includes:

- Introduction of a hybrid approach for calculating zone-to-zone journey times through the utilisation of established node-based concepts (in section 2).
- Adaptation of the optimisation procedure used in [60] to work with zone-based demand (in section 3).
- Extension of the instance generation procedure introduced in [60] to data required for zone-based algorithms (in section 4).

This paper further introduces two more general additions to the methods used in [60]:

- An improvement in the generation of routes under consideration of restricted start and end points (in section 3.3).
- A methodology to include trips across the boundaries of the study area into the generation of demand matrices (in section 4.4.2).

Furthermore, the instances described in section 4 and an evaluation procedure have been published online in order to increase the attractiveness of working with zone-based trip representations for other researchers.

1.2 Problem formulation - node-based and zone-based

Studies on the UTRP are usually based on an undirected graph $G = (N, E)$ representing the available transport infrastructure. Its nodes N represent access and interchange points and are connected by links E, that represent connecting infrastructure (e.g. streets for bus travel).

In such a graph, the public transport network can be represented as a set of routes $R = \{r_1, r_2, ..., r_{|R|}\}$. Each route r constitutes a list of directly connected nodes. The routes are considered undirected, assuming that vehicles after finishing one journey start a journey in the opposite direction. In an urban setting, this requires routes to begin and end on one of the designated terminal nodes $U \subseteq N$ which allow the performing of U-turns.

Optimising route sets requires criteria to evaluate the performance of different sets. A criterion used in many studies is the (average) passenger journey time [64]. Calculating passenger journey times in a transport model requires estimating the path passengers take through the available network. One of the factors[1] impacting these estimations is the representation of passenger journeys and travel demand. This study compares the two most commonly used concepts, which are illustrated in figure 1.

The most common approach in UTRP research is to present travel demand as trips between pairs of nodes. This node-based concept assumes that travellers use the same pair of beginning and end nodes for their PuT journeys (independent of R). Under this approach only in-vehicle travel and transfers of a journey are considered.

The alternative approach is to divide the stud area into zones each represented by a centroid. This allows to simplify[2] a trip with an origin in a zone O and a destination in a zone D as a trip between the centroids of O and D. Travellers use connectors to move between centroids and the nodes of G to board PuT services. As centroids can be connected to several nodes, passengers can often choose where they start/end their PuT journey.

In the present study the zonal division is carried out in two separate layers[3] for origin zones Z^O and destination zones Z^D (with connectors also being separated into C^O and C^D). To include a simple form of mode choice modelling, direct walking connections W between origin and destination centroids are added. All network objects are summarised in info box 1.

1.3 Background

Zone-based approaches are the most common concept in macroscopic transport modelling and the basis of many real-world planning processes. On the one hand can zone demand matrices be generated relatively simple, e.g. from mobile phone data (see e.g. [54]), or survey data (see section 4.4). Further, do trip distribution models and mode choice models require a zonal set-up as a common base of tips with all modes. Such

[1] A fully realistic modelling of the passengers' paths choice through the PuT network would also need to consider frequency- and capacity differences between routes. As mentioned in section 1.1, these aspects are not taken into account in the present study. Nevertheless, the arguments made here can be applied to models which include frequency and capacity differences.

[2] Aggregating trips on zonal level results in simplification errors which depend on the size and the layout of the zones.

[3] The majority of zone-based approaches use only one layer of zones. The separation into origin and destination zones used here is due to the travel demand data (see section 4.4). However, the methods presented in this study can be applied to models with only one zone type by setting $Z^O = Z^D = Z$. In this case, the walking vs. PuT mode choice is optional as very short trips will usually be excluded as inner-zone travel.

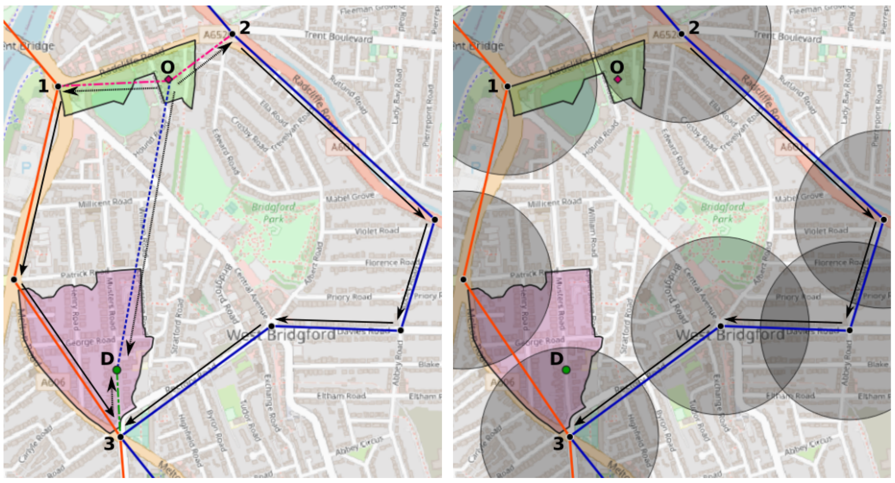


Fig. 1: Example of the impact of different trip representations on trip options. Assumed is a trip between two zone centroids *O* and *D*. Two PuT routes can be used: "orange" over nodes 1 and 3, and "blue" over nodes 2 and 3.
In a zone-based approach (left), travellers can choose from three options: a) walk from *O* to 1, ride on route "orange" to 3, and walk from 3 to *D*; b) walk from *O* to 2, ride on route "blue" to 3, walk on from 3 to *D*; c) use the direct walking connection from *O* to *D*.
In the node-based approach (right) the trip is assigned to nodes based on their catchment areas (grey circles). Travellers have only one option: travelling on route "blue" from 2 to 3. No walking is considered.
(The zones displayed in the figure were taken from the sets of zones described in section 4.1. The map underlying the image is taken from https://www.openstreetmap.org.)

models are an integral part of many more complex transport modelling processes, e.g. the standard 4-step model [88, 110, 113].

Nevertheless, the majority of researchers working on the UTRP prefer the node-based concept. Surveying more than a hundred publications on the UTRP using journey time calculation in their evaluation revealed that more than 80% of them use node-based travel demand. For the full list please see appendix A.

The main advantage of the node-based concept is that they are less complex and allow efficiently calculating the transit times through the use of standard shortest path algorithms (e.g. Floyd's algorithm [51] or Dijkstra's algorithm [40]). However, the fact that travel demand is assigned to the nodes themself, leads to parts of the demand being unserved if the designed PuT network does not include all nodes. Such unserved demand complicates the calculation of average journey times and almost all[4] studies instead constrain route sets to include all nodes. Unfortunately, this con-

[4] Of the 92 node-based studies listed in appendix A, only two ([33, 67]) were found to allow the exclusion of nodes.

$G = (N,E)$:	Graph structure representing the available infrastructure of the PuT system to be optimised. Composed of nodes E and links N.
$N = \{n_1, n_2, ..., n_{\|N\|}\}$:	Nodes of graph G.
$E = \{e_1, e_2, ..., e_{\|E\|}\}$:	Links of graph G connecting nodes N.
$U = \{u_1, u_2, ..., u_{\|U\|}\}$:	Terminal nodes $\in N$ where routes r are allowed to start or end.
$Z^O = \{z_1^O, z_2^O, ..., z_{\|Z^O\|}^O\}$:	Origin zones, usually represented by their centroids.
$Z^D = \{z_1^D, z_2^D, ..., z_{\|Z^D\|}^D\}$:	Destination zones, usually represented by their centroids.
$C^O = \{c_{a,i}^O, ..., c_{o,k}^O\}$:	Connectors between centroids of origin zones and nodes. Defined by connector matrix T^O.
$C^D = \{c_{i,b}^D, ..., c_{k,q}^D\}$:	Connectors between nodes and centroids of destination zone. Defined by connector matrix T^D.
$W = \{w_{a,b}, ..., w_{o,q}\}$:	Direct walking connections between origin and destination zones. Defined by connector matrix T^W.

Info box 1: Summary of instance objects (for the referenced travel time matrices see info box 2 on page 6).

straint can exclude otherwise advantageous route sets and restricts comparisons between optimised and existing PuT networks[5].

For determining the path and journey times of zone-to-zone trips, the literature knows several different approaches (for the respective publications see appendix A). A favourable option is to utilise professional transport modelling software[6], such as EMME [66] or PTV VISUM [107]. Unfortunately, not all researchers or planners have access to such software packages or the resources to replicate the complex assignment algorithms used in them.

Many UTRP studies with zone-based trip representations calculate journey times by using a “travel graph”. This extends on the graph G by including zone centroids and connectors as special types of nodes and links. Such a set up allows employing the same shortest path algorithms as commonly used in node-based approaches.

The present study takes a slightly different approach by introducing a new procedure to calculate zone-to-zone journey times. Its main advantage is in the way it can be utilised for the adaptation of the optimisation procedure (see section 3.3). Ad-

[5] When all nodes need to be included, optimisation results can only be compared to representations of the existing PuT network when both contain the same nodes. In [60] this forced the generation of a separated “reduced” instance to generate such route sets.

[6] To make interfacing UTRP algorithm and macroscopic transport modelling software more accessible, the author and others recently introduced an interface for the coupling of UTRP algorithms and PTV Visum in [59].

ditionally, it is straightforward to implement as it is based in established node-based concepts.

One other possible reason why UTRP researchers prefer using node-based demand is the availability of instances. Many researchers prefer to use publicly available instances. Doing so also allows a direct comparison of results and avoids the time-consuming work of generating own datasets. There are several published node-based instances (see, e.g. [4], [60], [83] or [91]). However, to the best of the author's knowledge, no zone-based instance has yet been made publicly available. This issue will be addressed by publishing the instance described in section 4 of this paper.

$D^Z = \left(d^Z_{a,b}\right)$:	Zone-based demand matrix of size $\lvert Z^O\rvert \times \lvert Z^D\rvert$. Gives number of trips from zone z^O_a to zone z^D_b.
$T^N = \left(t^N_{i,j}\right)$:	Node travel time matrix of size $\lvert N\rvert \times \lvert N\rvert$ (symmetric). Gives direct vehicle journey times between nodes n_i and n_j. If $t^N_{i,j} < \infty$: link e_k exist connecting nodes n_i and n_j
$T^O = \left(t^O_{a,i}\right)$:	Origin connector matrix of size $\lvert Z^O\rvert \times \lvert N\rvert$. Gives walking time between zone z^O_a and node n_i. If $t^O_{a,i} < \infty$: connector $c^O_{a,i}$ exists.
$T^D = \left(t^D_{i,b}\right)$:	Destination connector matrix of size $\lvert N\rvert \times \lvert Z^D\rvert$. Gives walking time between node n_i and zone z^D_b. If $c^D_{i,b} < \infty$: connector $c^D_{j,b}$ exists.
$T^W = \left(t^W_{a,b}\right)$:	Walking matrix of size $\lvert Z^O\rvert \times \lvert Z^D\rvert$. Gives direct walking time between zones z^O_a and z^D_b. If $t^W_{a,b} < \infty$: walking connection $w_{a,b}$ exists.

Info box 2: Summary of instance matrices for zone-based optimisation.

2 Calculating journey time

2.1 Defining journey time

The calculation of zone-to-zone journey times introduced in the following assumes that passengers will always use the path they perceive as the shortest. They also will choose the mode (walking or PuT) after this criterion.

The perceived length of a PuT journey can be expressed via a weighted sum. In professional modelling software (e.g. [66, 107]) a multitude of different walking-, in-vehicle-, and waiting times is considered. However, in line with the simplifications mentioned in section 1.1, the present study uses a reduced formulation: in the following the perceived PuT journey time $\theta^{PuT}_{a,b}$ between an origin zone z^O_a and a destination zone z_b^D is defined as

$$\theta_{a,b}^{PuT} = q_1 \cdot t_{a,i}^O + q_2 \cdot t_{i,j}^{InV} + q_3 \cdot t_{i,j}^{TP} + q_1 \cdot t_{j,b}^D \tag{1}$$

where $t_{a,i}^O$ is the walking time from z_a^O to a graph node n_i, $t_{i,b}^{InV}$ is the (total) in-vehicle travel time for the shortest PuT journey between nodes n_i and n_j, $t_{i,j}^{TP}$ is a cumulative time penalty for transfers necessary on that journey, and $t_{j,b}^D$ is the walking time from a graph node n_j to z_b^D. The factors q_1, q_2 and q_3 reflect that different time factors are weighted differently in the travellers' perception[7].

Once $\theta_{a,b}^{PuT}$ is calculated (see section 2.2) the final journey time $\theta_{a,b}$ can be determined by comparing $\theta_{a,b}^{PuT}$ to the direct walking time $t_{a,b}^W$:

$$\theta_{a,b} = \begin{cases} \theta_{a,b}^{PuT}, & \text{if } \theta_{a,b}^{PuT} < q_1 t_{a,b}^W \\ q_1 \cdot t_{a,b}^W, & \text{if } \theta_{a,b}^{PuT} \geq q_1 t_{a,b}^W \end{cases} \tag{2}$$

This process is repeated for every non-zero demand pair[8] to create a matrix Θ of size $|Z^O| \times |Z^D|$. Thereafter, Θ is then used in the calculation of the average journey time (as described in section 3.1).

2.2 Calculating zone-to-zone journey time

The first step in calculation $\theta_{a,b}^{PuT}$ is the generation of the node-to-node transit time matrix $\Lambda(R)$. The transit time $\lambda_{i,j}$ is defined as the combination of in-vehicle and transfer time for the shortest possible PuT transit between two nodes n_i and n_j:

$$\lambda_{i,j} = q_2 t_{i,j}^{InV} + q_3 t_{i,j}^{TP} \tag{3}$$

In a node-based concept, the transit time is the only travel time considered. Therefore, suitable methods to determine $\Lambda(R)$ are available in the literature. The present study uses the method from [45] (also used e.g. in [3, 60, 91]). This approach is based on the extended graph $\tilde{G}(R) = (\tilde{N}(R), \tilde{E}(R))$ (illustrated in figure 2). The nodes $\tilde{N}(R)$ are equal to those in N; however, they are multiplied every time they are used in a route $r \in R$. The links $\tilde{E}(R)$ represent all connections within R plus the transfer connections between the duplicate versions of nodes in $\tilde{N}$. While the length of the regular edges is as given in the travel time matrix T^N, the length of the transfer links is equal to the fixed transfer penalty. In this study, it is set as $t^{trans} = 5$ min[9]. This graph extension allows calculating $\Lambda(R)$ as the all-pairs shortest path matrix

$$\Lambda(R) = S\left(\tilde{G}(R)\right) \tag{4}$$

[7] θ^{PuT} is therefore formally referred to as "perceived journey time". However, unless the configuration explicitly deviates from $q_1 = q_2 = q_3 = 1$, the term "journey time" will be used for the sake of simplicity.

[8] If there is no travel between two zones z_a^O and z_a^D ($D_{a,b} = 0$) it is possible to skip the calculation of $\theta_{a,b}$ to safe computing time. In these cases, the value of $\theta_{a,b}$ will have no influence on the calculation of the average journey time (see equation 8 on page 9).

[9] This is in line with the definition of "frequent services" given by the UK Department for Transport, which constitutes having a maximum of 10 minutes between buses [123]. The vast majority of bus services in the study area that are active during the considered time period (see section 4) fall into this category.

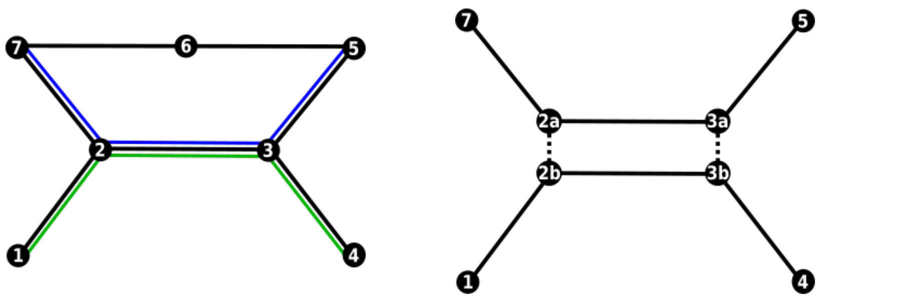


Fig. 2: A simple example of an extended transit graph $\tilde{G}(R)$, as described in [45]. The left side shows a regular graph G with seven nodes, used by a route set R with two routes (green: [1-2-3-4], blue: [5-3-2-7]). The right side shows the resulting extended graph $\tilde{G}(R)$. The nodes used in both routes are duplicated, and transfer links of length t^{trans} are inserted in between them (dashed lines). The nodes and link which are not used by any route, do not exist in $\tilde{G}(R)$.

After calculating $\Lambda(R)$ it can be used to construct the all-combinations journey time matrix $H_{a,b}(\Lambda(R))$, which gives the combined walking and transit times for a trip between zones z_a^O and z_b^D for all possible zone-node combinations:

$$H_{a,b}(\Lambda(R)) = \begin{bmatrix} q_1 t^O_{a,1} + \lambda_{1,1} + q_1 t^D_{1,b} & \cdots & q_1 t^O_{a,|N|} + \lambda_{|N|,1} + q_1 t^D_{1,b} \\ q_1 t^O_{a,1} + \lambda_{1,2} + q_1 t^D_{2,b} & \cdots & q_1 t^O_{a,|N|} + \lambda_{|N|,2} + q_1 t^D_{2,b} \\ \vdots & \ddots & \vdots \\ q_1 t^O_{a,1} + \lambda_{1,|N|} + q_1 t^D_{|N|,b} & \cdots & q_1 t^O_{a,|N|} + \lambda_{|N|,|N|} + q_1 t^D_{|N|,b} \end{bmatrix}$$

This equation can be written as a sum of three matrices:

$$H_{a,b}(\Lambda(R)) = \left(q_1 \tilde{T}^O_a + \Lambda(R) + q_1 \tilde{T}^D_b\right) \tag{5}$$

where the matrix $\tilde{T}^O_a$ is composed of $|N|$ copies of the a'th row vector of the connector matrix T^O, and the matrix $\tilde{T}^D_b$ of $|N|$ copies of the b'th column vector of the connector matrix T^D:

$$\tilde{T}^O_a = \begin{bmatrix} t^O_{a,1} & \cdots & t^O_{a,|N|} \\ & \vdots & \\ t^O_{a,1} & \cdots & t^O_{a,|N|} \end{bmatrix}, \quad \tilde{T}^D_b = \begin{bmatrix} t^D_{1,b} & & t^D_{1,b} \\ \vdots & \cdots & \vdots \\ t^D_{|N|,b} & & t^D_{|N|,b} \end{bmatrix} \tag{6}$$

With this are all three matrices ($\tilde{T}^O_a$, $\tilde{T}^D_b$, and $\Lambda(R)$) of size $|N| \times |N|$.

The PuT journey time can then be determined as the minimal value in $H_{a,b}$:

$$\theta^{PuT}_{a,b}(R) = \min\left(H^{a,b}(\Lambda(R))\right) \tag{7}$$

Once determined, $\theta^{PuT}_{a,b}(R)$ is compared to $q_1 t^W_{a,b}$ in order to obtain the final value for the journey time $\theta_{a,b}(R)$ (see equation 2). The execution of equations 7 and 2 are repeated for every non-zero demand pair to generate the complete zone-to-zone journey travel matrix $\Theta(R)$. However, $\Lambda(R)$ has to be calculated only once per R. A brief discussion on the resulting run times can be found in appendix B.

3 Optimisation procedure

The zone-based optimisation procedure described in the following is derived from the node-based approach utilised by the author and others in [60]. It centres on a genetic algorithm (GA) optimising route sets generated by a heuristic initialisation procedure.

The main adaptations are based on selecting the pair of nodes which form the beginning and end of the optimal PuT journey between two nodes. The basic process for this is a variation of the journey time calculation described in section 2.2. It is outlined in section 3.3. Additional adaptations were carried out to the route selection in step three of the initialisation procedure (section 3.4.3) and in the crossover operation of the GA (section 3.5.1), as well as to the repair operations (section 3.5.3).

3.1 Optimisation objectives

The optimisation procedure described here uses two competing objectives. Their formulation is, in principle, identical to [60] and other studies (e.g. [3, 91, 73]).

The first objective, i.e. the passenger objective, is to reduce the average passenger journey time[10], which is given by:

$$C_P^{\Theta}(R) = \frac{\sum_{a,b=1}^{|N|} d_{o,d}^{Z} \theta_{a,b}(R)}{\sum_{a,b=1}^{|N|} d_{a,b}^{Z}} \tag{8}$$

where $d_{a,b}^{Z}$ is the number of passengers travelling from zone z_a^O to zone z_b^D and $\theta_{a,b}(R)$ is the journey time between z_a^O and z_b^D.

The second objective, i.e. the operator objective, is to reduce the length of all routes as a simple[11] proxy for the cost of the operator:

$$C_O(R) = \sum_{j=1}^{|R|} \sum_{i=1}^{|r_j|-1} t_{i,i+1}(r_j) \tag{9}$$

with $t_{i,i+1}(r_j)$ referring to the travel time between two adjacent nodes i and $i+1$ in the route r_j. This formulation does not depend on the used demand representation.

3.2 Optimisation constraint

There is a list of constraints that all route sets R and their routes r have to fulfil during both the generation and the optimisation processes:

[10] [60] and other node-based studies use average transit time as the passenger objective. The mathematical formulation is identical to 8; however, D^Z and Θ are replaced by a node-based demand matrix D^N and the transit time matrix Λ.

[11] A more realistic calculation of such costs would require techno-economic data, e.g. on the fleet composition and vehicle crowding. Such approaches are out of the scope of this paper; however, have been proposed in other publications (see, for example, [69, 89]).

1. A route set R consists of a predefined number of routes $|R|$.
2. Each route r has minimal l_{min} and maximal l_{max} nodes.
3. No route r fully overlaps with any other route in R.
4. R is connected - every node in R is connected to all other nodes in R.
5. Nodes appear only once in any route r - there are no loops or cycles.
6. The first and last nodes of each route is a terminal node $\in U$.
7. Each zone centroid is connected to at least one node in R.

Besides constraint 7, these constraints are identical to those used in [60].

3.3 Determining optimal node pairs for shortest zone-to-zone travel

On several occasions during the optimisation process it is required to determine the nodes forming the beginning and end of the shortest path between two demand sources, e.g. to establish a new route between them. When adopting a node-based approach this is trivial as nodes and demand sources are identical. For a zone-based approach it is required to first identify the node pair $(n_i, n_j)_{a,b}$ for the beginning and end of the overall optimal PuT journey between two zones z_a^O and z_b^D. Utilising the journey time calculation procedure introduced in section 2.2, this can be carried out by determining the indices of the smallest entry of the all-combinations journey time matrix $H_{a,b}$:

$$(n_i, n_j)_{a,b}^G = \begin{cases} \underset{i,j \in [0,|N|]}{\operatorname{argmin}} \left(H_{a,b}(S(G)) \right) & \text{if } q_1 t_{a,b}^W > \min\left(H_{a,b}(S(G)) \right) \\ \emptyset & \text{if } q_1 t_{a,b}^W \leq \min\left(H_{a,b}(S(G)) \right) \end{cases} \tag{10}$$

where $S(G)$ is a matrix with shortest path node-to-node travel times on graph G[12]. If the minimum of $H_{a,b}(S(G))$ is larger than the direct walking connection ($q_1 \cdot t_{a,b}^W$), no optimal node pair exists.

One use of $(n_i, n_j)_{a,b}^G$ is the generation of a route $r_{a,b}$ optimally connecting the zones z_a^O and z_b^D. This route is then established as the shortest path between $(n_i, n_j)_{a,b}^G$ on G. In case one (or both) of the nodes is not a terminal node, $r_{a,b}$ is extended to a close terminal node using a guided random walk[13].

This technique to generate routes between non-terminal nodes, which complies with constraint 6, marks a general improvement from the approach presented in [60] where routes could only be generated between terminal nodes. It does not depend on the demand representation used.

Further, it relatively straightforward to transfer the technique described here to other models, as equation 10 is independent of the way $S(G)$ is calculated. For example, it can be used to add operations generating new routes to the optimisation proce-

[12] As G does not change during the optimisation it is possible to determine the $(n_i, n_j)_{a,b}^G$ and for all zone pairs in advance of the optimisation process in order to save run time.

[13] The process to extend routes to the next available terminal node via a guided random walk is described in detail in appendix B of [60] as part of the mutation operation “Add nodes”. Nevertheless, in [60], this process was not used in the generation of routes.

dure described in [59] which is interfaced with the transport modelling software PTV Visum[14].

3.4 Heuristic construction of route sets

Before starting the GA optimisation, an initial population of $|P|$ route sets needs to be generated. For this, the construction heuristic introduced in [60] is adapted for the zone-based approach. The process can be divided into the following steps:

3.4.1 1st step: constructing the reversed usage graph Ω

The process begins by noting the usage of each link, assuming that all travellers can travel on their shortest path. This is done by determining the shortest paths between the optimal node pairs $(n_i, n_j)^G_{a,b}$ of each origin-destination pair (z^O_a, z^D_b). Next, the reversed usage graph Ω is constructed as a copy of G with the travel times of the links being replaced by the total demand minus the usage of the links. Thus, the most used link becomes the shortest in reversed usage distance, and vice versa.

3.4.2 2nd step: generating candidate routes

The second step is to generate a palette of candidate routes. For this, the algorithm iterates through the zone pairs in order of demand, starting with the highest. For each pair (z^O_a, z^D_b), a route is generated as the shortest path on Ω between $(n_i, n_j)^G_{a,b}$.

Following the creation of each potential candidate route, the reversed usage distance of the links that it used is increased by 10%. This increases the likelihood of routes created later using less high-demanding links. If the generated route fulfils constraint 2, it enters the palette. The generation of routes continues until every zone is connected to at least c_z routes in the palette[15].

3.4.3 3rd Step: forming route sets by combining routes from the palette of candidate routes

In this final step, $|P|$ route sets are assembled from the palette of candidate routes. For the first route set R_1, the procedure begins by selecting the first route in the palette. The second route is chosen from all other routes in the palette which have at least one node in common with the first. Of these, the route with the highest coverage extension ratio $c_e(r)$ is selected.

For the node-based approach in [60] $c_e(r)$ is defined as ratio between $|n_{new}(r)|$ (the number of nodes in r which are not yet part of any other route in R_1) and $|r|$ (the

[14] Further information on combining the Visum interface described in [59] and the process to determine $(n_i, n_j)^G_{a,b}$ described here, are outlined in [58].

[15] The parameter c_z is set arbitrarily. The present study used $c_z = 10$, following some sensitivity analysis.

length of a route r in number of nodes) to spread the network coverage while maintaining a balance between shorter and longer routes in R_1. For zone-based approach the definition of $c_e(r)$ is modified to

$$c_e(r) = \frac{|n_{new}(r)|}{|r|} \cdot \frac{|z^O_{new}(r)| + |z^D_{new}(r)|}{|Z^o| + |Z^d|} \tag{11}$$

where $|z^O_{new}(r)|$ and $|z^D_{new}(r)|$ is the number of origin and destination zones connected to r but not to any other route in R_1.

The process to select new routes repeats, adding one route at a time until all zones are connected to R_1. Further routes are added at random until R_1 contains $|R|$ routes. After R_1 has been successfully generated the same process is repeated for the second route set R_2, however starting from the second route in the palette. This continues until $|P|$ route sets are assembled.

3.5 Genetic algorithm optimisation

The general structure of the genetic algorithm used in this study is that of Non-dominated Sorting Genetic Algorithm II (NSGAII). This genetic algorithm optimises a population of solutions (i.e. route sets) for two competing objectives simultaneously. It was first introduced in [38] and has since been used in multiple UTRP studies (e.g. [4, 60, 73]). Figure 3 presents a flow diagram of NSGAII. All changes are within the crossover and mutation operations and described in the following sections.

3.5.1 Crossover Step

During the crossover step an offspring population Q_k of size $|R|$ is generated. Each offspring route set Q^i_k is either a directly copied parent route set or, with a probability of $\rho_{cross} = 0.9$, constructed in a crossover operation from two parent route sets.

In the crossover operation, route sets are selected from both parents in alternation. The first route is selected at random from one parent. In the following, the routes of the other parent which include at least one node that is already part of Q^i_k are ranked according to their coverage extension ratio c_e (see section 3.4.3) and the route with the highest c_e is added to Q^i_k. This process repeats until all zones are connected to Q^i_k. Thereafter, routes are selected at random until Q^i_k consists of $|R|$ routes. After its generation, Q^i_k undergoes a feasibility test (see section 3.5.3). If it passes, it is inserted into Q_k. Otherwise, the crossover step restarts.

3.5.2 Mutation operations

After their generation in the crossover phase, each offspring route set undergoes mutations. The number of mutations in each route set is determined by a binomial distribution $B(|R|, \frac{1}{|R|})$. For every mutation, one of the following mutation operations is selected at random:

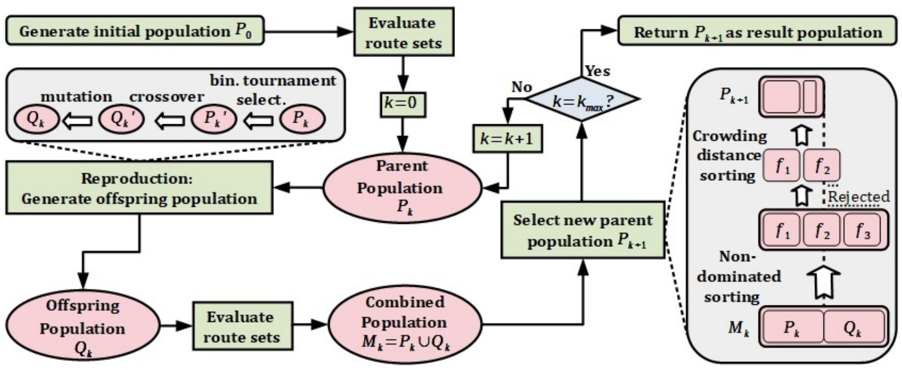

Generate initial population P_0
Evaluate route sets
Return P_{k+1} as result population
Yes
No
$k=k_{max}$?
bin. tournament select.
mutation
crossover
Q_k
Q_k'
P_k'
P_k
$k=0$
$k=k+1$
Reproduction: Generate offspring population
Parent Population P_k
Select new parent population P_{k+1}
P_{k+1}
Crowding distance sorting
f_1 f_2
Rejected
f_1 f_2 f_3
Non-dominated sorting
M_k
P_k Q_k
Offspring Population Q_k
Evaluate route sets
Combined Population $M_k=P_k \cup Q_k$

Fig. 3: Flow diagram of the NSGAII genetic algorithm setup. The $|P|$ route sets in the initial population form the first parent population P_0. In the reproduction phase, route sets from P_0 are selected in binary tournaments in order to generate the offspring populations Q_0 (also of size $|R|$) via crossover and mutation operations (as described in sections 3.5.1 and 3.5.2.) Q_0 is then combined with P_0 to form M_0. For the selection of the next parent population P_1, M_0 is divided into sets f based on domination amongst one another. Starting with the nondominated solutions, these sets are then added to P_1 until one set f_x cannot be added completely. The remaining route sets are selected from f_x in such a way as to achieve a more even spread of solutions. The process will repeat for a predetermined number of generations. Further details can be found in [38].

- "Delete nodes"[16]: deletes nodes from the end of randomly selected routes until they again end on a terminal node. In total, at least C nodes are deleted[17]
- "Add nodes"[16]: adds nodes at the end of randomly selected routes until a new terminal node is reached. In total, at least C nodes are added[17]
- "Exchange"[18]: splits two randomly selected routes at a common vertex. The divided parts are recombined into two new routes replacing the originals.
- "Merge"[19]: randomly selects two routes with a common terminal node and merges them into one route. Thereafter, a new route is generated.
- "Replace"[19]: replaces the route satisfying the lowest demand with a new route.

After every mutation, the changed route set needs to pass a feasibility test (see section 3.5.3). If it fails, the mutation is undone and a new mutation operation is selected. More information on the mutation operations can be found in [60].

The mutation operations "Delete nodes", "Add nodes" and "Exchange" do not require any changes for use in a zone-based optimisation. In "Merge" and "Replace" new routes are generated as described in section 3.3. The routes are generated between the pair of not optimally connected zones with the highest demand. A pair of

[16] This mutation operation was first proposed in [91]. In [60] it was adapted to constraint 6.
[17] $C \in [0, \frac{n_{max}}{2}]$ is set randomly at the beginning of the operation.
[18] This mutation operation was first proposed in [83].
[19] This mutation operation was first proposed in [73]. In [60] it was adapted to constraint 6.

zones (z^O_a, z^D_b) is considered to be not optimally connected in a route set R if no single route includes its optimal node pair $(n_i, n_j)^G_{a,b}$.

3.5.3 Feasibility test

Every route set generated in a crossover or changed in a mutation operation is subject to a feasibility test in order to check whether all of the constraints listed in section 3.2 are obeyed. Repair operations are called in case of two common constraint violations:

– "Replace overlapping"[20]: called by a violation of constraint 3. It replaces the overlapped route with a new route generated as described in section 3.3.
– "Add missing nodes"[16]: called by a violation of constraint 7. It connects unconnected nodes to randomly selected routes terminating once all zones are connected. (In [60] the process first stopped when all nodes were included in R.)

4 Instance datasets

4.1 Study area and data sources

An instance dataset for the zone-based optimisation described in this paper includes the instance matrices listed in info box 2 as well as information on terminal nodes. The following sections describe how these data can be generated.The primary study area for this process is the southern part of the metropolitan area of Nottingham, UK (including the areas of Clifton and West Bridgford and the village of Ruddington). It is presented in figure 4. Travel patterns in this area are significantly influenced by trips across boundaries, especially to the north to Nottingham city centre. To capture this cross-boundary flow, origins and destinations in an extended study area are also taken into account, as described in section 4.4.2. This extended study area is the travel-to-work area[21] of Nottingham and is presented figure 6.

Corresponding to the demand data used in section 4.4.1, the zonal division of the study area is taken from datasets of 2011 UK Census conducted by the UK Office for National Statistics (ONS). The low-level Census geography types[22] "Output Areas" (OA) and "Workplace Zones"(WZ) are used to divide the study area into origin zones and destination zones respectively. Both are designed by the ONS by aggregating postcode areas for spatial analysis of Census results. OAs are designed for residential statistics with each zone including between 40 and 250 households. WZs are designed for employment statistics and based on workplace counts [124]. In addition to the zone layout, the ONS also generates population-weighted centroids for every zone, which here are used as zone centroids.

The primary study area contains 248 Output Areas and 56 Workplace Zones, and the extended study area 2390 Output Areas and 647 Workplace Zones.

[20] Originally proposed in [73] as a mutation operation it was used in [60] as a repair operation.

[21] Travel-to-work areas are designed by the ONS as a collection of lower Census geographies in which "at least 75% of the area's resident workforce work in the area and at least 75% of the people who work in the area also live in the area" [125].

[22] The spatial layout of zones and centroids can be downloaded from: https://census.ukdataservice.ac.uk/.

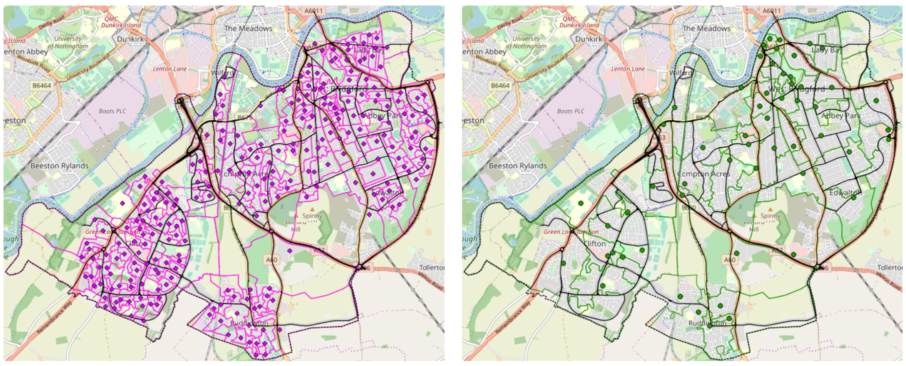


Fig. 4: Maps of the study area with boundaries and centroids of Output Areas (left) and Workplace Zones (right). Streets used as basis for the graph G are highlighted in black. (Sources: street data, zone layout, and centroid locations from UK Ordnance Survey. Underlying map from https://www.openstreetmap.org).

4.2 Node travel time matrix and terminal nodes

The primary study area is a subsection of the study area used in [60]. Therefore, the respective subset of the instance generated and published in [60] provides the nodes and links of the graph G as well as information on the terminal nodes U. In [60] the positions of the nodes were mainly determined by street junctions and the distances between them. Adjacency relations between the nodes and travel times along the links were determined via shortest path searches. Further, terminal nodes U were identified using data on existing bus services. For the study area, the graph G includes 60 nodes and 94 edges, with 28 nodes being classified as terminal nodes (see figure 5).

Also converted from the dataset generated in [60] is a set of routes representing existing bus services in the primary study area. In order to fit with the travel-to-work data used to generate the demand (see section 4.4.1) only services in operation during the morning rush hour (7:30 a.m. to 10:30 a.m.) are considered. This “real-world route set” includes 54 nodes in 18 routes, the shortest of which has three nodes and the longest 12 nodes. It will be used in section 5 for comparisons with the optimisation results.

Details of all procedures used to generate these data, as well as the underlying data sources can be found in [60].

4.3 Zone connectors and walking matrix

Connector matrices T^O and T^D, and the walking matrix T^W need to be generated based on walking accessibility of the zones and nodes. For this the 2011 version of the UK Ordnance Survey’s urban path layer[23] was used. It allows calculating the short-

[23] Researchers with UK institutional access can download Ordnance Survey’s datasets from http://digimap.edina.ac.uk/. Furthermore, the procedure can be used with data from other sources, e.g. OpenStreetMap (https://www.openstreetmap.org).

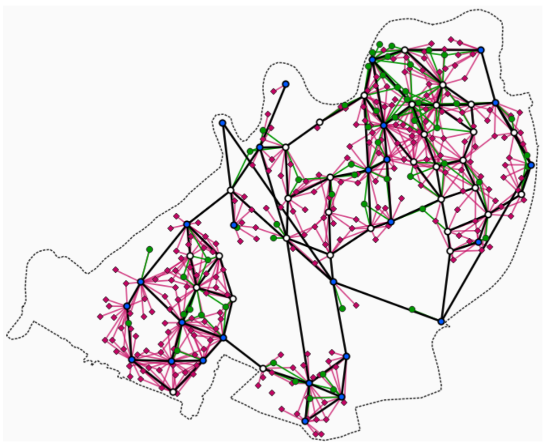


Fig. 5 Graph G generated by application of the generation procedure from [60] to the primary study area presented in figure 4 (terminal nodes marked in blue, regular nodes in white). Also displayed are the locations of origin centroids (purple squares), destination centroids (green circles) and the connectors. The direct walking connections between centroids are not displayed for clarity sake.

est path distance between zone centroids and nodes through the use of specialised geographic information systems[24].

The entries of matrices T^O,T^D and T^W are determined by taking the calculated shortest path distances and dividing them by a walking speed of 1.4m/s (as recommend in [61]). All entries which are above a cap distance d_c are set as ∞. For T^O and T^D the cap distance is set to $d_c = 758$m (approx. 9 minutes walking time). This is the largest distance between a zone and its nearest node in the study area. For T^W the cap distance is set as twice that of the connector matrices.

4.4 Travel demand

4.4.1 Data sources and classification

The demand data for this study are taken from the travel-to-work flow data[25] of the 2011 UK Census[26]. This dataset contains the number of commuters travelling from OAs to WZs. The considered trips can be grouped into four segments:

1. Both origin and destination inside of the primary study area.
2. Origin inside of the primary study area and destination inside of the extended study area.
3. Origin inside of the extended study area and destination inside of the primary study area.

[24] The present study used ArcGIS with Network Analyst. Equivalent calculations can also be executed in QGIS with the QNEAT3 plugin

[25] This dataset is used in this study because it is easy to access and has already been used in [60]. It represents only a subset of all trips; for example, trips for shopping or leisurely purposes are not included. However, it is sufficient for a proof-of-concept work such as this study. The comparisons to real-world routes in section 5 are limited to the morning rush hour, wherein travelling to work dominates the overall travelling pattern.

[26] The flow data can be downloaded from https://census.ukdataservice.ac.uk/. The same methodologies can be used in similar ways with data from other sources, such as data from other surveys, datasets generated via estimation models (see for example [128]), or mobile phone data (see for example [54]).

4. Both origin and destination inside of the extended study area.

Using OAs as origin zones and WZs as destination zones, the trips in segment 1 can be filled directly in the demand matrix D^Z. The process for trips in the other segments is discussed in the following.

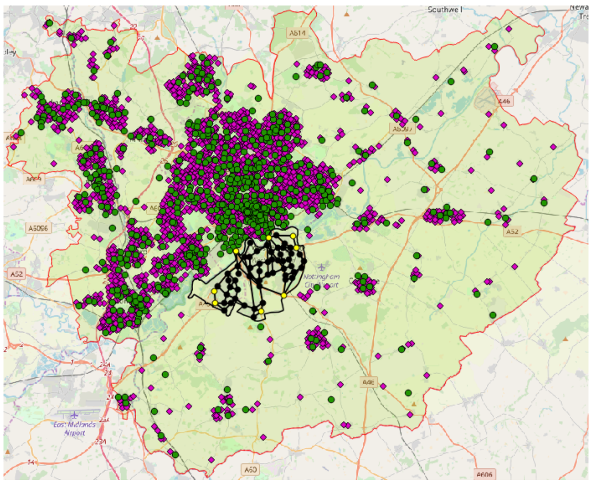


Fig. 6 Map of the extended study area with the locations of OA centroids (purple squares) and WZ centroids (green dots) (sources as in figure 4). The primary study area with the graph G is shown black, the eight gate nodes beeing marked in yellow.

4.4.2 *Cross-boundary flow*

The PuT services outside of the primary study area are not part of the optimisation process. Therefore, a trip between two zones, z_l^p inside of the primary study area and z_k^e inside of the extended study area, can be treated as a trip between z_l^p and the point at which this trip crosses the boundary. In theory, this point would need to be determined based on R, which would add another layer of complexity to the evaluation process[27]. However, if primary- and extended study area are sufficiently separated[28] and there are only a small number of distinct crossing points, as is the case here, the following simplification can be made: every zone z_k^e in the extended study area, independent of origin or destination zone, can be associated with a "gate node" $n_k^g \in N$, where all trips from/to z_k^e enter/leave the graph G and thereby the study area. This allows simplifying every trip between a zone z_l^p in the primary study area and zone z_k^e as a trip between z_l^p and n_k^g.

To use this concept in the generation of a zone-demand matrix, every gate node n_k^g generates a virtual origin zone z_{k*}^O and a virtual destination zone z_{k*}^D. Each virtual origin zone expands the matrix D^Z by one row, and each virtual destination zone by

[27] Theoretically, variations of equation 10 could be used to determine the gate nodes based on R. However, given the large number of zones in the extended study area, this would drastically increase the runtime of the optimisation.

[28] The extended area is sufficiently separated if there are no direct connections (shorter than d_c) between zones in the extended study area and nodes of G other than the zones gate nodes (described in the text).

one column. Concerning the connector times, virtual zones are instantly connected to their respective gate nodes ($t^{O}_{k*,k} = 0$ and $t^{D}_{k,k*} = 0$), while connectors to all other nodes do not exist. The direct walking times between regular zones and virtual zones are identical to the connector times from regular zones to the gate node.

The concept of virtual zones allows to inserting the demand of segments 2 and 3 as trips between zones inside of the primary study area and the virtual zones representing zones in the extended study area. Which node is the gate node for which zone is determined by a shortest path search on the Real-World Routes Graph (RWRG). The RWRG is a graph structure representing the public transport network in the extended study area. It is described in detail in appendix C.

Moreover, the RWRG can be used to filter out all trips from segment 4 which do not pass over the primary study area (also described in appendix C.). The remaining trips in segment 4 can be assigned to the demand matrix as follows: a trip from a zone z^{O}_{k}, with gate node n^{g}_{k}, to a zone z^{D}_{l}, with gate node n^{g}_{l}, is represented as a trip between the two virtual zones z^{O}_{k*} and z^{D}_{l*}.

For the presented study areas this process results in a demand matrix with of size 256×64 (248 origin zones, 56 destination zone, and 8 virtual zones each). In total, it has 5751 non-zero entries.

The gate-node approach can, of course, also be used to include cross-border demand in a node-based demand matrix. In this case, trips of segments 2 and 3 are considered to go between gate nodes and the nodes associated to the respective origins/destinations inside of the primary study area. Trips of segment 4 can be represented as trips between two gate nodes.

5 Experimental results

The following sections present the results of the optimisation procedure described in section 3 and applied to the instance generated in section 4. All experiments were conduced with a population size of $|P| = 50$ route sets. Each route set includes $|R| = 18$ routes, i.e the same number as real-world route set. The minimal and maximal numbers of nodes in a route is set as $l_{min} = 2$ and $l_{max} = 14$. The genetic algorithm runs for 200 generations.

5.1 The base optimisation

For the first experiment, the weighting factors in equation 1 are set as $q_1 = q_2 = q_3 = 1$. The results of this optimisation are shown in figure 7. Each of the displayed points gives the evaluation of one route set for total route length (C_O) and the average journey time (C^{Θ}_{P}). The evaluation results form a clear non-dominated front with several route sets surpassing the performance of the real-world route set[29](circle) in

[29] The real-world route set is evaluated with five origin connectors and one destination connector being added to C^{O} and C^{D} respectively. These connectors have a length between 772m and 1084m, longer than the otherwise used cap distance d_c. Their addition is necessary to connect all zones to at least one node included in the real-world route set. This gives a slight advantage to the real-world route set; however, it

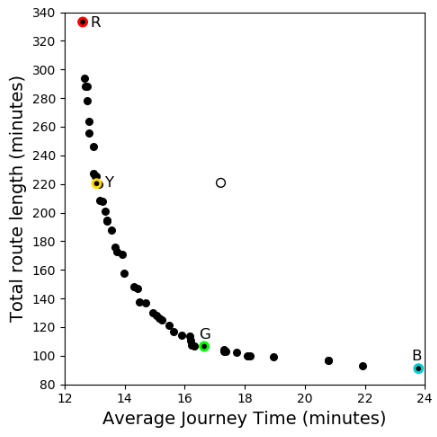


Fig. 7 Evaluation of the route sets resulting from zone-based optimisation with $q_1 = q_2 = q_3 = 1$ (dots) in comparison with the performance of the real-world route set (circle). Four results are highlighted: at the extremes, the one with the lowest average journey time (red, R) and the one with the lowest total route length (blue, B); from those route sets which surpass the real-world route set in both objectives, the one with the lowest average journey time (yellow, Y) and the one with the lowest total route length (green, G).

both objectives. Although such comparisons are limited by the assumptions made study set-up and instance generation, they indicate that the described optimisation procedure can generate route networks superior to those of pre-existing services.

To simplify the discussion, four critical positions are highlighted: at the extremes, the route sets with the lowest C_P^Θ in red and the one with the lowest C_O in blue; as for, the route sets which surpass the real-world route set in both objectives, the one with the lowest C_P^Θ in yellow, and the one with the lowest C_O in green.

The evaluation results of the route sets highlighted are shown in more detail in table 1 as the base case. In addition to the values displayed in figure 7, the table also presents the number of nodes included and the transfer statistic. The latter gives the percentage of travellers reaching their destination without transfers, with one, two or more transfers, or who do not use the service at all. The table shows, for example, that for the yellow route set, 74.1% of passengers undertake direct trips, 2.5 times that of the real-world route set. Furthermore, C_P^Θ reduced by 23.8%, while C_O is almost identical. The green route set has a 2.9% lower C_P^Θ than real-world route set, while C_O is reduced by 51.5%.

5.2 The impact of different weighting factors

The other results listed in table 1 are from optimisation experiments under variations q_1 and q_3 and selected after the same criteria as described above. The full results are shown in figure 8. Also displayed are the evaluation results of the real-world route set under the respective parameters, showing that, for all setups, the optimisations generated results which were superior .

The left-hand side of figure 8 shows the results of different weightings for the walking time. As expected, the fronts move farther to the right with higher values for q_1. By contrast, the evaluation results of optimisations with an increased transfer

is an improvement upon the separate, reduced instance which was necessary for the comparisons between real-world and optimised route sets in [60].

Route set	C_P^Θ	K	C_O	d_0	d_1	d_2	d_{3+}	d_W
Base case: $q_1 = 1$, $q_3 = 1$								
Real routes	17.2 min	54	221 min	29.2%	45.4%	18.7%	0.7%	6.0%
Red	12.6 min	57	333 min	82.3%	11.4%	0.1%	0.0%	6.1%
Yellow	13.1 min	58	220 min	74.1%	19.1%	0.5%	0.0%	6.3%
Green	16.7 min	53	107 min	35.5%	46.0%	10.6%	1.1%	6.8%
Blue	23.8 min	51	91 min	25.0%	21.0%	18.6%	28.2%	7.3%
Variation of walking weight: $q_1 = 1.5$								
Real routes	19.8 min	54	221 min	28.5%	45.7%	19.4%	0.9%	5.5%
Red	15.1 min	59	348 min	81.4%	13.2%	0.1%	0.0%	5.3%
Yellow	15.7 min	58	218 min	72.9%	20.9%	0.6%	0.0%	5.6%
Green	19.5 min	52	107 min	37.6%	36.9%	12.6%	6.3%	6.5%
Blue	26.6 min	52	90 min	17.2%	21.9%	26.9%	27.4%	6.6%
Variation of walking weight: $q_1 = 2$								
Real routes	22.4 min	54	221 min	28.2%	46.1%	19.7%	0.9%	5.3%
Red	17.5 min	59	352 min	77.5%	17.4%	0.2%	0.0%	4.9%
Yellow	18.1 min	59	220 min	69.7%	24.3%	1.0%	0.0%	5.0%
Green	21.5 min	53	120 min	38.7%	42.0%	12.4%	1.2%	5.6%
Blue	26.2 min	52	94 min	20.3%	25.3%	35.8%	12.3%	6.3%
Variation of walking weight: $q_1 = 2.5$								
Real routes	24.9 min	54	221 min	28.0%	46.5%	19.7%	0.9%	5.0%
Red	19.9 min	59	373 min	77.4%	17.3%	0.5%	0.0%	4.8%
Yellow	20.5 min	57	217 min	68.7%	25.0%	1.3%	0.0%	4.9%
Green	24.6 min	51	117 min	38.2%	33.5%	18.5%	3.7%	6.0%
Blue	44.3 min	51	88 min	12.5%	17.8%	8.9%	54.7%	6.0%
Variation of walking weight: $q_1 = 3$								
Real routes	27.4 min	54	221 min	27.7%	46.2%	20.3%	0.9%	4.9%
Red	22.4 min	60	335 min	73.1%	21.5%	0.6%	0.0%	4.8%
Yellow	23.0 min	58	217 min	67.2%	26.9%	1.0%	0.0%	4.9%
Green	27.4 min	52	106 min	33.6%	37.7%	18.8%	4.3%	5.7%
Blue	41.2 min	50	94 min	17.0%	24.9%	21.1%	30.7%	6.2%
Variation of transfer weight: $q_3 = 2$								
Real routes	21.2 min	54	221 min	33.8%	42.2%	17.8%	0.2%	6.0%
Red	13.0 min	59	341 min	86.6%	7.3%	0.0%	0.0%	6.1%
Yellow	13.9 min	56	211 min	79.7%	13.6%	0.2%	0.0%	6.6%
Green	20.4 min	49	106 min	44.0%	35.6%	8.8%	4.2%	7.5%
Blue	39.3 min	52	89 min	15.1%	19.5%	14.4%	43.8%	7.2%
Variation of transfer weight: $q_3 = 3$								
Real routes	25.1 min	54	221 min	34.7%	43.2%	16.2%	0.0%	6.0%
Red	13.3 min	59	372 min	88.7%	5.3%	0.0%	0.0%	6.0%
Yellow	15.0 min	57	219 min	79.5%	14.2%	0.2%	0.0%	6.2%
Green	23.7 min	50	115 min	38.5%	41.5%	11.8%	0.4%	7.7%
Blue	47.0 min	49	95 min	12.5%	21.3%	20.2%	38.2%	7.8%
Variation of transfer weight: $q_3 = 4$								
Real routes	28.8 min	54	221 min	34.8%	45.6%	13.5%	0.0%	6.0%
Red	13.4 min	56	359 min	89.2%	4.5%	0.0%	0.0%	6.3%
Yellow	14.6 min	52	210 min	82.9%	10.3%	0.1%	0.0%	6.7%
Green	25.8 min	49	102 min	47.8%	33.1%	10.7%	1.2%	7.2%
Blue	44.5 min	48	88 min	27.8%	15.3%	26.3%	23.0%	7.5%

Table 1: Evaluation results of selected route sets resulting from optimisation with zone-based demand and different weighting factors. Categories are as follows: C_P^Θ: average journey time; C_O: total route length; K: number of nodes included; d_0: % of direct trips; d_1: % of trips with one transfer; d_2: % of trips with two transfers; d_{3+}: % of trips with three or more transfers d_W: % of pure walking trips. Route sets are selected as highlighted in figure 7 (only the base case) and figure 8. Results for the real-world route set are evaluated with the respective parameter combination.

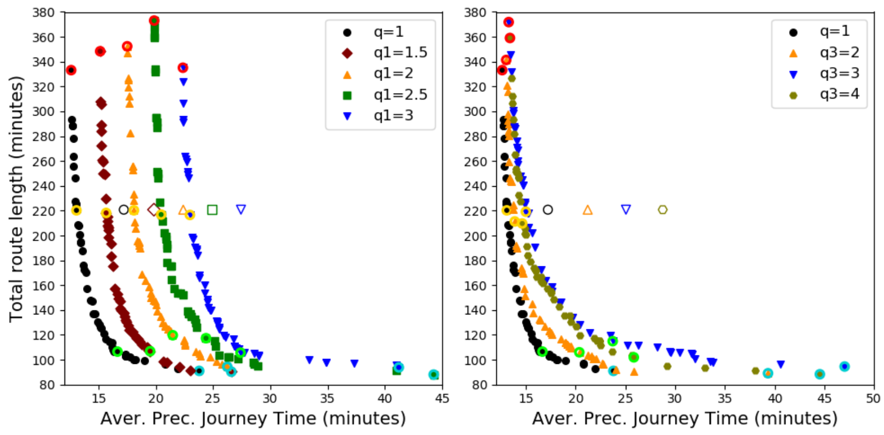


Fig. 8: Evaluations of the result route sets optimised for different weighting factors. The left side shows variations of q_1 (weight of walking time) the right side variations of q_3 (weight of transfer penalty) . Evaluations of the real-world route sets with the are presented as empty markers. Base case ($q_1 = q_2 = 1$) and highlighting of route sets is identical to in figure 7.

penalty on the right-hand side are closer together. This indicates that the optimisation algorithm effectively constructed more direct connections. However, the blue route sets show sharp increases in the average perceived journey time, as their low C_O comes at the cost of more transfers for passengers.

Figure 9 presents all transfer statistics of route sets resulting from the base case and the optimisation with $q_1 = 3$ and $q_3 = 4$. All these graphics show that the percentage of passengers undertaking direct trips increases with higher C_O. The percentages of trips with more transfers are consequently reduced, leading to the percentage of single-transfer trips peaking before then decreasing. This basic dynamic is the same for all setups. However, when q_1 is increased the percentages of trips with transfers decrease much more slowly, as passengers prefer less direct trips over longer walking times. By contrast, an increase of q_3 results in a general shift towards fewer transfers. Not only does the percentage of direct trips itself increase, the percentage of single-transfer trips peaks at a significantly lower level and decreases more quickly. Figure 9 further shows that the real-world route set offers significantly less direct travel than the optimised route sets with similar C_O under all configurations.

5.3 Comparison with node-based optimisation

The following section attempts to compare the results generated with the zone-based optimisation procedure presented in this study with those resulting from an equivalent node-based approach. For this, the procedure used in [60] is modified to include the generation of routes between non-terminal nodes, as described in section 3.3. The required node-based demand matrix is generated with the procedure described in

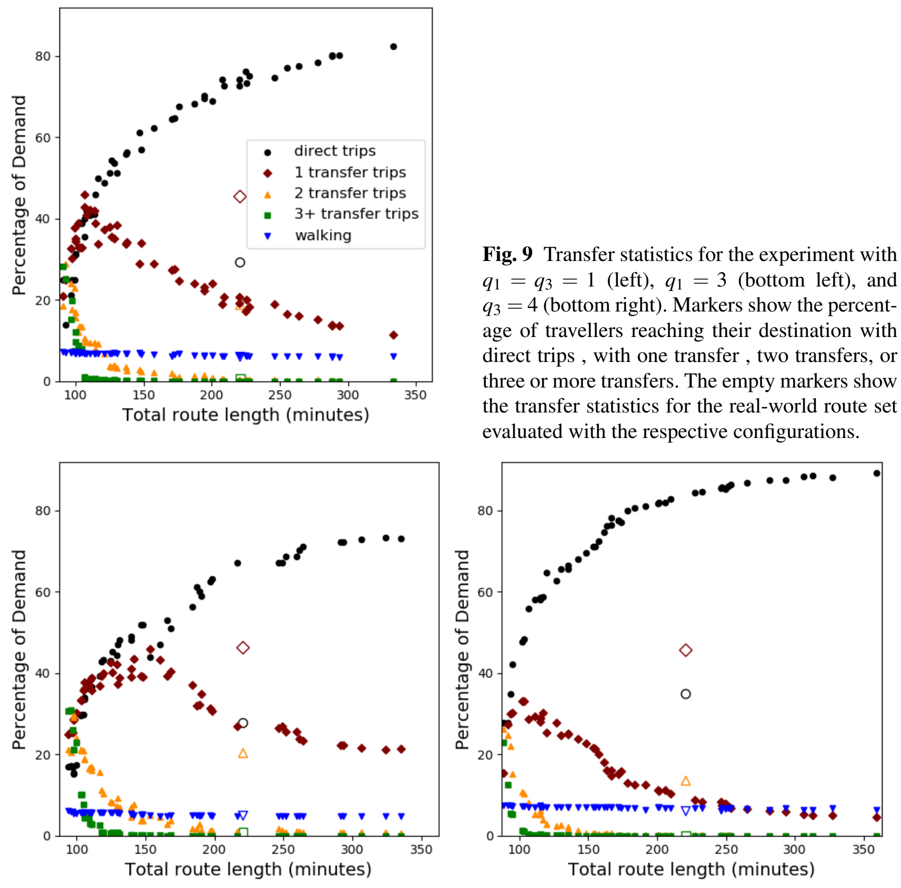


Fig. 9 Transfer statistics for the experiment with $q_1 = q_3 = 1$ (left), $q_1 = 3$ (bottom left), and $q_3 = 4$ (bottom right). Markers show the percentage of travellers reaching their destination with direct trips , with one transfer , two transfers, or three or more transfers. The empty markers show the transfer statistics for the real-world route set evaluated with the respective configurations.

[60], using the same data as used in section 4.4. Cross-border demand flow is included as described at the end of section 4.4.2.

The left-hand side of figure 10 shows the results of the zone-based optimisation from section 5.1 and of the node-based optimisation with the same parameters. Both are evaluated for their total route length C_O and average journey time C_P^Θ. This is possible because route sets resulting from node-based optimisation are required to include all nodes and, consequently, also all zones. The right-hand side of figure 10 shows the same results evaluated for average transit time. For each node-based result, two markers are displayed: one where the average transit time was calculated with zone-based demand[30], the other where node-based demand[31] was used.

[30] The calculation of transit times with zone-based demand uses equation 1 with $q_1 = 0$.

[31] This is the value used in the passenger objective of the node-based optimisation. For its calculation see footnote 10 on page 9.

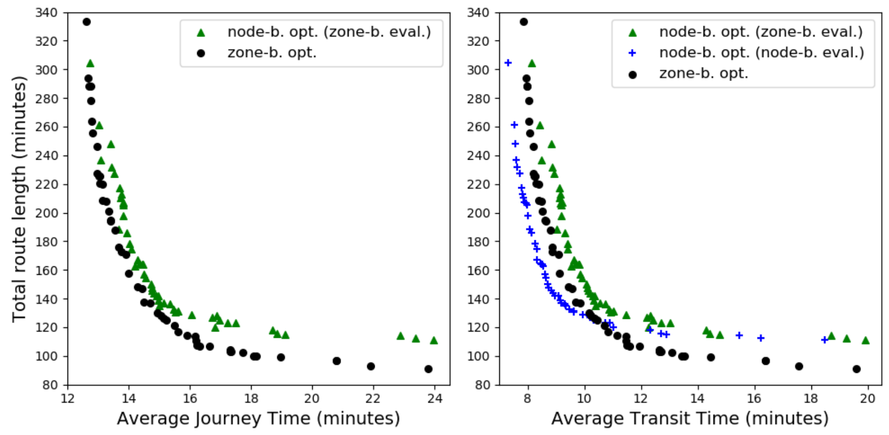


Fig. 10: Comparison of zone-based and node-based optimisation results for both the total route length vs. average journey time (left) and the total route length vs. average transit time (right). For the average transit time the route sets from node-based optimisation were evaluated two times: with the zone-based demand matrix (triangles) and with the zone-based demand matrix (pluses). Zone-based results (dots) are identical to those presented in figure 7.

The average transit times calculated with zone-based demand are, on average, 1.43 minutes (13.6%) longer than the one calculated with node-based demand[32]. These deviations are a result of the differences in aggregating trips between the two concepts (see figure 1 on page 4) and do not indicate the superiority of any concept. However, they highlight the importance of carefully choosing the used trip representation based on the available data.

The differences in the average transit time limit the conclusiveness of comparing both approaches for the given scenario. However, the results indicate that both approaches perform similarly well for small values of C_P^{Θ}, while for low C_O the results generated by zone-based optimisation are superior. This is expected as the constraint to always include all nodes hinders the node-based optimisation in reducing C_O.

6 Summary and conclusion

Different concepts for the representation of journeys and travel demand exist in the literature on automatic optimisation of public transport routes. The node-based concept, which considers only in-vehicle and transfer times, is used in the majority of studies because it is more straightforward to implement and has several input datasets that are publicly available. Zone-based concepts, which also take access times into ac-

[32] This shift also exists for route sets generated with zone-based optimisations which can be evaluated using node-based demand (e.g. the red route of the optimisation with $q_1 = 3$ in table 1).

count and are more often used in practical planning applications, however, feature in much fewer research studies.

This paper presented an adaptation of the methods used in [60], i.e. previous work by the author and others, from a node-based to zone-based approach. For this, it first introduced a hybrid procedure for the calculation of zone-based journey times. It first calculates the transit times between all node pairs and then identifies the connection offering the shortest overall zone-to-zone journey time for every zone pair.

This procedure can further be used to determine the “optimal node pair” for each zone pair. These form the beginning and end of the PuT journey with the shortest overall journey time between a specific pair of zones. These optimal node pairs form the basis of the majority of adaptations necessary to use the construction heuristic and genetic algorithm from [60] with zone-based demand. For example, they can be used to generate routes optimally connecting specific zone pairs. In cases where these nodes can not be route terminals, the route is extended until a possible terminal node is reached.

Further, this paper described procedures with which to generate the required input data based on freely available data sources. Included in this procedure is a method that considers cross-border demand flow in the generation of demand matrices. The procedure was applied to a subsection of the metropolitan area of Nottingham, UK.

Experimental results demonstrated the ability of the optimisation procedure to generate efficient route networks for different setups, as given by different weighting factors for walking and transfer times. Comparisons between optimisation results and representations pre-existing services are limited by the assumptions made; however, indicate that the presented optimisation procedure can generate superior route networks.

Independent of the results obtained, the methods presented bring about several advantages over the approach presented in [60]. These include the improved route generation and the ability to compare optimisation results and pre-existing routes without the need for reducing the instance. Further, the use of zone-based trip representation allows to more easily interface the presented optimisation procedure with macroscopic transport modelling software, which has the potential to drastically reduce barriers for practical application.

Further improvements are possible in several aspects. For example, it would be sensible to improve the calculation of the operator cost or to change the mutation operations to allow for a changing number of route sets. Moreover, the instance generation procedure can be further enhanced, potentially by measuring the connector length to existing stop points which can then be mapped to the graph nodes. However, additional research has to show whether such an approach is viable.

The instance dataset generated in this paper, the results presented, and a Python program for route set evaluation can be downloaded under
https://data.mendeley.com/datasets/jkz4bkb5j5 .

Acknowledgements

I am thankful for the technical support provided by Tönjes Heyken as well as for the helpful comments and suggestions with regard to various versions of this manuscript that I received from him as well as Dr Yong Mao, Prof. Terry Moore, Prof. Dario Landa-Silva (all three from the University of Nottingham), Dr. Christine Mumford, Dr. Rhyd Lewis (both from Cardiff University), and three anonymous reviewers. This work is part-funded by Leverhulme Programme Grant RP2013-SL-015.

A List of publications using node-based and zone-based demand

All publications listed in the following are dealing with the optimisation of PuT route networks (as defined in section 1.1 and 1.2) and use the passenger journey time as one of its evaluation criteria. This includes publications which extend the optimisation to other phases of PuT network design (e.g. frequency setting). In total 112 publication were found which meet these criteria.

91 of these studies use a node-based approach: [1, 2, 4, 3, 6, 7, 8, 9, 10, 11, 12, 14, 15, 17, 18, 19, 21, 22, 23, 24, 26, 25, 28, 27, 29, 30, 33, 37, 41, 42, 45, 44, 50, 52, 53, 55, 56, 57, 60, 62, 63, 65, 67, 68, 70, 71, 73, 72, 74, 76, 77, 78, 79, 80, 81, 82, 83, 85, 86, 87, 90, 91, 92, 93, 94, 97, 98, 99, 100, 101, 102, 103, 106, 108, 109, 114, 115, 116, 117, 118, 121, 122, 127, 129, 130, 131, 132, 133, 134, 135, 136]

The remaining studies can be separated into three groups based on the methods used to calculate the passenger journey times:

- 9 studies employ a professional transport modelling software (EMME [66] or PTV VISUM [107]): [5, 13, 34, 36, 35, 59, 104, 105, 112]
- 3 studies use specialised assignment algorithms: [16] (using [119]), [120] (using [95]), [20] (using [39])
- 9 studies use more regular shortest path algorithms (e.g Floyd's algorithm [51], or Dijkstra's Algorithm [40]): [31, 32, 43, 46, 47, 48, 49, 84, 111]

Further discussion on this literature review can be found in [58].

B Concerning run times

The computing time required for the evaluation is important as it needs to be executed many times during the optimisation process. In studies using node-based demand, the evaluation is typically dominated by the time $\tau(\Lambda)$, i.e. required for the generation of the transit time matrix Λ. The runtime of the hybrid-process described in section 2.2 requires additional time to calculate the journey time matrix Θ. It can therefore be described as $\tau(\Lambda+\Theta) = \tau(\Lambda) + \tau(\Theta)$.

Executed with Floyd's algorithm [51], the generation of Λ has a time complexity of $O(|\tilde{N}|^3)$. The time required to generate Θ depends largely on two factors. One is the number of non-zero demand pairs $|\Delta|$ (with $|\Delta| \leq |Z^O| \cdot |Z^D|$) which gives the number of times equation 5 needs to be executed. The other is the number of regular nodes $|N|$, which define the size of the matrices $\tilde{T}_a^O$, $\Lambda(R)$, and $\tilde{T}_b^O$ (see equation 5). As summing matrices has a (worst case) time complexity of $O(|N|^2)$[33], the time complexity of the complete process is

$$O\left(\,|\tilde{N}|^3 + |N|^2 \cdot |\Delta|\,\right) \tag{12}$$

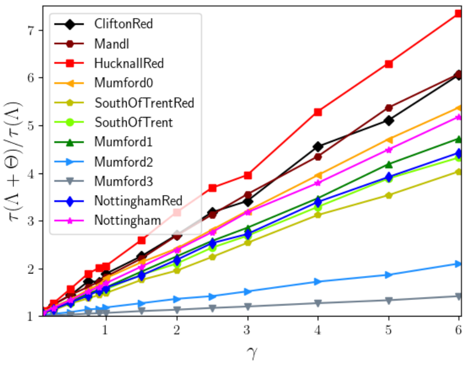
The ratio $\tau(\Lambda+\Theta)/\tau(\Lambda)$, therefore, depends on both the number of zone pairs and the relation between the number of regular nodes $|N|$ and the number of extended nodes $|\tilde{N}|$. The latter depends on the individual

[33] It should be noted that this time complexity gives the growth rate of the total number of operation. In practice, significantly lower increases in the observed run time can be achieved by utilising multi-core processors.

route set R. On one side, every node which is not included in R can be excluded from $\tilde{G}$ reducing $|\tilde{N}|$. On the other side, more transfer possibilities between routes increase $|\tilde{N}|$ (see section 2.2).

Figure 11 presents the runtime increase as ratio $\tau(\Lambda+\Theta)/\tau(\Lambda)$ for different values $\gamma = \frac{|\Delta|}{|N|^2}$, i.e. the number of non-zero demand pairs normalised by the number of node pairs. Every data point shows the average values for 50 calculations[34] of Λ and Θ.

As can be seen, the ratio increases linearly with the number of zone-pairs and the gradient of the increase depends largely on the relation between $|\tilde{N}|$ and $|N|$. The instance presented in section 4 is titled "SouthOfTrent" and has 5751 non-zero node pairs ($\gamma = 1.6$), resulting in $\tau(\Lambda+\Theta)/\tau(\Lambda) \approx 2$. This is consisted with the run times of the genetic algorithm optimisation[35].



Instance Graph	Ref.	$\|N\|$	$\|R\|$	$<\|\tilde{N}\|>$
CliftonRed	[4][36]	10	4	14.3
Mandl	[83]	15	5	23.6
HucknallRed	[4][36]	17	5	24.4
Mumford0	[91]	30	12	58.24
SouthOfTrentRed	[4][36]	54	18	144.66
SouthOfTrent	-	60	18	156.09
Mumford1	[91]	70	15	178.97
Mumford2	[91]	110	33	441.58
Mumford3	[91]	127	38	609.94
NottinghamRed	[60][36]	376	69	1818.28
Nottingham	[60][36]	428	69	2047.97

Fig. 11: Results of runtime experiments[34] with the graphs of different publicly available instances. Experiments were executed for different values of γ the ratio between zone- and node pairs. The y-axis gives the ratio between run times of the hybrid-process $\tau(\Lambda+\Theta)$ and only calculating transit times $\tau(\Lambda)$. The table on the right side shows source, number of regular nodes ($|N|$), size of the used route sets ($|R|$), and average number of extended nodes ($<|\tilde{N}|>$) for each instance.

For instances such as the one presented in section 4, run-times are in general sufficiently short[35]. For larger instances the procedure presented here can only be considered relatively efficient when $|\Delta| \gg |\tilde{N}|^2$. In other cases, it would be sensible to employ a different algorithm (e.g. Dijkstra's algorithm [40]) for the evaluation and use the hybrid procedure only for the process described in section 3.3. This would not impact the results of the optimisation.

It should further be noted that the individual executions of the equations 7 are independent from each other. This theoretically allows to significantly reduce the runtime of the hybrid-approach by parallelising the calculation of Θ.

[34] The route sets for these experiments were generated by the node-based initialisation procedure in [60], and therefore have all the same number of nodes to increase comparability. The matrices T^O,T^D and T^W required for the calculation of Θ were generated randomly for each data point. The run times were measured with the python module "timeit". These experiments took place on an Intel i5-6500 3.20GHz Quadcore CPU with 8GB RAM.

[35] For the experiments presented in section 5, calculating the objectives for a population of 50 route sets took on average 11.5s for the zone-based optimisation and 5.5s for node-based optimisation. The complete run with 200 generations required on average 42 minutes and 19.9 minutes respectively. The experiments were executed on an Intel i5-4300 2.60GHz CPU with 8GB RAM.

[36] These instances can be downloaded from https://data.mendeley.com/datasets/kbr5g3xmvk/1

C The real-world routes graph

The Real-World Routes Graph (RWRG) is a graph structure constructed with data on the existing PuT routes[37] in service in the extended study area during the morning rush hour. Nodes of the RWRG represent existing stop points, and two nodes are connected via a link if a direct connection between them exists within the selected PuT routes. Additional links are added between vertices which are closer than 100 metres together representing possible interchanges. The RWRG does not allow for an accurate calculation of journey times; however, it is sufficient for the tasks outlined in the following

The gate node n^g_k of zone z^e_k in the extended study is determined by calculating the shortest paths on the RWRG from z^e_k to all nodes $n_i \in N$. The n_i which is closest to selected as n^g_k.

The shortest path calculation can be carried out by building the RWRG as a shapefile through the use of Python library ArcPy or PyQGIS and converting it into a network dataset for use with the Network Analyst toolbox[38]. The Network Analyst function "Closest Facility" is then used to determine which node ni of the graph G is the closest to zone zi in the extended study area through the use of the RWRG.

The RWRG can also be used to filter the trips in the fourth demand segment (trips between outer zones) into those that go over the study area and those that do not. To do so, Network Analyst function "Closest Facility" is used to find the shortest paths from all origin zones outside of the study area to all destination zones outside of the study area. As a second step, ArcGIS Linear referencing tool "Locate Features along routes" is used to determine which of these shortest paths lead over nodes of the graph G. All origin–destination pairs in which this is not the case will be deleted and not considered further.

References

1. S. Afandizadeh, H. Khaksar, and N. Kalantari. Bus fleet optimization using genetic algorithm a case study of mashhad. *International Journal of Civil Engineering*, 11(1):43–52, 2013.
2. J. Agrawal and T. V. Mathew. Transit Route Network Design Using Parallel Genetic Algorithm. *Journal of Computing in Civil Engineering*, 18(3):248–256, 2004.
3. L. Ahmed, P. Heyken Soares, C. Mumford, and Y. Mao. Optimising bus routes with fixed terminal nodes: comparing hyper-heuristics with nsgaii on realistic transportation networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1102–1110. ACM, 2019.
4. L. Ahmed, C. L. Mumford, and A. Kheiri. Solving Urban Transit Route Design Problem using Selection Hyper-heuristics. *European Journal of Operational Research*, 274(2):545–559, 2019.
5. B. Alt and U. Weidmann. A stochastic multiple area approach for public transport network design. *Public Transport*, 3(1):65–87, 2011.
6. S. M. M. Amiripour, A. A. Ceder, and A. S. Mohaymany. Designing large-scale bus network with seasonal variations of demand. *Transportation Research Part C: Emerging Technologies*, 48:322–338, 2014.
7. S. M. M. Amiripour, A. A. Ceder, and A. S. Mohaymany. Hybrid Method for Bus Network Design with High Seasonal Demand Variation. *Journal of Transport Engineering*, 140(6):1–11, 2014.
8. S. M. M. Amiripour, A. S. Mohaymany, and A. A. Ceder. Optimal Modification of Urban Bus Network Routes Using a Genetic Algorithm. *Journal of Transportation Engineering*, 141(3):1–9, 2014.
9. R. O. Arbex and C. B. da Cunha. Efficient transit network design and frequencies setting multi-objective optimization by alternating objective genetic algorithm. *Transportation Research Part B: Methodological*, 81:355–376, 2015.
10. M. H. Baaj and H. S. Mahmassani. An AI-Based Approach for Tansit Route System Planning and Design. *Journal of Advanced Transportation*, 25:187–209, 1991.

[37] In the UK, information on PuT routes can be extracted from the National Public Transport Data Repository (NPTDR) downloadable from https://data.gov.uk/dataset/nptdr. Outside the UK similar datasets should be available from national transport authorities, local authorities or public transport operators. For operators who use General Transit Feed Specification (GTFS) these datasets can be downloaded from https://transitfeeds.com/.

[38] There are potentially also ways to extract the same information using QGIS, however, these have not been explored for this work.

11. Baaj, M. Hadi and Mahmassani, Hani S. Hybrid route generation heuristic algorithm for the design of transit networks. *Transportation Research C*, 3(1):31–50, 1995.
12. B. Bachelet and L. Yon. Enhancing theoretical optimization solutions by coupling with simulation. In *First Open International Conference on Modeling and Simulation (OICMS)*, 2005.
13. S. A. Bagloee and A. A. Ceder. Transit-network design methodology for actual-size road networks. *Transportation Research Part B: Methodological*, 45(10):1787–1804, 2011.
14. A. Barra, L. Carvalho, N. Teypaz, V.-D. Cung, and R. Balassiano. Solving the transit network design problem with constraint programming. In *11th World Conference in Transport Research-WCTR 2007*, 2007.
15. B. Beltran, S. Carrese, E. Cipriani, and M. Petrelli. Transit network design with allocation of green vehicles: A genetic algorithm approach. *Transportation Research Part C: Emerging Technologies*, 17(5):475–483, 2009.
16. M. Bielli, M. Caramia, and P. Carotenuto. Genetic algorithms in bus network optimization. *Transportation Research Part C: Emerging Technologies*, 10(1):19–34, 2002.
17. M. Bielli and P. Carotenuto. A new approach for transport network design and optimization. In *38th Congress of the European Regional Science Association*, 1998.
18. J. J. Blum and T. V. Mathew. Intelligent Agent Optimization of Urban Bus Transit System Design. *Journal of Computing in Civil Engineering*, 25(5):357–369, 2010.
19. R. Borndörfer, M. Grötschel, and M. E. Pfetsch. Models for line planning in public transport. In *Computer-aided systems in public transport*, pages 363–378. Springer, 2008.
20. P.-L. Bourbonnais, C. Morency, M. Trépanier, and É. Martel-Poliquin. Transit network design using a genetic algorithm with integrated road network and disaggregated o–d demand data. *Transportation*, pages 1–36, 2019.
21. A. T. Buba and L. S. Lee. Differential evolution for urban transit routing problem. *Journal of Computer and Communications*, 4(14):11, 2016.
22. A. T. Buba and L. S. Lee. A differential evolution for simultaneous transit network design and frequency setting problem. *Expert Systems with Applications*, 106:277–289, 2018.
23. H. Cancela, A. Mauttone, and M. E. Urquhart. Mathematical programming formulations for transit network design. *Transportation Research Part B*, 77:17–37, 2015.
24. S. Carrese and S. Gori. An urban bus network design procedure. In *Transportation planning*, pages 177–195. Springer, 2002.
25. A. Ceder and Y. Israeli. User and operator perspectives in transit network design. *Transportation Research Record*, 1623(1):3–7, 1998.
26. A. Ceder and N. H. M. Wilson. Bus network design. *Transportation Research B*, 20B(4):331–344, 1986.
27. P. Chakroborty. Genetic Algorithms for Optimal Urban Transit Network Design. *Computer-Aided Civil and Infrastructure Engineering*, 18(3):184–200, 2003.
28. P. Chakroborty and T. Wivedi. Optimal route network design for transit systems using genetic algorithms. *Engineering optimization*, 34(1):83–100, 2002.
29. J. S. C. Chew and L. S. Lee. A genetic algorithm for urban transit routing problem. In *International Journal of Modern Physics: Conference Series*, volume 9, pages 411–421. World Scientific, 2012.
30. J. S. C. Chew, L. S. Lee, and H. V. Seow. Genetic Algorithm for Biobjective Urban Transit Routing Problem. *Journal of Applied Mathematics*, 2013, 2013.
31. S. Chien and P. Schonfeld. Optimization of grid transit system in heterogeneous urban environment. *Journal of Transportation Engineering*, 123(1):28–35, 1997.
32. S. I.-J. Chien and L. N. Spasovic. Optimization of grid bus transit systems with elastic demand. *Journal of advanced transportation*, 36(1):63–91, 2002.
33. J. C. Chu. Mixed-integer programming model and branch-and-price-and-cut algorithm for urban bus network design and timetabling. *Transportation Research Part B: Methodological*, 108:188–216, 2018.
34. E. Cipriani, G. Fusco, S. Gori, and M. Petrelli. A procedure for the solution of the urban bus network design problem with elastic demand. *Advanced OR and AI Methods in Transportation*, pages 681–685, 2005.
35. E. Cipriani, S. Gori, and M. Petrelli. Transit network design: A procedure and an application to a large urban area. *Transportation Research Part C: Emerging Technologies*, 20(1):3–14, 2012.
36. E. Cipriani, M. Petrelli, and G. Fusco. A multimodal transit network design procedure for urban areas. *Advances in Transportation Studies*, 10:5–20, 2006.

37. I. M. Cooper, M. P. John, R. Lewis, C. L. Mumford, and A. Olden. Optimising large scale public transport network design problems using mixed-mode parallel multi-objective evolutionary algorithms. *Evolutionary Computation (CEC), 2014 IEEE Congress*, pages 2841 – 2848, 2014.
38. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
39. J. Dibbelt, T. Pajor, B. Strasser, and D. Wagner. Intriguingly simple and fast transit routing. In *International Symposium on Experimental Algorithms*, pages 43–54. Springer, 2013.
40. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.
41. D. Dubois, G. Bel, and M. Llibre. A set of methods in transportation network synthesis and analysis. *Journal of the Operational Research Society*, 30(9):797–808, 1979.
42. J. Duran, L. Pradenas, and V. Parada. Transit network design with pollution minimization. *Public Transport*, 11(1):189–210, 2019.
43. J. Enrique Fernández L, J. de Cea Ch, R. H. Malbran, et al. Demand responsive urban public transport system design: Methodology and application. *Transportation Research Part A: Policy and Practice*, 42(7):951–972, 2008.
44. L. Fan and C. L. Mumford. A metaheuristic approach to the urban transit routing problem. *Journal of Heuristics*, 16(3):353–372, 2010.
45. L. Fan, C. L. Mumford, and D. Evans. A simple multi-objective optimization algorithm for the urban transit routing problem. In *2009 IEEE Congress on Evolutionary Computation*, pages 1–7, May 2009.
46. W. Fan and R. B. Machemehl. Optimal transit route network design problem with variable transit demand: genetic algorithm approach. *Journal of transportation engineering*, 132(1):40–51, 2006.
47. W. Fan and R. B. Machemehl. Using a Simulated Annealing Algorithm to Solve the Transit Route Network Design Problem. *Journal of transportation engineering*, 132(2):122–132, 2006.
48. W. Fan and R. B. Machemehl. Tabu Search Strategies for the Public Transportation Network Optimizations with Variable Transit Demand. *Computer-Aided Civil and Infrastructure Engineering*, 23:502–520, 2008.
49. W. D. Fan and R. B. Machemehl. Bi-Level Optimization Model for Public Transportation Network Redesign Problem Accounting for Equity Issues. *Transportation Research Record*, 2263(1):151–162, 2011.
50. X. Feng, X. Zhu, X. Qian, Y. Jie, F. Ma, and X. Niu. A new transit network design study in consideration of transfer time composition. *Transportation Research Part D: Transport and Environment*, 66:85–94, 2019.
51. R. W. Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.
52. G. Fusco, S. Gori, and M. Petrelli. A heuristic transit network design algorithm for medium size towns. In *Proceedings of the 13th mini-euro conference*, 2002.
53. Z. Gao, H. Sun, and L. L. Shan. A continuous equilibrium network design model and algorithm for transit systems. *Transportation Research Part B: Methodological*, 38(3):235–250, 2004.
54. Golding, James. Best Practices and Methodology for OD-Matrix Creation from CDR-data. Technical report, University of Nottingham, Business School, N-LAB, 2018.
55. J. F. Guan, H. Yang, and S. C. Wirasinghe. Simultaneous optimization of transit line configuration and passenger line assignment. *Transportation Research Part B: Methodological*, 40(10):885–902, 2006.
56. G. Gutierrez-Jarpa, G. Laporte, V. Marianov, and L. Moccia. Multi-objective rapid transit network design with modal competition: The case of Concepción, Chile. *Computers and Operations Research*, 78:27–43, 2017.
57. G. Gutiérrez-Jarpa, C. Obreque, G. Laporte, and V. Marianov. Rapid transit network design for optimal cost and origin–destination demand capture. *Computers & Operations Research*, 40(12):3000–3009, 2013.
58. P. Heyken Soares. *Three Steps Towards Practical Application of Public Transport Route Optimisation in Urban Areas*. PhD thesis, University of Nottingham, Nottingham, UK, 2020.
59. P. Heyken Soares, L. Ahmed, C. L. Mumford, , and Y. Mao. Public Transport Network Optimisation in PTV Visum using Selection Hyper-Heuristics. In Review, 2019.
60. P. Heyken Soares, C. L. Mumford, K. Amponsah, and Y. Mao. An Adaptive Scaled Network for Public Transport Route Optimisation. *Public Transport*, 11(2):379–412, 2019.
61. Highways England. Design Manual for Roads and Bridges. Technical report, Highways England, 2019. accessed at 09.08.2019.

62. J. Hu, X. Shi, J. Song, and Y. Xu. Optimal Design for Urban Mass Transit Network. In *International Conference on Natural Computation*, pages 1089–1100, 2005.
63. D. Huang, Z. Liu, X. Fu, and P. T. Blythe. Multimodal Transit Network Design in a Hub-and- Spoke Network Framework. *Transportmetrica A: Transport Science*, 0(0):1–42, 2018.
64. C. Iliopoulou, K. Kepaptsoglou, and E. Vlahogianni. Metaheuristics for the transit route network design problem: a review and comparative analysis. *Public Transport*, 11(3):487–521, 2019.
65. C. Iliopoulou and I. Tassopoulos. Electric Transit Route Network Design Problem: Model and Application. *Transportation Research Record*, 2673(8):264–274, 2019.
66. INRO, Montreal, Canada. *Emme 4 User Manual*, 2018.
67. K. A. Islam, I. M. Moosa, J. Mobin, M. A. Nayeem, and M. S. Rahman. A heuristic aided Stochastic Beam Search algorithm for solving the transit network design problem. *Swarm and Evolutionary Computation*, 46(March 2018):154–170, 2019.
68. Y. Israeli and A. Ceder. Transit route design using scheduling and multiobjective programming techniques. In *Computer-aided transit scheduling*, pages 56–75. Springer, 1995.
69. S. R. Jara-Diaz and A. Gschwender. Towards a general microeconomic model for the operation of public transport. *Transport Reviews*, 23(4):453–469, 2003.
70. S. B. Jha, J. K. Jha, and M. K. Tiwari. Computers & Industrial Engineering A multi-objective metaheuristic approach for transit network design and frequency setting problem in a bus transit system. *Computers & Industrial Engineering*, 130(November 2018):166–186, 2019.
71. Y. Jiang, W. Y. Szeto, and T. M. Ng. Transit Network Design: a Hybrid Enhanced Artificial Bee Colony Approach and a Case Study. *International Journal of Transportation Science and Technology*, 2(3):243–260, 2013.
72. M. P. John. *Metaheuristics for Designing Efficient Routes & Schedules For Urban Transportation Networks*. PhD thesis, University of Cardiff, 2016.
73. M. P. John, C. L. Mumford, and R. Lewis. An improved multi-objective algorithm for the urban transit routing problem. In C. Blum and G. Ochoa, editors, *Evolutionary Computation in Combinatorial Optimisation*, pages 49–60. Springer Berlin Heidelberg, 2014.
74. P. N. Kechagiopoulos and G. N. Beligiannis. Solving the urban transit routing problem using a particle swarm optimization based algorithm. *Applied Soft Computing*, 21:654–676, 2014.
75. K. Kepaptsoglou and M. Karlaftis. Transit route network design problem. *Journal of transportation engineering*, 135(8):491–505, 2009.
76. F. Kiliç and M. Gök. A demand based route generation algorithm for public transit network design. *Computers & Operations Research*, 51:21 – 29, 2014.
77. M. Kim, S.-Y. Kho, and D.-K. Kim. A transit route network design problem considering equity. *Sustainability*, 11(13):3527, 2019.
78. W. Lampkin and P. Saalmans. The design of routes, service frequencies, and schedules for a municipal bus undertaking: A case study. *Journal of the Operational Research Society*, 18(4):375–397, 1967.
79. Y.-J. Lee and V. R. Vuchic. Transit network design with variable demand. *Journal of Transportation Engineering*, 131(1):1–10, 2005.
80. Y. Liu, N. Zhu, and S.-f. Ma. Simultaneous optimization of transit network and public bicycle station network. *Journal of Central South University*, 22(4):1574–1584, 2015.
81. F. López-Ramos, E. Codina, Á. Marín, and A. Guarnaschelli. Integrated approach to network design and frequency setting problem in railway rapid transit systems. *Computers & Operations Research*, 80:128–146, 2017.
82. S. H. Mahdavi Moghaddam, K. R. Rao, G. Tiwari, and P. Biyani. Simultaneous bus transit route network and frequency setting search algorithm. *Journal of Transportation Engineering, Part A: Systems*, 145(4):04019011, 2019.
83. C. E. Mandl. Applied network optimization. *Academic Press*, 1979.
84. Á. G. Marín and P. Jaramillo. Urban rapid transit network design: accelerated benders decomposition. *Annals of Operations Research*, 169(1):35–53, 2009.
85. B. Marwah, F. S. Umrigar, and S. Patnaik. Optimal design of bus routes and frequencies for ahmedabad. *Transportation Research Record*, 994:41–47, 1984.
86. A. Mauttone and M. E. Urquhart. A route set construction algorithm for the transit network design problem. *Computers and Operations Research*, 36(8):2440–2449, 2009.
87. A. Mauttone and M. E. Urquhart. A multi-objective metaheuristic approach for the transit network design problem. *Public Transport*, 1(4):253–273, 2009.
88. M. G. McNally. The four step model. *Handbook of transport modelling*, 1:35–41, 2000.

89. L. Moccia, D. W. Allen, and E. C. Bruun. A technology selection and design model of a semi-rapid transit line. *Public Transport*, 10(3):455–497, 2018.
90. K.-H. Müller. *Ein mathematisches Modell für die Bestimmung von Endknotenzuordnungen in Nahverkehrsnetzen*. PhD thesis, Bergakademie Freiberg, 1967.
91. C. L. Mumford. New heuristic and evolutionary operators for the multi-objective urban transit routing problem. *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, pages 939–946, 2013.
92. M. A. Nayeem, M. K. Rahman, and M. S. Rahman. Transit network design by genetic algorithm with elitism. *Transportation Research Part C: Emerging Technologies*, 46:30–45, 2014.
93. H. Nebelung. *Rationelle Umgestaltung von Straßenbahnnetzen in Großstädten*. Ministerium f. Wirtschaft, Mittelstand u. Verkehr Nordrhein-Westfalen, 1961.
94. S. Ngamchai and D. J. Lovell. Optimal Time Transfer in Bus Transit Route Network Design Using a Genetic Algorithm. *Journal of transportation engineering*, 129(5):510–521, 2003.
95. S. Nguyen and S. Pallottino. Equilibrium traffic assignment for large scale transit networks. *European journal of operational research*, 37(2):176–186, 1988.
96. G. Nielsen, J. Nelson, C. Mulley, G. Tegner, G. Lind, and T. Lange. HiTrans Best Practice Guide 2: Public transport - Planning the networks. *HiTrans*, 2005.
97. M. Nikolic and D. Teodorovic. Transit network design by Bee Colony Optimization. *Expert Systems with Applications*, 40(15):5945–5955, 2013.
98. M. Nikolic and D. Teodorovic. A simultaneous transit network design and frequency setting: Computing with bees. *Expert Systems with Applications*, 41(16):1–10, 2014.
99. M. Owais, G. Moussa, Y. Abbas, and M. El-Shabrawy. Simple and Effective Solution Methodology for Transit Network Design Problem. *International Journal of Computer Applications*, 89(14):32–40, 2014.
100. M. Owais and M. K. Osman. Complete hierarchical multi-objective genetic algorithm for transit network design problem. *Expert Systems With Applications*, 114:143–154, 2018.
101. M. Owais, M. K. Osman, and G. Moussa. Multi-objective transit route network design as set covering problem. *IEEE Transactions on Intelligent Transportation Systems*, 17(3):670–679, 2016.
102. J. Pacheco, A. Alvarez, S. Casado, and J. L. González-velarde. A tabu search approach to an urban transport problem in northern Spain. *Computers & Operations Research*, 36:967–979, 2009.
103. S. Pattnaik, S. Mohan, and V. Tom. Urban Bus Transit Route Network Design Using Genetic Algorithm. *Journal of Transportation Engineering*, 124(4):368–375, 1998.
104. M. Petrelli. A transit network design model for urban areas. *WIT Transactions on The Built Environment*, 75:163–172, 2004.
105. H. Poorzahedy and F. Safari. An Ant System application to the Bus Network Design Problem: An algorithm and a case study. *Public Transport*, 3(2):165–187, 2011.
106. M. Pternea, K. Kepaptsoglou, and M. G. Karlaftis. Sustainable urban transit network design. *Transportation Research Part A: Policy and Practice*, 77:276–291, 2015.
107. PTV AG, Karlsruhe, Germany. *PTV Visum 17 User Manual*, 2018.
108. C. Quak. Bus line planning. Master's thesis, TU Delft, 2003.
109. M. K. Rahman, M. A. Nayeem, and M. S. Rahman. Transit network design by hybrid guided genetic algorithm with elitism. In *Proceedings of the 2015 conference on advanced systems for public transport (CASPT)*, 2015.
110. J. Rich. Transport models - from theory to practise. Department of Transport, Technical University of Denmark, Lyngby, Denmark, 2015.
111. M. Roca-riu, M. Estrada, and C. Trapote. The design of interurban bus networks in city centers. *Transportation Research Part A*, 46(8):1153–1165, 2012.
112. H. Sadrsadat, H. Poorzahedi, A. Haghani, and E. Sharifi. Bus network design using genetic algorithm. Technical report, University of Maryland, Department of Civil and Environmental Engineering, 2012.
113. J. Schlaich, U. Heidl, and P. Möhl. Multimodal macroscopic transport modelling: State of the art with a focus on validation & approval. In *Proceedings of the 17th IRF World Meeting & Exhibition, Riyadh, Saudi-Arabia*, 2013.
114. M.-C. Shih, H. S. Mahmassani, and M. H. Baaj. Planning and design model for transit route networks with coordinated operations. *Transportation Research Record*, 1623(1):16–23, 1998.
115. H. Shimamoto, J.-D. Schmöcker, and F. Kurauchi. Optimisation of a bus network configuration and frequency considering the common lines problem. *Journal of Transportation Technologies*, 2(03):220, 2012.

116. L. A. Silman, Z. Barzily, and U. Passy. Planning the route system for urban busses. *Computers & Operations Research*, 1(2):201–211, 1974.
117. S. Soehodo and M. Koshi. Design of public transit network in urban area with elastic demand. *Journal of advanced transportation*, 33(3):335–369, 1999.
118. H. Sonntag. Ein heuristisches verfahren zum entwurf nachfrageorientierter linienführung im öffentlichen personennahverkehr. *Zeitschrift für Operations Research*, 23(2):B15–B31, 1979.
119. H. Spiess and M. Florian. Optimal Strategies: a new assignment model for transit networks. *Transportation Research Part B*, 23(2), 1989.
120. W. Y. Szeto and Y. Jiang. Hybrid artificial bee colony algorithm for transit network design. *Transportation Research Record*, 2284(1):47–56, 2012.
121. W. Y. Szeto and Y. Wu. A simultaneous bus route design and frequency setting problem for tin shui wai, hong kong. *European Journal of Operational Research*, 209(2):141–155, 2011.
122. V. M. Tom and S. Mohan. Transit Route Network Design Using Frequency Coded Genetic Algorithm. *Journal of transportation engineering*, 129(2):186–195, 2003.
123. Turfitt, R. Statutory Document No. 14 Local bus services in England (outside London) and Wales. Technical report, Senior Traffic Commissioner (UK), 2018.
124. UK Office for National Statistics. Census geography. http://www.ons.gov.uk/ons/guide-method/geography/beginner-s-guide/census/index.html, 2016.
125. UK Office for National Statistics. Travel to work area analysis in great britain: 2016. https://www.ons.gov.uk/employmentandlabourmarket/peopleinwork/employmentandemployeetypes/articles/traveltoworkareaanalysisingreatbritain/2016, 2016. (22/9/2019).
126. S. Walter. Nachfrageorientierte liniennetzoptimierung am beispiel graz (demand orientated line optimisation at the example of graz). Master's thesis, Graz University of Technologie, 2010.
127. J. L. Walteros, A. L. Medaglia, and G. Riaño. Hybrid Algorithm for Route Design on Bus Rapid Transit Systems. *Trasportation Science*, 49(1):1–19, 2013.
128. A. G. Wilson. The Use of Entropy Maximising Models, in the Theory of Trip Distribution, Mode Split and Route Split. *Journal of Transport Economics and Policy*, 3(1):108–126, 1969.
129. R. Wu and S. Wang. Discrete wolf pack search algorithm based transit network design. In *7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 509–512. IEEE, 2016.
130. Y. Xiong and J. B. Schneider. Transportation network design using a cumulative genetic algorithm and neural network. *Transportation Research Record*, 1992.
131. G.-m. Xu, F. Shi, and P. Wang. Model and algorithm of optimizing bus transit network based on line segment combination. In *CICTP 2014: Safe, Smart, and Sustainable Multimodal Transportation Systems*, pages 1514–1525. ASCE, 2014.
132. F. Zhao. Large-scale transit network optimization by minimizing user cost and transfers. *Journal of Public Transportation*, 9(2):6, 2006.
133. F. Zhao and X. Zeng. Optimization of transit network layout and headway with a combined genetic algorithm and simulated annealing method. *Engineering Optimization*, 38(6):701–722, 2006.
134. F. Zhao and X. Zeng. Optimization of user and operator cost for large-scale transit network. *Journal of Transportation Engineering*, 133(4):240–251, 2007.
135. F. Zhao and X. Zeng. Optimization of transit route network, vehicle headways and timetables for large-scale transit networks. *European Journal of Operational Research*, 186(2):841–855, 2008.
136. H. Zhao, W. Ato, and R. Jiang. Expert Systems with Applications The Memetic algorithm for the optimization of urban transit network. *Expert Systems with Applications*, 42(7):3760–3773, 2015.

6 Public Transport Network Optimisation in PTV Visum using Selection Hyper-Heuristics

List of Authors

Philipp Heyken Soares, Leena Ahmed, Yong Mao, Christine L. Mumford

Status by thesis submission

Resubmitted to Springer Nature journal "Public Transport" on the 19th of April 2020 following corrections requested by the reviewers. As the paper is still under review by the time of thesis submission, its final version might deviate from the one presented here.

Authors' contributions

The principle idea for this paper, interfacing between of a UTRP algorithm and the macroscopic transport modelling software PTV Visum, was conceptualised by Philipp Heyken Soares.

The development and programming for Visum-SSHH interaction (Section 5.2) and the required interface procedures (Section 4), was carried out by Philipp Heyken

Soares. For the hyper-heuristics part, Leena Ahmed programmed the core of the algorithm of SSHH and SR (Section 5.1) based on her work in [166], the programming of the low-level heuristics was also carried out by Leena Ahmed based on her work [166], and later modified by Philipp Heyken Soares who adjusted the implementation to adjacency relations and added additional low-level heuristics (Section 5.3). The evaluation procedures (Section 5.4) were programmed by Philipp Heyken Soares. He further, prepared the network models for the computer experiments (Appendix A), and analysed results of the city-size experiments (Sections 6.2 and 6.2). The results for the small-scale experiments were performed and analysed by Leena Ahmed (Section 6.1). Christine Mumford and Yong Mao provided guidance throughout the project.

The first drafts of the Sections 1, 2.2, 2.3, 3, 4, 5.4, 5.5, 6.1.1, 6.2, 6.3 and 7, as well as Appendix A were written by Philipp Heyken Soares. The first drafts of the Sections 5.1, 5.3 and 6.1.2 were written by Leena Ahmed, and for those of the Sections 2.1 and 5.2 Leena Ahmed and Philipp Heyken Soares have worked collaboratively. After the initial draft, all authors contributed in proofreading and editing of all parts of the manuscript. Most figures were generated by Philipp Heyken Soares. The exceptions are the Figures 3 and 5, which were generated by Leena Ahmed.

Summary of content

This paper introduces an interface between an UTRP algorithm and a macroscopic transport modelling software. When fully developed, such an interface can drastically reduce the barriers for UTRP algorithms to be used in practical planning processes as no new data sets would need to be generated. The transport modelling software used in this project is PTV Visum, which is used by professional planning agencies around the world. It was selected for this project, as it has an easy-to-use python interface and as obtaining an academic license for it was relatively straightforward.

The main difficulty for the interface is to translate adjacency and route information between Visum network models, which are likely to include directed connections between stop points, and the standard UTRP graph model, which is based on the undirected connections between nodes (Section 3). The alternative, the use of directed connections in the UTRP algorithm, would result in significantly higher runtimes and leed to unwelcome divergence between the courses of two directions of the same route. Instead, the cap between the two data structures is bridged by using two interface procedures: The first extracted the adjacency relations between stop points from a given Visum network model and used these to construct an undirected UTRP graph (Section 4.1). The other translates routes, consisting of lists of UTRP nodes, into directed lists of stop points to be implemented in Visum for evaluation (Section 4.2).

In contrast to Chapter 4 and Chapter 5, the optimisation procedure used in this work is based on Selection Hyper-heuristics. The main reason for this change is that the method used here only evolves a single solutions (Section 5.1). This allows to minimise the number of time-consuming changes to the Visum network model and, therefore, significantly reduces the run-time of the optimisation.

The capability of the interface and the optimisation procedure are demonstrated in several computer experiments (Section 6). Two network models from Visum training examples serve as instances for these experiments. One of them represents a small town of approximately 18,000 inhabitants, and the other, a city with more than 200,000 inhabitants. They were selected, as they come with implemented assignment procedures, which significantly reduces the workload for preparing the experiments. Both of them originate from real-world planning processes, which helps to showcase that the proposed interface can be applied such network models without the necessity of significant modifications.

Two different optimisation modes (i.e., sets of objectives) are used in the experiments: The first, the global optimisation, uses the same formulations of passenger and operator objective as used in the Chapters 4 and 5. Because of the nature of the used optimisation procedure, they were combined in a weighted sum. The second, the local optimisation, aims to reduce the number of private cars driving on a selected street. It is realised by utilising Visum's ability to combine regular assignment procedures with mode choice models. This sequence of procedures estimates which mode and path travellers between two zones are going to use, depending on the available PuT network[1]. The results of these estimations can be accessed on the level of individual links in the form of a prediction of the number of private cars using the link. Reducing these locally estimated carloads can be used as an objective for the UTRP algorithm. The resulting optimisation might include public transport connections that offer more direct connections for travellers who would otherwise drive over the selected street, as well as other modifications on the network, which are predicted to redirect the flow of private cars.

The local optimisation concept has, to the best of the author's knowledge, not been used before. From a theoretical point of view, it only marks a minor contribution, as it does not require changes to the interface or the optimisation procedure. However, the fact that the realisation of the local optimisation is possible with the same procedures used for the more standard global optimisation, highlights the adaptability of the presented interface and the potential for its use in a variety of planning tasks. (e.g. for the reduction of air pollution in heavily polluting areas).

Disclaimer

This paper uses the terms "vertex" and "edge" to describe the elements of the UTRP graph. This was done to better differentiate between these and the nodes and links used to described the Visum network model in the Visum manual [180].

[1]The mode choice procedure also takes into account factors unrelated to public transport; however, for the application described here these are kept constant.

Transportation manuscript No.
(will be inserted by the editor)

Public Transport Network Optimisation in PTV Visum using Selection Hyper-Heuristics

Philipp Heyken Soares · Leena Ahmed · Yong Mao · Christine L Mumford

Received: date / Accepted: date

Abstract Despite the progress in the field of automatic public transport route optimisation in recent years, there exists a clear gap between the development of optimisation algorithms and their applications in real-world planning processes. In this study, we bridge this gap by developing an interface between the Urban Transit Routing Problem (UTRP) and the professional transport modelling software PTV Visum. The interface manages the differences in data requirements between the two models and allows the optimisation of public transport lines in Visum network models. This is demonstrated with the application of Selection Hyper-heuristics on two network models representing real world urban areas. The optimisation objectives include the passengers' average travel time and operators' costs. Furthermore, we show how our approach can be combined with a mode choice model to optimise the use of public transport in relation to other modes. This feature is applied in a special optimisation experiment to reduce the number of private vehicles on a selected set of links in the network. The results demonstrate the successful implementation of our interface and the applied optimisation methods for a multi-modal public transport network.

P.Heyken Soares
Laboratory of Urban Sustainability
Nottingham University, NG7 2RD
E-mail: philipp.heyken@nottingham.ac.uk

L.Ahmed
School of Computer Science & Informatics
Queen's Buildings, The Parade, Cardiff University
E-mail: ahmedlh@cardiff.ac.uk

Y. Mao
School of Physics & Astronomy
Nottingham University, NG7 2RD
E-mail: yong.mao@nottingham.ac.uk

C.L. Mumford
School of Computer Science & Informatics
Queen's Buildings, The Parade, Cardiff University
E-mail: MumfordCL@cardiff.ac.uk

Keywords Route Optimisation · PTV Visum · Selection Hyper-heuristics

1 Introduction

Public transportation systems form an essential part of the infrastructure in urban areas. Their efficiency is vital to many stakeholders and therefore require precise design and planning. The generation of efficient route networks for public transport systems is known in many studies as the urban transit routing problem (UTRP)[1]. This problem has been addressed in the literature by a considerable number of studies that vary in their application on different instances, objectives and solution methodologies.

Despite intensive research into the UTRP, there is a vast gap between the often purely academic studies and the application of their findings in real-world planning processes. The reasons for the gap have not been thoroughly researched so far [42]. However, one possible explanation could be the differences in data requirements between the algorithms used in UTRP research and the commonly used planning tools. Planning agencies usually base their decisions on simulations made with professional transport modelling software, such as PTV's Visum [37] or INRO's Emme [20]. Models built with software packages such as these require detailed information about the street layout, transport infrastructure, and travel demand, making them time-consuming to be implemented and calibrated. The resulting model, however, is very powerful and allows planners to study and simulate a variety of transport related phenomena. Researchers working on the UTRP on the other hand, apply their design algorithms to abstract models that simplify many aspects of real-world transport networks. A graph structure with interchange points as vertices and direct connections between them as undirected edges is the common model used by most previous studies.

In this work, we bridge the gap between the two worlds of theoretical research on the UTRP and real-world transportation planning. We focus here on Visum transportation modelling software and compare its transport network structure to a common UTRP model used previously by many researchers, pointing out the key differences between them. Based on these findings, we outline a process to translate Visum network components into the UTRP graph structure and vice versa by implementing a middle layer interface through which the transport network information passes between the models. Utilising this interface, selection hyper-heuristics are used to optimise Visum public transport routes while taking advantage of Visum analysis tools to evaluate a given candidate solution. The primary contributions of this study are:

- A comparative study of network structures used in the UTRP research and Visum, and a methodology for bridging between them.

[1] The UTRP is, in fact, a sub-problem of a larger task to optimise public transport networks, which consists of five phases: 1) Route Design 2) Vehicles Frequency Setting 3) Timetables development 4) Vehicles Scheduling and 5) Crew Scheduling [9]. As the phases are interconnected, the problem has very high complexity, and most studies only deal with subsets of these phases using simplifications. In the case of the UTRP, which only considers the first stage, a fixed frequency on all routes is assumed, and other aspects such as the actual departure and arrival times of vehicles at stop points are not considered.

- Novel interface procedures between Visum and the UTRP network models to facilitate transferring and translating data between them.
- Selection hyper-heuristic algorithm adapted to optimise Visum public transport routes utilising the interface procedures.
- The integration of Visum tools for the evaluation of candidate solutions.

The following sections are organised as follows: section 2 outlines the historical background of the UTRP in planning applications and provides a brief introduction to PTV Visum and its applications; section 3 describes the UTRP and Visum network models in more details; section 4 summarises the interface processes; section 5 outlines the optimisation procedure using selection hyper-heuristics and the applied evaluation tools. Finally, section 6 presents our experimental results and section 7 highlights the key findings and conclusion.

2 Background

2.1 A Brief History of the UTRP

Attempts to automate the optimisation of urban public transport routes date back many decades using a variety of optimisation methodologies and algorithms. They generally fall into two categories: exact or heuristic based methods. Exact mathematical approaches [8, 10, 16, 41] have been tested in many studies. The major limitation of such methods is the difficulty of finding optimal solutions in large size networks given the combinatorial nature of the problem, and this has resulted in their failure to scale up to practical size instances.

Because of these shortcomings, research on the UTRP then shifted towards adopting heuristic based methods to solve the problem, due to their ability to tackle large-size problem instances. Some heuristic methods are based on heuristic construction procedures, which attempt to build optimal public transport route sets from scratch. An example of such method is the heuristic algorithm developed by Simonis in 1981 [39]. This method starts by generating a route using the shortest path between the highest density demand points and then deletes the demand satisfied by this route. The process iterates to the next highest demand points until a maximal number of routes is reached. Other heuristic construction methods are described in [4, 14, 38, 40].

The second group of heuristic methods attempt to improve a given input of route sets. These optimisation methods are usually based on meta-heuristic methods such as tabu search (e.g. [24, 31]), simulated annealing (e.g. [12, 13]) ant colony otimisation (e.g. [32, 43]), bee colony optimisation (e.g. [28, 29]), or genetic algorithms (GAs) (e.g. [19, 21, 27, 33]). The initial route sets are derived in some studies from the existing public transport network of the study areas. However, a heuristic construction procedure is often used to generate initial route sets. Examples of such a combination can be found in [2, 19, 24, 26].

Despite the huge amount of research on computer-based solutions for the UTRP, there are few studies that have been actually used in real-world planning processes [42]. One rare example is the work by Pacheco et al. in 2009 [31] which describes the optimisation of the bus network in Burgos, Spain, with regard to waiting times and trip duration. The solutions obtained improved significantly over those generated by planners of the city transport authorities using the same transit network data. Another example is a study from 2012 by Cipriani et al. in cooperation with the mobility agency of Rome, Italy [11]. It included the application of a genetic algorithm on an undirected graph representing the street network of Rome. The results show improvements over the existing bus route network in terms of waiting times, operator costs and unsatisfied demand. According to the authors, the mobility agency of Rome had started implementing their results in 2012. Many other studies compare their results against the real existing routes, showing that their methods can lead to improvements over an existing service (e.g. see [1, 5, 6, 19]). However, the results of these studies have not been verified in real-world planning processes.

2.2 Interfacing UTRP Algoirthms and Transport Modelling software

The key challenge for interfacing UTRP algorithms and macroscopic transport modelling software such as Visum or Emme, is to handle the differences between the undirected connections used in UTRP algorithms and the directed connections used in urban scale macroscopic modelling (more on this in section 3).

All UTRP studies which have so far interfaced their algorithms with macroscopic transport modelling software have bypassed these problems by only using network representations with undirected connections. One such case is mentioned above, the study on Rome [11], where the applied algorithm is interfaced with Emme. Other examples [3, 5] used Visum and Emme, respectively, and included procedures to construct suitable network representations.

So far, the most general tool for automatic public transport network optimisation in a transport modelling software is the line proposal algorithm for Visum developed by PTV itself in 2006 [30]. This tool applies a heuristic algorithm to an existing Visum network model. The algorithm first generates a palette of candidate routes between pre-selected stops. These candidate routes are then individually evaluated in Visum and the route taking up the most demand is then permanently added to the network. This step can be repeated as often as required. The tool only requires a minimal interface as the algorithm simply controls the end points of routes and initiates their evaluation. All other aspects are handled by Visum and the algorithm does not modify the pre-existing routes. Although the algorithm has been successfully used in at least two academic studies [25, 42], it remained a prototype and was never fully developed into an official Visum extension. Unfortunately, it no longer works on recent versions of Visum.

2.3 *Optimising Public Transport in Visum Through Selection Hyper-Heuristics*

The aim of this work is to create an interface between the UTRP network model and the macroscopic transport modelling software Visum, and utilise the interface to optimise Visum public transport routes using the general search methodology of selection hyper-heuristics.

Visum is a macroscopic transport modelling software package that combines both private (PrT) and public transport (PuT) into a single model and provides a rich suite of methodologies. It is a product of the PTV company based in Karlsruhe, Germany, and has been available for commercial use since the late 90s [15]. It is used at present by transport planners and analysts throughout the world. The results presented in this paper have been produced using Visum 17, which was the latest released version at the start of this project.

In addition to the variety of analysis tools, one of the main reasons to use Visum for the present work is the ability to control it via Python scripts over the Visum COM-API [34]. This library provides a number of interface functions to control Visum via scripts and extract any required information. Such scripts form the basis of the present work.

Given the complexity of Visum software and its huge number of available features and tools, we have to limit the descriptions of its structure and capabilities to those which are vital for the present work. We will discuss the relevant data structures of Visum network models in section 3.2 and the analysis options we use in section 5.5. For further details we refer the interested reader to the Visum Manual [37].

Selection hyper-heuristics are search methodologies motivated by the idea of generalising search techniques to several problem domains with minimal adaption. They have been applied to solve the UTRP for the first time in [2]. In this work thirty selection hyper-heuristics combining different combinations of selection and move acceptance methods with varying characteristics are tested on a set of known benchmark instances. A series of experiments and statistical comparisons between the thirty methods revealed the success of an online selection method inspired by the hidden Markov model over other selection methods. Furthermore, the best performing algorithm was successful in finding single solutions of high quality in very short run times compared to the best published solutions. The same winning approach was applied in [1] on a set of benchmark instances representing urban areas, and proved that it can surpass the performance of the genetic algorithm, NSGAII.

There are many advantages of a selection hyper-heuristic framework that makes it a good candidate for application on this work. First, it is a single point based framework, meaning it only require a single initial solution. This allows us to extract the existing public transport network from a given Visum network model and use it as the initial solution. Second, maintaining a single solution while improving it iteratively during the search, makes the interaction with Visum through the interface procedures straightforward. Also the relatively short run time of hyper-heuristic methods significantly adds to their attractiveness. Further description of the selection hyper-heuristics framework is outlined in section 5.

PuT: Refers to public transport.
PrT: Refers to private transport.
G: Graph of connected vertices and edges. Used to represent a UTRP graph.
v: Vertex of the UTRP graph belonging to the set of vertices V. Represents a stop in a Visum network model. (The terms stop and vertex can be used interchangeably in the UTRP graph.)
u: Terminal vertex ($\in V$), a vertex that route r is allowed to start and end with.
A: Adjacency matrix defines the connectivity of graph G. $A_{i,j} = 1$ means a direct connection exists between vertex i and vertex j. In the case of $A_{i,j} = 0$ the vertices are not directly connected.
r: Route in the UTRP graph as a path connecting a number of vertices. In Visum the route corresponds to a line. It is assumed to be bi-directional.
s: Stop in Visum network. Represents a vertex in the UTRP Graph.
sp: Stop point in Visum network associated to exactly one stop. Can be used to define the course of the line route.
l: PuT line in Visum. The spatial course of the line is defined by its line routes.
lr: Line route in Visum. Belongs to exactly one line and specifies the spatial course of this line for one travel direction. Line routes are defined by stop points.

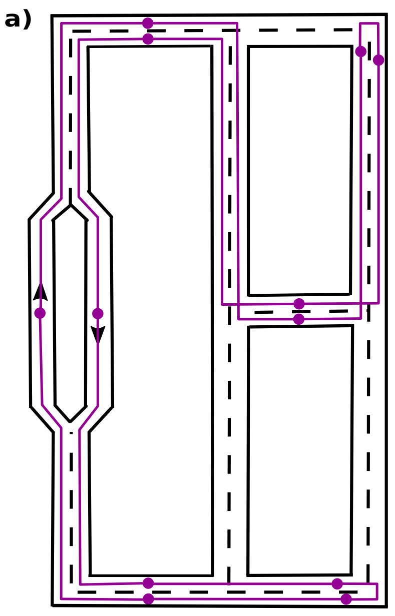
InfoBox 1: Summary of important concepts used in the following sections

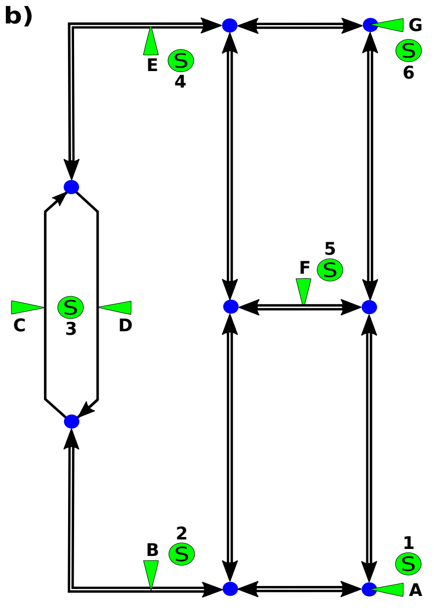
3 Data structures

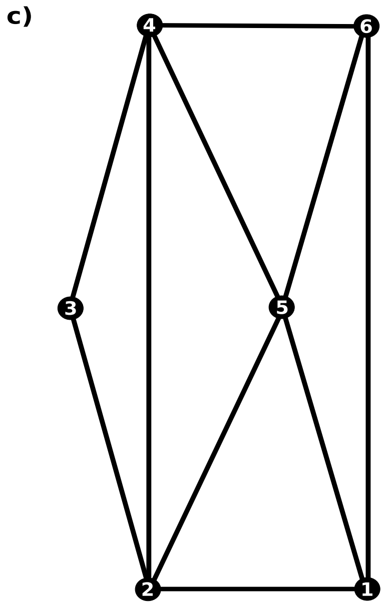
3.1 Defining the UTRP

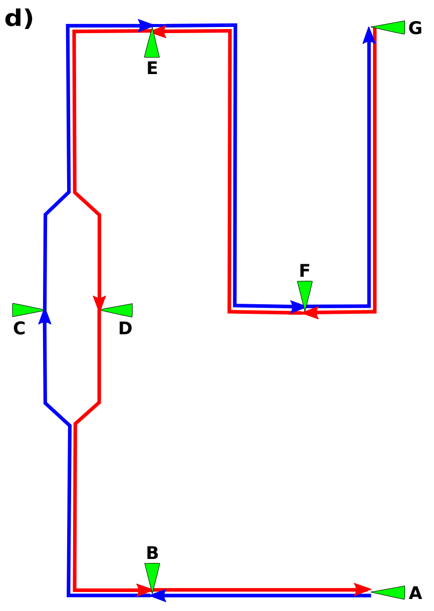
Almost all previous approaches to the UTRP choose to represent the available street (or rail) network as a graph $G = (V, E)$. The vertices $V = \{v_1, v_2, ..., v_{|V|}\}$ represent access and interchange points, and the edges $E = \{e_1, e_2, ..., e_{|E|}\}$ are direct connections between them. Using these definitions, a PuT route r (e.g. a bus line) can be represented by a list of directly connected vertices. The route is usually assumed to be bi-directional, meaning that vehicles after completing a journey in one direction, can turn around and make the same journey in the opposite direction. For this to be possible in an urban setup it is important that all routes begin and end at designated terminal vertices $U = \{u_1, u_2, ..., u_{|U|}\} \in V$ to allow U-turns. Following these definitions, the PuT network of a certain transportation mode can be described as a set of connected routes $R = \{r_1, r_2, ..., r_{|R|}\}$, which represents a solution to the UTRP as an optimisation problem. This “route set” has to be connected, in such a way that each route has at least one vertex in common with one or more other routes in the route set.

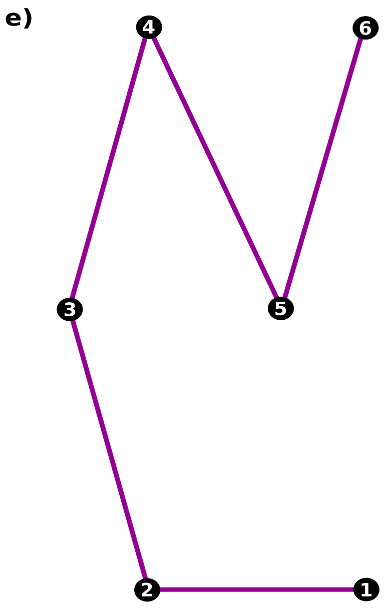
Fig. 1 **a)** Schematic street network with one bus service (purple) and its stop locations (purple dots). **b)** Representation of a) as a Visum network model. Shown are links (black arrows) and nodes (blue dots), as well as stop points (triangles) and their stops (green circles) The stop points C and D are both part of stop 3. **c)** UTRP graph with the stops in b) as vertices and the possible direct connections between them as edges. **d)** Schematic display of two line routes in the same network model as b) representing the bus service shown in a). The path of a line route is only defined by the stop points, although routing takes place along the nodes and links of the underlying network model with all the restrictions defined on it. Together both line routes form one line. **e)** The line routes from d) as a bi-directional list of vertices in the UTRP graph.











In its simplest form, the structure of the graph G is given by an adjacency matrix $A_{|V|\times|V|}$. This matrix defines the connectivity of the graph as follows: If two nodes i and j are directly connected, the respective entry in the adjacency matrix equals $A_{i,j} = 1$ otherwise $A_{i,j} = 0$. This matrix must be symmetrical following the rule that the routes are assumed bi-directional. The main advantage of such graphs for solving the UTRP is that they allow constraining routes to be made up of only directly connected vertices. This excludes many possible flawed solutions and thereby drastically reduces the search space.

Furthermore, information about the travel demand and the travel time between any two vertices in the graph is required for evaluation. This information is typically provided in the form of two-dimensional matrices: the travel time matrix, and the demand matrix. Both matrices are usually symmetrical. The travel time matrix gives the travel time between directly connected vertices. Differences in travel times between directions (e.g. resulting from one way streets) are usually not considered. The demand matrix determines the number of travellers between any two demand sources. In the vast majority of studies, the vertices V are used as demand sources. Only about one in five studies, use a zone demand structure similar to the structure used in macroscopic transport modelling (described below) [17].

3.2 Visum Data Structure

The inputs to Visum transport models also include travel demand and the network model of the available transport infrastructure.

As a standard practice in macroscopic transport simulations, the travel demand in Visum is aggregated at the level of zones ($Z = \{z_1, z_2, ..., z_{|Z|}\}$) and given in the form of an origin-destination matrix. A demand matrix $D_{|Z|x|Z|}$ defines how many trips are originating from zone z_i, and travelling to zone z_j. The demand data is given as separate matrices for PuT and different modes of PrT (e.g. cars). However, it is possible to initially give the demand as one matrix aggregating all trips and then use a mode choice procedure to split the demand into separate matrices for different travel modes (more on this in section 5.5.2).

The Visum network model is also based on a graph structure, though it is more detailed than the UTRP graph described above. Streets or rail segments are represented by links usable for specified modes of transport. Links are composed of two separate network objects, one for each travel direction. Each one of these objects can have different attribute values such as the allowed speed, and capacity in terms of the number of vehicles. One-way streets can be represented by blocking one direction of a link to all modes. Nodes at the beginning and end points of each link define the positions of intersections and junctions in the network. Which turns are permitted for which mode at the represented junctions can be defined in the properties of the nodes. Each zone ($z \in Z$) must be connected to at least one node by a connector, for exiting and entering the zone to the link network through this connection node.

The Visum network model further includes other elements to represent the available PuT lines and interchange points. PuT interchange points are defined in three layers. The “stop point” is the lowest layer indicating the actual stopping location of the vehicles (e.g. a segment of a train platform). Stop points can be placed either on nodes or links and can be used by all PuT lines traversing these network objects. A stop point placed on a link can either be accessible from both directions or can be restricted for use in a specific direction. The “stop area”, the middle layer, incorporates one or more stop points (e.g. a platform on a train station). All the stop points of the same stop area are considered instantly connected while transfers between stop areas require a certain walking time. Further, a node can be allocated to a stop point area to allow access to the wider network model, most importantly to connectors connecting nodes to zones. The “stop”, the highest level, incorporates one or more connected stop areas (e.g. a train station) and is the level on which walking time between different stop areas is assigned. Stop areas do not play a major role in this work, and we will assume that stops and stop points are directly connected to simplify explanations.

PuT lines are also defined in several different layers. On the highest level is the “line”. Each line in Visum belongs to exactly one transport system. The spatial course of the line is defined by its line routes. A single line can aggregate several line routes that are defined by lists of stop points to be traversed in the given order[2]. Line routes are thereby directed and most lines will have at least two line routes, one for each direction of travel. A special case are ring lines defined by single line routes starting and ending at the same stop point. Each line route is given a “time profile” describing, among other attributes, the travel times between the single stop points. “Vehicle journeys” with concrete departure times can be added to the line routes to create detailed timetables that specify departure and arrival times at the individual stop points.

3.3 Interfacing between the UTRP and Visum data structures

Based on the comparison of the standard graph structure of the UTRP with the respective data structure of Visum, there are key questions to be answered before constructing an interface between them. First we need to specify whether the optimisation is going to be performed on the level of line routes or lines. Optimising the line routes individually can lead to significant deviations between the line routes that belong to the same line. This would not be desirable in practice. Therefore, we have chosen to perform the optimisation at the level of lines.

The second question is whether the vertices of the UTRP graph should represent stops or stop points in the Visum network model. To answer this question we need to consider that some stop points in the Visum network model might be restricted for use in specific directions only. This would conflict with the fact that routes in the UTRP structure are assumed bi-directional. However, in the majority of such cases[3],

[2] Technically the list defining a line route contains both stop points and nodes. However, it is sufficient to only provide the stop points as Visum can add the required nodes automatically via the shortest path search.

[3] There are cases where a stop can only be reached from one direction. However, in all the networks examined in this work, these cases were very few in number and did not seem to have a significant impact on the results. We therefore neglected them at the present stage of the work. For the future, we plan to implement additional procedures that adequately address this problem.

the stops which these directed stop points belong to also incorporate other stop points that are accessible from several directions (see section 3.2 and figure 1). We therefore decided that the UTRP graph vertices should represent stops in the Visum network model rather than stop points.

Finally, it has to be decided which stops (i.e. vertices) to be considered as terminal vertices in the UTRP graph. It is recommended in [37] to exclusively use stop points that are placed on nodes as the beginning or end points of line routes. However, there are many existing line routes in the network models we used that do not follow this recommendation. We therefore select as terminal vertices, the stops which fulfil at least one of the following conditions[4]: a) Having at least one of their stop points placed on a node. b) Having at least one of their stop points being the beginning or end of an existing line route.

4 The Interface Processes

Based on the discussion in section 3.2, we can formulate two main interface processes to translate routing information between UTRP algorithms and Visum. The first, is to extract the adjacency relations between stops to be used as the basis for route alterations during the optimisation. This process will be introduced in section 4.1. The second is a process to convert the modified routes (i.e. undirected lists of stops) into pairs of line routes (i.e. directed lists of stop points) to allow the implementation in Visum for evaluation. This process will be outlined in section 4.2. The outcomes of both interface procedures are mode-specific, as some links in a Visum network model may only be accessible to specific modes.

4.1 Extracting a UTRP graph from a Visum network model

The process to extract the adjacency relations between the stops for a PuT mode m from a given Visum network model is outlined in algorithm 14. It begins by creating a stop point connectivity matrix C^m. This matrix has three types of entries: $C^m_{i,j} = 0$ indicates that there is no connection between the two stop points i and j, $C^m_{i,j} = 1$ means a direct connection between stop points i and j exists, and $C^m_{i,j} = 0.5$ indicates that a connection exists but it is indirect (i.e. the connection goes over other stop points). Additionally, a distance matrix Δ^m which records the travel time between each pair of stop points is also created in this process (its usage is described in section 4.2). A python script utilising functions from Visum COM-API is used to extract the required information from the given Visum network model. This script is demonstrated by algorithm 1. For every possible combination of stop points i, j, the algorithm attempts to build a test line route beginning at i and ending at j by finding the shortest path along the links open to mode m. The building is only successful if Visum manages to find a shortest path from i to j considering all the constraints of the given network

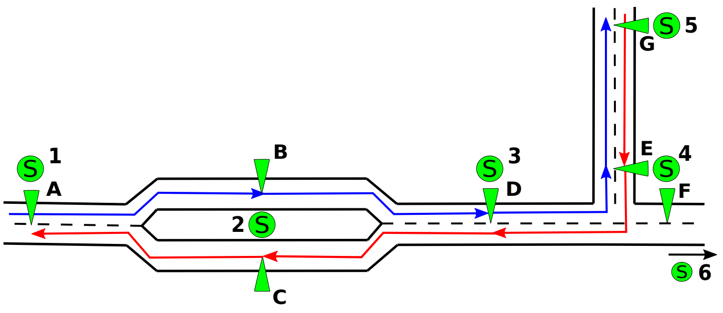
[4] The conditions listed here were selected for the optimisation of bus lines described in section 6. For other scenarios or in case of optimising other PuT modes it might be necessary to adapt these conditions. Also, a manual selection of terminal vertices is possible.

Algorithm 1: Algorithm to generate the stop point connectivity matrix and distance matrix for a mode *m* in a given Visum network.

```
   Data: Visum Network
   Result: Connectivity Matrix $C^m$, Distance Matrix $\Delta^m$
1  begin
2      foreach Stop point i in Visum network do
3          foreach Stop point j in Visum network do
4              Build test line route $l$ of the mode $m$ between $i$ and $j$ .
5              if $l$ can be built then
6                  $\Delta^m_{i,j} \leftarrow$ travel time between $i$ and $j$
7                  $n \leftarrow$ number of stop points in $l$
8                  if $n = 2$ then
9                      $C^m_{i,j} = 1$
10                 else
11                     $C^m_{i,j} = 0.5$
12             else
13                 $C^m_{i,j} = 0$
14     return $C^m$, $\Delta^m$, Visum network back to initial state.
```

model. All other stop points located on this path are automatically connected to the built line route. If the test line route contains more than two stop points (not only *i* and *j*), the connection between *i* and *j* is considered indirect and recorded in the connectivity matrix as $C^m_{i,j} = 0.5$. If no further stop points are connected to the test line route, the connection is considered direct and recorded as $C^m_{i,j} = 1$. In either case, the travel time of the built line route is calculated by Visum, and recorded in the distance matrix as $\Delta^m_{i,j}$. If the process of finding the shortest path between *i* and *j* fails, these stop points are considered as not connected ($C^m_{i,j} = 0$). Once the algorithm has iterated through all stop point pairs, the Visum network model is returned to its initial state.

After the stop points connectivity matrix C^m is constructed, the adjacency matrix $A^m_{|V|\times|V|}$ of the undirected UTRP graph is built using the following rules: two stops X and Y are considered connected ($A^m_{X,Y} = A_{Y,X} = 1$) if and only if at least one stop point x belonging to stop X is directly connected to at least one stop point y^* belonging to stop Y ($C^m_{x,y*} = 1$), and at least one stop point y belonging to stop Y is indirectly connected to at least one stop point x^* belonging to stop X ($C^m_{y,x^*} \geq 0.5$). In any other case, even if all stop points of the two stops X and Y are indirectly connected, X and Y are considered as not adjacent ($A^m_{X,Y} = A^m_{Y,X} = 0$).



X	Y	Z	SP
-	1	2	A
2	1	-	A
1	2	3	B
3	2	1	C
2	3	4	D
4	3	2	D
3	4	5	E
5	4	3	E
3	4	6	F
6	4	3	F
4	5	-	G
-	5	4	G

Fig. 2: Schematic street network with five stops (1 to 5) and seven stop points (A to G). A sixth stop is assumed along the street to the right. The UTRP route [1,2,3,4,5] is converted into two stop points lists using the conversion table on the right. The conversion table gives the correct stop point for a single stop using a combination of three stops (i.e. vertices) containing this stop and its adjacent stops. The resulting stop points lists are displayed in blue (from A, over B to D, E and G) and red (from G, E and D over C to A). For the terminal nodes, 1 and 5, a vertices triple with empty fields are given to be used at the beginning and end of a route respectively.

4.2 Transforming UTRP Routes to Visum Line routes

During the optimisation process, the changed routes have to be implemented into Visum for evaluation. To do so, the lists of vertices (i.e. stops) in these routes need to be converted into two lists of stop points, one for each direction of travelling[5]. As described in section 3, a single stop can have multiple stop points. Therefore, the correct combination of stop points that represent a specific combination of vertices has to be selected. This is done with the help of a conversion table Υ^m . Its use is illustrated in figure 2.

Assume we have a route r_i that contains the following set of vertices in this order: $\{1,2,3,4,5\}$ and vertices 2 and 4 are associated with multiple stop points as illustrated in figure 2. To convert the route r_i to a stop point list, we need to select the correct combination of stop points that represent a particular direction of travel through a vertex. For the vertices associated with a single stop point this selection is trivial. However, for the vertices associated with multiple stop points, the selection process depends on the adjacent vertices and the order of these vertices in the route. In the example given in figure 2, the stop point representing stop 2 can be either B (if the order is 1,2,3) or C (if the order is 3,2,1).

The conversion table Υ^m used to select the correct stop points for every vertex is shown in figure 2. The first three columns give the possible triple set of connected vertices, one combination in each row. Every combination is considered twice, once for each travel direction. The fourth column specifies the stop point choice for the middle vertex in the combination. For terminals, additional triple sets with empty fields are used to determine the stop points at the beginning and end of a line route.

[5] Ring lines, where the first and last stops are identical require only one line route.

Algorithm 2: Algorithm to generate the conversion table for mode m

```
Data: Set of all vertices $V$, Adjacency Matrix $A^m$, Connectivity Matrix $C^m$,
      Distance Matrix $\Delta^m$
Result: Conversion Table $\Upsilon^m$
begin
    foreach $Y \in V$ do
        $K \longleftarrow$ all Vertices $K \in V$ with $A^m(Y,K) = 1$
        foreach $X \in K$ do
            foreach $Z \in K$ do
                $\Xi \longleftarrow$ Stop Points of Stop represented by Vertex $X$
                $\Phi \longleftarrow$ Stop Points of Stop represented by Vertex $Y$
                $\Omega \longleftarrow$ Stop Points of Stop represented by Vertex $Z$
                $q \longleftarrow$ 3 dim. matrix for travel times between stop point
                    triples. (Default value $\infty$)
                foreach $\chi$ in $\Xi$ do
                    foreach $\varphi$ in $\Phi$ do
                        foreach $\omega$ in $\Omega$ do
                            if $C^m_{\chi,\varphi} \geq 0.5$ and $C^m_{\varphi,\omega} \geq 0.5$ then
                                $q_{\chi,\varphi,\omega} = \Delta^m_{\chi,\varphi} + \Delta^m_{\varphi,\omega}$
                $\varphi^* = \underset{\varphi}{\operatorname{argmin}}(q_{\chi,\varphi,\omega})$
                add row [ $X$ , $Y$ , $Z$ , $\varphi^*$ ] to $\Upsilon^m$
    return $\Upsilon^m$
```

With the conversion done, the stop point lists can be implemented in Visum, replacing the old line routes by using Visum COM-API [34]. To allow PuT assignments to be run for the evaluation (see section 5.5), time profiles and vehicle journeys can be generated as well.

The process to construct the conversion table Υ^m is outlined in algorithm 2. It builds on data produced during the generation of A^m (algorithm 14), specifically the connectivity matrix C^m and the distance matrix Δ^m. For every vertex Y, the algorithm identifies the vertices K adjacent to it (line3). With this, it is possible to define triples of connected vertices $\{X,Y,Z\}$ where $(A^m_{X,Y} = A^m_{Y,Z} = 1)$. The algorithm loops over every possible combination of the stops triple $\{X,Y,Z\}$ (line 5) and extracts the corresponding stop point triple defined as $\{\chi,\varphi,\omega\}$, where χ is a stop point of X, φ a stop point of Y, and ω a stop point of Z (line 6). If any of the stops $\{X,Y,Z\}$ is associated with multiple stop points, several stop points triples are generated. The connection between the stop points for every extracted stop points triple $\{\chi,\varphi,\omega\}$ is checked against the stop points connectivity matrix C^m. If according to C^m, these stop points are connected $(C^m_{\chi,\varphi} \geq 0.5, C^m_{\varphi,\omega} \geq 0.5)$ (line 14), the travel time is calculated and stored in a three dimensional travel time matrix: $q_{\chi,\varphi,\omega} = \Delta^m_{\chi,\varphi} + \Delta^m_{\varphi,\omega}$ (line 15). After calculating the travel times for every possible stop point triple of the stops

$\{X,Y,Z\}$, the one with the minimum travel time is selected $q_{\chi^*,\varphi^*,\omega^*} = \min(q_{\chi,\varphi,\omega})$ (line 16), where φ^* is the stop point that will represent the vertex $Y \in \{X,Y,Z\}$ in the conversion table.

5 Optimisation

5.1 Selection Hyper-heuristics

Hyper-heuristics have emerged to raise the level of generality of search techniques for difficult computational problems. While heuristics work directly in the solution space, hyper-heuristics work at a higher level controlling a set of low-level heuristics which perform direct operations on the solution. This way, hyper-heuristics do not require direct knowledge of the underlying implementation of the solution domain, and this allows simple adaptation to different problem domains.

Hyper-heuristics are classified in [7] according to the nature of the heuristic search space, into generation and selection hyper-heuristics. The former generates new heuristics from existing components of other heuristics, while the latter (i.e. the approach we use here) selects existing heuristics in their entirety from an existing set of heuristics. The selection hyper-heuristics operate as two components: the selection and move acceptance components. At each decision point, the selection method selects a heuristic or sequence of heuristics from an existing repository of low-level heuristics and applies it to the solution at hand, to generate a new solution. Afterwards, an evaluation step decides whether to accept the new solution based on an acceptance criterion. The two components are iterated until a termination condition is met.

Selection hyper-heuristics can embed a non-learning or a learning mechanism depending on whether there is feedback received during the search. This feedback helps to improve the selection decisions made by the selection component.

In this work, we have tested two selection methods: a "Simple Random" (SR) selection with no learning and a "Sequence-based Selection Hyper-heuristic" (SSHH) with online feedback. For both, we used the acceptance criterion "improve or equal" (IE) meaning that new solutions are only accepted if they are equal to or better than the current solution.

SR randomly selects a heuristic based on a uniform probability distribution. It is considered the simplest selection method and can be effectively used as a reference for more complex selection methods. SSHH resembles the Hidden-Markov model, where the low-level heuristics represent the hidden states of the model and the transition between these different states forms sequences of heuristics applied to the solution. The goal is to generate and learn good sequences of heuristics that are likely to improve the solution. In [2] a series of experiments and comparisons through statistical methods proved that sequence-based selection is more successful in solving the UTRP than other single-based heuristic selection mechanisms. We will briefly summarise this algorithm here and outline its application for the optimisation of Vi-

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

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

(a) Matrices initial values

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

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

(b) Matrices updated values

Fig. 3: Example of updating the values in the transition and sequence construction matrices: We assume the application of the sequence [llh_0, llh_1, llh_3] improved the best solution. The scores of these low-level heuristics in the "Transition Matrix" and the "Sequence Construction Matrix" are updated. This update increases the probability of selecting this sequence in later steps.

sum PuT networks in the following sections. Further details for this algorithm can be found in [2, 22, 23].

Each low-level heuristic is associated with a number of scores, which are used to derive the probability of moving from this heuristic to another low-level heuristic in the sequence. Assuming we have the set of n low-level heuristics: [llh_0, llh_1, $\ldots$, llh_n], a "transition matrix" of size $n \times n$ stores these scores. Another matrix called the "sequence construction matrix" of size $n \times 2$ associates each low-level heuristic with two states: "continue", and "end" to determine whether the sequence should end or continue at this point. Both matrices initially start with equal scores.

A sequence of low-level heuristics is constructed with the guidance of the two matrices. The sequence is initialised with a randomly selected heuristic. The probability for a heuristic to be selected next in the sequence increases with its score in the transition matrix (i.e. the higher the score, the higher the selection chance). Each time a new heuristic enters the sequence, the sequence status is checked in the sequence construction matrix to determine whether to stop building the sequence or to add another heuristic.

The values in the matrices are updated if the new solution generated by applying a specific sequence proves to be superior to the current solution. Updating the scores in the matrices increases the chance of selecting this successful sequence in later steps. An example of this update is demonstrated in figure 3. Over the duration of the optimisation process, the update mechanism helps identifying successful sequences.

5.2 PuT line route optimisation using selection hyper-heuristics

At the beginning of the optimisation process, a set of routes is initialised from the Visum network model by transforming Visum line routes into lists of stops (vertices) to construct an initial solution. The initial solution (S_{init}) is introduced to the hyper-heuristic as the current solution (S_{curr}) and the iterative optimisation process

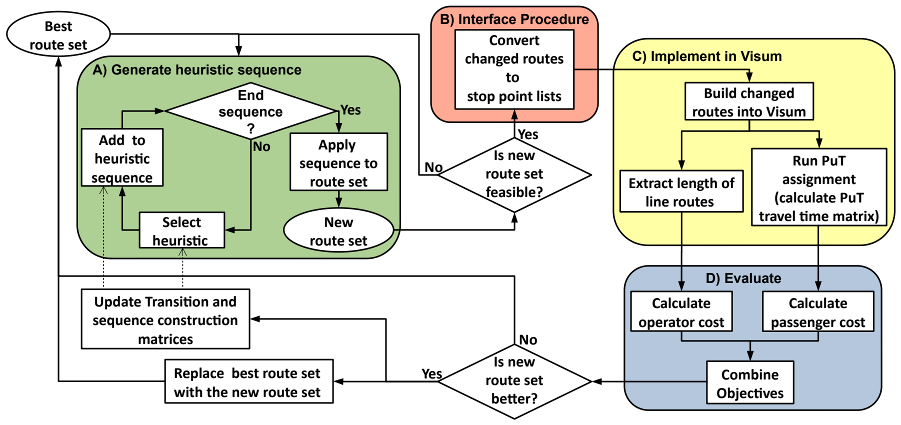

Best route set
A) Generate heuristic sequence
End sequence ?
Yes
No
Add to heuristic sequence
Apply sequence to route set
Select heuristic
New route set
B) Interface Procedure
Convert changed routes to stop point lists
Yes
No
Is new route set feasible?
C) Implement in Visum
Build changed routes into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

Fig. 4: Description of one iteration of the SSHH algorithm application in the global optimisation. Each iteration begins with box A: The generation of sequence of heuristics (see section 5.1) and applying it to the current route set (see section 5.3) to create a new route set. The new route set is tested for its feasibility (see section 5.4). If the new route set is not feasible, it is rejected and a new heuristic sequence is generated. Otherwise, the new route set is converted to stop point lists (Box B, see 4.2). The stop point lists are implemented in Visum as line routes and an assignment is executed to extract the information necessary for the evaluation (Box C, see section 5.5). The evaluation includes combining the objectives of passenger and operator costs (Box D, see section 5.5.1). If according to the evaluation, the new route set improves the current best route set, it replaces it and the transition and sequence construction matrices are updated (see section 5.1).

begins. One iteration of the SSHH algorithm is illustrated in figure 4. Depending on the selection mechanism, either a single heuristic is selected in case of the simple random selection (SR), or a heuristic sequence is constructed in case of the sequence based selection (SSHH). The application of this heuristic/heuristics sequence to S_{curr} creates the new solution S_{new}, which is tested for its feasibility (see section 5.4). If S_{new} is not feasible, it is rejected and a new heuristic/heuristics sequence is selected. Otherwise, S_{new} is converted into lists of stop points and implemented in the Visum network model (see section 4.2) for evaluation (see section 5.5). The evaluation automatically takes into account the interplay between different PuT modes. If configured accordingly, the impact of private transport modes can also be considered.

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

As the adjacency matrix and conversion table have to be mode-specific, solutions can only include routes for one PuT mode. If multiple modes are to be optimised

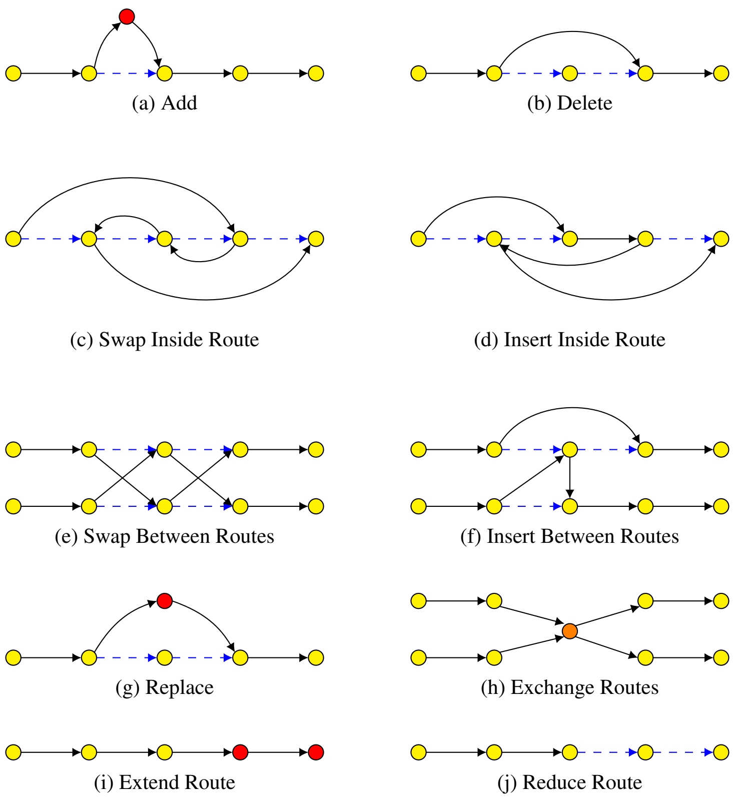


Fig. 5: Low-level heuristics set description. Straight arcs are edges in the route or added after applying the heuristic, dashed arcs are edges removed after applying the heuristic, red nodes are nodes added after applying the heuristic.

the solutions for different route networks have to be evolved independently. Theoretically, this can happen either in alternation, by changing the mode to be optimised in each step, or sequentially in a hierarchical process. However, such an undertaking is beyond the scope of this paper.

5.3 Low-Level Heuristics

The hyper-heuristic algorithm manages a set of low-level heuristics to modify a given route set. All the low-level heuristics have been designed to follow the adjacency relations defined by the adjacency matrix A when performing operations. If applying the operation on the selected routes and positions would create invalid connections, new routes and positions are selected instead. This increases the chance of generating feasible solutions.

We give here a full list of the low-level heuristics applied in this work (also illustrated in figure 5):

- llh_0 **(Add)**: Selects a random route and a random position in this route. A new vertex is selected and added in this position.
- llh_1 **(Delete)**: Selects a random route and random position and deletes the vertex in this position.
- llh_2 **(Swap Inside Route)**: Selects a random route and two random positions. The two vertices in these positions swap with each other.
- llh_3 **(Insert Inside Route)**: Selects a random route and two random positions. The vertex in the first position is inserted in the second position.
- llh_4 **(Swap Between Routes)**: Selects two random routes and two random positions on each of them. The vertices in these positions swap with each other.
- llh_5 **(Insert Between Routes)**: Selects two random routes and two random positions on each of them. The vertex in the first position of the first route is inserted in the second position of the second route.
- llh_6 **(Replace)**: Selects a random route and a random position. The vertex in this position is replaced by another selected vertex.
- llh_7 **(Exchange)**: Selects two random routes and splits them at a common vertex. The parts of the two routes are exchanged to create two new routes. If the selected routes do not have a common vertex, a new pair of routes is selected.
- llh_8 **(Extend Route)**: Selects a random route and adds vertices to the end of the route until reaching another terminal.
- llh_9 **(Reduce Route)**: Selects a random route and deletes vertices starting from the last vertex in the route until reaching another terminal node.

5.4 Solution Feasibility

Before implementing S_{new} into the Visum network model, a feasibility test is applied to ensure that the route set satisfies all defined constraints in order to avoid wasting time in generating and evaluating many infeasible solutions. The feasibility constraints are defined with respect to the specifications in the Visum network model, for instance, backtracks and cycles are tolerated in the Visum network setup, while it is commonly considered a violation in most UTRP models. The full list of constraints is as follows:

- The order of vertices on each route must follow the adjacency relations defined by the matrix A.
- Routes must start and end at the defined terminal vertices.
- Ring routes are allowed, but must start and end on the same vertex.
- The length of each route is within specified limits defined as input parameters by the user.
- No routes should fully overlap (this includes one route being sub-part of another).
- Each zone must be connected to the route set: each zone in the Visum network model must have at least one connector that is connected to an active stop point (i.e. a stop point used by at least one PuT line).

5.5 Evaluation

Depending on the objectives of the study and the available data, there are different ways to evaluate a route set. Transport modelling software packages like Visum come with a multitude of options to analyse the impact of changes in a transport network which can be used to construct a vast array of evaluation functions. Most notable among Visum's tools to model the impact of changes in a transport model are the assignment procedures used to calculate the flow of vehicles (PrT assignments) or PuT passengers (PuT assignments) through the network model. They provide output such as zone-to-zone travel time matrices (used in section 5.5.1) or the load of vehicles on links (used in section 5.5.2).

Assignment procedures, in general, start by determining all possible paths through the network model between all pairs of zones and their impedance. For PrT assignments, a specific PrT mode (e.g. private cars) has to be selected and the possible paths calculated are constrained to the links which are open to this mode. PuT assignments determine the available paths on the network of all PuT lines. Possible interchanges between different PuT modes (e.g. between bus and train) are automatically taken into account.

The impedance of a path includes all time costs[6] associated with the traveller. For PrT, the impedance is defined by factors like the allowed speed on the links, turn penalties and time losses due to congestion effects. For PuT, the main factor of the path impedance is the perceived journey time, which is defined as weighted sum of several factors[7] such as:

- Access time from origin zone to origin stop point (usually via the mode "walking").
- Waiting time at origin stop point.
- Total in-vehicle travel time.
- Penalty for the number of transfers.
- Walking time between stop points at transfers.
- Waiting time at transfers.
- Access time from destination stop point to destination zone (usually via the mode "walking").

Of particular importance are the transfer waiting times, which are the factors that distinguish between the two distinctive PuT assignment models used in this work: the headway-based and the timetable-based assignment. While the former assumes a fixed waiting time between vehicles for all routes, the latter derives the waiting times from a concrete timetable. Once all paths are determined, the assignment distributes[8] the trips given in the demand matrix over the available paths based on their

[6] Other costs, such as monetary costs, can be defined as well. However, these do not play a role in the present study.

[7] Other factors, such as boarding time, can also be added, but it does not play a role in the present study. The weightings of the different factors can be set freely. In this study we used Visum default configurations where all time factors are weighted equally and the transfer penalty is set to 10 minutes per transfer.

[8] The used distribution models are one of the main differences between the available assignment procedures. More details of these procedures and their distribution models are explained in Visum manual [37].

impedance. The resulting trips distribution allows us to estimate, for example, the number of vehicles that use a certain link (i.e. the link load).

The results of the assignments can be accessed in a number of different ways to generate evaluation functions. In the following, we introduce two evaluation methods: the global optimisation and the local optimisation. The former uses a travel time matrix generated from the perceived journey time. The latter accesses the vehicle loads on selected links.

5.5.1 Global Optimisation

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

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

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

where $D_{i,j}$ is the PuT travel demand from a zone i to a zone j, and $|Z|$ is the total number of zones. $\Theta_{i,j}$ is the shortest perceived journey time from zone i to zone j using the PuT network defined by the solution S. The matrix Θ can be generated during the execution of the PuT assignment.

The second objective is the reduction of the operator costs. We have used a simple approximation for the operators expenditures given by the total sum of travel times for travelling all the line routes in the PuT network:

$$C_O(S) = \sum_{i=1}^{|lr|} \tau_i(S) \tag{2}$$

where τ_i is the total travel time of line route i and $|lr|$ is the total number of line routes. The value of τ_i can be easily extracted from the network model using Visum COM-API.

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

$$f_{global}(S) = \alpha \frac{C_P(S)}{C_P(S_0)} + \beta \frac{C_O(S)}{C_O(S_0)} \tag{3}$$

where S_0 is the initial solution. The two weighting factors α and β can be adjusted in relation to one another to generate solutions that are more favourable for either operators or passengers.

5.5.2 Local Optimisation

For the local optimisation, we take advantage of two features in Visum. The first is that the assignment results can be easily accessed at a very localised level, e.g. the vehicle load of a particular transport mode on an individual link. The second is the ability to combine PuT and PrT assignments with a mode choice procedure. The mode choice procedure takes as input a demand matrix with all trips, and distributes them into PuT trips and PrT trips. The distribution[9] is based on the impedance of the possible paths in the respective modes [37]. The results are separated into PuT and PrT demand matrices which are then used by the respective assignment procedures to assign the trips to travel paths. This combined procedure sequence allows us to analyse the effects of changes in the infrastructure on different transportation modes.

For the local objective function, we select a group of links L (e.g. links in a specific neighbourhood), run the above mentioned combined procedure sequence and sum up the load of private cars (i.e. PrT mode) on these links. The objective is to minimise the load of private cars on the selected links given by:

$$C_L(S) = \sum_{i=1}^{|L|} v_i(S) \tag{4}$$

$v_i(S)$ is the load of private cars on link $i \in L$ while the travellers in the network can choose between travelling via the public transport network defined by solution S, or by private cars. The link load is a standard output for most PrT assignments available in Visum 17[37].

However, optimising a single objective can lead to very extreme solutions that are undesirable from other perspectives[10]. In order to avoid this we have used the global objectives introduced in the previous section as bounding factors:

$$f_{local}(S) = \begin{cases} \frac{C_L(S)}{C_L(S_0)} & \text{if } C_P(S) \leq \lambda_P \cdot C_P(S_0) \text{ and } C_O(S) \leq \lambda_O \cdot C_O(S_0) \\ \infty & \text{otherwise} \end{cases} \tag{5}$$

where S_0 is the the initial solution and λ_P and λ_O are factors to determine how much of an increase in the global objectives of solution S over their initial values is deemed acceptable to consider solution S.

We have chosen a relatively simple measure to show the validity of the concept of local optimisation. However, it is straightforward to generalise this concept and apply it to other scenarios. For example, it is possible to estimate noise and air pollution with the HBEFA extension module in Visum[11] and access the results on link level similarly to the PrT load. This would extend our methods to design PuT networks that improve the situation in heavily polluted areas.

[9] For the experiment in section 6.3 a nested logit distribution function was used. However, other distribution functions are available in Visum, including a gravity model. More details on them can be found in the Visum manual [37].

[10] For an example see the results from the passenger perspective presented section 6.1 where the operator costs increase by a factor of 8.

[11] Unfortunately, such a scenario could not be included in this study, as the HBEFA module was not included in our academic license.

6 Empirical Results

6.1 Test on a small instance

6.1.1 Setup

For the first set of experiments, we used the transport model from the Visum quick start tutorial. This network model was built in 2006 as a Visum training exercise, and is loosely based on the small town of Pfullingen, Germany, with around 18 thousand inhabitants. The network model is relatively small, containing 652 nodes and 1782 links, 81 zones, 35 stops and stop points, and only five bus lines. We slightly modified this network model to be able to test various aspects of the interface procedure. It was, for example, necessary to create one stop with more then one stop point to test the use of the conversion table (see section 4.2). A detailed description of these changes can be found in appendix A.1.

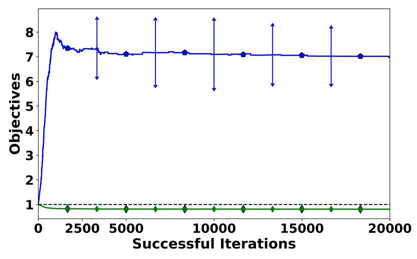
The optimisation in these experiments is based on the global evaluation method (section 5.5.1). To calculate the perceived journey time matrix Θ, we used the headway-based PuT assignment procedure. A fixed frequency of 10 minutes is defined for all lines. This assignment model does not require the generation of vehicle journeys for each changed line route, which improves the run time.

The termination condition on these experiments is defined by the number of successful iterations. A successful iteration consists of generating a new feasible solution, implementing it in Visum and evaluating it. The number of successful iterations before the hyper-heuristic terminates is set to 20000.

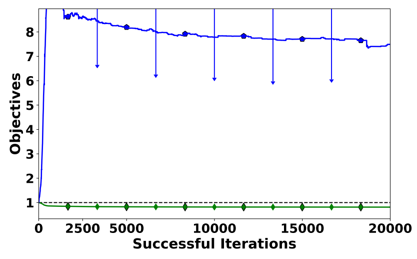
6.1.2 Results

In the first set of experiments we tested the two selection methods SR and SSHH in three distinctive scenarios: the passenger perspective, the operator perspective, and the balanced perspective. Each of these scenarios is defined by a different set of parameters in equation 3: For the operator perspective, effectively only the operator cost was considered as we set $\alpha = 10^{-6}$ and $\beta = 1 - 10^{-6}$. The opposite in the passenger perspective, where the focus is set on the passenger cost by setting $\alpha = 1 - 10^{-6}$ and $\beta = 10^{-6}$. In the balanced perspective, we create a balance between the two objectives by setting both parameters to $\alpha = \beta = 0.5$.

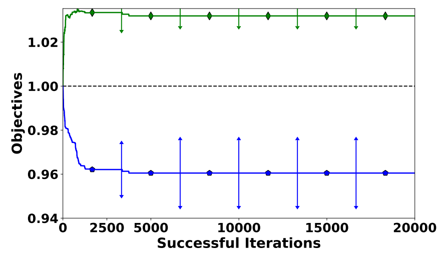
Figure 6 displays the change of the average passenger cost C_p(green line), average operator cost C_O (blue line), and the combined objective f_{global} (black line) calculated by equation 3. For each of the three scenarios the passenger and operator costs have been normalised using their initial values for better interpretation of their performance. The averages are calculated from the ten runs for each successful iteration.



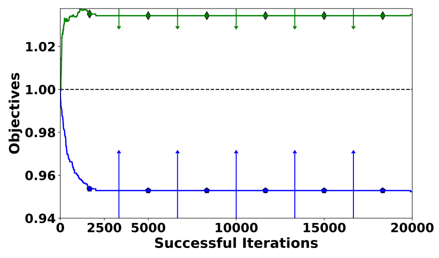
(a) SR passenger perspective



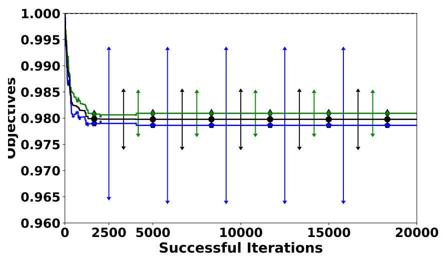
(b) SSHH passenger perspective



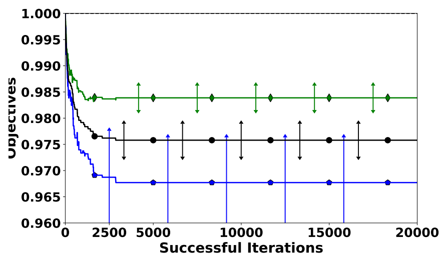
(c) SR operator perspective



(d) SSHH operator perspective



(e) SR balanced perspective



(f) SSHH balanced perspective

Fig. 6: Results of the global optimisation on a small network model for three scenarios: passenger perspective, operator perspective, and balancing the two objectives using two selection hyper-heuristics (SR and SSHH). Each figure displays the development of the normalised passenger objective C_P averaged for ten runs (green with rectangles[12]) the normalised averaged operator objective C_O (blue with pentagons[12]) and the averaged combined optimisation function f_{global} (black with circles[12]). The averages were calculated for every successful iteration from ten independent runs. The bars in the middle represent the standard deviation between the runs.

[12] The position of the markers (rectangle, pentagon and circle) have no meaning other than distinguishing the lines.

From the figures, it is clear that, from either the passenger or the operator perspective, the objective that the optimisation is focusing on decreases rapidly from its initial value, while the other objective increases. This improvement starts to slow down at the 2000 iterations stage.

In the case of balancing the two objectives, both the passenger and operator costs show similar behaviour by dropping quickly at the beginning of the search and slowing down after 2000 iterations. The SSHH selection method was more successful in this case in improving the operator cost, while both selection methods reduced the passenger cost at a similar rate.

Table 1 summarises the results of the passenger and operator costs for the ten runs normalised and averaged. The best results and the standard deviation are also recorded. From this table, the most notable improvement is in the passenger perspective with a reduction of 20% in the passenger cost from the initial values, although this was at the expense of significantly increasing the operator costs. The operator perspective runs reduced the operator cost on average by almost 7%, and the balanced runs recorded an improvement of nearly 5% on the operator cost while the passenger cost is also improved by 3%.

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

Table 1: Results from passenger, operator, and balanced configurations for the two selection hyper-heuristics. The results are normalised and averaged over the ten runs. The standard deviation and the best results are also recorded.

The performance difference between the two selection hyper-heuristics is very small as can be seen from the table, but two observations were made during the experiments: the SSHH selection method was able to improve more in fewer iterations compared to simple selection, and this fact is critical for working with larger networks. Second, the SSHH recorded better individual results for the runs in many cases, especially from the operator perspective. Based on these facts we have selected the SSHH to be applied in the next set of experiments on a larger network model and to test the concept of local optimisation.

6.2 Application on City Size Network

6.2.1 Setup

In order to show the validity of the presented methods on a larger scale, another set of experiments has been performed on a network model originating from a real-world planning process. It was generated in the 1990s for the city of Halle, Germany, with over 200 thousand inhabitants. The model is made up of 1934 nodes, 4832 links, 81 zones, 288 stops and 313 stop points, and in total 41 PuT lines of which 18 are bus lines. Since 1996, this model has formed the basis of many Visum training examples, and therefore it is currently included in all Visum installations. Although this model has been modified over time, its size and layout are sufficient to represent a real-world network model. We only made very small modifications for the purpose of this work which can be found in appendix A.2.

The Halle network model includes several modes of public transport: bus, tram and train. The optimisation is only applied on the bus lines, leaving the lines of the other modes unchanged. The frequency of the bus lines is set to 10 minutes. In these experiments, we used the timetable-based PuT assignment, which bases the interchange waiting times on a timetable generated from departure and drive times. This way, the frequencies of the unchanged PuT lines are accurately reflected. However, the use of the timetable-based assignment requires adding vehicle journeys to the modified bus lines to define the bus timetable.

The termination condition for these experiments is set to run time rather than iterations, where each experiment is run for 16 hours before it terminates. This was done for practical reasons and resulted in an average of 8919 successful iterations (i.e. on average 6.6 seconds per successful iteration[13]) for each run.

6.2.2 Results

Ten runs were applied on this transport network using the sequence based selection with the balanced configuration. This configuration was chosen as an example of a planning process which requires a compromise between passengers and operators. Figure 7 shows the development of the average value of the passenger cost C_P (green with rectangles[12]), the operator cost C_O (blue with pentagons[12]), and the objective calculated by the combined optimisation function f_{global} (equation 3) (black with circles[12]) over time. The error bars show the standard deviation between the different runs.

For this network, it can be observed that at the early stages of the search, the passenger objective steadily decreases, while the very early solutions show an increase in the operator cost. However, over the search time, the operator cost drops below its initial values but hovers around a value of 0.95, unlike the passenger cost which continues to drop until the end time of the search for most runs. After 16 hours, the passenger cost reaches on average a value of 0.877.

[13] We used a desktop PC with an Intel i3-4150 3.50GHz Dual Core CPU and 16GB RAM for these experiments.

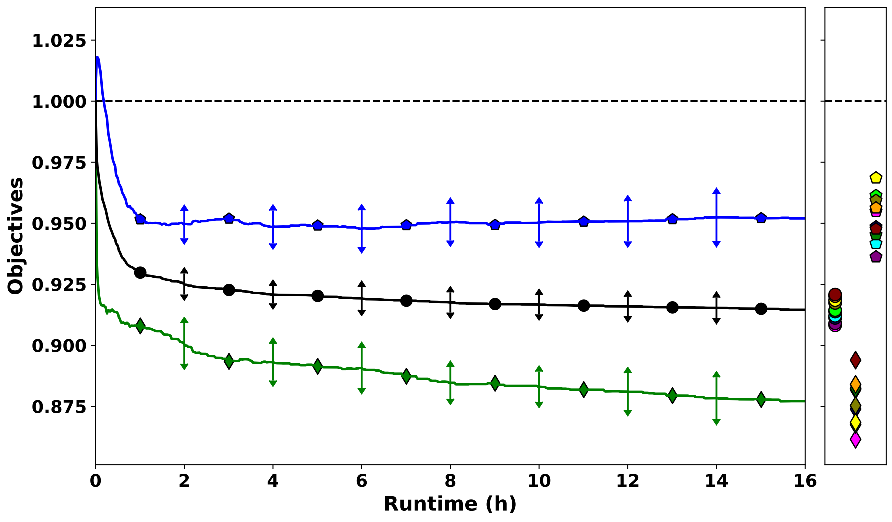


Fig. 7: Results for PuT line global optimisation with SSHH on a city-sized network. Displayed is the development of the normalised average passenger objective C_P (green with rectangles[12]), the operator objective C_O (blue with pentagons[12]) and the combined optimisation function f_{global} (black with circles[12]). The averages were calculated in steps of one minute from ten independent runs. The bars show the standard deviation between the runs. The markers at the right side bar show the distribution of the final values of the individual runs after 16 hours of run time (also shown in table 2). Each marker represents the final value of either f_{global} (circles), C_P (rectangles), or C_O (pentagons) for each run. Each colour uniquely identifies one of the ten experiments.

	1	2	3	4	5	6	7	8	9	10	Avg
f_{global}	0.912	0.917	0.914	0.911	0.919	0.914	0.909	**0.908**	0.921	0.920	0.915
C_P	0.882	0.875	0.868	0.874	0.868	0.882	0.882	**0.862**	0.894	0.884	0.877
C_O	0.942	0.959	0.961	0.949	0.969	0.945	**0.936**	0.955	0.948	0.956	0.952

Table 2: Final values of ten runs with global SSHH optimisation. Values normalised on initial values and minimal values are highlighted in bold.

The right side bar in figure 7 displays the final values of C_P (rectangles), C_O (pentagons), and the combined objective f_{global} (circles) for each of the ten runs. The objective values belonging to one run are given the same marker colours. These values are also shown in table 2. We see that while for the operator cost C_O, the reduction is between 3.1% and 6.4%, for the passenger cost C_P larger reductions between 10.6% and 13.8% are achieved.

Interestingly, the two runs with the highest reduction in C_O and C_P, respectively, are also the two best runs in terms of combined reduction of both objectives. The run reducing C_O by 6.4% reduced C_P by 11.8%, and the run which reduced C_P by 13.8%, also reduced C_O by 4.5%. This shows that improvements in both objectives are not mutually exclusive.

6.3 Application of Local Optimisation

6.3.1 Setup

For testing the local optimisation we used the transport model of the Visum 17 example “Demand NestedLogit” which comes with an implemented mode choice procedure for the mode choice between the PrT and PuT modes. The network model is identical to the one used in section 6.2 and was subjected to the same modifications.

The predefined assignment procedures are split to separate assignments for the morning and evening peak traffic using different demand matrices. However, only the output from the morning peak (with the settings unchanged) is used in our study, to keep the application of our methodology as simple as possible. The applied assignment procedure for PuT is the timetable-based assignment, and vehicle journeys were set to start with an interval of 10 minutes.

As stated in section 5.5.2, the target of the local optimisation is to reduce the load of private cars on a selected number of links. For this purpose, we selected the links with the ID-numbers 178, 186, and 4048, which represent the street Wörmlitzer Straße in the city of Halle. This street is an important connector about 1km south of the main city centre. As bounding factors, we have chosen $\lambda_P = 1.05$ and $\lambda_O = 1.01$[14]. We again used the 16 hours run time. However, due to the more complex sequence of procedures required for the evaluation, the average length of a successful iteration during the experiments was 83.8 seconds, leading to an average number of successful iterations of 765.

6.3.2 Results

Ten independent runs were performed and the results are displayed in figure 8. The data lines show the time development of the average of the global passenger objective C_P (green with rectangles), and the operator objective C_O (blue with pentagons), respectively. The black line with circles shows the average of the local objective C_L. The bars show the standard deviation between the different runs.

It can be seen from the figure that the main objective of these experiments, which is reducing the load of private cars on the selected links has been successfully achieved with an average reduction of C_L to 50.7% from its initial value. We also see a broad spread of solutions as the standard deviation between the results starts to increase with the progression of search time. For the least successful run, C_L was reduced by 30.3% while for the most successful one, the car load on the selected links is reduced by 70.8%, more than two thirds of the initial value.

[14] The values of λ_P and λ_O were chosen arbitrarily. They represent a maximal allowed increase of the operator cost of 1% and an increase of the perceived journey time of 5%.

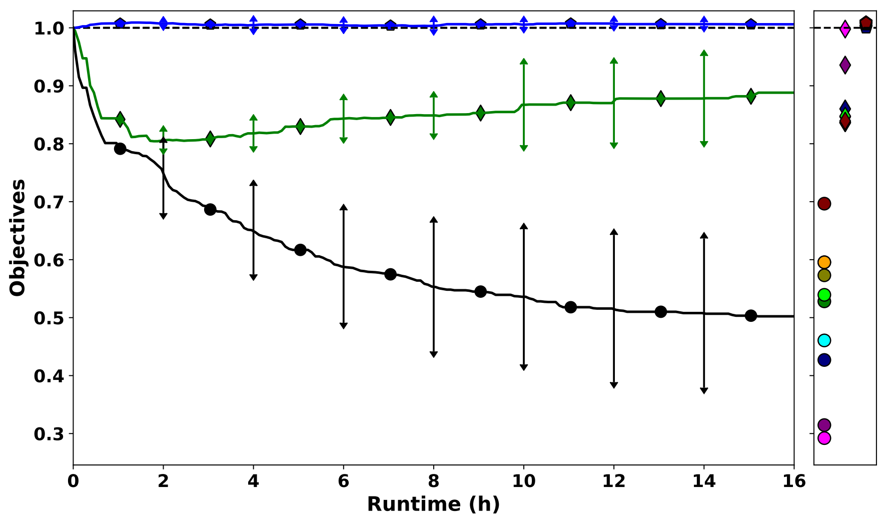


Fig. 8: Results of PuT line local optimisation with SSHH to reduce the number of private cars using a specific street in a city size model. The elements of this figure are similar to the description of figure 7. Here the black line with circles shows the average reduction in the local objective C_L. On the right side the circle markers show the final values of C_L for the individual runs (also shown in table 3).

	1	2	3	4	5	6	7	8	9	10	Avg
C_L	**0.292**	0.426	0.315	0.528	0.461	0.573	0.596	0.539	0.697	0.596	0.502
C_P	0.998	0.860	0.936	0.847	1.049	0.836	0.837	0.848	0.839	**0.833**	0.888
C_O	1.001	**1.00**	1.005	1.005	1.006	1.008	1.01	1.009	1.009	1.008	1.006

Table 3: Final values of ten runs with local optimisation using SSHH. Values are normalised on initial values and the best results are highlighted in bold.

Further, we see that the average value of C_P drops at the beginning of the search, even more than that which occurred in the global optimisation in some cases. This suggests that the initial reductions of C_L are the result of decreasing the average travel time in general for all passengers and in turn increasing the attractiveness of public transport, causing a reduction in the use of private cars. Further reductions of C_L come at the cost of increases in C_P, probably by creating a network that specifically favours passengers who otherwise would use private cars on paths over the selected links. After 16 hours C_P is on average still 13% lower than its initial value. However, there are significant differences in the final values of C_P between the runs. The runs with the most substantial reduction in C_L show the highest final values in C_P. This indicates that some runs developed highly specialised networks which, for this specific task, out-compete other networks with more global improvements. The values of C_O are very similar in all the runs without any significant development. All runs end up with an increase in C_O between a minimum of 0.003% and a maximum of 0.96%.

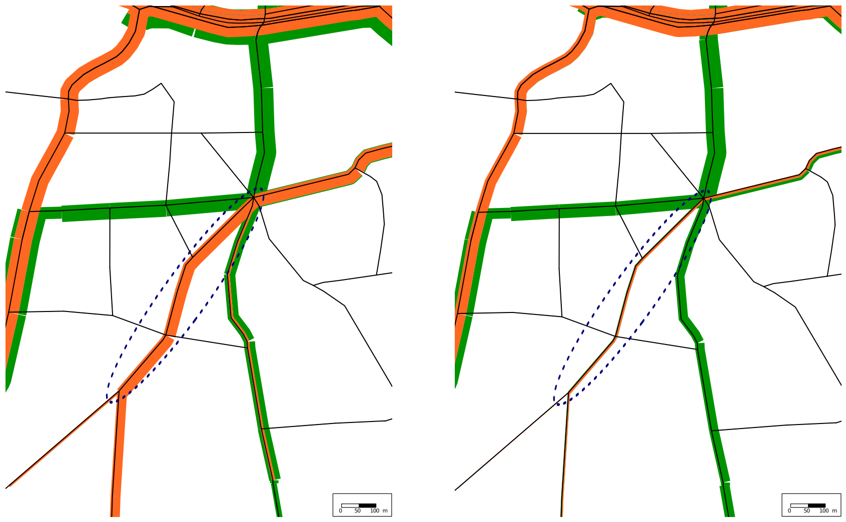


Fig. 9: Parts of the Halle network model before (left) and after (right) the most successful run of the local optimisation experiments. Links are displayed in black and the link loads are displayed by orange bars (PrT load in terms of number of vehicles) and green bars (PuT load in terms of number of passengers) along the links. The thickness of the bars indicates the relative volume of the load. The dashed blue circle marks the links selected for optimisation.

7 Conclusion and Future Work

In this work, we have compared a data structure used by many researchers on the urban transit route optimisation problem (UTRP), and the professional transportation modelling software PTV Visum. Hence, we developed an interface to translate network structures, and other important information between the two models. This interface procedure is then employed in the optimisation of Visum public transport lines using selection hyper-heuristics.

The optimisation was applied using two different optimisation modes: the global optimisation which aims to minimise the overall operator cost and the average passenger travel time, and the local optimisation which aims to reduce the load of private car users on selected street(s). The results of the global optimisation showed the validity of hyper-heuristics in a small as well as a city-sized example network. In both cases high reduction rates in the passenger and operator costs are achieved simultaneously. Additionally, the local optimisation was tested on the city size network, reducing the rate of private car users on the targeted streets by up to 70%.

This work shows the new opportunities arising for planners. The use of optimisation algorithms allow the design of near optimal PuT networks which would not be discovered by more traditional planning methods. The described interface allows the simple application of such algorithms with existing network models and accessing the results in a familiar format. For academic researchers, this work shows the advantages of using professional transport modelling software like PTV Visum in UTRP research, as it allows easy access to a vast pool of powerful evaluation tools. Furthermore, the interface procedures described in this work can easily be used to run other UTRP algorithms using a similar data structure[15]. Variations on the evaluation functions used in this work can easily be implemented to suit specific planning or research scenarios.

For future work, the optimisation capabilities can be extended to include other phases of transit network optimisation (see footnote 1 on page 2), especially the setting of route frequencies. It would be worthwhile to explore and improve the process of optimising routes of multiple modes. Moreover, there are some design aspects in the interface procedures that can be further improved, such as the handling of isolated stop points on directed links. The hyper-heuristic performance can also be improved by adding new low level heuristics, especially interesting would be to include heuristics which add and delete routes[16], to allow for varying numbers of routes.

Although this work is merely a proof of concept, it demonstrates the potential of the full implementation of such an interface in real-world public transport planning. Given the anticipated impact of innovations like connected autonomous vehicles on the public transport landscape, we believe that tools like the one presented here can help planners in necessary adaptations of public transport networks. Therefore, we hope this work will be a stepping stone on the path to a widespread real-world application of network design algorithms to public transportation.

Acknowledgements

We thank PTV for providing us with two academic licences for PTV Visum and especially to Devrim Kara for his long lasting support. Additionally thanks go to Dr. Klaus Nökel, from PTV, and Dr. Anton Marauli, from the engineering office planma, for their very helpful literature suggestions. Further we are thankful for the very helpful comments and suggestions of two anonymous reviewers as well as Dr. Rhyd Lewis, from Cardiff University, and Prof. Dario Landa-Silva, from the University of Nottingham.
This work is part funded by the Leverhulme Programme Grant RP2013-SL-015.

[15] It should be noted that most UTRP algorithms do not use a zone-based demand structure, hindering their applicability on zone-based set ups such as the once used here (see section 3). However, a method to overcome such problems has recently been proposed in [18].

[16] One problem in designing route generation heuristics is to identify optimal beginning and end nodes. An approach to do this is presented in [18]. (Additional discussions on the integration in interface presented here in Chp. 7 of [17])

A Network Modifications

A.1 Small Network

The Visum network model used for the experiments in section 6.1 is a modified version of Visum Version file “430_VisumTutorial.ver”, used in the last stage of Visum17 quick start tutorial [36]. This network model, although loosely based on the small town of Pfullingen, Germany, was built as a pure training exercise. We have modified it to be able to test the different aspects of our interface procedures.

In the original version, all stops have exactly one stop area with one stop point, and there are no one-way streets for buses. In order to properly test if the conversion table (see section 4.2) works correctly, we needed to create a situation where one stop has two stop points, accessible from opposite directions. In order to achieve this we blocked the links representing the street “Friedrichstraße” in the town centre (link numbers: 53167233, 53167376, 53167410, 53167440, 53167475, 53167523, 540910046 and 563879509) for bus travel in a southerly direction and the links representing its parallel street “Seitenstraße” (link numbers: 53167201, 53167240, 53167275, 53167371, 53167383, 53167421, 53167524 and 78579745) for bus travel in a northerly direction. Further, we deleted the stop with the ID 106062573 on the “Friedrichstraße” and connected its stop area and stop point to the stop with the ID 106071832 on the “Seitenstraße”. The transfer time between two stop points was set to 2.5 minutes. The only pre-existing line routes affected by these changes were those belonging to the line “Bus 5”. They were rerouted accordingly.

Additionally, we added some connectors to the network. In the original network, the connectors are placed in such way that all stop areas are necessary to keep all zones connected to the PuT network. As all stop areas have only one stop point, all these stop points are required by the PuT line network in order to have all zones connected to the PuT network. In order to test if the optimiser finds better solutions if it is able to leave out some stop points, we added additional connectors for the Transport System “PUTW” (walking to PuT) between the following zone node pairs:
- Zone 16 and Node 106062573,
- Zone 17 and Node 106062529,
- Zone 24 and Node 106062529,
- Zone 32 and Node 106062573,
- Zone 33 and Node 106063464,
- Zone 37 and Node 106063464,
- Zone 52 and Node 106062293,
- Zone 62 and Node 106061623.

A.2 Halle Network

The transport model we used for the experiments in section 6.2 and section 6.3, are based on two Visum 17 training examples (“3D_Visualization.ver” and “NestedDM_absoluteResult.ver” respectively [35]). Both use the same network model and differ only in the demand data they use and the predefined modelling procedures.

The network model is based on the city of Halle, Germany, with over 200 thousands inhabitants. As our aim is to keep the experiments as close as possible to the real-world example we only made minimal changes to this network: We added the transport system “Walk” (walking to PuT) to connectors between a zone and a node connected to a stop point area. This allows the respective 23 connectors to be used in PuT assignments in addition to PrT assignments. Further, we deleted the bus line “B33”, as it fully overlaps with the bus line “B38”. Finally, we changed the vehicle journeys of all bus services to depart at 10-minute intervals. Vehicle journeys of other PuT modes were left unchanged.

References

1. L. Ahmed, P. Heyken Soares, C. Mumford, and Y. Mao. Optimising bus routes with fixed terminal nodes: comparing hyper-heuristics with nsgaii on realistic transportation networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1102–1110. ACM, 2019.

2. L. Ahmed, C. Mumford, and A. Kheiri. Solving urban transit route design problem using selection hyper-heuristics. *European Journal of Operational Research*, 274(2):545–559, 2019.
3. B. Alt and U. Weidmann. A stochastic multiple area approach for public transport network design. *Public Transport*, 3(1):65–87, 2011.
4. Baaj, M Hadi and Mahmassani, Hani S. Hybrid route generation heuristic algorithm for the design of transit networks. *Transportation Research C*, 3(1):31–50, 1995.
5. S. A. Bagloee and A. A. Ceder. Transit-network design methodology for actual-size road networks. *Transportation Research Part B: Methodological*, 45(10):1787–1804, 2011.
6. M. Bielli, M. Caramia, and P. Carotenuto. Genetic algorithms in bus network optimization. *Transportation Research Part C: Emerging Technologies*, 10(1):19–34, 2002.
7. E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward. A classification of hyper-heuristic approaches: revisited. In *Handbook of metaheuristics*, volume 272, pages 453–477. Springer, 2019.
8. M. Bussieck. *Optimal lines in public rail transport*. PhD thesis, Citeseer, 1998.
9. A. Ceder and N. H. Wilson. Bus network design. *Transportation Research Part B: Methodological*, 20(4):331–344, 1986.
10. S. Chien and P. Schonfeld. Optimization of grid transit system in heterogeneous urban environment. *Journal of Transportation Engineering*, 123(1):28–35, 1997.
11. E. Cipriani, S. Gori, and M. Petrelli. Transit network design: A procedure and an application to a large urban area. *Transportation Research Part C: Emerging Technologies*, 20(1):3–14, 2012.
12. L. Fan and C. L. Mumford. A metaheuristic approach to the urban transit routing problem. *Journal of Heuristics*, 16(3):353–372, 2010.
13. W. Fan and R. B. Machemehl. Using a Simulated Annealing Algorithm to Solve the Transit Route Network Design Problem. *Journal of Transportation Engineering*, 132(2):122–132, 2006.
14. H.-D. Franz. Untersuchung zur Planung von Verkehrsnetzen unter besonderer Berücksichtigung des öffentlichen Personennahverkehrs. *Forschung Straßenbau und Straßenverkehrstechnik*, 182, 1975.
15. M. Friedrich, T. Haupt, and K. Noekel. Planning and Analyzing Transit Networks: An Integrated Approach Regarding Requirements of Passengers and Operators. *Journal of Public Transportation*, 2(4):19–39, 1999.
16. J. Guan, H. Yang, and S. C. Wirasinghe. Simultaneous optimization of transit line configuration and passenger line assignment. *Transportation Research Part B: Methodological*, 40(10):885–902, 2006.
17. P. Heyken Soares. *Three Steps Towards Practical Application of Public Transport Route Optimisation in Urban Areas*. PhD thesis, University of Nottingham, 2020.
18. P. Heyken Soares. Zone-based public transport route optimisation in an urban network. *Public Transport*, 2020.
19. P. Heyken Soares, C. L. Mumford, K. Amponsah, and Y. Mao. An adaptive scaled network for public transport route optimisation. *Public Transport*, 11(2):379–412, 2019.
20. INRO, Montreal, Canada. *Emme 4 User Manual*, 2018.
21. M. P. John, C. L. Mumford, and R. Lewis. An improved multi-objective algorithm for the urban transit routing problem. In C. Blum and G. Ochoa, editors, *Evolutionary Computation in Combinatorial Optimisation*, pages 49–60. Springer Berlin Heidelberg, 2014.
22. A. Kheiri and E. Keedwell. A sequence-based selection hyper-heuristic utilising a hidden markov model. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 417–424. ACM, 2015.
23. A. Kheiri and E. Keedwell. A hidden markov model approach to the problem of heuristic selection in hyper-heuristics with a case study in high school timetabling problems. *Evolutionary computation*, 25(3):473–501, 2017.
24. F. Kılıç and M. Gök. A demand based route generation algorithm for public transit network design. *Computers & Operations Research*, 51:21–29, 2014.
25. A. Marauli. Nachfrageorientierte Verkehrsmodellbasierte ÖPNV-Planung. *TU Graz*, 2011.
26. C. L. Mumford. New heuristic and evolutionary operators for the multi-objective urban transit routing problem. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 939–946. IEEE, 2013.
27. M. A. Nayeem, M. K. Rahman, and M. S. Rahman. Transit network design by genetic algorithm with elitism. *Transportation Research Part C: Emerging Technologies*, 46:30–45, 2014.
28. M. Nikolić and D. Teodorović. Transit network design by bee colony optimization. *Expert Systems with Applications*, 40(15):5945–5955, 2013.
29. M. Nikolić and D. Teodorović. A simultaneous transit network design and frequency setting: Computing with bees. *Expert Systems with Applications*, 41(16):7200–7209, 2014.

30. K. Nökel. Network Design for Public Transport. *PTV AG (not published)*, 2006.
31. J. Pacheco, A. Alvarez, S. Casado, and J. L. González-Velarde. A tabu search approach to an urban transport problem in northern spain. *Computers & Operations Research*, 36(3):967–979, 2009.
32. H. Poorzahedy and F. Safari. An Ant System application to the Bus Network Design Problem: An algorithm and a case study. *Public Transport*, 3(2):165–187, 2011.
33. M. Pternea, K. Kepaptsoglou, and M. G. Karlaftis. Sustainable urban transit network design. *Transportation Research Part A: Policy and Practice*, 77:276–291, 2015.
34. PTV AG, Karlsruhe, Germany. *Introduction to the PTV Visum COM-API*, 2014.
35. PTV AG, Karlsruhe, Germany. *PTV Visum 17 - Overview of examples in the Visum installation*, 2017.
36. PTV AG, Karlsruhe, Germany. *Vision Traffic Suite - Tutorial PTV Visum 17 Quick Start*, 2017.
37. PTV AG, Karlsruhe, Germany. *PTV Visum 17 User Manual*, 2018.
38. L. A. Silman, B. Z., and U. Passy. Planning the route system for urban busses. *Computers & Operations Research*, 1:201–211, 1974.
39. C. Simonis. Optimierung von Omnibuslinien. *Berichte des Instituts für Stadtbauwesen*, (26), 1981.
40. H. Sonntag. *Linienplanung im öffentlichen Personennahverkehr*. PhD thesis, Technical University Berlin., 1977.
41. D. van Oudheusden, S. Ranjithan, and K. Singh. The design of bus route systems — an interactive location-allocation approach. *Transportation*, 14(3):253–270, 1987.
42. S. Walter. Nachfrageorientierte liniennetzoptimierung am beispiel graz (demand orientated line optimisation at the example of graz). Master's thesis, Graz University of Technologie, 2010.
43. B. Yu, Z.-z. Yang, P.-h. Jin, S.-h. Wu, and B.-z. Yao. Transit route network design-maximizing direct and transfer demand density. *Transportation Research Part C*, 22:58–75, 2012.

7 Conclusion

7.1 Summary

This thesis began with an introduction of the Urban Transit Routing Problem (UTRP) as an optimisation problem to find the optimal network of bus routes for a given area. It was further argued that, despite the huge number of publications in this field in recent years, almost none of the results are used in real-world planning processes. After Chapter 2 introduced basic concepts, a review of the available literature on the UTRP in Chapter 3 identified several possible reasons for this:

- Methods to generate the required input datasets struggle when applied to irregular urban networks and do not include urban specific aspects like information on potential terminal nodes.
- Many approaches do not include important aspects for urban applications, also because there are no publicly available instances which include these aspects.
- The absence of an interface between the developed algorithms and transport modelling software.

The Chapters which followed present publications addressing these issues. These are summarised in table 7.1 and will be reviewed in the following. Unless stated

Table 7.1: Summary of papers presented in this paper. Despite column "Chapter", the columns are as in table 3.1.

Year	Chapter	Method	Instance	Term.	Objective	TR
2019	Chp. 4	GA	ui, ui	$\surd$	PT, OC	N
2019	App. A	GA/HH	ui, ui, ui, ui-Chp.4	$\surd$	PT, OC	N
2020	Chp. 5	GA	ui	$\surd$	PT, OC	Z*
submitted	Chp. 6	HH	ui, ui	$\surd$	PT, OC, oth	Zp

otherwise, the reviewed work was conducted by the authors of this thesis. More details on the contributions of the individual authors can be found in the respective sections at the beginning of the publication-chapters

Chapter 4 introduced a new methodology for generating instance datasets for urban areas. The described procedure scales down an urban street network to a manageable size while preserving its characteristics. The process involves a systematic placement of nodes based on the street layout and extracting the travel times between the nodes via GIS software. Information on potential terminal nodes is derived from existing bus routes. Further, a node-based demand matrix is generated from census data. Although the presented application of the method used UK specific data sources, the procedure can be executed using freely available data sources from around the world.

The Chapter further demonstrated an optimisation procedure adapted for the use of restricted terminal nodes. It consists of a new initialisation heuristic, developed by Christine Mumford, and a genetic algorithm, for which the author modified some of the genetic operations. This procedure is applied on two versions generated instance: the full version and a reduced version where all nodes not visited by the real-world bus routes were excluded. The reduced version is necessary to allow a comparison between the performance of the optimisation results and the actual bus routes of the study area. This comparison showed that the optimisation procedure was able to generate several route sets superior to the real-world bus routes in both objectives used.

The concepts introduced in Chapter 4 are also used in the conference paper presented in Appendix A. This conference paper, which is in largely the work of Leena Ahmed, presents a comparison between the genetic algorithm approach used in Chapter 4 and the Selection Hyper-Heuristics approach used in Chapter 6. The results illustrate the superiority of the latter in terms of both run time and optimisation results. The respective experiments were conducted using the reduced instance from Chapter 4, as well as three smaller instances generated with the procedure introduced in Chapter 4.

Chapter 5 expanded the methods introduced in Chapter 4 for the use of a zone-based representation of passenger journeys, as is often used in real-world planning processes. For the instance generation, this extension introduced zones, walking connections between zones, and between zones and nodes. The extended procedure was applied to a subsection of the area used in Chapter 4. This area was selected chosen to demonstrate another addition to the procedure: the inclusion of demand flow over the study area boundary into the demand matrix.

Chapter 5 also introduced a new procedure for calculating journey times between zones. This procedure first calculates a matrix with the PuT transit times (in-vehicle and transfer times) between all node-pairs. Together with the connector matrices, this matrix is then used to identify, for each zone-pair, which combination of nodes results in the shortest overall journey time. Further, the calculated PuT journey times are compared against direct walking connections between the respective zones to model a simple mode choice option. This procedure can further be used to identify the pair of nodes at the beginning and end of the optimal PuT journey between a specific zone pairs. These optimal node-pairs form the basis for most adaptations necessary to adapt the optimisation procedure from Chapter 4 to the zone-based set-up.

This adapted optimisation procedure was applied to the generated instance in several optimisation experiments. These analysed the impact of different weighting factors for walking and waiting times on the optimisation, and proved the ability of the optimisation, procedure to generate optimised results for different set-ups. Using zone-based demand further allowed comparing the optimisation results with the pre-existing routes without the need for a reduced instance as was necessary in Chapter 4. These comparisons showed that the optimisation generated superior solutions for all tested configuration.

Finally, Chapter 6 introduces an interface for data exchange between a UTRP algorithm and the professional transportation modelling software PTV Visum. The Chapter compares the data structure of the Visum network model with the standard data requirements of UTRP algorithms and uses this analysis as the basis for two interface processes. The first generates an undirected graph structure from the directed connections in a Visum network model, and extracts the existing PuT network as an initial solution. The second translates new candidate solutions from undirected lists of nodes to directed lists of stop points and implements them into Visum for evaluation.

The interface processes were employed in an optimisation procedure optimising the PuT routes in a given Visum network model. The algorithm on its core is a sequence-based Selection Hyper Heuristics which was developed by Leena Ahmed. This procedure was applied to two instances taken from Visum training examples, one of them resembling an existing city with 200,000 inhabitants. The objectives were to minimise the operator cost and the average passenger travel time. The results of this process showed that the presented procedure can create solutions with improved performance in both objectives. The best solution reduced both the operator cost and the average passenger travel time by 4.5% and 15.8%, respectively.

An additional optimisation experiment tested utilising modal-choice functions and localised evaluation of results, by optimising the PuT network for the reduction of the number of private cars using a selected street. Results show a reduction by up to 70%. This demonstrated how features like modal-choice functions available in Visum can be utilised for the evaluation. All these results showcase the high potential the application of such an optimisation procedure offers for real-world planning processes.

7.2 Discussion

The instance generation procedure introduced in the Chapters 4 and 5 fulfils the requirements which were listed in Chapter 1: it can be executed with freely available data, includes a data-driven procedure to identify potential terminal nodes and can be applied to urban areas with an irregular layout. Using the classifications from Section 3.2.3 it can be best described as a layout-based generation procedure as node locations are primarily based on street layouts.

The procedure was used to generate seven new public instances (see table 7.2). All include terminal nodes, and two of these are significantly larger than previously published instances. One instance introduced in Chapter 5 features a zone-based travel demand and is, to the best of the author's knowledge, the first such instance to be made publicly available . Not included in table 7.2 are the two Visum network models used in Chapter 6. Nevertheless, the basic forms of these network models are part of every Visum installation and are, therefore, accessible to everyone with a Visum license. The modifications made for the experiments in Chapter 6 are detailed in the Chapters' Appendix and can be easily reproduced.

The fact that the experiments used in Chapter 6 did not require simplifications on the network models demonstrates the interface's ability to deal with the complexity of real-world urban areas. In contrast to approaches used in other publications, the interface presented in Chapter 6 is not restricted to the application of network models with only uses undirected connections. Further, Chapter 6 showed opportunities of

Table 7.2: Publicly available instances introduced in the various publications presented in this thesis. The columns "$|N|$","$|E|$", and "$|T|$" give the numbers of nodes, links, and potential terminal nodes. Column "$|Z^O|/|Z^D|$" gives the number of origin- and destination zones.

Chapter	Year	Name	$\|N\|$	$\|E\|$	$\|U\|$	$\|Z^O\|/\|Z^D\|$
Chp. 4	2019	Nottingham	428	703	168	-
Chp. 4	2019	Nottingham (red.)	376	656	159	-
Chp. 5	2020	South of Trent (zone)	60	94	28	256/64
Chp. 5	2020	South of Trent (node)	60	94	28	-
App. A	2019	South of Trent (red.)	54	86	25	-
App. A	2019	Hucknall (red.)	17	28	10	-
App. A	2019	Clifton (red.)	10	15	7	-

the interface and the adaptability of the employed optimisation procedure in the exemplary reduction of the private vehicle load on a selected street. The described set-up can be easily adapted to more practical planning scenarios such as reducing noise- or air pollution at critical locations. Additionally, the interface also opens up avenues for several research projects branching-out from the definition of the UTRP. Ideas for two such projects can be found in Appendix C.

As the low-level heuristics used in Chapter 6 were relatively simplistic it is worth pointing out that the described interface procedures also allow implementing more complex mutation operations used in Chapter 5 with only minimal adaptations. The key for this is the procedure to identify the node-pair at the beginning and end of the optimal PuT journey between specific pairs of zones which was introduced in Chapter 5-section 3.3. The data required to execute this process are a node-to-node transit time matrix and the connector times. The latter can be easily extracted from the network model, while the former can be generated together with the conversion table described in Chapter 6-section 4.2. This allows, for example, identifying the node-pairs between a new route to be created. These nodes can then be translated into stop points by an extended version of the above-mentioned conversion table[1]. Afterwards the route can be implemented using Visum's inbuilt shortest path procedure. These possibilities demonstrate how the publications presented in this thesis are interlinked, and highlight the importance of considering aspects like zone-based trip representations in the designing of UTRP algorithms.

It can be concluded that the publications included in this thesis open up many opportunities for planners and researchers alike. The results of the various experiments show the high potential the presented algorithms have for real world applications. To realise such applications, planners can either generate the required input data with the procedures described in the Chapters 4 and 5 to generate the required input data, or, where a Visum network model is available, employ the interface processes introduced in Chapter 6. Further, the introduced public instances listed in table 7.2, as well as the Visum interface, allow researchers to more efficiently develop further improved optimisation algorithms for such applications. The presented publications, therefore, form important steps towards practical application of automatic public transport route optimisation.

7.3 Future Work

There are several pathways to take the research presented over the past Chapters further. One focus of this thesis was to include urban specific aspects into the generation of instances and, by extension, the optimisation procedures. Further steps on this path would include adding static public transport modes to the generated instance. For the study areas used in Chapter 4, this would be the two lines of the Nottingham tram network. They would be represented in the same way as bus routes. However, they would stay unchanged throughout the optimisation process. Their representation would require additional links which would be blocked for the bus routes. This can be realised by using different travel time matrices for

[1]For the described process the conversion table from 6-section 4.2 needs to be extended so it not only includes the optimal stop points for all possible node triples but also the stop points to begin and end routes between all possible node pairs. This is easily possible as all the required data is extracted by the algorithm described in 6-section 4.1.

route manipulation of routes and evaluation. Such a set-up was in fact originally planned[2] for Chapter 4.

While Chapter 5 introduced an extension of the instance which allowed to evaluate the passenger objective in a more realistic way, the operator objective remained in a very simplified formulation in all publications. There are several proposals on how PuT operator costs can be represented more realistically (see, e.g. [171, 172]). These require information such as vehicle capacities and fleet composition, which would have to be included in the instances. Such information has to be acquired by the operator. Unfortunately, this task is especially complicated in the UK, where the local bus market is separated into many different private operators with overlapping service areas.

Further, there is a possibility of removing more unnecessary constraints forced by the used optimisation procedures. Of particular interest is the removal of the constraint of a constant number of routes. A straightforward approach to do so would be to divide the mutation operation ”Replace”, used in Chapters 4 and 5, to delete one route and generate a new one, into two operations. Such operations can also be used as low-level heuristics for the Selection Hyper-Heuristic approach discussed in Chapter 6.

Finally, there is the possibility of extending the concepts introduced in this thesis to the remaining phases of the PuT network optimisation mentioned in Section 1.2 (Vehicle Frequency Setting, Timetabling, Vehicle Scheduling and Crew Scheduling). Many such studies already exist, especially on a combination of route design and frequency setting [11, 49]. However, as each of these has its own challenges in

[2]Unfortunately, including the Nottingham Tram Network into the instances presented in Chapter 4 had to be abandoned due to space- and time constraints. However, such a set-up is de-facto realised in Chapter 6 in the experiments using the Halle network model as it includes tram and rail services that were left unchanged during the optimisation.

terms of increasing realism, another thorough literature review would be required to identify possible obstacles for real world-applications.

Appendices

A
Optimising Bus Routes with Fixed Terminal Nodes on Realistic Networks

List of Authors

Leena Ahmed, Philipp Heyken Soares, Christine L. Mumford, Yong Mao

Status by thesis submission

Published in Proceedings of the Genetic and Evolutionary Computation Conference 2019

Authors' contributions

The principle idea for this paper, the comparison of results of NSGAII- and SSHH-optimisation on new instance data sets, was conceptualised by Leena Ahmed and Christine Mumford.

Leena Ahmed adapted the SSHH algorithm to the use with terminal nodes (Section 3) by modifying code from previous research ([166]). She further executed the SSHH experiments. Philipp Heyken Soares generated the instances (Section 4) with the procedures described in Chapter 4 and executed the genetic algorithm experiments with the code available from the work on the same chapter. The analysis and comparison of the results (Section 5) was conducted by Leena Ahmed. Christine Mumford and Yong Mao provided guidance throughout the project.

For most sections of this paper, the initial draft was written by Leena Ahmed. The exception is Section 4, which was initially written by Philipp Heyken Soares, and later modified by Leena Ahmed. After the initial draft, all authors contributed in proofreading and editing of all parts of the manuscript. The Figures 1 and 4 were generated by Leena Ahmed, Figure 2 by Christine Mumford, and Figure 3 by Philipp Heyken Soares.

Summary of content

This conference paper presents the comparison of an optimisation procedure based on sequence-based Selection Hyper-Heuristics (SSHH) and the NSGAII-based optimisation procedure used in chapter 4. The SSHH procedure is based on the one described in [166]; however, modified to work with restricted terminal nodes. It is similar to the procedure used in Chapter 6.

For the comparison, both optimisation procedures are applied to four instances generated with the procedure described in chapter 4. The largest of them is identical to the reduced instance introduced in Chapter 6. The smaller three were generated specifically for this work.

The results of the comparison show that the SSHH optimisation can outperform the genetic algorithm procedure in both the speed and performance of the obtained solutions.

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

Leena Ahmed
Cardiff University
ahmedlh@cardiff.ac.uk

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

Christine Mumford
Cardiff University
mumfordcl@cardiff.ac.uk

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

ABSTRACT

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

CCS CONCEPTS

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

KEYWORDS

Public transport, routing, benchmarks, hyper-heuristics

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

1 INTRODUCTION

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

One of the key elements in public transportation systems planning is the design of routes over a given network to provide an efficient service for passengers and network operators. This problem is referred to as the urban transit routing problem (UTRP). The UTRP is considered an enormous challenge for optimisation algorithms, because of the huge complexity imposed by the multiple constraints which define the criteria for accepting feasible solutions, and the many conflicting objectives that the designed network should satisfy. This makes finding near optimal solutions extremely difficult.

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

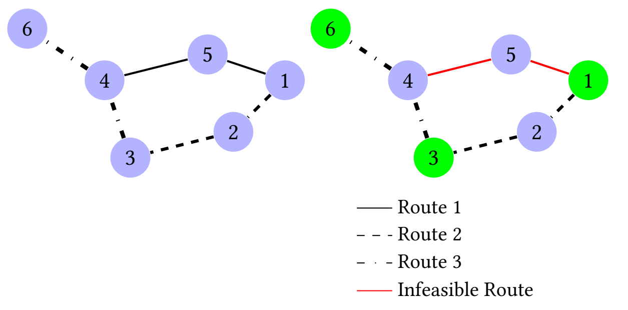


Figure 1: Feasible route network becomes infeasible by introducing three terminal points (green)

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

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

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

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

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

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

2 PROBLEM FORMULATION

We will use a simplified formulation for the UTRP utilised by several previous studies [2, 5, 13, 15, 18], which is the graph representation for a given road network with identified stop locations. The road network comprises a set of stops connected by road segments, this can be mapped into an undirected graph $G = \{V, E\}$, where the graph vertices $V = \{v_1, v_2, \ldots, v_n\}$ are access points (i.e. bus stops), and the graph edges $E = \{e_1, e_2, \ldots, e_m\}$ are direct transport links. Some of the vertices are identified as terminal points $U = \{u_1, u_2, \ldots, u_k\}$, such that $U \subseteq V$. These terminal points allow u-turns to make the reverse trip in the opposite direction. A public transport route, according to this graph definition, is a path in the graph that connects a set of vertices, and starts and finishes at terminal points $r = \{u_i, v_{i_1}, \ldots, v_{i_q}, u_j\}$. The *route network*, which is the solution to this model, results from devising a set of routes $R = \{r_1, r_2, \ldots r_{|N|}\}$ to form the transportation network, where $|N|$ is the total number of routes in the network. To evaluate a given route network, the following information is required for every pair of vertices in the transport network $(v_i, v_j) \in V$: the time to travel between the two vertices, and the number of passengers travelling. Travel time and demand between each pair of vertices is given in the form of two dimensional matrices (i) travel time matrix (ii) demand matrix. t_{v_i,v_j}, a single entry in the travel time matrix refers to the time in minutes required to travel from v_i to v_j and d_{v_i,v_j} in the demand matrix refers to the number of passengers travelling between v_i and v_j. The travel time matrix records travel times between directly connected nodes, with travel times of zero between a node and itself, and ∞ between pairs of nodes that are not directly connected in the graph. On the other hand, the demand matrix records travel demand between pairs of source-destination nodes for the travellers. The two matrices are assumed to be symmetrical for simplicity, meaning that the inbound and outbound journeys along the route will have the same travelling duration. We also assume that the demand level remains the same and does not change for the duration of the day. The terminal points are identified using a one dimensional vector $Ur_{n \times 1}$, where n is the number of vertices in the road network and each entry Ur_i has a value of one if v_i is a valid terminal, or zero otherwise.

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

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

3 OPTIMISATION PROCEDURE

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

3.1 Creating Initial Route Set Using a Heuristic Construction Procedure

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

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

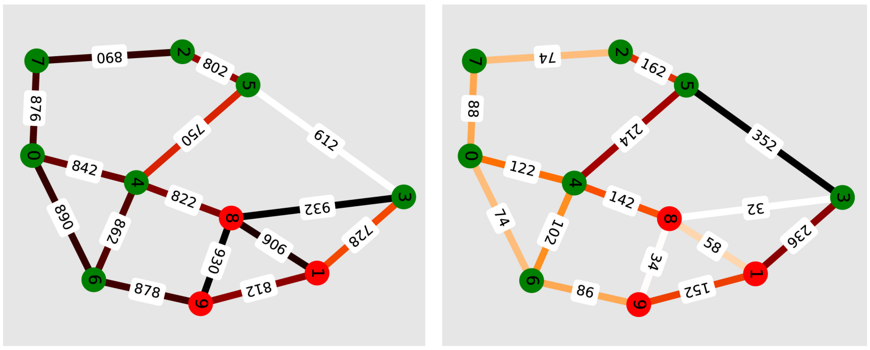
Next we perform a simple transformation on this map to convert the usages into distances so that the largest usage becomes the shortest distance and vice-versa. This is done by subtracting the usage on each edge from some arbitrary large number. We have chosen to use the total demand for the whole network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

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

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

3.2 Objectives and Evaluation

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



(a) Usage map (b) Transformed usage map

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

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

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

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

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

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

3.3 Optimising Route Sets Using Selection Hyper-heuristics

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

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

SSHH is an online selection method based on the hidden Markov model that constructs sequences of heuristics to apply at each decision point. A probabilistic scheme is utilised in this method to increase the chance of choosing successful sequences in later steps, and the selection method maintains and learns these sequences during the search. The great deluge acceptance method uses a threshold value to determine an acceptance range for the solutions. The threshold value equals the initial solution at the beginning of the search and decreases with time in a linear rate. At each step, the threshold value is recalculated using the following formula: $\tau_t = f_0 + \Delta F \times (1 - \frac{t}{T})$ where ΔF is the maximum change in the objective value, f_0 is the final expected objective value, T is the time limit, and t is the time at the current step. Improved solutions are always accepted, while worsening solutions are accepted if their objective value is less than or equal to the calculated threshold value at the current step. The details of the winning algorithm and the analysis are in [2]. This algorithm will be used to optimise the proposed data set and will be known by the name SS-GD in the rest of the paper.

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

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

3.4 The Low Level Operators

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

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

4 NOTTINGHAM INSTANCES

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

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

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

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

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

The travel times associated with the network edges are defined by calculating the shortest paths between pairs of nodes. For every nodes pair, the shortest path is found for each direction of travelling. If these paths are direct, they are averaged, and recorded as the travel time between this pair. If the paths pass by at least one other node, the connection is considered indirect and assigned to the travel time matrix as ∞. The final step is to determine the travel demand between pairs of nodes in the network. For this, travel to work data from 2011 UK census is used. It gives the number of commuters between different census zones, and can be converted into a matrix of passengers travelling between different nodes by assigning zones demand to the network nodes. It is done by assigning the zone demand to a node if the zone's centroid is not more than 400 meters away from the node position. In case that the zone centroid is close to more than one node, the travel demand is divided equally between these nodes. Table 1 summarises the features of the data set. Note that the instances generation procedure ensures producing symmetrical demand and travel time matrices matching the problem description (section 2). The Journey patterns used in generating the real route sets are also modified to satisfy the problem constraints, in order to ensure fair comparison to the optimisation results.

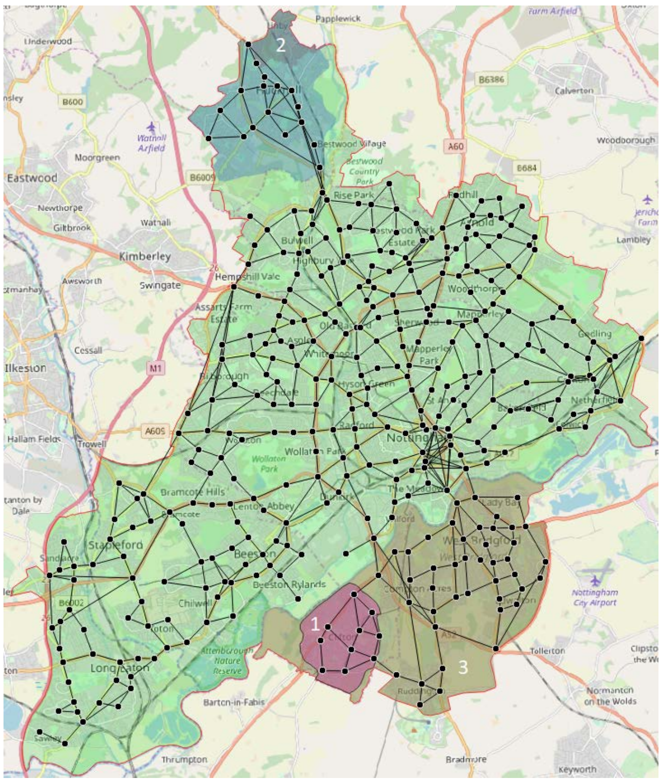


Figure 3: Map of the study area together with nodes and network edges generated with the method described in section 4. (map source: https://www.openstreetmap.org.) The colours and numbers indicate the areas of the instances: 1: Clifton (red), 2: Hucknall (blue), 3: South of Trent (brown) and 4: Nottingham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

Table 1: Features of the data set

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

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

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

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

5 EXPERIMENTAL RESULTS

5.1 NSGAII Optimisation

NSGAII [7] is an elitist non-dominated sorting algorithm used very widely in multi-objective optimisation. The idea is to generate a parent population of size N_{pop} and use it to generate an offspring population of size N_{pop} through crossover and mutation operations. The parent and the offspring populations are combined to produce a population of size $2 \cdot N_{pop}$ from which the population for the next generation is selected by applying non-dominated sorting algorithm and crowding distance and choosing the first N_{pop} solutions of the sorted population. The NSGAII is applied in [13] to solve the UTRP problem in Mandl and Mumford data sets, and in [11] it has been tailored to adapt to the presence of terminal nodes. In this work we have used the algorithm applied in [11] that found preliminary optimised results for Nottingham instance. We will give a brief outline here for the crossover and mutation operators of this algorithm. The crossover operator generates an offspring route set from pairs of parent route sets, where the routes from the two parents are selected alternately such that the proportion of unseen vertices in the offspring is maximised. The generated offspring route set has then a certain chance to undergo mutation. For the mutation, one of the following mutation operators is selected randomly: delete nodes, add nodes, exchange two routes, replace route, merge two routes. These operators are similar to the ones implemented in [13] with some modified to the presence of terminals. Note that the NSGAII is subject to the same constraints described in section 2, and a feasibility test after both crossover and mutation ensures that all the offspring route sets obey these constraints. Repair operators to add the missing nodes, and replace overlapping routes are also implemented to avoid rejecting too many solutions. Further, NSGAII attempts to minimise the same objectives as the hyper-heuristics, using the equations 1 and 2. The parameters for route lengths and numbers of routes in the route set are are also the same as in the hyper-heuristics experiments. We will be comparing the results of NSGAII with SS-GD in the following subsection.

5.2 Comparison of SS-GD and NSGAII

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

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

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

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

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

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

5.3 Comparison with Real World Route Sets

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

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

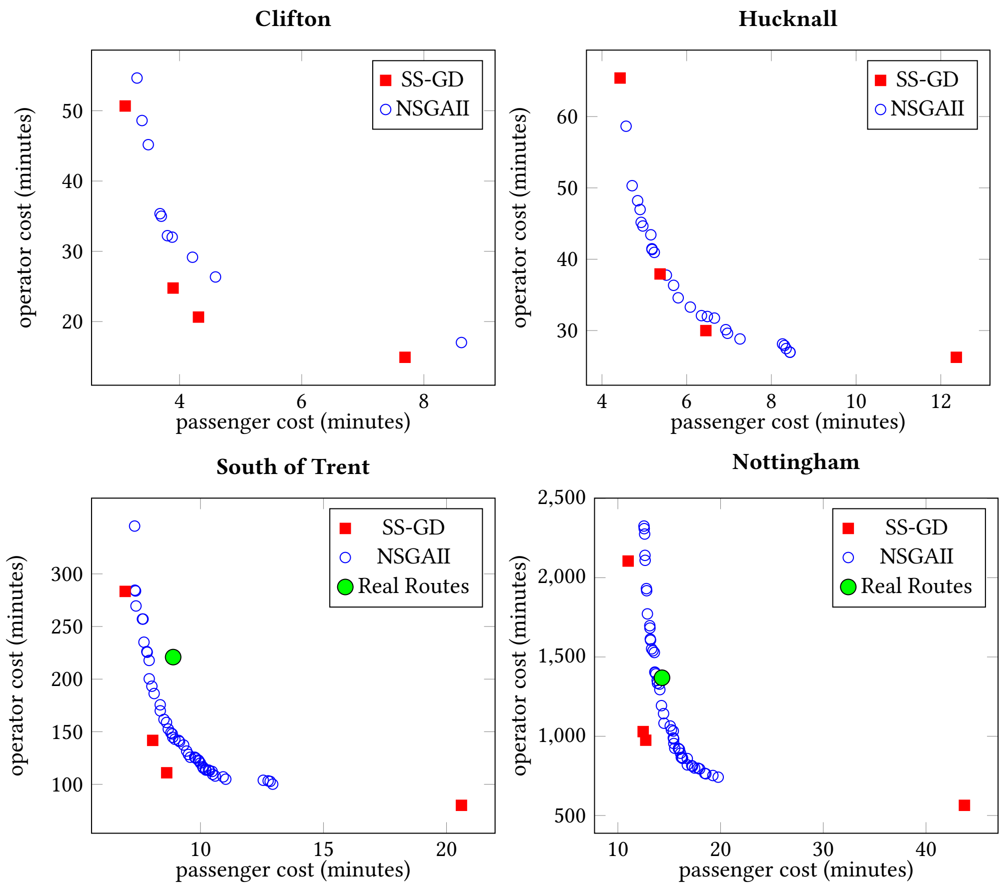


Figure 4: SS-GD results plotted against the evaluation results of the final population. The blue dots are the results of the population evaluation and the red dots are SS-GD results.

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

6 CONCLUSIONS

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

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

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

B
Additional results: Impact of differing route numbers

In all of the optimisation procedures presented in this thesis, the number of routes in a route set was always fixed. This constraint has been rightly criticised as unrealistic by several people, including one of the anonymous reviewers of the publication presented in Chapter 4. On his/her request, a set of experiments was conducted estimating the impact of the number of routes on the optimisation results. Because of the limited time available, these experiments could not be conducted on the relatively large instances introduced in Chapter 4. Instead, the reduced "South of Trent"-instance, introduced in Appendix A, was used, as it was the next smaller instance available at the time. However, since Appendix A was already completed, and Chapter 4 would have required major additions to introduce the smaller instance, these experiments could not be included in either of the two publications. Consequently, they are published here for the first time.

All of the five experiments used the same general set-up as that used in both Chapter 4 and Appendix A: a population of $|P| = 50$ route sets optimised over 200 generations with a crossover probability of $\rho_{cross} = 0.9$. Further, for all the experiments, the minimal and the maximal length of the routes were set to the

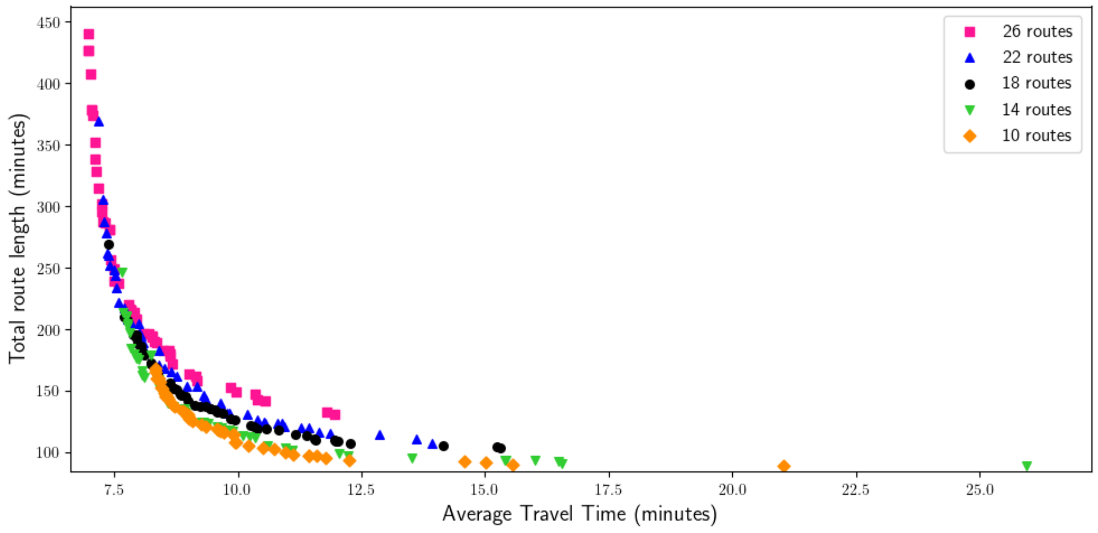


Figure B.1: Evaluation results of optimisation with different numbers of routes. The experiments were conducted on the reduced "South of Trent" instance an optimised 50 route sets over 200 generation. The minimal route length was set to 3, the maximal route length to 12.

values of the real-world route set: $l_{min} = 3$ and $l_{max} = 12$. However, in each experiment, the number of routes per route set was varied. Specifically, experiments with 10, 14, 18, 22, and 26 routes were conducted.

The results are presented in Figure B.1. It shows the approximate Pareto fronts formed by evaluating the result route sets for average travel time and total route length (as in Chapter 4 and Appendix A). The positioning of the fronts reveals that no specific number of routes leads to clearly superior results as no front is fully dominated by another one. Instead, larger number of routes shift the front towards shorter average travel times for higher total route length, while a smaller number have the opposite effect.

The observed behaviour can be explained by the fact that a larger number of routes allows for more transfer options with potentially better connections for passengers. In contrast, a lower number of routes allows to build more minimal networks, which would be otherwise blocked by length or duplication constraints. Consequently, allowing the number of routes per route set to vary is likely to lead to

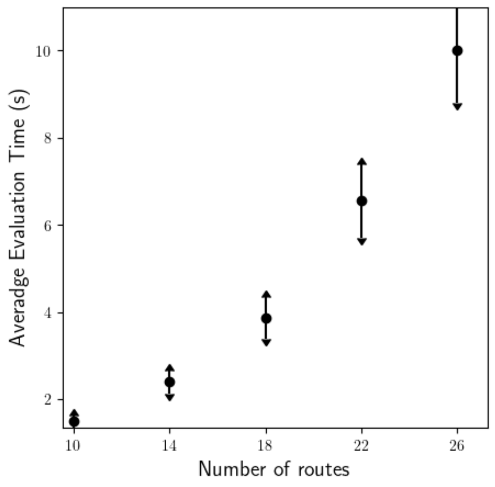


Figure B.2: Average time to required evaluate a population of 50 route sets during the here described optimisation experiments. The error bars give the respective standard deviation.

improved optimisation results, particularly on the extreme ends of the result fronts. As discussed in Section 7.3, relaxing this constraint can be realised by modifying the set of genetic operations.

An additional observation of this set of experiments was the differences in the runtime. As shown in Figure B.2, a linear increase in the number of routes causes an exponential increase in the average time required for evaluation. Again, this can be explained by the higher number of transfer possibilities in route sets with more routes, as these increase the number of node duplications required for the evaluation of the route set (see Chapter 5 - Section 2.2). Consequently, it can be expected that allowing the number of route sets to vary will lead to uncertainties in the run times of the optimisation process.

C Out-branching research

The following sections outline two potential research projects. These branch out of the direct definition of the UTRP but are enabled by the concepts introduced in this thesis, most especially the Visum interface described in chapter 6.

C.1 Coordination between classical PuT and autonomous vehicle fleets.

In the wake of recent advancement in the development of self-driving vehicles, several recent studies have analysed the impact fleets of shared autonomous vehicles (SAVs) could have on urban mobility. The results of these studies suggest that SAVs can significantly reduce the number of regular car trips but would also replace large parts of the public transport system [183, 184]. If driven to the extreme, this can even increase congestion and CO_2 emissions [185]. However, multiple studies also suggest that well-managed coordination between SAV fleets and traditional PuT can have a substantial net positive impact[184–186]. It is clear from these analyses that the advent of SAVs will have a significant impact on future redevelopments of urban PuT networks.

The latest version of Visum (Visum2020) includes an experimental functionality to simulate the coordination of PuT and SAVs for feeder services (first/last-mile concepts) [187]. A combination between this and the interface procedure presented in chapter 6 would allow to develop algorithm for the optimisation of a PuT network for working in coordination with an SAV fleet. The optimisation would, for example, automatically reduce those PuT connections which can best be substituted by SAV feeder services, and establish better connections between often used pick-up points. Such a set up would most likely only require minor changes in the interface and the described optimisation procedure. In return, it would allow for the exploration of many different scenarios and should be of great interest for both planners and researchers.

C.2 Optimising the urban form

Density pattern and structural layout of an urban area, also known as the urban form, have a significant impact on its functioning. In past decades, both empirical- and theoretical studies concluded that increases in urban density reduce the overall transport energy consumption [188]. To some extent, this is due to the effects of origins and destination coming closer together. Further, there is evidence that increased urban density has an additional positive impact on public transport use [189, 190]. However, multiple studies have also pointed out that densification needs to be carefully planned for an urban area to achieve positive impacts [191–193].

One good example of successfully implemented densification is the retrofit for Perth, Australia, after the network city concept[194]. In this concept, multiple points in the city, so-called activity centres, are subject to zoning rules encouraging higher densities. These centres are connected with efficient public transport services, turning them into transit-oriented developments. Such densification concepts are promising, however, are subject to a complicated planning process as the optimal position of the activity centres is hard to determine.

For the available evidence, it is possible to formulate the task of finding the optimal urban form for a given urban area as an optimisation problem. The objective of this optimisation would be to identify the density pattern, which results in the lowest total (transport) energy consumption. A density pattern would be given in the form of residential- and workplace densities in a zonal representation of the study area[1].

As changes in the densities would have a substantial impact on the PuT usage in an area, it is vital to estimate an optimal PuT network for the respective solution as part of the evaluation. A possible evaluation procedure would be as follows:

1. Estimate origin-destination matrices based on residential- and workplace locations.
2. Connect origins and destinations to trips and generate an origin-destination matrix.
3. Optimise PuT network based on calculated origin-destination relations.
4. Determine modes used for trips.
5. Calculate travel times and estimate transport energy use.

Despite point 3, this structure is very close to the standard four-step model used in macroscopic transport modelling [173]. The algorithms necessary to construct a four-step model are well studied and available professional transport modelling software packages, like Visum or EMME [180, 181]. The Visum interface outlined in chapter 6, therefore, forms the missing link to realise the described evaluation procedure[2]. This would then allow to construct heuristic algorithms solving the

[1]As alternative to having the solution define all densities; it would also be possible only to fix some key elements and heaving the rest determined by residential choice modelling these can increase the viability of the generated solutions [195].

[2]Alternatively, it is possible to implement all necessary algorithms all from scratch and represent the transport infrastructure with an instance generated by the procedures outlined in chapters 4 and 5. To incorporate private transport, the instance generation procedure needs to be extended to include road capacities and speed limits. For individual streets, these can be derived from official street classifications, however, have this information has to be aggregated on link level.

optimal urban form problem.

In the long run, it would also be possible to include other aspects effected by the urban form into the evaluation. In addition to the transport system, there is clear evidence of the urban form affecting building energy consumption [196], the urban sub-climate [197], and even impact on the number of social conflicts in an area[198]. To include such aspects when evaluating a proposed urban form, the procedure described above could be coupled with other urban simulations. An earlier project with the participation of the LUCAS has already proven the potential for coupling of transport- and building energy simulations [199]. It would also be possible to extend such a framework to other simulations like urban climate models [200, 201], or social simulations [202].

References

[1] United Nations Department of Economic and Social Affairs. *World urbanization prospects: The 2018 revision.* 2019.

[2] Sanjay K Singh. “Review of Urban Transportation in India”. In: *Journal of Public Transportation* 8.1 (2005), pp. 79–97.

[3] Patrick Miller et al. “Public transportation and sustainability: A review”. In: *KSCE Journal of Civil Engineering* 20.3 (2016), pp. 1076–1083.

[4] UN-Habitat. *State of the World’s Cities Reports 2012/2013.* 2013.

[5] G. Nielsen et al. “HiTrans Best Practice Guide 2: Public transport - Planning the networks”. In: *HiTrans Best Practice Guide* (2005).

[6] Stefan Walter. “Nachfrageorientierte Liniennetzoptimierung am Beispiel Graz (Demand Orientated Line Optimisation at the example of Graz)”. MA thesis. Graz University of Technologie, 2010.

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

[8] Avishau Ceder. *Public Transport Planning and Operation.* 2nd ed. CRC Press - Taylor and Francis Group, 2016.

[9] Fang Zhao and Albert Gan. *Optimization of transit network to minimize transfers.* Tech. rep. Florida International University, Department of Civil and Environmental Engineering, 2003.

[10] Avishai Ceder and Nigel H. M. Wilson. “Bus network design”. In: *Transportation Research B* 20B.4 (1986), pp. 331–344.

[11] O.J. Ibarra-Rojas et al. “Planning , operation , and control of bus transport systems : A literature review”. In: *Transportation Research Part B* 77 (2015), pp. 38–75.

[12] Konstantinos Kepaptsoglou and Matthew Karlaftis. “Transit route network design problem”. In: *Journal of transportation engineering* 135.8 (2009), pp. 491–505.

[13] R. Z. Farahani, E. Miandoabchi, and H. Szeto W. Y.and Rashidi. “A review of urban transportation network design problems”. In: *European Journal of Operational Research* 229.2 (2013), pp. 281–302.

[14] Christina Iliopoulou, Konstantinos Kepaptsoglou, and Eleni Vlahogianni. “Metaheuristics for the transit route network design problem: a review and comparative analysis”. In: *Public Transport* 11.3 (2019), pp. 487–521.

[15] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms.* Vol. 16. John Wiley & Sons, 2001.

[16] Bernard Roy. “Transitivité et connexité”. In: *Comptes Rendus Hebdomadaires Des Seances De L Academie Des Sciences* 249.2 (1959), pp. 216–218.

[17] Stephen Warshall. “A theorem on boolean matrices”. In: *Journal of the ACM*. Citeseer. 1962.

[18] Robert W. Floyd. “Algorithm 97: shortest path.” In: *Communications of the ACM* 5.6 (1962), p. 345.

[19] Edsger W Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271.

[20] William J Cook William H Cunningham and William R Pulleyblank Alexander Schrijver. *Combinatorial optimization*. 1997.

[21] Matthew P. John. “Metaheuristics for Designing Efficient Routes & Schedules For Urban Transportation Networks”. PhD thesis. University of Cardiff, 2016, p. 188.

[22] CB Quak. “Bus line planning”. MA thesis. TU Delft, 2003.

[23] SM Sinha. *Mathematical Programming: Theory and Methods*. Elsevier, 2005.

[24] Michael O Ball. “Heuristics based on mathematical programming”. In: *Surveys in Operations Research and Management Science* 16.1 (2011), pp. 21–38.

[25] Leonora Bianchi et al. “A survey on metaheuristics for stochastic combinatorial optimization”. In: *Natural Computing* 8.2 (2009), pp. 239–287.

[26] Dušan Teodorović. “Swarm intelligence systems for transportation engineering: Principles and applications”. In: *Transportation Research Part C: Emerging Technologies* 16.6 (2008), pp. 651–667.

[27] Edmund K Burke et al. “Hyper-heuristics: A survey of the state of the art”. In: *Journal of the Operational Research Society* 64.12 (2013), pp. 1695–1724.

[28] M. Hadi Baaj and Hani S. Mahmassani. “An AI-Based Approach for Tansit Route System Planning and Design”. In: *Journal of Advanced Transportation* 25 (1991), pp. 187–209.

[29] Y Jiang, W Y Szeto, and T M Ng. “Transit Network Design: a Hybrid Enhanced Artificial Bee Colony Approach and a Case Study”. In: *International Journal of Transportation Science and Technology* 2.3 (2013), pp. 243–260.

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

[31] Hossain Poorzahedy and Farshid Safari. “An Ant System application to the Bus Network Design Problem: An algorithm and a case study”. In: *Public Transport* 3.2 (2011), pp. 165–187.

[32] Wei Fan and Randy B Machemehl. “Optimal Transit Route Network Design Problem with Variable Transit Demand: Genetic Algorithm Approach”. In: *Journal of Transport Engineering* 132.January (2006), pp. 40–51.

[33] Joaquín Pacheco et al. “A tabu search approach to an urban transport problem in northern Spain”. In: *Computers & Operations Research* 36 (2009), pp. 967–979.

[34] Christina Iliopoulou and Ioannis Tassopoulos. “Electric Transit Route Network Design Problem: Model and Application”. In: *Transportation Research Record* 2673.8 (2019), pp. 264–274.

[35] Gabriel Gutiérrez-Jarpa et al. “Multi-objective rapid transit network design with modal competition: The case of Concepción, Chile”. In: *Computers and Operations Research* 78 (2017), pp. 27–43.

[36] Francisco López-Ramos et al. “Integrated approach to network design and frequency setting problem in railway rapid transit systems”. In: *Computers and Operations Research* 80 (2017), pp. 128–146.

[37] S. N. Kuan, H. L. Ong, and K. M. Ng. “Solving the feeder bus network design problem by genetic algorithms and ant colony optimization”. In: *Advances in Engineering Software* 37.6 (2006), pp. 351–359.

[38] Steven Chien, Zhaowei Yang, and Edwin Hou. “Genetic algorithm approach for transit route planning and design”. In: *Journal of transportation engineering* 127.3 (2001), pp. 200–207.

[39] Junhyuk Park and Byung-In Kim. “The school bus routing problem: A review”. In: *European Journal of operational research* 202.2 (2010), pp. 311–319.

[40] A Patz. “Die richtige Auswahl von Verkehrslinien bei großen Straßenbahnnetzen”. In: *Verkehrstechnik* 50 (1925), p. 51.

[41] Héctor Cancela, Antonio Mauttone, and María E Urquhart. “Mathematical programming formulations for transit network design”. In: *Transportation Research Part B* 77 (2015), pp. 17–37.

[42] Ralf Borndörfer, Martin Grötschel, and Marc E. Pfetsch. “A Column-Generation Approach to Line Planning in Public Transport”. In: *Transportation Science* 41.1 (2007).

[43] Gabriel Gutiérrez-Jarpa et al. “Rapid transit network design for optimal cost and origin–destination demand capture”. In: *Computers & Operations Research* 40.12 (2013), pp. 3000–3009.

[44] Hermann Nebelung. *Rationelle Umgestaltung von Straßenbahnnetzen in Großstädten.* Ministerium f. Wirtschaft, Mittelstand u. Verkehr Nordrhein-Westfalen, 1961.

[45] L. A. Silman, Z. Barzily, and U. Passy. “Planning the route system for urban busses”. In: *Computers & Operations Research* 1.2 (1974), pp. 201–211.

[46] Herbert Sonntag. “Ein heuristisches Verfahren zum Entwurf nachfrageorientierter Linienführung im öffentlichen Personennahverkehr”. In: *Zeitschrift für Operations Research* 23.2 (1979), B15–B31.

[47] Baaj, M. Hadi and Mahmassani, Hani S. “Hybrid route generation heuristic algorithm for the design of transit networks”. In: *Transportation Research C* 3.1 (1995), pp. 31–50.

[48] Valérie Guihaire and Jin-kao Hao. “Transit network design and scheduling : A global review”. In: *Transportation Research Part A* 42.10 (2008), pp. 1251–1273.

[49] Francisco López-Ramos. “Integrating network design and frequency setting in public transportation networks: a survey”. In: *SORT* 38.December (2014), pp. 181–214.

[50] Uwe Pape, Yean-Suk Reinecke, and Erwin Reinecke. “Line network planning”. In: *Computer-Aided Transit Scheduling.* Springer, 1995, pp. 1–7.

[51] W Lampkin and PD Saalmans. “The design of routes, service frequencies, and schedules for a municipal bus undertaking: A case study”. In: *Journal of the Operational Research Society* 18.4 (1967), pp. 375–397.

[52] Karl-Heinz Müller. “Ein mathematisches Modell für die Bestimmung von Endknotenzuordnungen in Nahverkehrsnetzen”. PhD thesis. Bergakademie Freiberg, 1967.

[53] Christoph E. Mandl. “Applied network optimization”. In: *Academic Press* (1979).

[54] D Dubois, G Bel, and M Llibre. “A set of methods in transportation network synthesis and analysis”. In: *Journal of the Operational Research Society* 30.9 (1979), pp. 797–808.

[55] BR Marwah, Farokh S Umrigar, and SB Patnaik. “Optimal design of bus routes and frequencies for Ahmedabad”. In: *Transportation Research Record* 994 (1984), pp. 41–47.

[56] DL van Oudheusden, S Ranjithan, and KN Singh. “The design of bus route systems — An interactive location-allocation approach”. In: *Transportation* 14.3 (1987), pp. 253–270.

[57] Rob van Nes, Rudi Hamerslag, and Ben H Immers. “Design of Public Transport Networks”. In: *Transportation Research Board* 1202 (1988), pp. 74–83.

[58] Yihua Xiong and Jerry B Schneider. “Transportation network design using a cumulative genetic algorithm and neural network”. In: *Transportation Research Record* (1992).

[59] Yechezkel Israeli and Avishai Ceder. “Transit route design using scheduling and multiobjective programming techniques”. In: *Computer-aided transit scheduling.* Springer, 1995, pp. 56–75.

[60] Steven Chien and Paul Schonfeld. “Optimization of grid transit system in heterogeneous urban environment”. In: *Journal of Transportation Engineering* 123.1 (1997), pp. 28–35.

[61] S.B. Pattnaik, S. Mohan, and V.M. Tom. “Urban Bus Transit Route Network Design Using Genetic Algorithm”. In: *Journal of Transportation Engineering* 124.4 (1998), pp. 368–375.

[62] Giuseppe Bruno, Gianpaolo Ghiani, and Gennaro Improta. “A multi-modal approach to the location of a rapid transit line”. In: *European Journal of Operational Research* 104.2 (1998), pp. 321–332.

[63] John R Current, Charles S Revelle, and Jared L Cohon. “The median shortest path problem: A multiobjective approach to analyze cost vs. accessibility in the design of transportation networks”. In: *Transportation Science* 21.3 (1987), pp. 188–197.

[64] Michael Bussieck. “Optimal lines in public rail transport”. PhD thesis. Technischen Universitat Braunschweig, 1998.

[65] Mao-Chang Shih, Hani S Mahmassani, and M Hadi Baaj. “Planning and design model for transit route networks with coordinated operations”. In: *Transportation Research Record* 1623.1 (1998), pp. 16–23.

[66] Avishai Ceder and Yechezkel Israeli. “User and operator perspectives in transit network design”. In: *Transportation Research Record* 1623.1 (1998), pp. 3–7.

[67] Maurizio Bielli and Pasquale Carotenuto. “A new approach for transport network design and optimization”. In: *38th Congress of the European Regional Science Association.* 1998.

[68] S. Soehodo and M. Koshi. “Design of public transit network in urban area with elastic demand”. In: *Journal of advanced transportation* 33.3 (1999), pp. 335–369.

[69] Larry J LeBlanc, Edward K Morlok, and William P Pierskalla. “An efficient approach to solving the road network equilibrium traffic assignment problem”. In: *Transportation research* 9.5 (1975), pp. 309–318.

[70] Steven I-Jy Chien and Lazar N Spasovic. “Optimization of grid bus transit systems with elastic demand”. In: *Journal of advanced transportation* 36.1 (2002), pp. 63–91.

[71] Gaetano Fusco, Stefano Gori, and Marco Petrelli. “A heuristic transit network design algorithm for medium size towns”. In: *Proceedings of the 13th mini-euro conference, Bari.* 2002.

[72] S Carrese and S Gori. “An urban bus network design procedure”. In: *Transportation planning.* Springer, 2002, pp. 177–195.

[73] Maurizio Bielli, Massimiliano Caramia, and Pasquale Carotenuto. “Genetic algorithms in bus network optimization”. In: *Transportation Research Part C: Emerging Technologies* 10.1 (2002), pp. 19–34.

[74] Partha Chakroborty and Tathagat Wivedi. “Optimal route network design for transit systems using genetic algorithms”. In: *Engineering optimization* 34.1 (2002), pp. 83–100.

[75] Partha Chakroborty. “Genetic Algorithms for Optimal Urban Transit Network Design”. In: *Computer-Aided Civil and Infrastructure Engineering* 18.3 (2003), pp. 184–200.

[76] V M Tom and S Mohan. “Transit Route Network Design Using Frequency Coded Genetic Algorithm”. In: *Journal of transportation engineering* 129.2 (2003), pp. 186–195.

[77] Somnuk Ngamchai and David J Lovell. “Optimal Time Transfer in Bus Transit Route Network Design Using a Genetic Algorithm”. In: *Journal of transportation engineering* 129.5 (2003), pp. 510–521.

[78] Quentin K Wan and Hong K Lo. “A mixed integer formulation for multiple-route transit network design”. In: *Journal of Mathematical Modelling and Algorithms* 2.4 (2003), pp. 299–308.

[79] M Petrelli. “A transit network design model for urban areas”. In: *WIT Transactions on The Built Environment* 75 (2004), pp. 163–172.

[80] Ziyou Gao, Huijun Sun, and Lian Long Shan. “A continuous equilibrium network design model and algorithm for transit systems”. In: *Transportation Research Part B: Methodological* 38.3 (2004), pp. 235–250.

[81] Joaquin De Cea and Enrique Fernández. “Transit assignment for congested public transport systems: an equilibrium model”. In: *Transportation science* 27.2 (1993), pp. 133–147.

[82] Jitendra Agrawal and Tom V Mathew. “Transit Route Network Design Using Parallel Genetic Algorithm”. In: *Journal of Computing in Civil Engineering* 18.3 (2004), pp. 248–256.

[83] Fang Zhao and Ike Ubaka. “Transit Network Optimization: Minimizing Transfers and Optimizing Route Directness”. In: *Journal of Public Transportation* 7.1 (2004), pp. 63–82.

[84] Jianming Hu et al. “Optimal Design for Urban Mass Transit Network”. In: *International Conference on Natural Computation.* 2005, pp. 1089–1100.

[85] Ernesto Cipriani et al. “A procedure for the solution of the urban bus network design problem with elastic demand”. In: *Advanced OR and AI Methods in Transportation* (2005), pp. 681–685.

[86] Bin Yu et al. “Optimizing bus transit network with parallel ant colony algorithm”. In: *Proceedings of the Eastern Asia Society for Transportation Studies.* Vol. 5. 2005, pp. 374–389.

[87] Fang Zhao, Ike Ubaka, and Albert Gan. “Transit network optimization: Minimizing transfers and maximizing service coverage with an integrated simulated annealing and tabu search method”. In: *Transportation research record* 1923.1 (2005), pp. 180–188.

[88] Young-Jae Lee and Vukan R Vuchic. “Transit network design with variable demand”. In: *Journal of Transportation Engineering* 131.1 (2005), pp. 1–10.

[89] John C Rea. *Designing urban transit systems: an approach to the route-technology selection problem.* Urban Transportation Program, Univ. of Washington, Seattle, 1972.

[90] Bruno Bachelet and Loıc Yon. “Enhancing theoretical optimization solutions by coupling with simulation”. In: *First Open International Conference on Modeling and Simulation (OICMS).* 2005.

[91] F Zhao and X Zeng. “Optimization of transit network layout and headway with a combined genetic algorithm and simulated annealing method”. In: *Engineering Optimization* 38.6 (2006), pp. 701–722.

[92] Wei Fan and Randy B Machemehl. “Optimal transit route network design problem with variable transit demand: genetic algorithm approach”. In: *Journal of transportation engineering* 132.1 (2006), pp. 40–51.

[93] Fang Zhao. “Large-scale transit network optimization by minimizing user cost and transfers”. In: *Journal of Public Transportation* 9.2 (2006), p. 6.

[94] Wei Fan and Randy B Machemehl. “Using a Simulated Annealing Algorithm to Solve the Transit Route Network Design Problem”. In: *Journal of transportation engineering* 132.2 (2006), pp. 122–132.

[95] Ernesto Cipriani, Marco Petrelli, and Gaetano Fusco. “A multimodal transit network design procedure for urban areas”. In: *Advances in Transportation Studies an international Journal* 10 (2006), p. 5.

[96] Fang Zhao and Xiaogang Zeng. “Simulated annealing–genetic algorithm for transit network optimization”. In: *Journal of Computing in Civil Engineering* 20.1 (2006), pp. 57–68.

[97] Bin Yu and Zhongzhen Yang. “Model and algorithm for iterative design of bus network”. In: *Applications of Advanced Technology in Transportation.* 2006, pp. 731–736.

[98] J. F. Guan, Hai Yang, and S. C. Wirasinghe. “Simultaneous optimization of transit line configuration and passenger line assignment”. In: *Transportation Research Part B: Methodological* 40.10 (2006), pp. 885–902.

[99] Alexandre Barra et al. “Solving the transit network design problem with constraint programming”. In: *11th World Conference in Transport Research-WCTR 2007.* 2007.

[100] Fang Zhao and Xiaogang Zeng. “Optimization of user and operator cost for large-scale transit network”. In: *Journal of Transportation Engineering* 133.4 (2007), pp. 240–251.

[101] Zhongzhen Yang, Bin Yu, and Chuntian Cheng. “A Parallel Ant Colony Algorithm for Bus Network Optimization”. In: *Computer-Aided Civil and Infrastructure Engineering* 22 (2007), pp. 44–55.

[102] Ralf Borndörfer, Martin Grötschel, and Marc E Pfetsch. “Models for line planning in public transport”. In: *Computer-aided systems in public transport.* Springer, 2008, pp. 363–378.

[103] J Enrique Fernández L, Joaquin de Cea Ch, R Henry Malbran, et al. “Demand responsive urban public transport system design: Methodology and application”. In: *Transportation Research Part A: Policy and Practice* 42.7 (2008), pp. 951–972.

[104] Fang Zhao and Xiaogang Zeng. “Optimization of transit route network, vehicle headways and timetables for large-scale transit networks”. In: *European Journal of Operational Research* 186.2 (2008), pp. 841–855.

[105] Wei Fan and Randy B Machemehl. “Tabu Search Strategies for the Public Transportation Network Optimizations with Variable Transit Demand”. In: *Computer-Aided Civil and Infrastructure Engineering* 23 (2008), pp. 502–520.

[106] Quentin K Wan and Hong K Lo. “Congested multimodal transit network design”. In: *Public Transport* 1.3 (2009), p. 233.

[107] Ángel Marın and Ricardo Garcıa-Ródenas. “Location of infrastructure in urban railway networks”. In: *Computers & Operations Research* 36.5 (2009), pp. 1461–1477.

[108] Ángel G Marın and Patricia Jaramillo. “Urban rapid transit network design: accelerated Benders decomposition”. In: *Annals of Operations Research* 169.1 (2009), pp. 35–53.

[109] Borja Beltran et al. “Transit network design with allocation of green vehicles: A genetic algorithm approach”. In: *Transportation Research Part C: Emerging Technologies* 17.5 (2009), pp. 475–483.

[110] Antonio Mauttone and Marıa E Urquhart. “A multi-objective metaheuristic approach for the transit network design problem”. In: *Public Transport* 1.4 (2009), pp. 253–273.

[111] Antonio Mauttone and Maria E. Urquhart. “A route set construction algorithm for the transit network design problem”. In: *Computers and Operations Research* 36.8 (2009), pp. 2440–2449.

[112] L. Fan, C. L. Mumford, and D. Evans. “A simple multi-objective optimization algorithm for the urban transit routing problem”. In: *2009 IEEE Congress on Evolutionary Computation.* May 2009, pp. 1–7.

[113] Lang Fan and Christine L. Mumford. “A metaheuristic approach to the urban transit routing problem”. In: *Journal of Heuristics* 16.3 (2010), pp. 353–372.

[114] Jeremy J Blum and Tom V Mathew. “Intelligent Agent Optimization of Urban Bus Transit System Design”. In: *Journal of Computing in Civil Engineering* 25.5 (2010), pp. 357–369.

[115] Wei David Fan and Randy B Machemehl. “Bi-Level Optimization Model for Public Transportation Network Redesign Problem Accounting for Equity Issues”. In: *Transportation Research Record* 2263.1 (2011), pp. 151–162.

[116] Saeed Asadi Bagloee and Avishai Avi Ceder. “Transit-network design methodology for actual-size road networks”. In: *Transportation Research Part B: Methodological* 45.10 (2011), pp. 1787–1804.

[117] Wai Yuen Szeto and Yongzhong Wu. “A simultaneous bus route design and frequency setting problem for Tin Shui Wai, Hong Kong”. In: *European Journal of Operational Research* 209.2 (2011), pp. 141–155.

[118] Bernhard Alt and Ulrich Weidmann. “A stochastic multiple area approach for public transport network design”. In: *Public Transport* 3.1 (2011), pp. 65–87.

[119] Anton Marauli. “Nachfrageorientierte Verkehrsmodellbasierte ÖPNV-Planung”. In: *TU Graz* (2011).

[120] Hiroshi Shimamoto, Jan-Dirk Schmöcker, and Fumitaka Kurauchi. “Optimisation of a bus network configuration and frequency considering the common lines problem”. In: *Journal of Transportation Technologies* 2.03 (2012), p. 220.

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

[122] Wai Yuen Szeto and Yu Jiang. “Hybrid artificial bee colony algorithm for transit network design”. In: *Transportation Research Record* 2284.1 (2012), pp. 47–56.

[123] Ernesto Cipriani, Stefano Gori, and Marco Petrelli. “Transit network design: A procedure and an application to a large urban area”. In: *Transportation Research Part C: Emerging Technologies* 20.1 (2012), pp. 3–14.

[124] Mireia Roca-riu, Miquel Estrada, and César Trapote. “The design of interurban bus networks in city centers”. In: *Transportation Research Part A* 46.8 (2012), pp. 1153–1165.

[125] Bin Yu et al. “Transit route network design-maximizing direct and transfer demand density”. In: *Transportation Research Part C* 22 (2012), pp. 58–75.

[126] Sh Afandizadeh, H Khaksar, and N Kalantari. “Bus fleet optimization using genetic algorithm a case study of Mashhad”. In: *International Journal of Civil Engineering* 11.1 (2013), pp. 43–52.

[127] Chao Chen et al. “B-planner: Planning bidirectional night bus routes using large-scale taxi GPS traces”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.4 (2013), pp. 1451–1465.

[128] Milos Nikolic and Dusan Teodorovic. “Transit network design by Bee Colony Optimization”. In: *Expert Systems with Applications* 40.15 (2013), pp. 5945–5955.

[129] Yadan Yan et al. “Robust optimization model of bus transit network design with stochastic travel time”. In: *Journal of Transportation Engineering* 139.6 (2013), pp. 625–634.

[130] Jose L. Walteros, Andrés L. Medaglia, and Germán Riaño. “Hybrid Algorithm for Route Design on Bus Rapid Transit Systems”. In: *Trasportation Science* 49.1 (2013), pp. 1–19.

[131] J S C Chew, L S Lee, and H V Seow. “Genetic Algorithm for Biobjective Urban Transit Routing Problem”. In: *Journal ofApplied Mathematics* 2013 (2013).

[132] Mahmoud Owais et al. “Simple and Effective Solution Methodology for Transit Network Design Problem”. In: *International Journal of Computer Applications* 89.14 (2014), pp. 32–40.

[133] Fatih Kiliç and Mustafa Gök. “A demand based route generation algorithm for public transit network design”. In: *Computers & Operations Research* 51 (2014), pp. 21–29.

[134] W. Y. Szeto and Y. Jiang. “Transit route and frequency design: Bi-level modeling and hybrid artificial bee colony algorithm approach”. In: *Transportation Research Part B: Methodological* 67 (2014), pp. 235–263.

[135] S M Mahdi Amiripour, Avishai Avi Ceder, and Afshin Shariat Mohaymany. “Hybrid Method for Bus Network Design with High Seasonal Demand Variation”. In: *Journal of Transport Engineering* 140.6 (2014), pp. 1–11.

[136] Ian M Cooper et al. “Optimising large scale public transport network design problems using mixed-mode parallel multi-objective evolutionary algorithms”. In: *Evolutionary Computation (CEC), 2014 IEEE Congress* (2014), pp. 2841–2848.

[137] Matthew P. John, Christine L. Mumford, and Rhyd Lewis. “An Improved Multi-objective Algorithm for the Urban Transit Routing Problem”. In: *Evolutionary Computation in Combinatorial Optimisation*. Ed. by Christian Blum and Gabriela Ochoa. Springer Berlin Heidelberg, 2014, pp. 49–60.

[138] Baozhen Yao et al. “Transit network design based on travel time reliability”. In: *Transportation Research Part C: Emerging Technologies* 43 (2014), pp. 233–248.

[139] S M Mahdi Amiripour, Afshin Shariat Mohaymany, and Avishai Avi Ceder. “Optimal Modification of Urban Bus Network Routes Using a Genetic Algorithm”. In: *Journal of Transportation Engineering* 141.3 (2014), pp. 1–9.

[140] Guang-ming Xu, Feng Shi, and Pu Wang. “Model and Algorithm of Optimizing Bus Transit Network Based on Line Segment Combination”. In: *CICTP 2014: Safe, Smart, and Sustainable Multimodal Transportation Systems*. ASCE 2014, 2014, pp. 1514–1525.

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

[142] Milos Nikolic and Dusan Teodorovic. “A simultaneous transit network design and frequency setting: Computing with bees”. In: *Expert Systems with Applications* 41.16 (2014), pp. 1–10.

[143] Muhammad Ali Nayeem, Md Khaledur Rahman, and M. Sohel Rahman. "Transit network design by genetic algorithm with elitism". In: *Transportation Research Part C: Emerging Technologies* 46 (2014), pp. 30–45.

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

[145] Yang Liu, Ning Zhu, and Shou-feng Ma. "Simultaneous optimization of transit network and public bicycle station network". In: *Journal of Central South University* 22.4 (2015), pp. 1574–1584.

[146] Hang Zhao, Wangtu Ato, and Rong Jiang. "Expert Systems with Applications The Memetic algorithm for the optimization of urban transit network". In: *Expert Systems with Applications* 42.7 (2015), pp. 3760–3773.

[147] Md Khaledur Rahman, Muhammad Ali Nayeem, and M Sohel Rahman. "Transit network design by hybrid guided genetic algorithm with elitism". In: *Proceedings of the 2015 conference on advanced systems for public transport (CASPT), Rotterdam.* 2015.

[148] Moschoula Pternea, Konstantinos Kepaptsoglou, and Matthew G. Karlaftis. "Sustainable urban transit network design". In: *Transportation Research Part A: Policy and Practice* 77 (2015), pp. 276–291.

[149] Ruirui Wu and Shengsheng Wang. "Discrete wolf pack search algorithm based transit network design". In: *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS).* IEEE. 2016, pp. 509–512.

[150] Mahmoud Owais, Mostafa K Osman, and Ghada Moussa. "Multi-objective transit route network design as set covering problem". In: *IEEE Transactions on Intelligent Transportation Systems* 17.3 (2016), pp. 670–679.

[151] Ahmed Tarajo Buba and Lai Soon Lee. "Differential evolution for urban transit routing problem". In: *Journal of Computer and Communications* 4.14 (2016), p. 11.

[152] Luis Cadarso and Ángel Marın. "Improved rapid transit network design model: considering transfer effects". In: *Annals of Operations Research* 258.2 (2017), pp. 547–567.

[153] Amir Khakbaz et al. "Urban bus fleet routing in transportation network equipped with park-and-ride: a case study of". In: *Transport* 32.1 (2017), pp. 55–65.

[154] James C Chu. "Mixed-integer programming model and branch-and-price-and-cut algorithm for urban bus network design and timetabling". In: *Transportation Research Part B: Methodological* 108 (2018), pp. 188–216.

[155] Mahmoud Owais and Mostafa K Osman. "Complete hierarchical multi-objective genetic algorithm for transit network design problem". In: *Expert Systems With Applications* 114 (2018), pp. 143–154.

[156] Di Huang et al. "Multimodal Transit Network Design in a Hub-and- Spoke Network Framework". In: *Transportmetrica A: Transport Science* 0.0 (2018), pp. 1–42.

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

[158] Shashi Bhushan Jha, J K Jha, and Manoj Kumar Tiwari. “Computers & Industrial Engineering A multi-objective meta-heuristic approach for transit network design and frequency setting problem in a bus transit system”. In: *Computers & Industrial Engineering* 130.November 2018 (2019), pp. 166–186.

[159] Kazi Ashik Islam et al. “A heuristic aided Stochastic Beam Search algorithm for solving the transit network design problem”. In: *Swarm and Evolutionary Computation* 46.March 2018 (2019), pp. 154–170.

[160] Myeonghyeon Kim, Seung-Young Kho, and Dong-Kyu Kim. “A Transit Route Network Design Problem Considering Equity”. In: *Sustainability* 11.13 (2019), p. 3527.

[161] Xuesong Feng et al. “A new transit network design study in consideration of transfer time composition”. In: *Transportation Research Part D: Transport and Environment* 66 (2019), pp. 85–94.

[162] Leena Ahmed et al. “Optimising bus routes with fixed terminal nodes: comparing hyper-heuristics with NSGAII on realistic transportation networks”. In: *Proceedings of the Genetic and Evolutionary Computation Conference.* ACM. 2019, pp. 1102–1110.

[163] Pierre-Léo Bourbonnais et al. “Transit network design using a genetic algorithm with integrated road network and disaggregated O–D demand data”. In: *Transportation* (2019), pp. 1–36.

[164] SM Hassan Mahdavi Moghaddam et al. “Simultaneous Bus Transit Route Network and Frequency Setting Search Algorithm”. In: *Journal of Transportation Engineering, Part A: Systems* 145.4 (2019), p. 04019011.

[165] Javier Duran, Lorena Pradenas, and Victor Parada. “Transit network design with pollution minimization”. In: *Public Transport* 11.1 (2019), pp. 189–210.

[166] Leena Ahmed, Christine L. Mumford, and Ahmed Kheiri. “Solving Urban Transit Route Design Problem using Selection Hyper-heuristics”. In: *European Journal of Operational Research* 274.2 (2019), pp. 545–559.

[167] B R Mahwah, Farokh S Umrigar, and S B Patnaik. “Optimal Design of Bus Routes and Frequencies for Ahmedabad”. In: *Transportation Research Record* 994 (1984), pp. 41–47.

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

[169] Hadi Sadrsadat et al. *Bus network design using genetic algorithm.* Tech. rep. University of Maryland, Department of Civil and Environmental Engineering, 2012.

[170] Bernhard Alt. “Investigation of space-time structures in public transport networks and their optimization”. PhD thesis. ETH Zurich, 2010.

[171] Sergio R. Jara-Diaz and Antonio Gschwender. “Towards a general microeconomic model for the operation of public transport”. In: *Transport Reviews* 23.4 (2003), pp. 453–469.

[172] Luigi Moccia, Duncan W. Allen, and Eric C. Bruun. “A technology selection and design model of a semi-rapid transit line”. In: *Public Transport* 10.3 (2018), pp. 455–497.

[173] Michael G McNally. “The four step model”. In: *Handbook of transport modelling* 1 (2000), pp. 35–41.

[174] Jeppe Rich. *Transport Models - From Theory to Practise*. Department of Transport, Technical University of Denmark. Lyngby, Denmark, 2015.

[175] Johannes Schlaich, Udo Heidl, and Peter Möhl. “Multimodal macroscopic transport modelling: State of the Art with a focus on validation & approval”. In: *Proceedings of the 17th IRF World Meeting & Exhibition, Riyadh, Saudi-Arabia.* 2013.

[176] A. G. Wilson. “The Use of Entropy Maximising Models, in the Theory of Trip Distribution, Mode Split and Route Split”. In: *Journal of Transport Economics and Policy* 3.1 (1969), pp. 108–126.

[177] Heinz Spiess and Michael Florian. “Optimal Strategies: a new assignment model for transit networks”. In: *Transportation Research Part B* 23.2 (1989).

[178] Sang Nguyen and Stefano Pallottino. “Equilibrium traffic assignment for large scale transit networks”. In: *European journal of operational research* 37.2 (1988), pp. 176–186.

[179] Julian Dibbelt et al. “Intriguingly simple and fast transit routing”. In: *International Symposium on Experimental Algorithms*. Springer. 2013, pp. 43–54.

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

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

[182] Kalus Nökel. “Network Design for Public Transport”. In: *PTV AG (not published)* (2006).

[183] International Transport Forum. *Urban Mobility System Upgrade - How shared self-driving cars could change city traffic*. Tech. rep. OECD, 2015.

[184] Fábio Duarte and Carlo Ratti. “The Impact of Autonomous Vehicles on Cities: A Review”. In: *Journal of Urban Technology* 25.4 (2018), pp. 3–18.

[185] International Transport Forum. *Shared Mobility Simulations for Helsinki*. Tech. rep. OECD, 2017.

[186] Adriano Alessandrini et al. “Automated Vehicles and the Rethinking of Mobility and Cities”. In: *Transportation Research Procedia* 5 (2015), pp. 145–160.

[187] *PTV Visum 2020 - New Features at a glance*. PTV AG. Karlsruhe, Germany, 2019.

[188] Ruth L Steiner. “Residential Density and Travel Patterns: Review of the Literature”. In: *Transportation Research Record* 1446.2 (1994), pp. 37–43.

[189] Peter W. G. Newman and Jeffrey R. Kenworthy. “Gasoline Consumption and Cities”. In: *Journal of the American Planning Association* 55.1 (1989), pp. 24–37.

[190] Peter Rickwood, Garry Glazebrook, and Glen Searle. “Urban Structure and Energy — A Review”. In: *Urban Policy and Research* 26.1 (2008), pp. 57–81.

[191] Randall Crane. “The Influence of Urban Form on Travel : An Interpretive Review”. In: *Journal of Planning Literature* 15.1 (2000), pp. 3–23.

[192] Roberto Camagni, Maria Cristina, and Paolo Rigamonti. “Urban mobility and urban form: the social and environmental costs of different patterns of urban expansion”. In: *Ecological Economics* 40 (2002), pp. 199–216.

[193] Michael Neuman. “The Compact City Fallacy”. In: *Journal of Planning Education and Research* 25 (2005), pp. 11–26.

[194] Carey Curtis. “Network City: Retrofitting the Perth Metropolitan Region to Facilitate Sustainable Travel”. In: *Urban Policy and Research* 24.2 (2006), pp. 159–180.

[195] Qingxu Huang et al. “A review of urban residential choice models using agent-based modeling”. In: *Environment and Planning B: Planning and Design* 41.4 (2014), pp. 661–689.

[196] Yekang Ko. “Urban Form and Residential Energy Use: A Review of Design Principles and Research Findings”. In: *Journal of Planning Literature* 28.4 (2013), pp. 327–351.

[197] Brian Stone and Michael O. Rodgers. “Urban Form and Thermal Efficiency: How the Design of Cities Influences the Urban Heat Island Effect”. In: *Journal of the American Planning Association* 67.2 (2001), pp. 186–198.

[198] Glen Bramley. “Urban form and social sustainability : the role of density and housing type”. In: 36.1 (2009), pp. 30–49.

[199] M Gargiulo et al. “An integrated planning framework for the development of sustainable and resilient cities – the case of the InSMART project”. In: *Procedia Engineering* 198.September 2016 (2017), pp. 444–453.

[200] A John Arnfield. “Two decades of urban climate research: a review of turbulence, exchanges of energy and water, and the urban heat island”. In: *International Journal of Climatology: a Journal of the Royal Meteorological Society* 23.1 (2003), pp. 1–26.

[201] Sarah Chapman et al. “The impact of urbanization and climate change on urban temperatures: a systematic review”. In: *Landscape Ecology* 32.10 (2017), pp. 1921–1935.

[202] Frédéric Amblard et al. “Analyzing social conflict via computational social simulation: A review of approaches”. In: *Complex Societal Dynamics: Security Challenges and Opportunities*. Ed. by Martinás Katalin, Dario Matika, and Armano Srbljinovic. Vol. 75. IOS Press, 2010, pp. 126–140.