Metaheuristic Methods for the Urban Transit Routing Problem

Thesis submitted to Cardiff University in candidature for the degree of Doctor of Philosophy.

Lang Fan

School of Computer Science
Cardiff University
2009

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate) Date .....

STATEMENT 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of PhD.

Signed ..... (candidate) Date .....

STATEMENT 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ..... (candidate) Date .....

STATEMENT 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organizations.

Signed ..... (candidate) Date .....

ABSTRACT

The urban transit routing problem (UTRP) is NP-Hard and involves devising routes for public transport systems. It is a highly complex multiply constrained problem and the evaluation of candidate route sets can prove both time consuming and challenging, with many potential solutions rejected on the grounds of infeasibility. Due to the problem difficulty, metaheuristic algorithms are highly suitable, yet the success of such methods depends heavily on: 1) the quality of the chosen *representation*, 2) the effectiveness of the *initialization procedures* and 3) the suitability of the chosen *neighbourhood moves*.

In our research, we focus on these three issues, and concentrate on developing a metaheuristic framework for solving the UTRP. Embedding simple metaheuristic algorithms (hill-climbing and simulated annealing) within this framework, we have beaten previously best published results for Mandl's benchmark problem, which is the only generally available data set. Due to the lack of "standard models" for the UTRP, and a shortage of benchmark data it is difficult for researchers to compare their approaches. Thus we introduce a simplified model and implement a data set generation program to produce realistic test data sets much larger than Mandl's problem. Furthermore, some lower bounds and necessary constraints of the UTRP are also researched, which we use to help validate the quality of our results, particularly

those obtained for our new data sets. Finally, a multi-objective optimization algorithm is designed to solve our urban transit routing problem in which the operator's cost is modelled in addition to passenger quality of service.

ACKNOWLEDGEMENTS

I would like to give my great thanks to my supervisor Dr. Christine L. Mumford, who spent much time consulting with me. Thanks for her great enthusiasm and patience with me. Without her invaluable support and encouragement, this thesis would have not been accomplished. I feel lucky to have been inspired by her extraordinary motivation, great intuition and hard work. She is an example for my future career and has my deepest respect professionally and personally.

I would also like to thank my second supervisor Dr. Dafydd Evans, who has given a great help for my research. The thanks also go to Dr. R.M. Whitaker and Professor R.R. Martin who have been my review panel for three years, Dr Li Yu and Professor Huw Williams from School of City and Regional Planning.

Finally, on a personal note, I would like to thank my parents and friends for their continuous support, understanding and encouragement.

PUBLICATIONS

- Lang Fan and Christine L. Mumford. *A Simplified Model of the Urban Transit Routing Problem.* The 7th Metaheuristics International Conference (MIC), Montreal, Canada, 2007.

- Lang Fan and Christine L. Mumford. *A Metaheuristic Approach to the Urban Transit Routing Problem.* Journal of Heuristics, ISSN 1381-1231 (Print) 1572-9397 (Online). Published by Springer Online 22 July 2008.

- Lang Fan, Christine L. Mumford and Dafydd Evans. *A Simple Multi-Objective Optimization Algorithm for the Urban Transit Routing Problem.* IEEE Congress on Evolutionary Computation (CEC), Trondheim, Norway, 2009.

CONTENTS

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Background

Today, the urban transportation system is a key component of the social, economic and physical framework of an urban area. With the development of modern cities, and concerns about pollution and the environment, the design of urban transportation systems has become an urgent problem. Urban transportation systems can be divided into two sub-systems: the public transportation system and the private transportation system [102]. Compared with the public transportation system, the private transportation system has many advantages [73]:

- The road network is much better developed (more nodes, more links) than public transport networks. In particular, fewer nodes (representing small areas within a city) mean that people using the public transport (e.g., buses) usually have to walk more than people using cars.

- Public transport passengers usually have to wait for a vehicle at the beginning of their trip, and may also have to change vehicles part way through their journey, which involves more waiting.

- Travelling by a public transportation system can be rather incon-

venient compared to a comfortable trip in a car.

However, despite the common perception that the private car is the faster, more comfortable and convenient option, there are many negative factors that are increasingly challenging this view. The growth in the number of private cars on our roads has produced more traffic congestion, leading to increased travel times within urban areas. Additionally, more cars produce more air pollution and noise, and lead to higher energy consumption and more accidents. The importance of providing good urban public transport systems is increasingly recognized throughout the world.

Generally, the public transportation system may include various modes of service, with buses, trains and underground or metro services being the best known. Good public transport systems can substantially reduce the negative effects of the private car network: more people can be carried with fewer vehicles, which can reduce fuel consumption, traffic congestion, pollution and accidents. In addition, good public transport systems can invigorate city centres, eliminating traffic jams in pedestrian shopping areas, reducing the need for inner city parking, increasing job opportunities and promoting retail sales. Unfortunately public transport in the UK and some other countries has suffered underfunding for many years. As a consequence, many people are reluctant to give up the comfort and privacy of their own vehicles, and they are prepared to tolerate traffic congestion and parking difficulties rather than use a service they perceive as inconvenient or unreliable. In order to attract more people from their cars, firstly good services need to be provided and secondly, the public will need to be told about them. According to [47] poor information and marketing was partly to blame

for poor patronage following bus deregulation in the UK in 1984.

There is no doubt that bus services form a very important well-established component of a public transportation system. Essential features of good customer service include the operation of frequent reliable services, that minimize waiting and in-vehicle travel times, and avoid the need to change vehicles en route where possible. Ideally, a truly efficient service will satisfy customers needs, while keeping the operator costs in check, such as the total number of buses, bus running distances and operation hours [105].

In the UK, it is usually the bus companies that design the precise bus routes and schedules for an area (a notable exception is London, where Transport for London determines the routes). However, in some other parts of the world (for example China) it is customary for local government to determine the routes and schedules. Clearly, the main goal for a bus company is to maximize its profits. In contrast, local authorities will prioritize the needs of the travelling public. Nevertheless, local transport policies and regulations must also be observed by all [102], and a satisfactory service must be provided, otherwise it will not be used. Furthermore, local authorities are usually accountable to the local community who provide the funding, and may thus find it hard to justify maintaining excessively underutilized routes. In addition, all bus operators have to work within their budgets, and this may impose certain constraints, such as limiting the number of buses available, or the number and lengths of bus routes that can be operated. On the other hand, local authorities have a responsibility to ensure an acceptable level of customer service, and at the same time limit the negative effects of operating buses, e.g. strictly controlling exhaust emissions

from buses. In some parts of the world it is customary for public transit systems run by the private sector to be subsidized, and even in the UK a local authority will occasionally subsidize a service on a non-profit making bus route, to ensure a minimum level of service is maintained even if demand is fairly low (for example, in some rural areas) [92]. Every so often bus routes and schedules will be reviewed. This may be initiated by a bus company or local authority. On occasion, the whole public transport system will be reviewed in an area and costly alterations made, to facilitate the introduction of a new pedestrian precinct, a one way system or to make way for trams or "bendy buses". Most of the time, though, adjustments are likely to be relatively minor, to accommodate minor changes in levels of demand, or comply with new government policies or regulations.

Through our review of transit route and schedule design, it is clear that no common methodology is utilized in practice. As far as we know, in the early years, route networks and schedules were entirely "hand crafted", and most of the development procedures were highly dependent on the transit planners' experience, judgment and knowledge of existing demand patterns, land use and resource constraints [37]. These manual methods cannot solve large network problems efficiently. More recently, commercial software packages such as VISUM, Emme2/3, SATURN (Simulation and Assignment of Traffic to Urban Road Networks) and Cube Voyager have been developed and widely used in the transporting planning industry, mostly as interactive tools for decision support and visualization. Indeed, with the emergence of computing technology, various heuristic and metaheuristic approaches such as genetic algorithms, simulated annealing and tabu search have been de-

veloped to solve urban routing and scheduling problems [73–75], [19], [11–13], [88], [20,21], [106], [38] and [39]. Nevertheless, a good scientific framework is lacking, and researchers have developed their own (often very specialized) models to solve specific variations of the problem, using data that is not generally available. Thus it is very difficult to evaluate the effectiveness of many of the approaches, or make comparisons between them.

1.2 Problem Statement

The problem of designing urban transit routes and schedules, while adhering to practical constraints, is often referred to in the literature as the urban transit network design problem (UTNDP). The two major components of the UTNDP are the urban transit routing problem (UTRP) and the urban transit scheduling problem (UTSP) [20]. At the same time, the UTNDP is an example of a broader class of optimization problems: vehicle routing problems (VRPs) (introduced in Section 2.1).

Generally, the urban transit routing problem (UTRP) involves the development of efficient transit routes (e.g., bus routes) on an existing transit network, with predefined pick-up/drop-off points (e.g., bus stops). On the other hand, the urban transit scheduling problem (UTSP) is charged with assigning the schedules for the passenger carrying vehicles. In practice, the two phases are usually implemented sequentially, with the routes determined in advance of the schedules.

The UTNDP must optimize many criteria in order to efficiently meet the needs of passengers, while at the same time minimizing the costs to the service provider. From a passenger's viewpoint, an ideal

public transport system will provide frequent services and rapid travel times between source and destination, with a minimum of transfers between vehicles on the way. Operators, on the other hand, aim to minimize their costs, yet a low cost option may provide a poor service to the customer. Operator costs usually depend on the fleet size, transit vehicle size, transit vehicle miles, and vehicle operation hours required for a particular route configuration [107]. In addition, there are other stake-holders involved, including national and local government as well as taxpayers and local business. While many parties will benefit from an efficient public transport service, each one will evaluate its service from their own perspective.

1.2.1 Urban Transit Routing Problem (UTRP)

As mentioned above, the urban transit routing problem (UTRP) involves developing a set of routes for an existing urban transit network, following certain constraints. It can be defined as the *physical* design of the UTNDP [69]. In a transit network, adjacent nodes (e.g., bus stops) are linked by an arc or edge, and a route will consist of several nodes connected by edges to form a path. One or more such routes can be combined to form a route set, and when all the routes in a route set are superimposed, this will form a route network. A route network should contain all the nodes, but may not contain all the edges present in the original transit network - i.e., the route network is a subgraph of the original transit network. Ideally, the route network subgraph should be connected, so that there is a path in the route network connecting every node with every other node, in order to satisfy the travel demand. Accurate estimates of travel demand are essential, and a good route set

will ensure that travel requirements with a heavy demand are satisfied, with short travel paths and few vehicle transfers. This will probably be at the expense of less popular travel locations, which may be less well served. Travel demand can be estimated in several ways: for example, by examining current ticket sales, carrying out a survey on the local population, or undertaking a public and private vehicles analysis [14]. On the other hand, such predictions are notoriously difficult in practice, not least because demand is dynamic and highly sensitive to factors such as pricing and quality of service (see Section 1.2.3 for more information). In addition to satisfying customer demand, design guidelines are determined by many additional factors, including the street environment in the local area and the transport management policies of the local government [33].

In our research we will concentrate on the transit routing problem, and adopt the criteria set out by Chakroborty [20] and Yu et al. [10] to define an efficient route set for an urban transit network:

1. The entire transit demand is served, that is, the percentage of unsatisfied demand is zero;

2. A large percentage of transit demand is served through direct connections, that is, the percentage of demand satisfied with zero transfers is high;

3. The average travel time per transit user is as low as possible;

4. High network efficiency, i.e. prioritizing the layout of those transit routes with the highest demand.

At the same time, real world constraints have to be satisfied. It is usual, for example, for planners to set maximum and minimum route

lengths, to aid bus schedule adherence [107]. In addition, in order to simplify the UTRP we will adopt the following constraints in our research:

- A pre-defined number of routes in the route set.
- No cycles or backtracks will be allowed in individual routes.

1.2.2 Urban Transit Scheduling Problem (UTSP)

The urban transit scheduling problem (UTSP) aims to develop schedules for public vehicles (e.g., buses), to travel along predefined routes. More specifically, it involves defining arrival times and departure times at each node on each route in the route set. It can also be defined as the *operational* design of the UTNDP [69]. A good schedule will minimize the time that a passenger has to wait at each node (bus stop) within the operating resource and service constraints. The total waiting time accumulated over all passengers has two components [20]:

1. The total initial waiting time of passengers — the sum of the waiting times of all passengers at their points of origin;
2. The total transfer waiting time — the sum of the transfer times of all transferring passengers.

In addition, the resource and service constraints may include [20]:

- Limited fleet size — only a fixed number of buses are available for operating on the different routes.
- Limited bus capacity — each bus has a finite capacity.
- Maximum and minimum stopping time — buses should not stay at stops for very short or very long times.

- Policy headway — on a given route, a minimum frequency level needs to be maintained.
- Maximum and minimum transfer time — no passenger should have to wait too long for a transfer.

In addition, transfers play a significant role in the transit operations, therefore some research has been carried out on the transit route transfer coordination problem. Transit route transfer coordination forms part of the transit scheduling problem in the daily transit system. Basically it is a function of two decision variables, namely the common headway of the connecting vehicles (i.e., the time between two vehicles passing the same point travelling in the same direction on a given route), and the slack times (bus holding times at the transfer stops) added to their schedule to increase the probability of a successful connection [27]. An increase in the common headway, for instance will result in longer waiting times for non-transfer passengers waiting at the stop and higher passengers volumes and consequently longer stoppage time and longer overall travel time. If the common headway is decreased, the waiting time times will be reduced but the operator costs will increase due to the increase in the operational fleet size. For the slack times, an increase in the slack time will increase the probability of a successful connection and will minimize the transfer time but at the same time will increase the stopping time and the overall travel time of the in-vehicle passengers and will also increase the operator's costs.

1.2.3 Difficulties in Solving the UTNDP

The urban transit network design problem (UTNDP) is a very challenging problem:

1. Due to the need to search for optimal solutions from a large number of possible solutions, the UTNDP is an NP-hard problem. The term "NP-hard" (nondeterministic polynomial-time hard), in computational complexity theory, is a class of problems informally "at least as hard as the hardest problems in NP" [45].

2. There are so many variants on the UTNDP, and no commonly agreed "standard models".

3. Constraints of the UTNDP can be difficult to model and satisfy. For example, in the transit routing problem, the feasibility of the route set (i.e., whether the route network is connected) needs to be ensured, which can involve considerable computation.

4. Different parts of the solution are highly interdependent. For instance, in the UTRP, the transit routes cannot be evaluated in isolation. The performance of a route is dependent on the other routes in the route set. In other words, the entire route set needs to be evaluated as a whole.

5. Many important tradeoffs among conflicting objectives need to be addressed, making the UTNDP an inherently multi-objective problem [37]. For example, minimization of operator costs, maximization of coverage of transit service area and service hours, and minimization of the number of transfers, are objectives that can conflict with each other, since increasing the transit service coverage area or reducing passengers' transfers will increase the operator costs.

6. Accurate data for designing route sets can be difficult to obtain

- particularly travel demand, as previously mentioned. For this reason, designs will be seriously flawed if the data is of poor quality, no matter how good the optimization techniques. In reality, the demand is quite different at every hour of the day, and this can make the problem enormously more complex [31]. In addition, passengers could become confused and dissatisfied with too many changes to travel routes at different times of day.

Ideally one would like to solve the UTNDP in one go, and produce a route network and an associated set of vehicle frequencies simultaneously. In practice, the nature of a route network means that, once established it is much more stable and difficult to change than a vehicle schedule. As mentioned above, travel demand varies considerably at different times of the day, and it is relatively easy to schedule more buses at busy times. According to [10], the level of service requirement is highly sensitive to factors such as passenger flow, weather and road conditions, and needs to be adjusted in accordance with the different situations. Therefore, the quality of the network design may be adversely influenced if transit route network and frequencies are simultaneously optimized. We take the same view and tackle the UTRP as a strategic problem, dealing with averaged demands.

1.3 Research Statement

Through our review of research on the UTNDP, we note that most researchers have focussed on specific real world instances, each one requiring a highly specialized model to comply with specific local regulations and conditions (for example, see [88], [107], [94], [57] and [38]). While good solutions to real-world instances are obviously the ultimate

goal, more generic (often simplified) models are required to gain insight into a problem at a scientific level, and to make it possible to compare different methodologies for optimization. These models are in short supply in the literature. Furthermore, the data used by most researchers is not generally available to others. Only one small network of 15 nodes (Mandl's network, see below) seems have been used by a few researchers to make comparisons.

For the reasons stated above, we believe that establishing a more generic framework for basic research into the UTNDP is timely. We are fortunate that a limited number of researchers, such as Mandl [73–75], Baaj and Mahmassani [12], Kidwai [64] and Chakroborty [21], have already laid the foundations for a more fundamental approach to the problem. In our research we concentrate on the UTRP and build on the work of these earlier authors. Our main research contributions are summarized below:

- A simplified model of the UTRP, building on the work of Mandl etc., but applying a different objective function that explicitly penalizes the number of times a passenger changes vehicles in addition to minimizing the total travel time.

- A basic metaheuristic framework, focussing on four key components of the UTRP: 1) representation, 2) initialization 3) neighborhood moves and 4) feasibility checks. Further, we have implemented hill-climbing and simulated annealing within our framework, and applying this to our simple model we have beaten some previously best published results for Mandl's 15 node problem, using common assessment criteria to compare our results with

others. Our results have been published in an international journal ***(reference to JOH paper)***.

- Software for generating new test instances has been built, matching user requirements, and using the principle of minimum spanning trees to ensure connectedness. From this software, we have generated a new suite of test problems, of various sizes, with realistic properties (informed by our study of some real bus route networks).

- We have demonstrated the scalability of our metaheuristic approach using instances from our test suite.

- We have established good lower bounds to make it possible to effectively assess the quality of the solutions obtained by our metaheuristic approach on our test suite instances.

- Given the ease with which infeasible solutions are generated by "random" methods, especially as the problem size increases, we have demonstrated that our procedures "scale up" by recording the ratio of feasible/infeasible solutions that are generated by our methods.

- In addition, some important relationships are investigated, such as the number of routes in a route set, in relation to the maximum and minimum numbers of nodes in each route.

- Finally, a prototype multi-objective optimization algorithm is implemented, in which the operator's cost and the passenger quality of service are both considered.

We have noted that many key publications in the literature lack important detail regarding vital procedures that they rely on in their optimization processes. For example, in Chakroborty's papers [20, 21] details of his "external procedure" ensuring the feasibility of route sets have not been given. In contrast, we will state the precise details of all the processes we use in our research, to ensure our experiments are repeatable. A further limitation of many previous works is, in our view, their reliance on shortest path algorithms to optimize individual routes (for example, see [88], [94] and [57]). We believe (like Chakroborty [20,21,81]) that it is better to allow a heuristic or metaheuristic method determine the routes, as a shortest path may be inefficient if there is low demand between its terminal vertices.

Throughout our research we will evaluate our work against the following criteria, wherever possible;

- Solution quality - using common assessment criteria to compare our results with previously published results on Mandl's benchmark, and also ensuring that the techniques "scale up" by assessing results for larger data sets against lower bounds.

- Efficiency - we will record run times, and address the scalability issue and the limitations of the current approach in Chapter 6.

- Robustness - To demonstrate reliability, we will carry out 10 replicate runs per experiment, recording average, best and standard deviation. We will also ensure that our main routines "scale up" by recording % feasible/infeasible route sets generated for instances of various sizes.

The main purpose of our research is to explore and better under-

stand the underlying scientific concepts involved in the URTP. For this reason, we simplify the UTRP so that the differences between our model and realistic problems will inevitably exist. Firstly as mentioned before, travel demand is a key variable in the UTRP. However, detailed investigations into measuring and predicting travel demand is an enormously complex research problem in its own right, and is beyond the scope of this thesis. Demand values are all provided for Mandl's instance ***[reference]***, and in the case of our generated instances, we use randomized demand from a uniform distribution, imposing upper and lower bounds. It is worth mentioning that *absolute values* for demand are not important for our model - it is the only the *relative values* that influence the quality of the solution. Clearly temporal variations in demand are important in practice. Yet, bus routes tend to result from strategic planning, with variations in demand largely catered for by scheduling more or less vehicles at different times of the day, rather than implementing a modified route set for busy or slack periods. Nevertheless, our techniques are capable of providing modified route sets if required, simply by rerunning with different demand data.

Another important practical issue is the consideration of the geographical regions in which people live, work and shop, etc.. For example, in some areas city planners stipulate that a bus stop must be positioned where local people reach it within 10 minutes by walking [92]. Basing travel demand on geographical regions is common practice when undertaking surveys of the local population. Clearly, if more realistic travel demand can be provided in our research, better solutions can be obtained using our approaches.

1.4 Organization

This thesis is divided into six chapters. In the present chapter, we have introduced the UTNDP and described some of the difficulties in solving it. We then outlined our research and summarized the motivation for the work. Chapter 2 presents a comprehensive literature review of previous approaches to solve the UTNDP, such as manual approaches, mathematical approaches, and heuristic and metaheuristic methods. Furthermore, we put our research in context with some related problems, including various vehicle routing problems. In addition, we introduce the software tools widely used today in transportation planning. Chapter 3 presents our simple model of the UTRP and describes our data set generation method and our test set of problem instances. Moreover, a lower bound is introduced, against which we later evaluate the results obtained using our metaheuristic approach. Finally, necessary constraints for the UTRP are also discussed. In Chapter 4 we introduce our metaheuristic framework. Hill-climbing and simulated annealing are each tried within the framework and the results compared. Finally, our results are evaluated against state-of-the-art algorithms from the literature. Chapter 5 introduces an improved route set initialization procedure and a simple multi-objective optimization algorithm for the UTRP, including two objectives, namely, passengers' costs and operators' costs. Finally, Chapter 6 summarizes the general conclusions from the study and provides recommendations for future work.

1.5 Summary

This chapter has introduced the background and definition of the urban transit network design problem (UTNDP), and the difficulties of solving it. Furthermore, our research statement has been presented and the structure of the thesis has been described.

Chapter 2

LITERATURE REVIEW

In this chapter we first put the UTNDP in context with other vehicle routing problems. Next, many approaches used to solve the urban transit network design problem (UTNDP) are summarized, focussing particularly on the urban transit routing problem (UTRP). In addition, software tools popularly used in transportation planning are also introduced. Therefore, the literature review will be carried out under six headings:

1. Vehicle routing problems;
2. Manual approaches using service standards and practical guidelines;
3. Mathematical approaches;
4. Heuristic approaches;
5. Metaheuristic approaches;
6. Commercial software packages in transport planning.

First we survey vehicle routing problems and put the UTNDP in context. For the manual approaches, common guidelines adopted in public transport planning are reviewed and summarized. Next, we survey key research on mathematical approaches and heuristic approaches

for the UTNDP, and this is followed by an overview of important meta-heuristic techniques that have been applied to the problem, such as genetic algorithms, simulated annealing and tabu search. In addition, formulations of the UTNDP, objectives and feasible constraints in these works are also reviewed. Finally, commercial software packages popularly used in today's transport planning industry such as VISUM, Emme2/3, SATURN and Cube Voyager are briefly introduced.

2.1 Vehicle Routing Problems

The urban transit routing problem is an example of broader class of problems called vehicle routing problems (VRPs). However, it is not so clearly defined as most other VRPs, and differs insofar as solutions to the UTRP tend to involve long-term strategic planning based on estimated demand, while most other vehicle routing problems are solved on a daily basis to satisfy known demands. Routes for buses and trains often remain unchanged for decades. Logistics companies, on the other hand, may deliver to different customers every day and thus need to travel by different routes. Nevertheless, there is some commonality, as all vehicle routing problems involve determining a set of routes for a fleet of vehicles based at one or more depots for a number of geographically dispersed cities or customers [85]. The main objective of "standard" VRPs is to make deliveries to (or pick ups from) a set of customers with known demands on minimum-cost vehicle routes originating and terminating at one or more depots. The VRP is NP-hard, and many variations exist. Some of these are summarized below [95]:

- The capacitated Vehicle Routing Problem (with or without Time Windows): CVRP or CVRPTW. Goods are delivered to a number

of customers using homogeneous vehicles with limited carrying capacity.

- Multiple Depot Vehicle Routing Problem (MDVRP): The vendor uses many depots to supply the customers.

- Vehicle Routing Problem with Pickup and Delivery (with or without Time Windows) (VRPPD): A number of goods need to be moved from certain pickup locations to other delivery locations. The goal is to find optimal routes for a fleet of vehicles to visit the pick-up and drop-off locations.

- The Dial-a-Ride Problem (DARP) consists of designing vehicle routes and schedules for users who specify pick-up and drop-off requests between origins and destinations. The aim is to plan a set of minimum cost vehicle routes capable of accommodating as many users as possible, under a set of constraints.

- Split Delivery Vehicle Routing Problem (SDVRP): Individual customers may be served by more than one vehicle.

- Arc Routing Problems (ARPs) are a special kind of vehicle routing problem in which the vehicles are constrained to traverse certain arcs, rather than visit certain nodes as in the standard VRP. Typically, the arcs represent streets which require some kind of treatment or service. Examples include the Chinese Postman Problem, the Rural Postman Problem, garbage collection, winter gritting and street cleaning.

- Stochastic Vehicle Routing Problem (SVRP): Some values (like the number of customers, theirs demands, serve time or travel

time) are unpredictable.

Over the years, many techniques have been developed to solve vehicle routing problems. For example, exact approaches (e.g., Branch and Bound [40] and Branch and Cut [15]), heuristics approaches (e.g., these can be seen in the work of Clarke and Wright [23], Gillet and Miller [46] and Fisher and Jaikumar [41], etc.), metaheuristics approaches such as simulated annealing, genetic algorithms and tabu search (e.g., these can be seen in the work of Arbelaitz et al. [79], Czech and Czarnas [26], Jih and Hsu [60], Tan et al. [63], Toth and Vigo [96] and Amberg et al. [5], etc.) and multi-objective approaches (e.g., these can be seen in the work of tan et al. [62] and Saadah and Paechter [87], etc.).

In the current research on the VRPs, some benchmark data are available for researchers, these can be seen in some websites, such as TSPLIB and OR-Library. However, for the UTNDP, Mandl's network seems the only benchmark instance popularly used by researchers.

The Dial-a-Ride Problem (DARP) has some similarities with the UTNDP, insofar as passengers embark at the start of their journey and and alight at the end. However, the DARP operates to fulfil the precise needs of individual travellers on a day to day basis. Because of the unique nature of the UTNDP, specific algorithms designed for other VRPs are not generally useful for UTNDP. However, broad classes of approaches, such as mathematical, heuristic and metaheuristic methods (described above) have been applied equally to the UTNDP and other VRPs, and these are introduced in the following sections.

2.2 Manual Approaches to the UTNDP

For many years transport planners devised reasonable bus route networks and schedules entirely manually, relying on past experience, following practical guidelines and utilizing local knowledge. Regarding service patterns and service levels, important practical guidelines include service area and route coverage, route structure and spacing, route directness, route length, service period, policy headway and road speeds, etc. [80]. A summary of these planning guidelines is presented below:

- Service area and route coverage is usually defined by the local authorities, and the public transport system should serve major employment concentrations, schools and hospitals. Besides, the transit route set has to cover areas of high density population [72].

- For route structure and spacing, the transit routes must fit into major streets and comply with the land use patterns in the local area. On the other hand, the urban development goals of the local government need to be met in the design process [103].

- For route directness, usually routes should avoid circuitous routing and should be not be significantly longer than could be achieved by car [80].

- The length of the route should be as short as possible to serve their markets; excessively long routes should be avoided. Long routes require more travel time because of the difficulty in maintaining reliable schedules [80].

- For the service period, different countries have different criteria.

For example, in the UK buses usually operate between 5 a.m. and 12 p.m. on weekdays and between 7 a.m. and 10 p.m. at weekends [29]. However, in the USA buses tend to operate between 6 a.m. and 12 p.m. on weekdays and between 7 a.m. and 7 p.m. at weekends [80].

- Policy headway is the minimum frequency level that needs to be maintained on a bus route [20].

- For road speeds, they are determined in different areas according to different situations. For example, in the UK bus speed is usually not allowed to exceed 30 mph in urban areas [29], but in the USA bus speed is between 10 mph and 12 mph in urban areas [80].

- In addition, all transit routes need to be designed and operated in a safe manner [7]. The negative effects on the urban environment such as car pollution emission and noise pollution must be controlled [51].

Historically, transit planners have done a reasonable job without the aid of scientific tools or systematic procedures, just using their experience and professional judgement, while adhering to planning guidelines. However, as pointed out in [107], for a really large network it is almost impossible to design an efficient transit route network configuration and bus schedules relying only on past experience and guidelines: in a large urban area the number of bus routes may be over a hundred and the number of bus stops in the thousands, for example. In order to overcome this problem, research efforts have increased in recent decades, coinciding with developments in information and computer technology.

2.3 Mathematical Approaches to the UTNDP

In 1980, Scheele [89] dealt with the UTNDP using mathematical resources. A nonlinear model was proposed with the objective of minimizing the total passenger travel time, including the passenger travel in-vehicle time and the transfer time. At the same time, the passenger trip assignment was solved simultaneously with setting the frequencies. Furth and Wilson [43] in 1982 presented another mathematical method for the UTNDP. The objective was to maximize the network social benefit, consisting of the passenger's benefit (reducing the passenger's travel time) and waiting saving. Constraints were imposed on the fleet size, maximum headway and total budget. The problem was solved through an algorithm using the Kuhn-Tucker conditions on a relaxation of a nonlinear program, where the maximum headway and fleet size constraints were relaxed. The result was an optimal allocation of buses to routes. In a later study, in 1985 Koutsopoulos et al. [53] also proposed a mathematical modelling and resolution method for a simplified UNTDP. Passengers waiting time costs, operating costs and vehicle crowding costs constituted the objective function to minimize.

More recently, in 1995 Constantin and Florian [25] presented a model and resolution method for the UTNDP with the goal to minimize the passengers total expected travel and waiting time under fleet size constraints. A nonlinear nonconvex mixed integer programming model was formulated. A projected sub-gradient algorithm was then used to find optimal bus route frequencies considering the passenger's route choices. In 1998, Bussieck [18] also proposed mathematical models to create routes and frequencies that can more generally be applied to the mass transit system. In the first part, the objective consisted

of maximizing the number of direct passengers under resource-related constraints. In the second part, he sought to minimize the operator's costs with respect to a given level of service and quality. Mathematical programming methods such as relaxation and branch-and-bound were applied in combination with commercial solvers. In a complementary manner, Wan and Lo [99] in 2003 studied the problem of modifying the structure of an existing transit network. A mixed integer formulation was proposed and linearized so it can be solved by commercial solvers on small size instances.

Although some mathematical approaches have been used to solved the UTNDP, they tend to be limited in scope. The UTNDP is NP-hard [107], thus exact methods can be considered only for small instances. Further, as a constrained nonlinear optimization problem, traditional mathematical methods have difficulty solving the UTNDP and have to rely on successive linearizations, which add significantly to the computation burden [20]. In addition, mathematical methods cannot incorporate some external procedure-based declarations in the optimization process [20].

2.4 Heuristic Developments to the UTNDP

Heuristic methods are based upon intelligent search strategies for computer problem solving, using several alternative approaches [61]. Heuristics are typically used when there is no known exact method, or when it is prudent to give up searching for the optimal solution in favour of an improvement in run time [9]. As the UTNDP is an NP-hard combinatorial problem, heuristic methods seem appropriate for solving it.

Despite the enormous practical importance of the UTNDP, very little research appears to have been published prior to 1979. A few papers studied some operational research approaches to very specific instances. For example, Lampkin and Saalmans [68] in 1967 proposed an optimization model to design a transit network in an attempt to transport a maximum number of passengers, given a fixed travel demand matrix. In their approach, they considered trips without transfers first, and then assigned frequencies to the generated set of routes in a second stage. In 1974, Silmman el al. [67] also presented a two-staged approach to minimize the sum of journey times, accumulated over the total demand. However, their model was a little more sophisticated than that of Lampkin and Saalmans and included transfer time between vehicles and incorporated penalties to take account of passengers who could not find seats. Firstly, the candidate route set was established through several repetitions of a route addition and deletion process. Secondly, the frequencies were decided for the route set, constrained by a given number of available buses.

In 1979, the pioneering researcher Christoph Mandl [73–75] started to tackle the problem in a more generic form, and implemented his optimization techniques on a computer. Indeed, his common-sense account of the UTNDP in [73] makes remarkably contemporary reading, despite its early publication date. Mandl concentrated on the UTRP, and developed a solution in two stages: first, a feasible set of routes was generated, and then heuristics were applied to improve the quality of the initial route set. The route generation phase involved first computing shortest paths between all pairs of vertices by Dijkstra's algorithm [30] or Floyd's algorithm [42], and then seeding the route

set with those shortest paths that contained the most nodes, respecting the position of any nodes designated as terminals. Unserved nodes were then iteratively incorporated into routes in the most favourable way, or new routes created with unserved nodes as route terminals. In this first phase, Mandl considered only in-vehicle travel costs when assessing route quality. He went on to suggest several heuristic methods whereby improvements could be made to an initial route set, and used these in his second phase:

- Obtaining new routes by exchanging parts of routes at an intersection node;
- Including a node that is close to a route, if travel demand between this node and the nodes on the route is high;
- Excluding a node from a route that is already served by another route, if the travel demand between this node and the other nodes on the route is low.

In this second phase waiting costs were considered, in addition to in-vehicle travel costs. Waiting times were fixed at constant values, according to specified vehicle frequencies.

The above mentioned shortest path algorithms are very useful components in solving the UTNDP. Dijkstra's algorithm [30] works by visiting vertices in the graph starting with the object's starting point. It then repeatedly examines the closest not-yet-examined vertex, adding it to the set of vertices to be examined. It expands outwards from the starting point until it reaches the goal. Dijkstra's algorithm is guaranteed to find a shortest path from the starting point to the goal, as long as none of the edges have a negative cost. Floyd's algorithm [42] works

by looking for all non-direct paths between two vertices that have a less-expensive total cost than the best way yet found to move between said vertices. If such a path is found, it becomes the value against which future indirect paths between these vertices are tested. In the end, each element of the matrix represents the lowest-cost traversal between the vertices that its row and column represent. According to Van Vliet's [98]: for networks with more than 75 nodes the algorithm of Dijkstra is the fastest for computing shortest paths, while for networks with less than 75 nodes, Floyd's algorithm performs better. As researchers prefer to use methods that scale to solve large problems, Dijkstra's algorithm would appear to be the more suitable for solving the UTNDP.

Furthermore, Dijkstra's shortest path algorithm has also been improved by many researchers and specially tailored for transit route networks. For example, Wang and Li [101] in 2004 and Wang [100] in 2005 proposed best-routing algorithms by integrating a routine for finding the least transfers between two nodes in the network into Dijkstra's shortest path algorithm, in order to find a best path while considering the path length and the number of transfers simultaneously.

In 1981, Hasselstrom [52] tried to design a set of optimal bus routes and frequencies simultaneously. Firstly he employed a complex two-level optimization model to generate routes by assigning desired trips onto a network with all possible transit links, then he used certain criteria to form routes. A direct model was also used to estimate a demand matrix that could provide a service of high quality throughout the area. Note that the disadvantage of the models presented in this work was that although the bus routes and frequencies were determined simulta-

neously, two different optimization problems had to be formulated.

Ceder and Wilson [19] in 1986 and Israeli and Ceder [56] in 1989 published models for simultaneously solving the transit route design and scheduling problems. Appreciating the enormous complexity of real-world problems, they took a modular approach in an attempt to break down the problem into manageable and interrelated components. They considered multiple constraints and multiple objectives. However, their models were not implemented and only the simpler steps were tested on very small instances. More details of these models are given below.

First, the 1986 model [19] focussed on two routines for generating and testing candidate route sets: *Level I* considered only the passenger's viewpoint, and was aimed at minimizing the total travel time, while *Level II* considered both passengers' and operator's viewpoint, and balanced travel time and waiting time with the number of vehicles required. Vehicle frequencies and timetables were also set at *Level II*. The general idea of the route construction algorithms was to start from the terminal nodes having the largest demand and expand the routes incrementally by including more nodes. Ceder with Israeli [56] in their 1989 paper, introduced a much more complex seven-stage system. It included several steps to create routes, identify transfers, and calculate frequencies. Finally, various objectives such as travel time, waiting time, empty space and fleet size were identified as a set of multi-objective tradeoff solutions to be presented to a human decision maker.

More recently Baaj and Mahmassani [11–13] described an artificial intelligence-based solution approach including three major components.

Firstly, they implemented a heuristic route generation algorithm for the route network design. Generally it determined an initial set of skeletons and expanded them to form transit routes, which heavily depend on the travel demand matrix. In this algorithm, the designer's knowledge and experience were also used to reduce the search space. Secondly, an analytic procedure (TRUST) is used to compute an array of network-level, route-level, and node-level descriptors, as well as the frequencies of buses on all routes needed to maintain load factors under a predefined maximum. Thirdly, a route improvement algorithm is used to obtain feasible route networks.

Shih and Mahmassani [77, 90] also proposed a similar approach to that of Baaj, in which an artificial intelligence-based search approach guided by expert knowledge was used to solve the UTNDP. The approach consists of four components: a route generation procedure, a network evaluation procedure, a transit centre selection procedure and a network improvement procedure. Compared with Baaj's work, Shih's work incorporated three additional service concepts including route coordination, variable vehicle size, and a demand responsive service to solve the UTNDP, making the method more practical.

2.5 Metaheuristic Approaches to the UTNDP

A metaheuristic is a heuristic method for solving a very general class of computational problems by combining user-given procedures (usually heuristics themselves) in the hope of obtaining a more efficient or more robust procedure [49]. Some characteristics of metaheuristics are summarized as follows [49]:

- Metaheuristics are strategies that guide the search process.

- Metaheuristic algorithms are usually non-deterministic.
- The basic concepts of metaheuristics permit an abstract level description.
- Metaheuristics are not problem-specific.
- Metaheuristics make use of domain-specific knowledge controlled by the upper level strategy.

Generally, metaheuristics encompass and combine constructive methods (e.g., random, heuristic, adaptive, etc.), local search-based methods (e.g., tabu search, simulated annealing, iterated local search, etc.) and population-based methods (e.g., evolutionary algorithms, ant colony optimization, scatter search, etc.) [16]. At the same time, they have most frequently applied to combinatorial optimization problems and constraint satisfaction problems [58]. Hence metaheuristic approaches such as genetic algorithms, simulated annealing, tabu search and ant colony algorithm have all played important roles in recent research on the UTNDP.

2.5.1 Evolutionary Algorithms for the UTNDP

Genetic algorithms are particularly popular, and several researchers have used them to solve the UTNDP. Genetic algorithms are search algorithms that are based on concepts of natural selection and genetics [54]. A genetic algorithm is a local search algorithm, which starts with an initial collection of strings (a population) representing possible solutions to the problem in hand. Each string of the population is called a chromosome, and has associated with it a value called a fitness function. Offspring are created from members of the initial

and subsequent populations by processes of selection and reproduction, with genetic operators, such as crossover and mutation, ensuring that offspring are similar to yet subtly different from their parents. The purpose of the reproduction operator is to make more copies of fitter (better) individuals in a new population. In the crossover operation, a recombination process creates different individuals in the successive generation by combining material from two individuals of the previous generation [50]. Mutation adds new information in a random way to the genetic search process and ultimately helps to avoid getting trapped at local optima [50]. An individual position in a chromosome is called a gene. The genetic algorithm method differs from most other search methods in that it searches among a population of points and works with codings of a parameter set, rather than with the parameter values themselves [50]. Because of these features, genetic algorithms are being used as general purpose optimization algorithms. The basic structure of the genetic algorithm is illustrated as follows (see Algorithm 1):

Algorithm 1 A Generic Genetic Algorithm (GA)

```
Generate N random strings {N is the population size}
Evaluate and store the fitness of each string
repeat
  for i = 1 to N/2 do
    Select a pair of parents at random {The selection probability is
    in direct proportion to the fitness}
    Apply crossover with probability p_c to produce two offspring
    if no crossover takes place then
      Form two offspring that are exact copies of their parents
    Mutate the two offspring at a rate of p_m at each locus
    Evaluate and store the fitness for the two offspring
  Replace the current population with the new population
until stopping condition satisfied
```

Generally, the most important characteristic of genetic algorithms is the coding of variables that describe the problem efficiently.

In 1998, Pattnaik et al. [88] formulated the UTNDP with fixed transit demand as an optimization problem for minimizing the overall cost, composed of the user cost plus the operator cost. The user cost included components for the in-vehicle time, waiting time and also a transfer penalty, and the operator's cost was determined from the total bus running distance. Feasibility constraints consisted of minimum and maximum length of routes and bus frequencies, maximum load factor, allowable fleet size, etc. In the paper Pattnaik et al. used a genetic algorithm to determine the transit route network and associated frequencies simultaneously, based on their model. Firstly, they used a candidate route set generation algorithm (CRGA) to produce a set of candidate routes. The routes selected by the CRGA were heavily influenced by the demand matrix, the constraints for routes and the shortest path computations (here the Dijakstra's shortest path algorithm was used). Secondly, these candidate routes were listed and labelled. Next, a unique binary number was associated with every route in the list, and a predetermined number of the routes was then selected, at random, to form each chromosome (or string). On the chromosome binary codes for the selected routes were placed end-to-end. Sufficient chromosomes were generated in this way to make up the initial population. Following the generation of the initial population, the objective function of each individual was calculated. Next, the genetic operators - reproduction, crossover and mutation - were applied to find the best route set as the solution.

In 2002, Fusco et al. [44] combined methods developed by Baaj and Mahmassani [13] and Pattnaik et al. [88] and proposed their basic ideas to solve the UTNDP. Generally, three steps were included:

1. A heuristic algorithm to generate a set of feasible routes; for example, composed of the the shortest paths between the node pairs with demand flows larger than a given minimum value.

2. A genetic algorithm to find a good sub-set of routes with associated frequencies. Each bus route selected by the genetic algorithm can be verified according to the impact of the following actions: route extension, route shortening, and route expansion including other nodes based on some criteria.

3. Final improvement of the network configuration. The suitable route modifications are examined in terms of total demand served, network effectiveness and efficiency.

Chakroborty [21] in 2002 introduced his approaches based on a genetic algorithm to design an (optimal) route network for a transit system. First of all, his proposed methodology was presented as follow (see Algorithm 2):

Algorithm 2 Chakroborty's Proposed Algorithm

```
Input road network data, travel time data, and demand matrix
Determine a group of initial route sets using the procedure IRSG
repeat
    Evaluate each route set using the evaluation procedure EVAL
    Modify the group of route sets using the procedure MODIFY in
    order to evolve better route sets
until the optimal route sets are obtained
```

The initial route set generation procedure (IRSG) was used to generate efficient initial route sets. Three steps were included in the IRSG:

- Procedure to select the first node of a route

- Procedure to select any other node of route

- Termination of route and route set generation

In the procedure for selecting the first node of a route, firstly the activity level for each node was calculated; where the activity level refers to the number of transit trips culminating in this node or originating from it. Then, all nodes were listed in descending order of activity levels. Secondly, the user-defined number of nodes were chosen from the list to form the initial node set, INS. Finally, the first node of a route was obtained by randomly selecting a node from INS using a given probability. Once a node was selected it could be removed from the INS.

The procedure to select all other nodes of a route (given the first node has been chosen), is based on making a random choice from a node set called the vicinity node set VNS. Denote the most recent node to be added to a route as the previous node, PN. PN will be initialized as the first (and only) node in the route. A node can qualify as a vicinity node of PN if a single link joins PN to that node, provided that this node has not already been included in the route being currently generated. Then, based on a given probability, the next node of the route was obtained by randomly selecting a node from the VNS. This new node is then itself denoted PN, and the process is repeated. This process will iteratively add more nodes until one of the following stopping conditions becomes true:

1. the number of nodes in the route equals a pre-defined maximum number of nodes;

2. the route length (either in travel time or travel distance units) reaches a pre-defined maximum route length;

3. the cardinality of the VNS falls to zero.

In the evaluation procedure, EVAL, Chakroborty proposed some criteria to evaluate the route set, including the total travel time summed over all passengers, the percentage of satisfied demand with no vehicle transfers and with one and two transfers needed. Finally in the modification procedure, MODIFY, a genetic algorithm was utilized.

In 2003, Chakroborty [20] in his paper systematically introduced the urban transit network design problem, and divided the UTNDP into two components: the urban transit routing problem, UTRP and the urban transit scheduling problem, UTSP. At the same time, the definitions, characteristics, assessment criteria and feasible constraints of the UTRP and UTSP were described in detail (see details in Chapter 1). He also summarized the approaches for solving the UTRP and UTSP respectively based on the genetic algorithm in his previous publication [21, 81, 82]. Finally, he published the results obtained using his methods for the Mandl's network and compared them with other researchers'.

In 2002, Bielli et al. [70] also presented a genetic algorithm to solve the UTNDP. Their goal was to design the best bus network and associated frequencies satisfying both the customer demand and the requirements of operators. In their paper, they used a distinct representation which explicitly stored the frequencies of buses along each bus route, and also an on/off switch to enable or disable the use of that route in the corresponding network.

In papers by Tom and Mohan [94] and Agrawal and Mathew [57], a binary encoding scheme was used which identified candidate routes rather than individual nodes: i.e., a gene represents an entire route

(similar to Pattnaik et al. [88]). In this way candidate routes can be pre-determined and stored in a list, and it is the job of the genetic algorithm to select routes from this list to make up a route set. In general, their initial candidate route sets were produced using heuristic procedures, applying shortest path calculations moderated by user-defined guidelines. The genetic operators, mutation and crossover, produced new route set variations for selection, giving the population scope to improve over time, provided selection was biased towards saving the better solutions over the poorer ones. When the entity for encoding is an entire route, it is important that similar routes should be identified by similar binary codes, so that a simple mutation to a binary code for a particular route, for example, will tend to produce a mutated route with many nodes in common with its parent. Frequencies were also encoded as part of the chromosome in their methods.

2.5.2 Other Metaheuristic Approaches to the UTNDP

Though genetic algorithms seem to predominate in the literature, other metaheuristic approaches have been tried by some researchers. In 2004 and 2006, Zhao and Ubaka [106, 108] attempted the optimization of large-scale transit route networks. The objectives they used were to minimize the transfers and optimize route directness, while maximizing service coverage. For the constraints, the number of bus routes, the route lengths and the number of transit stops on individual routes were predetermined. For the solution search schemes, in 2004 Zhao and Ubaka [108] utilized a hill-climbing algorithm, and in 2006 they [106] used a simulated annealing algorithm. These two approaches are outlined below.

Hill-climbing

Hill-climbing is a mathematical optimization technique which belongs to the family of local search. It is relatively simple to implement, making it a popular first choice. Although more advanced algorithms may give better results, in some situations hill-climbing works just as well [86].

Generally, a hill-climbing algorithm begins with one initial solution to the problem at hand, usually chosen at random. This initial solution is then subjected to a neighbourhood move (mutation), and if the new solution is better than the current one, the new solution is kept; otherwise, the previous solution is retained. This process is then repeated until no neighbourhood move can be found that will improve the solution quality of the current solution, and this solution is returned as the result [59]. Hill-climbing are very good in finding local optima. However, difficulties arise when the global optima is different from the local optima. Since all the immediate neighbouring points around a local optima are worse than it in the performance value, local search can not proceed once trapped in a local optima point. Hence it is necessary to find some mechanism that can help us escape the trap of local optima.

Simulated Annealing

In contrast to hill-climbing, simulated annealing has the capability to escape local optima. The name of simulated annealing origins from the simulation of annealing process of heated solids. In simulated annealing [91], the concept of "temperature" is added. This is a global numerical quantity which gradually decreases over time. At each step of the algorithm, a neighbourhood move is used to generate a new solution. The solution quality of the new result is then compared with

the previous solution; if it is better, the new solution is kept and the old one discarded, in the same way as in hill-climbing. Otherwise, the algorithm makes a decision whether to keep or discard the new solution based on temperature and probability. If the temperature is high, as it is initially, even changes that cause significant decreases in solution quality may be kept and used as the basis for the next round of the algorithm, but as temperature decreases, the algorithm becomes more and more inclined to only accept improving changes. Finally, the temperature reaches zero and the system "freezes"; whatever configuration it is in at that point becomes the solution. As can be seen, simulated annealing consists of:

1. An initial value for the temperature T;
2. a cooling function (such as $T = T\alpha, 0 < \alpha < 1$);
3. a predefined number of iterations to be performed at each temperature;
4. a stopping criterion to terminate the algorithm.

The basic structure of the simulated annealing will be introduced in Section 4.2.

In addition to the work of Zhao and Ubaka [106], in 2006, Fan and Machemehl [38] also used a simulated annealing algorithm to solve their UTNDP. Similar to the work of Pattnaik et al. [88], they firstly generated candidate routes by finding all shortest paths for each node pair using the Dijkstra's shortest path algorithm and the Yen's k-shortest path algorithm [104], and checking them against the constraints, e.g. the minimum and maximum permitted route length. Then all candidate routes were kept and labelled. Finally the simulated annealing

algorithm was implemented to find the best route set from these candidate routes. On the other hand, in the paper it was claimed that the cost of unsatisfied demand is included in an objective function for the first time also including the sum of user cost and operator cost for the UTNDP. The user costs consisted of four components, including walking cost, waiting cost, transfer cost, and in-vehicle travel cost. The operator costs referred to the cost of operating the required buses. At the same time, some feasibility constraints such as the headway, load factor constraints, fleet size, trip length constraints, maximum number of routes constraint, and maximum allowed unsatisfied demand also needed to be satisfied in their model.

Tabu Search

Algorithm 3 Tabu Search Algorithm [6]

Set $k = 1$ and generate initial solution s
repeat
  Identify $N(s)$ (neighborhood set of solution s)
  Identify $T(s, k)$ (tabu set of solution s)
  Identify $A(s, k)$ (aspirant set of solution s)
  Choose the best solution $s*$ from $N(s, k) = \{N(s)$ - $T(s, k)\} + A(s, k)$
  Memorize $s*$ if it improves the previous best known solution
  $s = s*$
  $k = k + 1$
until a stopping condition is satisfied

In 1986, Glover [48] proposed a new approach, which he called tabu search, to allow local search methods to overcome local optima. The general ingredients of tabu search include [34]:

- A neighborhood is constructed to identify adjacent solutions that can be reached from the current solution.

- A tabu list records forbidden moves, which are referred to as tabu

moves.

- Tabu restrictions are subject to an important exception. When a tabu move has a sufficiently attractive evaluation where it would result in a solution better than any visited so far, then its tabu classification may be overridden. A condition that allows such an override to occur is called an aspiration criterion .

The basic structure of tabu search is shown in Algorithm 3. Recently, in 2008, Fan and Machemehl [39] also implemented a tabu search algorithm to solve their UTNDP.

Ant Colony Algorithm

Ant colony optimization, ACO, introduced by Dorigo [32] in 1992, is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. The principles of this algorithm can be illustrated by examining the food searching process of an ant colony. Along their way from the food source to the nest, ants communicate with each other with pheromone (a chemical substance). As the ants move, a certain amount of pheromone is deposited on the ground along the path they follow. Then the ants determine their movements by judging the density of the chemical substance on a path. The process can be described as loop of positive information feedback, in which the more ants that follow a given trail, the more pheromone is deposited on that trail, and the larger the probability that this trail will be followed by other ants.

Some researchers have used this algorithm to solve the UTRP (e.g. the work of Yu et al. [10]). In the paper by Yu et al., firstly they proposed some criteria to define an efficient transit network, such as reachability, low transfer rate, short travel time, and high network ef-

ficiency. Secondly, both the length of the route and the corresponding travel demand were taken into account as objectives, i.e. minimum transfer rate and maximum travel demand per length served. Finally their ant colony optimization procedures were introduced.

2.5.3 Limitations and Gaps in Previous Research

From the above literature review, it is clear that most heuristic and metaheuristic approaches rely very heavily on the use of standard shortest path algorithms to generate individual candidate routes. Most methods then make selections from this initial pool of shortest paths, to build their route sets, perhaps iteratively making some minor adjustments to some of the routes, to improve the overall solution or to ensure connectivity of the route network. We are not convinced, however, that building route sets from pre-computed shortest paths produces the best route sets in practice, thus we do not use this technique in our work. (Indeed we have beaten some best published results in this way). Depending on the pattern of travel demand, we believe that longer travel paths may be appropriate between some sources and destinations where travel demand is low, in the interests of efficiency. It is the quality of the route set as a whole that is important, rather than that of the individual routes.

Another problem is that many papers have not fully described their methodology for representation, initialization or neighbourhood moves. In particular, authors have skated over the important issue of ensuring route set feasibility. Chakroborty's [21], for example, simply refers to an "external procedure" for ensuring feasibility, without giving any details. In our work we describe all our routines and data structures in detail,

to ensure that our experiments are repeatable. Furthermore, we take particular care to make sure that route set feasibility is achieved with reasonable computation costs.

A lack of standard (simplified) models for the UTRP, is anther weakness in current literature. Fundamental research on such models is essential to gain a proper scientific understanding of the problem. To date, most work has and focussed on particular urban areas with unique properties. Although some excellent ideas have been put forward by various researchers, it is difficult to properly assess most of this work, or compare one approach with another. An absence of benchmark data is a related problem. The only data set used by several authors for the UTNDP is Mandl's 15 node network, which can be seen in the work of Mandl [73–75], Baaj and Mahmassani [12], Shih and Mahmassani [90], Kidwai [64], Chakroborty [20, 21] and Zhao and Ubaka [106, 108]. We initially evaluate our model and metaheuristic approach using Mandl's data, and demonstrate its effectiveness. Furthermore, we create new larger instances and make them available to other researchers (from the OR-library), developing good lower bound solutions to help researchers assess the quality of results they obtain on these new instances.

Finally, little practical work has been done on the potentially very fruitful area of multi-objective optimization for the UTRP. As previously discussed, Ceder and Wilson [19] and Israeli and Ceder [56] developed sophisticated models involving multiple objectives, however their routines were not implemented. Chapter 5 of this thesis covers our prototype multi-objective approach. This is a working model that can form a basis for further work.

2.6 Commercial Software Packages of the Transport Planning

In the last three decades, with the development of computer-aided techniques, various commercial software packages have been marketed and used in the transport planning industry for decision support. These tools are aimed broadly at all aspects of transport: cars, roads, traffic flows, traffic light positioning etc., and include such tools as VISUM, Emme2/3, SATURN and Cube Voyager.

First of all, these software tools allow GIS to be integrated into them in order to display the structure of the road network. GIS stands for Geographical Information System (e.g. MapInfo, ArcInfo), which can provide a user friendly environment and help the user manage, analyze and display geographical information, by connecting database tables with geographical objects [17]. In the field of transportation GIS, for example is used to build and maintain road databases or determine the accessibility of transit stops. Since standard GIS functionality does not cover specific transport aspects, transport planners all over the world are more or less successfully trying to adapt their GIS according to their planning requirements [71].

VISUM is a comprehensive, flexible software system for transportation planning, travel demand modelling and network data management. It is used on all continents for metropolitan, regional, statewide and national planning applications [76]. It also integrates all relevant modes of transportation (i.e., car, truck, bus, train, pedestrians and cyclists) into one consistent network model [76]. Generally, VISUM has a graphical user interface (GUI) which makes it easy to learn and relatively simple to use. For example, the user can simply mark the two desired terminals for a transit route by a mouse click. At the same time, VI-

SUM is able to merge GIS-data and transportation data into a common database. In addition, it also provides a COM interface based on MS Windows technology and can be integrated with other COM-compliant Windows products, such as MS Office and ArcGIS. Users can program applications using Python, Visual Basic (VBA, VBS, VB) or other programming languages (C, C++) [76].

In VISUM, methods to generate possible routes incorporate an objective function which minimizes the number of transfers. Using a set of predefined terminals for each route it generates and evaluates a set of possible routes for the planner. At the same time, the algorithm is based on a transit demand matrix and a link network which gives potential connections for a route [71]. On the other hand, it also provides methods to optimize the timetable, and minimize the transfer waiting time of passengers in a route network with a fixed headway. Based on the results of a public transport assignment, a genetic algorithm develops and evaluates "populations" of possible solutions by varying the departure time [71].

Emme2/3 is the powerful industry-standard travel demand forecasting software tool [93]. Emme2/3 is a versatile, professional toolkit with which planners can build their own planning models. At the same time, it provides an extensible, customizable, automated framework for developing transportation models, performing analysis, visualizing data and generating reports [93].

SATURN is the Simulation and Assignment of Traffic to Urban Road Networks, which is mainly used in designing highway transportation systems [8]. It has been continuously developed since 1976 by Institute for Transport Studies, Leeds University. Its features include [8]:

- Combined simulation and assignment for detailed representation of traffic behavior.
- Interactive network building and editing.
- Comprehensive graphical display and analysis options.
- Advanced matrix manipulation facilities.

Cube Voyager is designed to forecast personal travel and provides an open and user-friendly framework for modelling a wide variety of planning policies and improvements at the urban, regional and long-distance level, at the same time, it also provides a transit path-building and assignment function for highway design [22].

In addition, some researchers have also developed some auxiliary software packages to help them design transportation systems. For example, Zhao and Gan [107] in 2003 implemented their methodology of optimization of a transit network to minimize transfers in a prototype GIS based program called OPTNet (OPTimization Package for transit Network).

In general, the above commercial software tools used in real transport planning are efficient and useful. However, they are all limited in their capabilities, and are used primarily for visualization, simulation and decision support. None is able to automate the complete route set optimization procedure, which is the aim of our present research.

2.7 Summary

In this chapter, first vehicle routing problems have been simply introduced. Then we have summarized the guidelines and rules for designing

real route networks for the UTRP, noting that different counties or areas have different requirements for route network design. This is followed by a review of the literature covering the various techniques for solving the UTRP, including manual methods, mathematical approaches, and heuristic and metaheuristic algorithms. Throughout our review we have noticed a lack of generic models for the UTRP and an absence of benchmark data, making it very difficult to make useful comparisons between the different approaches published by researchers in the field. Finally, we have presented a brief review on some current software tools used for transport planning. According to our findings, current commercial software, where applicable to urban transit routing, is used for decision support, with only limited optimization functionality. For example, it may be used to find a shortest path between a source and a destination node, or given a set of terminals, it may be used to generate possible routes that minimize the number of transfers. On the other hand, some tools provide sophisticated simulation capability and are able to model complete transport systems with cars, buses, trucks, bicycles, and pedestrians. Other tools provide modelling and analysis capability, and some can be used to predict travel demand.

Chapter 3

RESEARCH FOUNDATIONS FOR THE UTRP

This chapter introduces our simplified model of the UTRP (published in [35]) and our data set generation method, which can produce random transit networks based on user-supplied parameters and constraints. We also define a lower bound for the urban transit routing problem, which is useful for assessing the quality of the route sets produced by our metaheuristic algorithms. Finally, we discuss setting some constraints, such as the number of routes and the maximum and minimum numbers of nodes in each route.

3.1 Our Simple Model of the UTRP

Given the practical importance of the UTRP, it is perhaps rather surprising that so little work has been done on extracting generic features, formulating simplified models and devising benchmark data sets, to facilitate comparative studies in order to identify which algorithms work best. Perhaps the lack of fundamental research can be explained by the enormous complexity of the UTRP. It may be difficult for researchers to agree which aspects of the problem are most important, and thus decide

which of these should extracted as "generic" to formulate a simplified model. Nevertheless, we attempt to do exactly this for the UTRP as part of our research.

First of all, different models of the UTRP have been characterized by different optimization criteria and special constraints. A set of criteria generally accepted by most researchers is outlined in [20]; these were discussed in Chapter 1 of this thesis, along with some basic real world constraints. On the other hand, the following evaluation parameters have been adopted to assess the quality of route sets by many researchers (e.g. Mandl [74], Baaj and Mahmassani [12], Kidwai [64], Chakroborty and Dwivedi [21], etc.):

d_0 - The percentage of demand satisfied without any transfers.

d_1 - The percentage of demand satisfied with one transfer.

d_2 - The percentage of demand satisfied with two transfers.

d_{un} - The percentage of demand unsatisfied.

ATT - Average travel time in minutes per transit user (mpu).

We also use these five parameters to assess the quality of the final results produced by our methods, to make it possible to compare our route sets with those produced by others. However, please note that we do not use these five measures directly in our objective function. Instead we use a weighted combination of travel times and transfers, as explained below.

For any given route set we can create a corresponding route network, simply by fusing together all routes in the route set. For example, in

the Figure 3.1, the left-hand graph is a transit network, while the right-hand graph represents a route network.

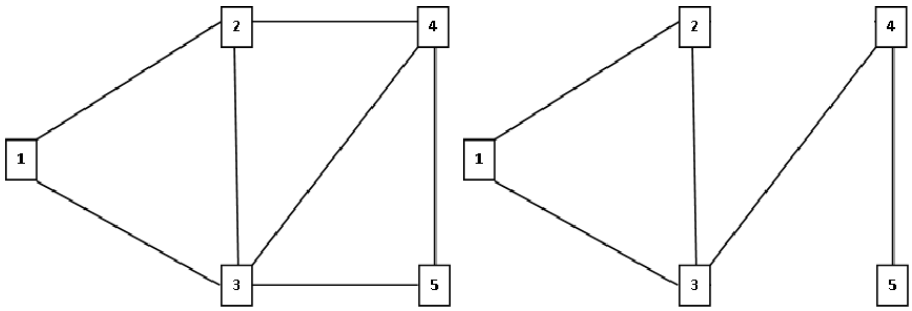

Figure 3.1. Transit Network and Route Network

A particular route network will differ from the original transit network from which it is derived, provided some links present in the transit network are absent from the route network. As a consequence, shortest path distances for travellers between the various node pairs will need to be recalculated for each new route set that is evaluated, using a distance or time matrix specific to that route network. We will assume that each traveller chooses the shortest path (in the route network) from source to destination node, without regard to the number of transfers. Waiting times are not included in our shortest path calculation. Instead transfers are dealt with separately in our objective function.

Our objective function is a weighted sum of two components: the total travel distance or time accumulated over all passengers, and the total number of transfers for the entire demand. Below, we present the key features of our simple model (introduced in [35]):

1. To represent the basic problem information we need:

    - An undirected graph, $G(V, A)$, consisting of N vertices (or nodes), $V = \{v_1, v_2, \ldots, v_N\}$, and m arcs, $A = \{a_1, a_2, \ldots, a_m\}$. This will store the transit network.

- A demand matrix, D, where d_{ij} = travel demand between nodes i and j (with d_{ii}=0).
- Routes in the current route set, stored as lists.
- A cost matrix, C, where c_{ij} = the travel cost (i.e., distance or time) between nodes i and j, where direct links exist in the current route network. (Note: travel cost is recorded as $+\infty$ between nodes that are not directly connected.)

2. We define the simple objective function:

$$Minimize : Z = A \sum_{i,j=1}^{N} d_{ij} p_{ij} + B \sum_{i,j=1}^{N} d_{ij} t_{ij} \qquad (3.1.1)$$

where:

p_{ij} is length of the shortest path between i and j for the current route network (calculated using Dijkstra's algorithm and the cost matrix, C);

t_{ij} is the minimum number of transfers required to traverse the shortest path for the current route set (obtained from the current routes and the cost matrix);

A and B are constants used to weight the two components of the objective function. (Please note that A and B are chosen to ensure the two parts of the objective function are of similar magnitude. More details are given in Chapter 4)

3. The objective function is subject to the following constraints:

- *each route in a given route set is free of cycles and backtracks.* This is easily checked when generating or modifying a route,

by ensuring that there are no repeated nodes in the route.

- *the route set is connected.* The connectivity of the route set is checked as part of the Feasibility Check Procedure (see details in Section 4.1.3).
- *there are exactly r routes in the route set to simplify the problem* (assume r is set by the planner or bus company).
- *the number of nodes in every route must be greater than one, and must not exceed a planner-defined maximum value, MAX.*

3.2 Data Set Generation Method (DSGM)

From the literature review on the urban transit routing problem (see Chapter 2), we saw that much previous work concentrates on specific real world problems, which are not in the public domain. Nevertheless, research workers need suitable data sets on which to test their algorithms and compare their results with those produced by competing approaches, and Mandl's 15 nodes network [73] would appear to be the only transit network instance readily available to researchers.

To meet the needs of our research programme, we have designed and implemented a data set generation method (DSGM) that will produce realistic transit networks, randomly positioning a given number of nodes and links within a rectangular framework. Our software will ensure that each transit network it produces is connected, and in addition it will produce demand values (within a given range) between each pair of nodes. Finally, both demand and distance or time matrices will be stored in a file for further use. In the next section, the principles of

our data set generation method will be introduced.

3.2.1 Basic Principles of Data Set Generation

Our overall aim is to produce realistic transit networks, according to user defined parameters. First of all, we constrain the transit network within the boundaries of an enclosing rectangle, and give the user control of its size, through inputs specifying the lengths of its two sides. At the same time, other information such as the total number of nodes (bus stops) in the transit network and the total number of transit links connecting the various nodes, also need to be defined by the user. The number of nodes and links gives the user control over the size and the complexity of the network, leaving the software to decide exactly where the nodes and links will be placed within the enclosing rectangle. Similarly, the general level of demand is determined by the user, but the actual demand values between each pair of nodes is randomly generated by our software. The user will supply an upper and a lower bound (i.e., a demand range), and this range will apply to every node pair in the network. The user may choose this range according to the scale of the transit network or the real situation in some areas.

We will assume that our transit networks needs to be connected; i.e., that at least one path will exist between every pair of nodes in the transit network, so that each part of the network is reachable from any given starting point. However, in our method the nodes and links are randomly generated, and it is not possible to ensure connectedness without introducing a specific mechanism to make this happen. Our method of choice is to construct a minimum spanning tree (MST), according to the Euclidean distances between each pair of nodes. We

construct our MST immediately following the generation of the nodes. It is then an easy matter to add additional links in a random fashion, to complete a transit network according to user determined parameters. (Note that the weight between each vertex pair in a transit network can represent either the distance or the travel time between each node pair.)

There are two popular algorithms used to obtain an MST, namely, Kruskal's algorithm [65] and Prim's algorithm [84]. In our DSGM, we choose Prim's algorithm [84] to generate the minimum spanning tree. The reason is that the efficiency of Prim's algorithm is not directly dependent on the total number of links in the network, thus it should be more efficient for the large networks of the type we need for testing our algorithms.

According to the property of a minimum spanning tree, if there are N nodes in the transit network, the total number of links is $N-1$. However, to simulate realistic transit networks, we need more links than this. Furthermore, additional links should be added sympathetically, so that the network retains a sensible appearance. In our method the concrete steps of each adding process are shown as follows:

1. A node is selected at random from the network of the minimum spanning tree;

2. The shortest of all the links between this node and any other nodes, not including the links already in the minimum spanning tree network, is selected;

3. Add this shortest link into the network.

Through repeating the above steps, the number of extra links re-

quired can be added to form a simulated transit network. Therefore, our DSGM can generate a range of realistic transit networks, with related demand, according to user-supplied parameters. In conclusion, the data set generation method can be summarized in the pseudocode presented in Algorithm 4:

Algorithm 4 Data Set Generation Method

```
Input Parameters: X, Y, Min-demand, Max-demand, N and ELINKS
Initialize a rectangle based on the X and Y values
Generate and distribute N nodes in the rectangle at random
Decide the demand for each node pair randomly within the Min-demand and Max-demand
Call Prim's Algorithm to find a minimum spanning tree
Initialize the transit network with these links
repeat
    Select a node randomly in the subgraph network
    Find the shortest link from the selected node, not currently included the network, if one exists (if not, choose another node)
    Add this link into the network
until ELINKS is achieved
Output final transit network and the distance and demand matrix
```

X and Y are the values of the X and Y coordinates measured from the origin, (0,0) defining the enclosing rectangle. Min-demand and Max-demand are the lower and upper bound demand for each node pair. N is the number of nodes and $ELINKS$ is the number of extra links required.

3.2.2 Implementation of the DSGM

Our data set generation method is implemented in the Java programming language. The user is presented with two application interfaces:

1. the minimum spanning tree generation interface, and

2. an interface for adding extra links.

In the minimum spanning tree generation interface, users can input the required parameters needed for generating their transit network: the total number of nodes, the enclosing rectangle dimensions, X and Y, and the maximum and minimum travel demand required. After the parameters are input, the minimum spanning tree can be generated and displayed on the interface, e.g. see Figure 3.2.

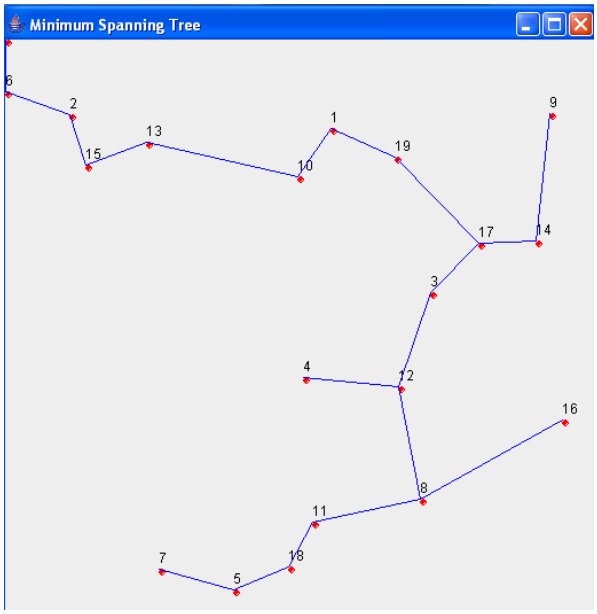

Figure 3.2. Minimum Spanning Tree

After the minimum spanning tree is generated by the application, users can add extra links through the "extra links" interface, to create a final simulated transit network, e.g. see Figure 3.3. Finally, the transit network is generated, and its demand and distance matrix are stored in a file.

3.3 Lower Bound for the UTRP

In the practical design process of an urban transit routing problem (UTRP), many criteria need to be optimized in order to efficiently

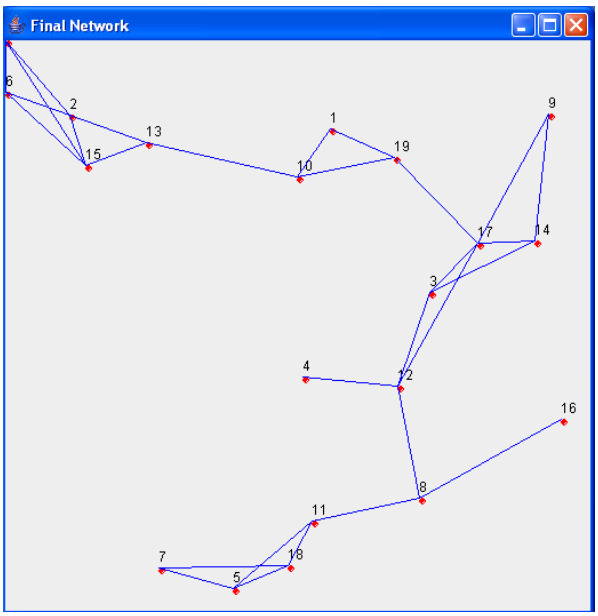

Figure 3.3. Final Network

meet the needs of the passengers, while at the same time minimizing the cost to the operators. In addition, there are other stake-holders involved: typically national and local government as well as taxpayers and local businesses. While all interested parties will benefit from an efficient public transport service, each one will be observing from their own perspective, and thus may have different notions of what efficiency means.

Nevertheless, as mentioned before both a simple model for the UTRP and benchmark data sets are currently missing from the literature. One of the main goals of the present thesis is to provide a simple framework for the UTRP, along similar lines to what has been provided for other well-known combinatorial problems. We believe that it is only by doing this that we can put the UTRP on a similar footing to, say, the Capacitated Vehicle Routing Problem, the Job-Shop Scheduling Problem, the Examination Timetabling Problem and the Quadratic Assignment Problem (to name but a few). We will present our solution methods in

Chapter 4, and deal with the issue of generating benchmark data in the present chapter using our DSGM. Firstly though, we will discuss the issues that arise when we use benchmark data for the first time - how can we use it to assess the quality of our route sets? We do not know what the optimum solutions are, and neither do we have any previous published results.

Recall the criteria for a good route set [20] (see Chapter 1):

- The entire transit demand is served, that is, the percentage of unsatisfied demand is zero;
- A large percentage of transit demand is served through direct connections, that is, the percentage of demand satisfied with zero transfers is high;
- The average travel time per transit user is as low as possible.

In order to overcome the above mentioned difficulty of evaluating route sets when using new benchmark data, we adopt the above criteria and propose a lower bound on the passenger's cost for the UTRP. Our lower bound is based on an ideal situation for passengers travelling on the transit network: namely, every passenger can travel to their destinations by the fastest (or shortest) path without any transfers. If the number of nodes, travel distance (or time) and travel demand between each node pair are already known, the ideal travelling path, between each pair of nodes can easily be found using Dijkstra's algorithm [30] (introduced in Section 2.4) on the entire transit network. Note that the network obtained by superimposing all the routes from a particular route set will consist of all the nodes but only a subset of the links from the entire transit network. Thus, "ideal travel paths" between various

pairs of nodes may or may not be attainable from a given route set.

In order to evaluate the lower bound, we calculate the Total-Demand, Total-Person-(Distance or Time) and Average-Travel-(Distance or Time). According to the mathematical formulation of our simple model of UTRP (introduced in Section 3.1) these can be mathematically formulated as follows:

1. Total-Demand:

$$\sum_{i,j=1}^{N} d_{ij} \tag{3.3.1}$$

2. Total-Person-(Distance or Time):

$$\sum_{i,j=1}^{N} d_{ij}p_{ij} \tag{3.3.2}$$

3. Average-Travel-(Distance or Time):

$$\sum_{i,j=1}^{N} d_{ij}p_{ij}(\sum_{i,j=1}^{N} d_{ij})^{-1} \tag{3.3.3}$$

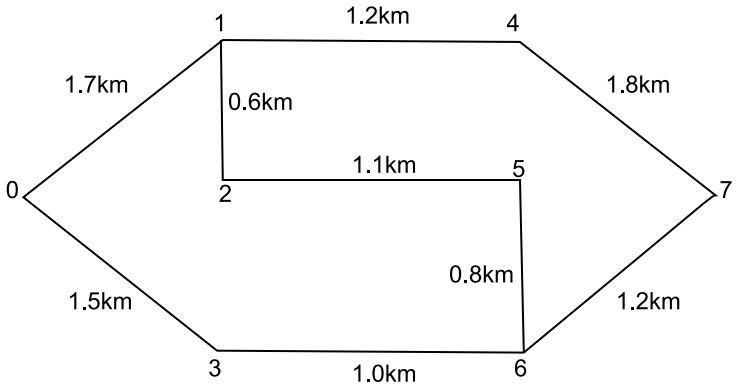

Figure 3.4. 8 Nodes and 9 Links Network

We implemented a program in Java to calculate the lower bound, and the following simple example is used to illustrate some of the re-

Table 3.1. Demand Matrix

8	0	1	2	3	4	5	6	7
0	0	95	120	58	112	177	125	184
1	95	0	64	167	59	166	167	144
2	120	64	0	71	173	50	78	166
3	58	167	71	0	59	95	58	119
4	112	59	173	59	0	49	86	84
5	177	166	50	95	49	0	86	135
6	125	167	78	58	86	86	0	93
7	184	144	166	119	84	135	93	0

sults calculated by our program. Figure 3.4 shows an 8 nodes and 9 links network from [88], with the distance between each node pair as shown. The travel demand matrix for this instance is presented in Table 3.1. Finally, the results - every exact shortest distance path for each node pair, Total-Demand, Total-Person-Distance and Average-Travel-Distance are presented in Table 3.2. It is easy to see in the lower bound situation the total number of shortest routes can be obtained by the following formulation:

$$N(N-1)/2 \tag{3.3.4}$$

(N is the total number of nodes in the transit network), at the same time, the Total-Transfer-Time is 0.

3.4 Constraints for the UTRP

For the urban transit routing problem, many criteria need to be optimized, while at the same time, many constraints also need to be satisfied. For example, in practical transit network design, the transport planners may decide the number of bus routes, based on the practical requirements of passengers and operators in a local area, and constrained by a limited number of buses. In addition, it is usual to limit

Table 3.2. Lower Bound Results

Route Number	Route Description
1	0-1
2	0-1-2
3	1-2
4	0-3
5	1-0-3
6	2-5-6-3
7	0-1-4
8	1-4
9	2-1-4
10	3-6-7-4
11	0-3-6-5
12	1-2-5
13	2-5
14	3-6-5
15	4-1-2-5
16	0-3-6
17	1-2-5-6
18	2-5-6
19	3-6
20	4-7-6
21	5-6
22	0-3-6-7
23	1-4-7
24	2-5-6-7
25	3-6-7
26	4-7
27	5-6-7
28	6-7
Total Demand	6080
Total-Person-Distance	14331200 m
Average-Travel-Distance	2357.11 m
Total-Transfer-Time	0

the maximum length of a route and restrict the number of bus stops on a route: for example, to maintain reliability and limit operators' costs. On the other hand, as previously mentioned, a bus route set designed by a planner should be a feasible route set, which means every node presented in the original transit network must be included in the bus route set, and every node needs to be connected either directly or in-

directly to every other node. Only on such a bus route network can passengers get to any destination point from any start point. In addition, in some areas a bus route is not allowed to backtrack or meander excessively. However, it is likely that setting appropriate constraints will significantly impact on the overall problem difficulty. For example, it may prove difficult to find *any* feasible route set if the problem is constrained by too few routes, or by having too few nodes on each route. On the other hand, if very many routes are allocated the search space increases, making it easy to find a feasible route sets but difficult to find an optimal one.

Given the obvious importance of setting constraints for the UTRP, it is perhaps strange to find that little attention has been paid to this problem in the literature. In an attempt to redress this balance, we will consider some of these issues in the remainder of this section, exploring the boundaries of feasibility and examining the affects of various constraint levels on problem difficulty. We will examine the following constraints in turn:

1. number of nodes in a route
2. number of routes
3. requirement for a feasible route set

3.4.1 Number of Nodes in A Route Constraint

First of all, it is easy to decide the minimum number of nodes for each bus route: a bus route should contain at least two nodes. To a large extent the maximum number of nodes will depend upon how close the nodes are together. For a long route, the operators' costs may be high

if demand is unevenly spread, and a large number of stops can make it difficult to maintain schedule. On the other hand, if a bus route is too short, passengers may need an increased number of transfers to reach their destinations. Hence, it is very important to set this constraint at the right level. We propose the following method for determining the maximum number of nodes for a route:

- Calculate all the shortest paths between each pair of nodes in the transit network for which the demand is greater than zero;
- Locate the "shortest path" with the most nodes, and record the number of nodes, MAX;
- Assign MAX as the maximum number of nodes for each route;

This number will ensure that the space of possible solutions will allow for travel plans that avoid vehicle transfers. If all bus routes are shorter than some passengers' best travel routes, for example, vehicle changes will be inevitable. It is appropriate to consider scenarios that provide opportunities for passengers to get to their destinations as quickly as possible and with as few transfers as possible.

3.4.2 Number of Routes Constraint

As mentioned above, transport planners will often decide the number of bus routes in advance, depending on the practical requirements of both passengers and operators in a local area. Operators may be constrained by a limited number of buses, and will certainly have a limited budget. More routes generally mean more buses and higher operating costs. Of course, there is a close interrelationship between the number of routes

and the number of nodes of those routes, and this has an impact on route set feasibility. We will consider this next.

3.4.3 Constraints for Feasible Route Sets

In the design process, it generally makes sense for transport planners to develop a *feasible route set*, which includes every node in the transit network, with every node directly or indirectly connected. In order to obtain a feasible route set, the relationship between the number of nodes of each route and the number of routes in the route set must satisfy some conditions.

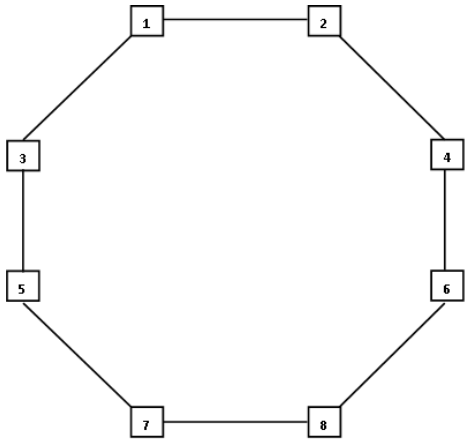

Figure 3.5. 8 Nodes and 8 Links Network

For example, Figure 3.5 illustrates a simple network with 8 nodes. If the maximum number of nodes for each route is set to 3, and the total number of routes in the route set is set to 2, it is clear that the maximum possible number of nodes that can be served by the network is 6, which leaves out 2 nodes. In Figure 3.6, two routes, *1-2-4* and *3-5-7*, make up the route set; nodes *8* and *6* can not be included. In such situations it is plainly impossible to develop a feasible route set.

On the other hand, if the maximum number of nodes for each route

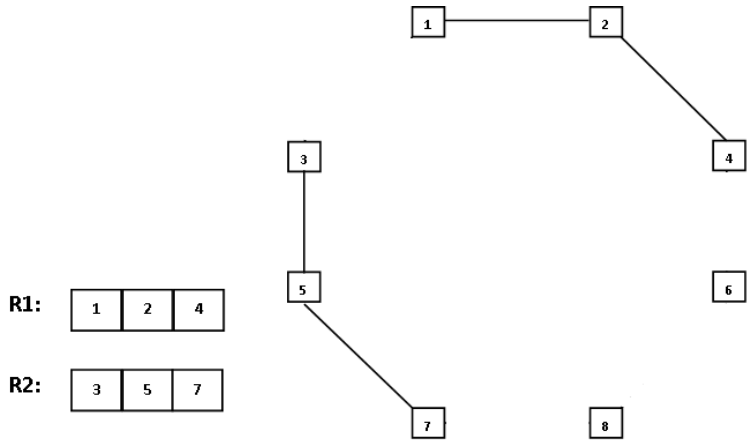

Figure 3.6. 2 Routes and Maximum 3 Nodes Example

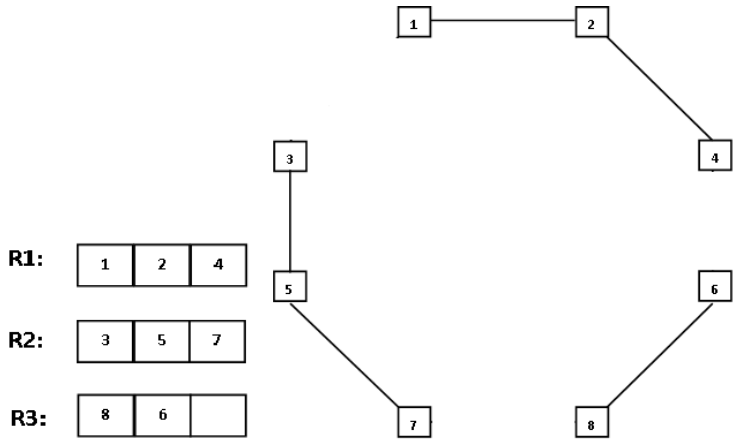

Figure 3.7. 3 Routes and Maximum 3 Nodes Example

is set to 3 and the number of routes in the route set is set to 3, then this route set can at most contain 9 nodes. If the route set covers all 8 nodes of the network, only one extra node position is available. However, one available position is not enough to make the route set connected (see Figure 3.7). The reason is that, for two routes to be directly connected, the routes must have at least one node in common. For routes to be indirectly connected, an additional intermediate route is required to join them together. This intermediate route must have at least one node in common with each of the routes it is connecting. Therefore, in this situation at least two available positions left in the

route set are needed to make the route set feasible, so that an extra route is needed (see Figure 3.8).

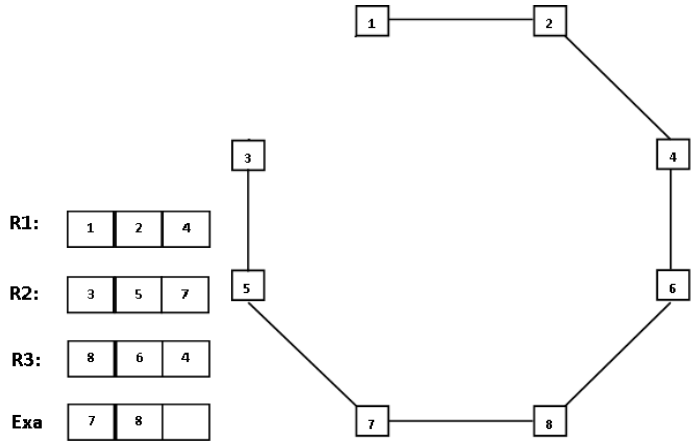

Figure 3.8. A Feasible Route Set

From the above discussion, we summarize the two necessary conditions for a feasible route set as follows:

1. all the nodes in the transit network must be represented in the route set

2. the route network (obtained by overlaying all the routes) must be connected

We will use the following terminology:

- The total number of nodes in the network is N;
- The maximum number of nodes in each route is MAX;
- The minimum number of nodes in each route is min (here the min is set to 2);
- The minimum number of bus routes needed to make a feasible route set is R_{min} ;

To satisfy Condition 1, we require each of the N nodes to appear at least once in the route set; to satisfy Condition 2, the route set needs to be connected. Without loss of generality, we will assume that all routes are connected end-to-end to form a directed path through all N nodes. Given each route is of length MAX, we will start at the beginning of the directed path. If $N \leq MAX$, we will need only one route. Otherwise, the first route will cover MAX previously uncovered nodes. The next path will start with the final node in route one, and discover $MAX - 1$ previously uncovered nodes (provided there are sufficient nodes remaining). Subsequent routes will also discover $MAX - 1$ new nodes, until we reach the final route, in which there may be less than $MAX - 1$ remaining nodes to discover. Thus, except for the first and last route, all other routes will contain $MAX - 1$ discovered nodes. From this it follows that the minimum number of routes needed to cover all the nodes is given by Equation 3.4.1.

$$R_{min} = \begin{cases} 1 & if\ N \leq MAX \\ \lceil (N-1)/(MAX-1) \rceil & otherwise \end{cases} \tag{3.4.1}$$

3.5 Creating Our Data Sets

Due to a lack of published data sets, we generate our own data to test our methods and see whether the efficiency of these methods are suitable for different sizes of networks.

3.5.1 Research Into Some Properties of Real World Data Sets

First of all, we examined some real world bus route maps, in order to extract typical properties such as the total number of nodes (bus stops)

and links, the number of bus routes and the maximum and minimum numbers of nodes in bus routes. In total, four different maps of urban areas in China and the UK have been studied, namely the city centre of Yubei district [3], which is a major part of Chongqing in China, the city centre of Beijing [2], which is the capital of China, the city centre of Cardiff [4], which is the capital of Wales, and the city centre of Brighton [1], which is a town on the south coast of England. The details of the (approximate) properties are shown in Table 3.3.

Table 3.3. Properties of Real Data Sets

Location	Number of Nodes and Links	Number of Routes	Route Nodes	Link /Node	AFN
Yubei	70 & 210	15	10 - 30	3	4.29
Beijing	1280 & 3550	170	15 - 45	2.77	3.98
Cardiff	127 & 425	60	12 - 25	3.35	8.98
Brighton	110 & 385	56	10 - 22	3.5	8.15

The route nodes indicates the minimum number of nodes (min) and the maximum number of nodes (MAX) for route map. The AFN is an estimate for the Average Frequency of bus routes visiting each Node. If the number of nodes (N) and the number of bus routes (B) are known, the AFN can be estimated by the following formulation:

$$\lceil (min + MAX)/2 \rceil B/N \tag{3.5.1}$$

It is interesting to note the link/node ratio is fairly consistent (close to 3) for each of the four data sets. The AFN varies between about 4 and 9.

Table 3.4. Data Sets Description

Network	Number of Nodes	Number of Links	Bounds of Demand
I	8	9	0 - 184
II (Mandl's)	15	21	0 - 880
III	70	175	0 - 500
IV	70	245	0 - 500
V	110	275	0 - 600
VI	110	385	0 - 600
VII	130	325	0 - 800
VIII	130	455	0 - 800

3.5.2 Our Data Sets

Table 3.4 lists features of the 8 different data sets we use for our research. Network I is a small instance obtained from Pattnaik et al.'s paper [88]; network II is Mandl's Swiss transit network [73]. Network III to network VIII are all new data sets generated by our DSGM application, with parameters guided by the real route networks that we examined in the previous section. The total number of nodes (70, 110 and 130) are related to the three places, Yubei, Brighton and Cardiff and the total number of links are chosen by using the factors 2.5 and 3.5 respectively, to obtain 6 new instances. We also generated a large network related to the Beijing system (see Chapter 6).

When generating the data sets, we apply the same range of coordinate axes: X coordinate and Y coordinate, to produce sides of an enclosing rectangle with lengths ranging between 0 and 500. At the same time, the travel time (in minutes) is used to measure the distance between each node pair. For example, if the distance between two nodes is "5", this is considered as 5 minutes travel time in the travel time matrix. The demand between every node pair is deter-

mined at random between the lower and upper bounds input by the user, as specified earlier. The details of these instances, including networks, demand matrices and distance matrices, are listed on our website (http://users.cs.cf.ac.uk/L.Fan/), and also published in OR-Library [**?**].

3.6 Summary

In this chapter we have introduced our simplified model of the UTRP, which evaluates routes according to the average travel time and the number of transfers between vehicles, and a data set generation method, which is able to produce random transit networks based on user-supplied parameters. In addition, we have explored various constraints and defined lower bounds for total transit times and also for the the minimum number of routes required to ensure coverage of a transit network. We have also explored the properties of four real world urban transit networks. Guided by our theoretical and practical studies, we have used our data set generation method to produce data sets for subsequent use.

Chapter 4

A METAHEURISTIC APPROACH TO THE UTRP

In this chapter (based on the work published in [36]), we describe our basic metaheuristic framework for solving the UTRP. This consists of a representation scheme, an initialization procedure, a feasibility check procedure and a set of simple neighbourhood moves. Furthermore, two simple search algorithms, hill-climbing and simulated annealing, are embedded into our metaheuristic framework. In addition, the assessment parameters for the bus route set are also discussed in this chapter. Finally we present some experimental results which improve upon published results for Mandl's benchmark problem [73], and also some further results for larger problem instances.

4.1 Methods of Representing and Improving the Route Set

Success in finding good route sets depends on devising the following: (1) a suitable representation scheme, (2) an effective initialization mechanism and (3) intelligent route improvement heuristics. In our method we use simple arrays to store the routes, and utilize three basic procedures, namely *Initialization*, *Feasibility Check* and *Make-Small-Change*.

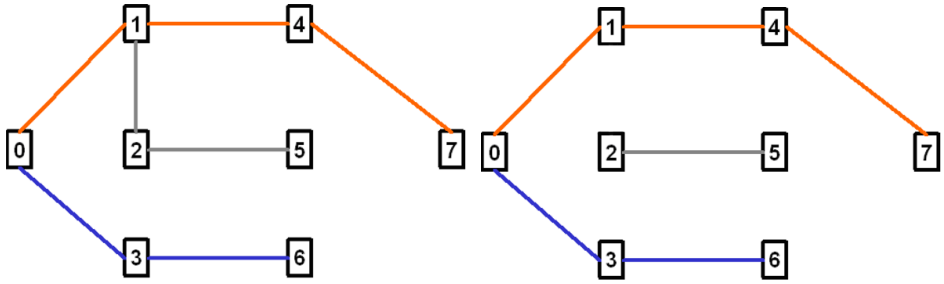

Figure 4.1. A Connected and An Unconnected 8 Nodes Network

4.1.1 Representation

R1	0	1	4	7
R2	0	3	6	*
R3	1	2	5	*

Figure 4.2. Two Dimensional Array

The representation we use to store the route set is a two dimensional array. The first location of each row stores the route number, which is useful for identification purposes. For example, consider the first graph in the Figure 4.1, if we set the maximum number of nodes in each route to 4 and the number of routes in the route set to 3 routes, the routes (i) *0-1-4-7*; (ii) *0-3-6*; (iii) *1-2-5* can be stored as shown in Figure 4.2 (where the * represents an empty array element).

We use a two dimensional array structure to represent the route set because it seems a very simple and natural way to represent a set of linear structures (the routes). While distance (or time) and demand matrices are used to represent the underlying graph structures, it is

convenient to use simple structures such as arrays, so that the routes are easily identifiable for route manipulation and feasibility checks, when applying a metaheuristic algorithm.

4.1.2 Initialization

Algorithm 5 Initialization Procedure I

Parameters: number of routes, minimum and maximum number of nodes for routes
Begin
Main loop
repeat
Route length selection:
Choose a length for the route between minimum and maximum number of nodes at random
Inner Loop
repeat
Start node selection:
Choose any node as the start node at random
label this node as the "previous node"
Next node selection:
Construct a node set consisting of all nodes directly connected to "previous node" that have not been selected for the current route, **if** this node set is not empty, choose a node from this set at random
Else invert the order of the route (see Section 4.1.4) and repeat the ***Next node selection***
If such route can not reach the route length, **then** delete this route and repeat ***Start node selection***
until route length is reached
until number of routes is reached
Output an initialization route set

The purpose of our *Initialization Procedure* is to construct an initial route set at random, according to the constraints listed in Section 3.1 and some user-defined parameters. In the initial route set, each route is a connected path containing no cycles or backtracks. However, the feasibility of the route set is not ensured at this stage. The main

structure of the initialization procedure is shown in Algorithm 5.

Unlike many researchers, we do not rely on shortest path algorithms to produce our initial routes. As we explained in Chapter 2, we do not believe that building route sets from pre-computed shortest paths necessarily produces the best route sets. The quality of the route set as a whole is the important factor, rather than that of the individual routes. Longer travel paths may be appropriate between some sources and destinations where travel demand is low, for example, in the interests of efficiency. In addition, the success of metaheuristic approaches is not generally very tightly related to the quality of the initial solution.

4.1.3 Feasibility Check

Algorithm 6 Feasibility Check Procedure

Input the route set, ***S***
Input N, the number of nodes in the transit network
Initialize ***found-node****[1...N]* = **0** {records nodes that have been found}
Initialize ***explored-node****[1...N]* = **0** {records nodes that have been explored}
Select an arbitrary node, i, present in at least one route
Set *feasibility* = *False*
while {*feasibility* == *False*} AND {there are unexplored nodes in ***found-node***} **do**
  Set ***explored-node****[i]* = ***found-node****[i]* = 1
  Find all routes containing node i
  Set flags in ***found-node*** to record all the nodes found in those routes
  Select any node from ***found-node*** that is absent from ***explored-node***
  That node becomes node i
  if all N nodes have been found and entered in ***found-node*** **then**
    feasibility = *True*
return *feasibility*

The Feasibility Check Procedure is necessary because finding feasible

route sets (that obey all constraints) using randomized methods is a huge challenge. The main purpose of the Feasibility Check routine is to ascertain whether candidate route sets are connected and include every node present in the original transit network. A connected route set means that the passengers can get to any destination point from any start point in the route set network. An unconnected route set means that some places or nodes in the network are not directly or indirectly linked, therefore passengers are not able to reach all points. For example, the first graph in Figure 4.1 is connected, but the route consisting of nodes 2 and 5 in the second graph is not linked to the rest of the route network. In a similar way, demand to and from nodes that do not appear in at least one route in the route set, cannot be met. The structure of our Feasibility Check Procedure is shown in Algorithm 6.

4.1.4 Make-Small-Change

Successful application of metaheuristic methods to the UTRP depends on whether suitable neighbourhood moves can be devised for the problem. It is clear that infeasible route sets may be too easily produced by random procedures on the UTRP. Careful reference needs to be made to the underlying transit network when adding or deleting nodes from a previously constructed route, or when moving a node from one route to another. Large random changes are clearly not desirable, as they are likely to destroy connectivity or produce solutions in which the routes meander excessively. Even a very small change can prove highly disruptive. Our "make small change" strategies cautiously adds or deletes nodes from individual routes, one at a time, checking and ensuring feasibility as an integral part of the process. Nevertheless, over a period of

time, we have observed that our approach is able to explore the search space thoroughly, accumulating "large changes" in small steps.

Recall that we store the individual routes as ordered lists in an array (see Figure 4.2). The Make-Small-Change procedure is responsible for making local neighbourhood changes to a route set. There are three possibilities:

1. *Adding a node to the last position in a route;*
   ensuring that there is a direct link in the transit network to connect the new node, and that no cycles or backtracks are produced.

2. *Deleting the first node in a route*;

3. *Inverting the order of nodes in a route;*
   i.e., the first node becomes last node and the last node becomes the first node. This method is used in place of the "adding a node" when no suitable nodes can be added (see below).

The structure of Make-Small-Change procedure is shown in Algorithm 7:

The above mentioned "add" and "delete" node operators are key in the Make-Small-Change procedure, with "inversion" used occasionally in place of "add", when it is not possible to add a node to the last position.

First of all, the procedure randomly selects one of the routes in the route set to act as a candidate route for change. Next, this route will be checked for its potential, with respect to possible application of the Make-Small-Change operators. In general, there are three situations. (I) the length of a route is between the maximum number of nodes and the minimum number of nodes defined by user. (II) the length of a

Algorithm 7 Make-Small-Change Procedure

Operators: "add", "delete" and "inversion"
Input a route set
repeat
  Randomly select one of routes from the route set, label this route as "selected route"
  Check the "selected route"
  If its length is between the maximum and minimum number of nodes
  then "add" or "delete" operator is selected at random
  in case "add" operator is selected, but no available node that can be added
  then "inversion" is selected
  If its length is equal to maximum number of nodes
  then "delete" operator is selected
  If its length is equal to minimum number of nodes
  then only "add" operator is selected
  in case no available node that can be added
  then "inversion" is selected
until Termination condition is satisfied

route is equal to the maximum number of nodes. (III) the length of a route is equal to the minimum number of nodes.

If a chosen route is in the (I) situation, the adding or deleting operator is randomly selected as the "small change" to be made. Unfortunately, a problem can occasionally arise, when the "add node" operator is selected and there is no available node that can be added to the end of the route, avoiding cycles and backtracks. For example, in the first graph in Figure 4.1, if a route *0-3-6* has been selected to add a node to the end, obviously no available node can be added to the route. Hence in this situation, the "inversion" operator will be applied instead to this route. In our example, the original route becomes *6-3-0* following inversion. Next, an alternative route will be selected at random from the remaining routes in the route set. This newly selected route will be identified as situation (I), (II) or (III), as before, and a Make-Small-

Change operator applied appropriately. This process will be repeated, as necessary, until a "small change" has been effected.

If a chosen route is in the (II) situation, the route cannot be made any longer so the deleting method will be applied to the route. In a similar way, if a chosen route is in the (III) situation, the route cannot be made any shorter so the adding method will be applied. Once again, in some circumstances there will be no available node to add, just as we saw in situation (I). Like before, this route will be inverted and another route selected.

4.2 Framework of Implementing HC and SA Algorithms

Algorithm 8 *Route-Hillclimber* or *Route-SimulatedAnnealing*

Parameters: D, C, r, MAX, {plus T_0 and L for SA}
Initialization:
Generate an initial route set of r routes, S
Outer loop - repeat until the stopping condition is satisfied
Inner loop - repeat L times {L=1 for HC}
Modification:
Call *Make-Small-Change* {to generate a near neighbourhood route set, S'}
Feasibility check:
repeat
  if the new route set is not connected **then**
    Call *Make-Small-Change*
until successful
Evaluation:
Calculate $\sum_{i,j=1}^{N} d_{ij}p_{ij}$, $\sum_{i,j=1}^{N} d_{ij}t_{ij}$, and the objective function, Z
Selection:
Select either S or S' as new focus of search following rules of Hill-Climbing or Simulated Annealing
Output Best route set and Z for the best route set

In order to validate the above framework, optimization algorithms need to be implemented. Two algorithms are chosen in our research, namely, hill-climbing and simulated annealing (introduced in Chapter

2). Due to their similar structure, the framework for the two algorithms is summarized in Algorithm 8.

Recall that D is the demand matrix, C the cost (distance or time) matrix for the current route network, r the number of routes in the route set and MAX is the maximum number of nodes per route. T_0 and L are parameters for the SA, to be discussed later.

Initialization: generates an initial route set based on the constraints and user-defined parameters.

Modification: calls the Make-Small-Change routine to generate a new neighbourhood route set.

Feasibility Check: is to check whether the new neighbourhood route set is connected, and contains all the demand nodes. If not, the Make-Small-Change routine is used iteratively until a feasible route set is produced.

Evaluation: Once a feasible route set has been obtained, it needs to be evaluated by calculating the objective function in Equation 3.1.1. We consider the route network obtained by fusing all the routes from a given route set, as explained in Section 3.1. (Recall that a *route* network is a subset of the specified *transit* network.) We assume that all demand is satisfied along the shortest path available (in the route network) between a given pair or nodes, regardless of whether or not this involves making transfers (no time penalty is added to the objective function when a transfer is made). All required shortest paths are calculated from the route network using Dijkstra's algorithm, and the first component of Equation 3.1.1, $\sum_{i,j=1}^{N} d_{ij}p_{ij}$, is calculated. This gives the total travel distance (or time), for the route network, summed over all passengers. Note that if there is more than one contender for

the shortest path between two nodes, the path with the highest demand is selected.

The total number of transfers, summed over the entire demand must also be calculated. This is done by checking every part of each shortest travel path, to identify the routes to which it belongs. In this way, the minimum number of transfers required along each shortest path is recorded, and this information is used to calculate the second term in the Equation 3.1.1, $\sum_{i,j=1}^{N} d_{ij}t_{ij}$. The basic idea can be seen in Algorithm 9.

Algorithm 9 Finding Minimum Number of Transfers Procedure

Input a travel path and the current route set
Parameters: the number of nodes in the travel path n, an index for nodes in the travel path i and minimum number of transfer j
Initialize $i = 1$ and $j = 0$
For the i node of the travel path, label j for these routes that contain this node in current route set
Main loop
repeat
  $i = i + 1$
  If the node i of the travel path does not exist in any marked routes in current route set
  then clean the label j for these marked routes, and label $j = j + 1$ to new routes contain node i
  else keep the label j for these routes still contain node i, and clean the label j for those routes do not contain node i
until i is equal to n
Output Minimum number of transfers j

(Note: in our research, we consider the minimum number of transfers needed to travel on the shortest path in current route set as an individual part in the objective function in order to evaluate the quality of the route set. While some researchers, e.g., Mandl [73] and Chakroborty [21], consider the transfer as the waiting time and insert it into the calculation of the total travel time for passengers.)

Selection: The selection rules are different for the HC and SA search methods. In the hill-climbing algorithm, a route set which has the smaller value of the objective function is kept as a current best result at each time-step. On termination, the best route set found during the entire run of the algorithm will be returned. In the simulated annealing algorithm a new neighbourhood route set will replace the "current" route set at a given time-step if it is better than the existing route set, similar to hill-climbing. However, if the neighbourhood route set is "worse" than the current route set, it is still possible that it may replace it as the new focus of the search. Acceptance will be determined using an "acceptance probability", and the value of this will depend on the current "temperature", and also on exactly how poor the new contender is, in relation to the route set currently occupying the focal position. Early in the execution of an SA algorithm the temperature is high and most neighbourhood moves will be accepted. As the search progresses, the temperature cools and poor solutions are accepted less frequently. As is the case with hill-climbing, the best route set found during the entire run of the algorithm will be returned when the algorithm terminates.

Values for the acceptance probability (*prob*) for a minimization problem, are evaluated using Equation (4.2.1) and (4.2.2). Δ represents the difference between the objective functions (or costs) of the new solution $C(S')$, and the focus solutions $C(S)$. Note that the value of *prob* depends on the value of Δ and also on T, the current tempera-

ture, which is determined by the cooling schedule.

$$\Delta = C(S') - C(S) \tag{4.2.1}$$

$$prob = min(1, e^{-\Delta/T}) \tag{4.2.2}$$

The new solution is accepted with probability 1 if $\Delta \leq 0$ (in other words, if the neighbourhood solution is better than S) and with probability $e^{-\Delta/T}$ if $\Delta > 0$ (that is, if the neighbourhood solution is worse than S). Throughout the execution of an SA algorithm, the temperature T is progressively lowered.

In the present study we determine the precise annealing schedule from user-specified values for the number of cooling steps and the initial and final solution acceptance probabilities. We use F cooling steps to correspond to the number of iterations, so that the temperature is decreased slightly between each iteration. Thus, knowing F and setting initial and final acceptance probabilities, P_0 and P_f, as well as an additional parameter M, that signifies an initial number of random trials, the starting temperature T_0, the final temperature T_f, and the cooling factor α can be calculated, as indicated below.

$$\Delta_i = C(S') - C(S) \tag{4.2.3}$$

$$\Delta_{ave} = \frac{\sum_{i=1}^{M} \mid \Delta_i \mid}{M} \tag{4.2.4}$$

$$T_0 = -\frac{\Delta_{ave}}{\log P_0} \tag{4.2.5}$$

$$T_f = -\frac{\Delta_{ave}}{\log P_f} \tag{4.2.6}$$

$$\alpha = \exp(\frac{\log T_f - \log T_0}{F}) \tag{4.2.7}$$

Note that Δ_{ave} (Equation (4.2.4)) is obtained by applying the Make-Small-Change procedure to construct M new neighbours (S') to the initial route set (S). In this way M values for $C(S')-C(S)$ are obtained, and their magnitude can be averaged to obtain an estimate for Δ_{ave}. We use this estimate to help determine the starting temperature, the final temperature and the cooling schedule. The neighbouring solutions generated during this parameter initialization phase are subsequently discarded.

In this study we use an "inner loop", with L iterations per temperature, in addition to the "outer loop". The outer loop implements the cooling schedule, while the inner loop gives the SA a chance to search the solutions space at each temperature.

4.3 Experimental Results

To the best of our knowledge, Mandl's network [73] is the only generally available benchmark problem instance (see Figure 4.3). In our first set of experiments we use Mandl's network to compare our results against those of other researchers. Although our objective function is different from those used by other researchers, we are nevertheless able to make direct comparisons on the basis of common criteria, discussed below.

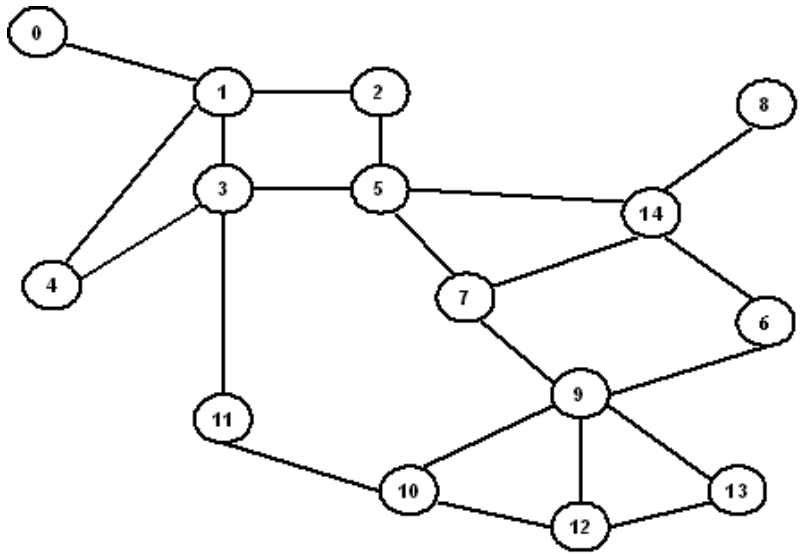

Figure 4.3. Mandl's Swiss Transit Network

4.3.1 Assessment Parameters

As mentioned before, the following parameters are used to compare the quality of our route sets with those obtained by Mandl [74], Baaj and Mahmassani [12], Kidwai [64], Chakroborty and Dwivedi [21].

d_0 - The percentage of demand satisfied without any transfers.

d_1 - The percentage of demand satisfied with one transfer.

d_2 - The percentage of demand satisfied with two transfers.

d_{un} - The percentage of demand unsatisfied.

ATT - Average travel time in minutes per transit user (mpu). This incorporates transfer waiting times, at 5 minutes per transfer.

The above parameters are quite easily calculated from the best route set generated at the end of an optimization run of our HC or SA algorithms. Recall that our objective function is composed of two components: 1) a component concerned with total travel time, accumulated

over all passengers, and 2) a similar accumulated term for the total number of transfers made between vehicles by passengers. An average travel time can be obtained simply by dividing the accumulated travel time by the total demand. However, unlike other researchers, the travel times we use in our optimization process do not make any allowance for transfer waiting times.

To obtain values for average travel times (ATT), comparable with other researchers, it is necessary to add 5 minutes for each person-transfer to our accumulated travel times before dividing by the total demand. However, this is not as straightforward as it seems. We have discovered that different values for ATT can be obtained, depending on whether or not transfer times are included in the shortest path calculations when determining the travel paths for passengers on the final route network. We tried two different ways of calculating ATT from a given route set:

1. Assume passengers ignore transfer waiting times when choosing their travel paths.

2. Assume passengers take account of transfer waiting times when choosing their travel paths.

Method 1 defines the mode of travel path selection used in our objective function. Evaluating ATT for our best route set at the end of a run (to make it comparable with values quoted by other researchers), involves adding five minutes for each person-transfer to the total travel time, before dividing by the total demand. Method 2, on the other hand, effectively gives the passengers fuller information. In these circumstances individuals will surely choose to avoid transfers, where this

will delay arrival at the final destination. As an added bonus, method 2 can reduce the total number of transfers. Thus ATT is calculated by accumulating all shortest travel paths, with transfer time included explicitly in the shortest path calculations.

In any case, our assessment routine will retrace the shortest paths from each source to destination node pair, using the distance (time) matrix computed for that particular route network, incorporating waiting times, or not, depending on the calculation model chosen. As each route is retraced, we can record which part of the shortest travel path belongs to which route in the best route set. Hence we can discover the number of transfers which passengers need to make to travel on their shortest path. Finally, with the demand of each path, we can respectively calculate the number of passengers who need 0, 1, 2 transfers.

To validate our calculations for the route set quality parameters, we examined Mandl's best route set (4 routes) from [73]. The routes (from the network shown in Figure 4.3) are listed below:

0-1-2-5-7-9-10-12

4-3-5-7-14-6

11-3-5-14-8

12-13-9

That we were able to replicate his values for d_0, d_1, d_2, d_{un} and ATT using method 2, is illustrated in Table 4.1. Thus, method 2 will be used to evaluate our final route sets in all our experiments. Interestingly (but not surprisingly) method 2 gives results that are at least as good (and probably better) than method 1, as can been seen in Table 4.1. Method

Table 4.1. Performance Measures for Mandl's Best Route Set

Parameters	Method 1	Method 2	Mandl's Results
d_0	66.67	69.94	69.94
d_1	26.33	29.93	29.93
d_2	7.00	0.13	0.13
d_{un}	0.00	0.00	0.00
ATT	13.29	12.90	12.90

2 produces a smaller value for ATT and a larger percentage of travellers reach their destinations with zero transfers.

4.3.2 Weighting Parameters for the Objective Function

As mention in Section 3.1, in our research the A and B (constants used to weight the two components of the objective function) are chosen to ensure the two parts of the objective function are of similar magnitude. In order to show the effect on the quality of the route sets by changing the A and B, we use different value of the A and B to test Mandl's network by considering the 4 routes and maximum 8 nodes for each route situation. At the same time, we evaluate the parameters d_0, d_1, d_2, d_{un} and ATT on the final results, as previously. calculated (see Table 4.2).

Table 4.2. Results of Different Weighting in Objective Function for Mandl's Network

	Travel time:Transfers contribution ratio				
Parameters	$A = 0$	$1:3$	$1:1$	$3:1$	$B = 0$
d_0	92.96	91.43	93.26	88.51	79.44
d_1	7.04	7.99	6.74	11.07	20.23
d_2	0.00	0.58	0.00	0.42	0.33
d_{un}	0.00	0.00	0.00	0.00	0.00
ATT	12.66	11.99	11.37	11.79	12.10

From the above experiments, it is clear that the two factors, better

results are obtained when A and B have equal importance. So we adopted this ratio for our remaining experiments.

4.3.3 Results for Mandl's Swiss Transit Network

In order to establish the viability of our approach we first compare the results (published in [36]) obtained by running our algorithms with those previously published by Mandl [74], Baaj and Mahmassani [12], Kidwai [64] and Chakroborty [20]. For consistency with the existing work, the route sets were developed for Mandl's network (see Figure 4.3) in four situations: 4 routes, 6 routes, 7 routes and 8 routes in each route set. In line with the previous authors, a transfer penalty of 5 minutes is added to the travel time of every passenger (on the final route set) for each time a transfer is made, as discussed in Section 4.3.1. We also set a maximum of eight nodes in each route. We carried out 10 replicate runs for each algorithm in each situation (i.e., 4, 6, 7, and 8 routes). For hill-climbing (HC) we used 100,000 iterations, and we performed 1,000 cooling steps, with 100 iterations within the inner loop, and $P_0 = 0.999$, $P_f = 0.001$, $M = F = 1,000$, and $L = 100$ for the simulated annealing (SA). In addition, we also compare our results with the lower bound result of the average travel time (ATT) on Mandl's network (introduced in Section 3.3). The difference between the ATT of our best route sets obtained and the "ideal" ATT (i.e., lower bound) is quoted as a percentage of the "ideal" quantity in our results called "ATT-error".

The results in Table 4.3 clearly show that competitive results have been found by our algorithms. Our results have better values for d_0, d_1 in 3 out of 4 cases. For the average travel time (ATT) our results beat

previous researchers' results for the 4 route and 8 route cases, and they are only marginally inferior to those published by Chakroborty [21] for the 6 and 7 route cases. Generally average solutions for the SA are slightly better than those obtained using the HC. This is perhaps to be expected, given the SA is the more sophisticated algorithm. The actual routes produced for our best solutions for each situation are presented in Table 4.4.

Table 4.5 gives the average run times for our hill-climbing and simulated annealing algorithm. Note: our computer platform is Windows XP with Intel(R) Pentium(R) D CPU 3.00GHz and 1GB of RAM. Clearly, the HC is slightly faster than the SA .

Table 4.3. Results for Mandl's Network

Number of Routes	Parameters	Mandl [74]	Baaj and Mahmassani [12]	Kidwai [64]	Chakroborty [20]	Our Best	ATT Error%	HC Average	SA Average
4	d_0	69.94	-	72.95	86.86	***93.26***	13.59	91.83	92.48
	d_1	29.93	-	26.92	12.00	***6.74***		8.17	7.52
	d_2	0.13	-	0.13	1.14	***0.00***		0.00	0.00
	d_{un}	***0.00***	-	***0.00***	***0.00***	***0.00***		0.00	0.00
	ATT	12.90	-	12.72	11.90	***11.37***		11.69	11.55
6	d_0	-	78.61	77.92	86.04	***91.52***	4.70	90.23	90.87
	d_1	-	21.39	19.68	13.96	***8.48***		9.26	8.74
	d_2	-	***0.00***	2.40	***0.00***	***0.00***		0.51	0.39
	d_{un}	-	***0.00***	***0.00***	***0.00***	***0.00***		0.00	0.00
	ATT	-	11.86	11.87	***10.30***	10.48		10.78	10.65
7	d_0	-	80.99	***93.91***	89.15	93.32	4.10	92.21	92.47
	d_1	-	19.01	***6.09***	10.85	6.36		7.13	6.95
	d_2	-	***0.00***	***0.00***	***0.00***	0.32		0.66	0.58
	d_{un}	-	***0.00***	***0.00***	***0.00***	***0.00***		0.00	0.00
	ATT	-	12.50	10.69	***10.15***	10.42		10.74	10.62
8	d_0	-	79.96	84.73	90.38	***94.54***	3.50	93.23	93.65
	d_1	-	20.04	15.27	9.62	***5.46***		6.18	5.88
	d_2	-	***0.00***	***0.00***	***0.00***	***0.00***		0.59	0.47
	d_{un}	-	***0.00***	***0.00***	***0.00***	***0.00***		0.00	0.00
	ATT	-	11.86	11.22	10.46	***10.36***		10.69	10.58

Table 4.4. Routes Obtained Using Our Methods

Situation	Number of Routes	Route Description
1	4	9-13-12-10-11-3-1-0
		11-10-9-7-5-2-1-0
		10-9-7-5-3-4-1-2
		1-2-5-7-9-6-14-8
2	6	12-13-9-10-11-3-5-7
		10-12-9-6-14-5-2-1
		8-14-5-2-1-3-11
		0-1-2-5-7-9-10-11
		4-3-11-10-9-6-14-8
		10-9-7-5-3-4-1
3	7	12-13-9-7-5-3-4-1
		11-10-12-13-9-6-14-8
		8-14-5-2-1-4
		3-1-2-5-14-6-9-12
		4-3-11-10-9-7-14-6
		9-10-11-3-5
		12-13-9-7-5-2-1-0
4	8	9-13-12-10-11-3-4
		6-9-7-5-3-4-1-0
		9-10-11-3-5-14-8
		8-14-6-9-10-11-3
		11-3-1-2-5-7-14
		9-6-14-5-2-1-3
		9-13-12-10-11-3-1-0
		0-1-2-5-7-9-12-13

Table 4.5. Average Run Times for the HC and SA Algorithms.

Number of Routes	HC Time (secs)	SA Time (secs)
4	254	315
6	244	302
7	232	289
8	221	267

4.3.4 Scalability Experiments

Because of a lack of published benchmarks, it is necessary to create our own data to establish whether the techniques would scale to larger instances. For these tests we used the data sets that have been generated

by the DSGM in Table 4.6 (introduced in Section 3.5).

Table 4.6. Test Data Sets

Network	Number of Nodes	Number of Links
I	70	175
II	70	245
III	110	275
IV	110	385
V	130	325
VI	130	455

Firstly, the lower bound and other parameters that we introduced in Chapter 3 are very useful in the present context because we are dealing with new data. We need to determine various constraints, such as the maximum number of nodes for each route in the route set, and also provide values against with we can evaluate the quality of the route sets generated by our algorithms. The details of the parameters and lower bounds for these data sets are shown in Table 4.7 (where "Max Nodes" represents the maximum number of nodes in any of the shortest paths).

Table 4.7. Lower Bound Parameters

Network	Total Demand	Total Person Travel Time	Average Travel Time	Max Nodes
Mandl's	15570	155790	10.01	8
I	1212620	36042456	29.72	17
II	1204596	34911176	28.98	13
III	3603360	121935974	33.84	29
IV	3613416	111143216	30.76	19
V	6695550	190533810	28.46	21
VI	6664344	175991432	26.41	20

Furthermore, it is also necessary to decide the constraints for these data sets before undertaking the experiments, such as the number of routes, and the minimum and maximum number of nodes for each

route. We considered these constraints from two separate viewpoints: 1) an analytical point of view and 2) a practical viewpoint based on the examination of the four real situations described in Chapter 3. From an analytical point of view, the minimum number of nodes for each route was set to 2 and the maximum number was determined by the number of nodes of the longest "shortest path" route (introduced in Section 3.4). Once the route length constraints had been decided, we used the real networks from Chapter 3 to help guide us regarding the number of routes to include in a route set. This was done by evaluating the average numbers of routes meeting/crossing at each node and using Equation 3.5.1 (introduced in Section 3.5). Problems 1, 3, 5, 7, 9 and 11 in Table 4.8 are guided from an analytical point of view. From a practical viewpoint, we simply identified the individual bus routes on our four route maps, counted these routes and then counted the nodes on each of the routes, registering the maximum and minimum node counts (problems 2, 4, 6, 8, 10 and 12 in Table 4.8).

Table 4.8. Experimental Conditions

Problem	Network	Number of Routes	Number of Nodes in Route	AFN
1	70 nodes and 175 links	35	2 - 17	5.00
2	70 nodes and 175 links	15	10 - 30	4.29
3	70 nodes and 245 links	44	2 - 13	5.00
4	70 nodes and 245 links	15	10 - 30	4.29
5	110 nodes and 275 links	55	2 - 29	8.00
6	110 nodes and 275 links	56	10 - 22	8.15
7	110 nodes and 385 links	80	2 - 19	8.00
8	110 nodes and 385 links	56	10 - 22	8.15
9	130 nodes and 325 links	98	2 - 21	9.00
10	130 nodes and 325 links	60	12 - 25	8.77
11	130 nodes and 455 links	106	2 - 20	9.00
12	130 nodes and 455 links	60	12 - 25	8.77

Hill-climbing and Simulated Annealing algorithms were both used to test the above problems. Before these experiments were carried out, parameters for implementing the two algorithms were determined (see Table 4.9). (Please note that the cooling schedule has been chosen to ensure an acceptable run time. It may be possible that if the different cooling schedule is used, different solutions may be obtained.)

Table 4.9. Experimental Parameters I

Problem	Objective Parameters	HC Parameters	SA Parameters
1,2	$A = 20000000$ $B = 1800000$	100000 Iterations	$P_0 = 0.999$ $P_f = 0.001$ $M = F = 1,000$ $L = 100$
3,4	$A = 19000000$ $B = 1500000$	100000 Iterations	$P_0 = 0.999$ $P_f = 0.001$ $M = F = 1,000$ $L = 100$
5,6	$A = 80000000$ $B = 8000000$	10000 Iterations	$P_0 = 0.99$ $P_f = 0.01$ $M = F = 100$ $L = 100$
7,8	$A = 70000000$ $B = 7000000$	10000 Iterations	$P_0 = 0.99$ $P_f = 0.01$ $M = F = 100$ $L = 100$
9,10	$A = 90000000$ $B = 10000000$	10000 Iterations	$P_0 = 0.99$ $P_f = 0.01$ $M = F = 100$ $L = 100$
11,12	$A = 90000000$ $B = 12000000$	10000 Iterations	$P_0 = 0.99$ $P_f = 0.01$ $M = F = 100$ $L = 100$

As in previous experiments, parameters such as d_0, d_1, d_2 and ATT were used to assess best route sets found by the hill-climbing or simulated annealing algorithm. At the same time, the average values and

the standard deviations (SD) of these parameters (10 runs), as well as the run time using the HC and SA, were also recorded. In addition, a simple method to assess the quality of the best route set found for each test problem is to compare the average travel time (ATT) per passenger with the lower bound results. The difference between the ATT of the best route set obtained and the "ideal" ATT (i.e., lower bound) for each problem is quoted as a percentage of the "ideal" quantity in our results called "best ATT-error". Hence the best ATT-error can be calculated thus:

$$(ATT_{best} - ATT_{ideal}) \times 100/ATT_{ideal} \tag{4.3.1}$$

Finally all the best route sets for these test problems can be seen on our website (http://users.cs.cf.ac.uk/L.Fan/) and their assessment results are shown in Table 4.10, 4.11, 4.12. It is clear that the "best ATT-errors" of the best route sets found for test problems have very similar values. The percentage errors are between 2.25% and 12.35%. On the other hand, comparing hill-climbing with our simulated annealing algorithm, the SA algorithm runs slower but finds better solutions for all the instances. In addition, the standard deviations of these parameters for 10 experiments are small, hence it demonstrates that our algorithms are robust.

In the final set of experiments we examined the efficiency of the Make-Small-Change procedure. Throughout the execution of our algorithms, each time a neighbourhood route set is generated, there is a chance that it will be infeasible (i.e., not connected). Hence the Make-Small-Change procedure will be called iteratively, until a connected

Table 4.10. Results for Network I and II

Problem	Assessment Parameters	Best Results	HC Average	SD for HC	SA Average	SD for SA	Best ATT Error%	SA Average Run Time (secs)	HC Average Run Time (secs)
1	d_0	53.70	50.68	1.07	52.12	1.18	9.72	36598	35344
	d_1	34.86	36.43	1.43	37.34	1.62			
	d_2	11.44	12.89	1.21	10.54	1.35			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	32.61	35.53	1.16	34.45	1.08			
2	d_0	43.26	40.12	1.56	41.56	1.85	12.35	45442	44379
	d_1	40.22	37.77	1.84	39.32	1.74			
	d_2	16.52	22.11	1.43	19.12	1.69			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	33.39	36.72	1.22	35.89	1.38			
3	d_0	53.68	50.44	2.05	52.32	1.94	9.52	24883	23617
	d_1	37.42	35.79	1.89	36.86	2.12			
	d_2	8.90	13.77	1.86	10.82	1.72			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	31.74	33.98	1.64	32.88	1.58			
4	d_0	56.66	53.33	1.87	54.46	2.19	9.25	14296	13029
	d_1	33.02	31.24	2.36	35.01	1.98			
	d_2	10.32	15.43	1.47	10.53	2.24			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	31.66	34.55	1.52	33.42	1.32			

Table 4.11. Results for Network III and IV

Problem	Assessment Parameters	Best Results	HC Average	SD for HC	SA Average	SD for SA	Best ATT Error%	SA Average Run Time (secs)	HC Average Run Time (secs)
5	d_0	72.91	69.98	1.85	70.64	1.56	2.25	15938	13726
	d_1	20.56	21.47	2.36	22.33	1.85			
	d_2	6.54	8.55	1.88	7.03	2.01			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	34.60	36.65	1.04	35.82	1.36			
6	d_0	71.21	68.49	2.42	69.94	1.45	5.44	19351	17456
	d_1	20.71	21.66	2.16	21.12	1.54			
	d_2	8.08	9.85	1.81	8.94	1.98			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	35.68	37.43	1.52	37.26	1.23			
7	d_0	74.82	70.21	1.76	72.24	2.33	5.10	14371	12564
	d_1	18.94	16.45	1.82	17.82	2.13			
	d_2	6.24	13.34	1.92	10.56	1.99			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	32.33	35.78	1.29	34.12	1.68			
8	d_0	76.17	71.89	1.78	73.34	1.78	4.71	12018	10878
	d_1	18.77	17.46	1.84	17.32	1.67			
	d_2	5.06	10.65	1.84	9.34	2.31			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	32.21	36.98	1.61	35.17	1.62			

Table 4.12. Results for Network V and VI

Problem	Assessment Parameters	Best Results	HC Average	SD for HC	SA Average	SD for SA	Best ATT Error%	SA Average Run Time (secs)	HC Average Run Time (secs)
9	d_0	78.35	74.62	1.82	76.51	1.63	4.60	49538	47326
	d_1	16.96	15.33	2.02	17.24	1.44			
	d_2	4.69	10.05	1.67	6.25	1.95			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	29.77	32.15	1.06	31.41	1.22			
10	d_0	73.58	70.64	1.76	71.92	1.43	5.31	15550	13412
	d_1	22.52	20.83	1.84	21.49	1.64			
	d_2	3.90	8.53	1.68	6.59	1.48			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	29.97	33.07	1.24	32.15	1.25			
11	d_0	75.13	71.48	1.72	72.55	1.88	5.72	20058	18193
	d_1	19.43	18.29	1.58	20.34	1.79			
	d_2	5.44	10.23	1.91	7.11	2.13			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	27.92	31.87	1.11	29.14	1.06			
12	d_0	75.11	71.89	2.14	73.45	1.97	5.45	14656	12318
	d_1	20.96	18.44	2.36	19.56	1.98			
	d_2	3.93	9.67	1.89	6.99	1.94			
	d_{un}	0.00	0.00	0.00	0.00	0.00			
	ATT	27.85	30.71	1.13	28.97	1.68			

Table 4.13. Make-Small-Change Procedure Tests

Network	Route Set Condition	Feasible Fraction	Time (milliseconds)
Mandl's Network	4 routes, minimum 2 nodes and maximum 8 nodes per route	1 / 7	0.020310
	8 routes, minimum 2 nodes and maximum 8 nodes per route	1 / 4	0.012255
70 nodes and 175 links	15 routes, minimum 2 nodes and maximum 17 nodes per route	1 / 22	0.064868
	35 routes, minimum 10 nodes and maximum 30 nodes per route	1 / 12	0.029962
70 nodes and 245 links	15 routes, minimum 2 nodes and maximum 13 nodes per route	1 / 21	0.059565
	44 routes, minimum 10 nodes and maximum 30 nodes per route	1 / 10	0.025583
110 nodes and 275 links	25 routes, minimum 2 nodes and maximum 29 nodes per route	1 / 31	0.096923
	56 routes, minimum 10 nodes and maximum 22 nodes per route	1 / 19	0.058643
110 nodes and 385 links	56 routes, minimum 2 nodes and maximum 19 nodes per route	1 / 24	0.071132
	80 routes, minimum 10 nodes and maximum 22 nodes per route	1 / 15	0.041556
130 nodes and 325 links	60 routes, minimum 2 nodes and maximum 21 nodes per route	1 / 28	0.081037
	98 routes, minimum 12 nodes and maximum 25 nodes per route	1 / 18	0.044735
130 nodes and 455 links	60 routes, minimum 2 nodes and maximum 20 nodes per route	1 / 27	0.079515
	106 routes, minimum 12 nodes and maximum 25 nodes per route	1 / 16	0.042464

route set is produced. Clearly, the efficiency of the Make-Small-Change procedure can be assessed by counting the number of iterations required before connectivity is achieved. Here we examine the average number of iterations needed in order for each new feasible route set to be generated. To do this we examine single runs of the HC algorithm on our problem instances. The results are presented in Table 4.13. Column seven of the table records the average run time required, per solution, for the Make-Small-Change routine to produce a feasible solution.

From the experimental results, we can observe some interesting patterns: the efficiency of the Make-Small-Change routine appears to improve with increasing numbers of routes in a route set and also when the maximum and minimum number of nodes allowed per route is increased. This is perhaps not a surprising result. More routes and longer routes introduce more scope for redundancy.

Table 4.14. Experimental Parameters II

Situation	Test Problem	HC Parameters	SA Parameters
I	1	1000 Iterations	$P_0 = 0.9$ $P_f = 0.1$ $M = F = 10$ $L = 100$
II	1	10000 Iterations	$P_0 = 0.99$ $P_f = 0.01$ $M = F = 100$ $L = 100$
III	2	1000 Iterations	$P_0 = 0.9$ $P_f = 0.1$ $M = F = 10$ $L = 100$
IV	2	10000 Iterations	$P_0 = 0.99$ $P_f = 0.01$ $M = F = 100$ $L = 100$

Table 4.15. Results for Different Cooling Schedule and Temperature Choices

Situation	Assessment Parameters	Best Results	HC Average	SA Average	SA Average Run Time (secs)	HC Average Run Time (secs)
I	d_0	32.41	30.12	31.21	402	378
	d_1	38.33	37.83	38.05		
	d_2	29.26	32.05	30.74		
	d_{un}	0.00	0.00	0.00		
	ATT	38.43	40.31	39.89		
II	d_0	44.06	42.54	43.27	3969	3777
	d_1	38.11	36.28	37.64		
	d_2	17.83	21.18	19.09		
	d_{un}	0.00	0.00	0.00		
	ATT	35.49	37.43	36.54		
III	d_0	24.18	22.56	23.33	442	423
	d_1	39.14	37.79	39.02		
	d_2	36.68	39.65	37.65		
	d_{un}	0.00	0.00	0.00		
	ATT	39.47	41.34	39.98		
IV	d_0	36.78	34.43	35.36	4563	4398
	d_1	37.77	35.45	36.82		
	d_2	25.45	30.12	27.82		
	d_{un}	0.00	0.00	0.00		
	ATT	36.69	38.09	37.21		

As mentioned before, the number of iterations for each HC and SA experiments and the cooling schedule for the SA was chosen to ensure good solutions in reasonable run times. However, it is clear that changing these parameters may effect the quality of the results obtained. In particular longer run times and slower cooling schedules are likely to produce better results. In order to show the effect, we use different parameter choices to test Problems 1 and 2 in Table 4.8. Similar to the above experiments, we also use 10 runs and record the values of the assessment parameters for the best solutions, average value for HC and SA and average run time for each algorithm. The experimental parameters and the results can be seen in Table 4.14 and 4.15.

From the the above results obtained by different numbers of iterations, cooling schedules and temperature choices, it is clear that longer run times and slower cooling schedules produce better results. This shows that there is a trade-off between run time and solution quality.

4.4 Summary

In this chapter, we have presented a framework for solving the UTRP, consisting of the following components: a *representation* for the problem, an *initialization procedure* to construct initial route sets, and a *Make-Small-Change* routine to generate neighbourhood moves. To test our techniques, we have implemented two simple algorithms: hill-climbing and simulated annealing, and embedded their simple search mechanisms into our metaheuristic framework. Furthermore, we have demonstrated the effectiveness of our scheme, by beating previously published results for the only benchmark problem we have been able to locate. In addition, the potential for solving larger problem instances

has been explored. We have also demonstrated the effectiveness of our Make-Small-Change procedure to generate feasible solutions to the UTRP. Finally, we have demonstrated the inevitable trade-off that occurs between run time and solution quality.

Chapter 5

A SIMPLE MULTI-OBJECTIVE OPTIMIZATION ALGORITHM FOR THE UTRP

The urban transit routing problem is an NP-Hard, highly constrained, multi-objective problem. In this chapter (based on the work published in [66]), we propose a simple evolutionary multi-objective optimization technique to solve the UTRP. Firstly we investigate an improved route set initialization procedure, which can be used in our simple multi-objective optimization algorithm instead of the previous one. Secondly we briefly introduce the idea of multi-objective optimization, then present our two key objectives, which are to minimize both passenger costs and operator costs. Following this, we describe a simple multi-objective optimization algorithm for the UTRP, then present experimental results obtained using the Mandl's benchmark data and some larger networks generated by ourselves.

5.1 Improving the Route Set Initialization Procedure

Our metaheuristic approach (introduced in Chapter 4), requires only a single initial route set to seed both hill-climbing and simulated annealing. For this reason even if an efficient initialization procedure is used, we would not expect the performance of our metaheuristic approach to be improved remarkably. However, in our simple multi-objective optimization algorithm (introduced later on), the initialization procedure is required to generate many initial feasible route sets. Hence it is clear that our multi-objective approach may benefit from a more efficient route set initialization procedure. Before considering the improvement of the route set initialization procedure, however, we will discuss some important issues relating to route set quality. We are particularly interested in assessing route sets for their level of infeasibility, i.e., how "close" a route set is to being "feasible". Following this discussion, we present a modified initialization routine, and finally we will compare it with the previous procedure used. Our comparisons include an assessment of the quality of the route sets, in terms of the proportion of feasible and "almost feasible" route sets generated by the two initialization procedures.

5.1.1 Assessing the Extent of Infeasibility of A Route Set

Recall that a feasible route set must contain every node present in the original transit network and provide passengers with travel routes from every source to every destination. Infeasible route sets will have one or more nodes missing, or some of the routes will not be connected to the rest of the framework. For our purposes though, infeasible route sets may still be useful, if they are easily made feasible by our *Make-*

Small-Change routine (introduced in Section 4.1.4). For this reason we are interested in assessing the level of infeasibility, and we consider this issue next.

We use a simple measure to assess the level of infeasibility - we count the number of disconnected components in the graph obtained by superimposing all the routes in a route set. A feasible route set will consist of just one component, containing every node. Infeasible route sets will be made up of two or more components, where each component consists either of one or more routes, or of an individual node which does not occur in any route. The procedure for checking begins by first identifying all the individual connected components, then every node of in the transit network is checked to see whether it is missing from the route set. Finally, the total number of individual components and missing nodes is accumulated to give us the number of components for the infeasible route set.

For example, Figure 5.1 illustrates a simple 5 nodes and 7 links transit network, and Figures 5.2 and 5.3 show two networks constructed by overlaying infeasible route sets based on the transit network. Both networks are in two components, but Figure 5.2 contains all nodes included in the transit network, while Figure 5.3 has a connected component and an isolated node, absent from the route network.

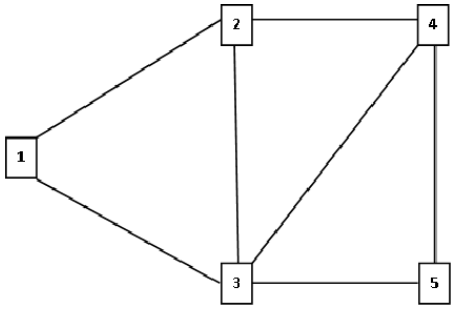

Figure 5.1. 5 Nodes and 7 Links Transit Network

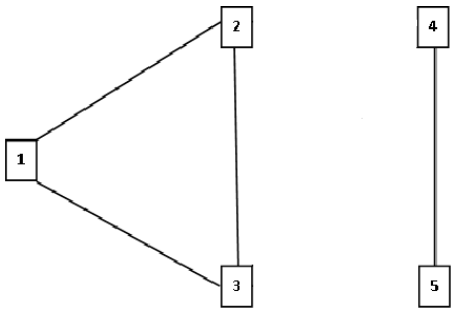

Figure 5.2. 2-component Infeasible Route Set Network I

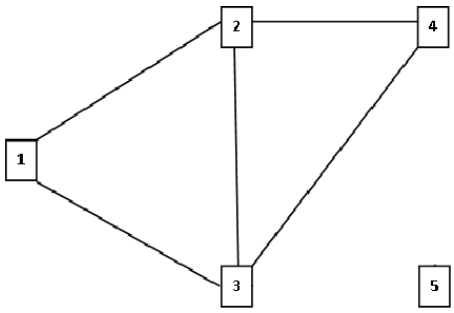

Figure 5.3. 2-component Infeasible Route Set Network II

Through the above description, it is clear that the number of components for a infeasible route set can be used to evaluate the degree of the infeasibility of that route set: assuming that the more components it has, the more difficult it will be to make it feasible. Thus, a 2-component infeasible route set should be relatively easy to convert. We will a call 2-component infeasible route set a "potentially feasible route set".

5.1.2 An Improved Route Set Initialization Procedure

As explained before, once our metaheuristic approach has generated a (random) route set, the *Feasibility Check* procedure is used to check the feasibility of the route set. If a route set is not feasible, then the *Make-Small-Change* procedure is utilized repeatedly until a feasible route set is obtained. It would appear that the successful generation

of a feasible route set relies heavily on the two procedures: *Feasibility Check* and the *Make-Small-Change*. On the other hand, the initial route set construction procedure guarantees that the individual routes are connected paths with no cycles or backtracks. Nevertheless, the collective feasibility of the route set as a whole is not considered at this stage. It is our conjecture that if we use an initial route set construction procedure capable of generating a high proportion of feasible and "potentially feasible" route sets, the initialization procedure as a whole should be more efficient, with fewer applications of the *Feasibility Check* and *Make-Small-Change* procedures needed. We will now consider an alternative route set construction procedure: *Initialization Procedure II*, based on the original version called *Initialization Procedure I* (see details in Section 4.1.2.)

In order to consider connectivity explicitly during the construction phase, and for the purpose of adding all nodes into the initial route set, a new initialization procedure is presented in Algorithm 10. Here, routes are constructed one at a time as before. However, this time we ensure the connectivity of the route set. Once the first route has been constructed, we will choose the start node of subsequent routes from the set of nodes present in previously constructed routes. Given that for two routes to be connected they must have at least one node in common, we can ensure the connectivity of the entire route set in this way. In addition, when selecting the next node for a route under construction, the new procedure favours nodes that do not already appear in the route set.

Algorithm 10 Initialization Procedure II

Parameters: number of routes, minimum and maximum number of nodes for routes
Begin
Main loop
repeat
Route length selection:
Choose a length for the route between minimum and maximum number of nodes at random
Inner Loop
repeat
Start node selection:
If this is the first iteration of the loop, **then** randomly choose any node as the start node,
Else Randomly choose a node from the previous route as a start node
label this node as the "previous node"
Next node selection:
Construct a node set consisting of all nodes directly connected to "previous node" that have not been selected so far for any route.
If this node set is not empty, choose a node from this set at random
Else construct a node set consisting of all nodes directly connected to "previous node" that have not been selected for the current route.
If this node set is not empty, choose a node from this set at random
Else invert the order of the route and repeat the ***Next node selection***
If such route can not reach the route length, **then** delete this route and repeat ***Start node selection***
until route length is reached
until number of routes is reached
Output an initialization route set

5.1.3 Comparison Experiments

It is clear that *Initialization Procedure I* is a very basic method, likely to produce unconnected route sets with missing nodes. We attempt to address these issues in *Initialization Procedure II*, by providing enhancements to improve the chances of route set connectivity, and reduce

the likelihood of missing nodes. Nevertheless it is important to establish the relative success of these routines empirically using comparison experiments.

For these experiments, seven data sets were used, and for each network, the two initialization procedures were each set up to produce 10,000 route sets under various user-selected constraints. The number of feasible route sets (1-component route sets) and "potentially feasible route sets" (2-component route sets) were then recorded for comparison. The constraints applied when generating the route sets include the number of routes and the maximum and minimum number of nodes for each route. The experimental conditions are summarized in Table 5.1. The results of the runs are shown in Figures 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10 (IP-I, IP-II represents the *Initialization Procedure I, II.*)

Table 5.1. Experimental Conditions

Problem	Network	Number of Routes	Number of Nodes in Route
I	Mandl's network	8	2-8
II	70 nodes and 175 links	35	2-17
III	70 nodes and 245 links	15	10-30
IV	110 nodes and 275 links	55	2-29
V	110 nodes and 385 links	56	10-22
VI	130 nodes and 325 links	98	2-21
VII	130 nodes and 455 links	60	12-25

For each experimental result, it is clear that *Initialization Procedure II* can produce more good route sets (feasible route sets plus 2-component route sets) and feasible route sets than *Initialization Procedure I*. At the same time, for *Initialization Procedure I*, the overall percentage of good route sets over all 7 problem instances is 71.80% while the the overall percentage of feasible route sets over all 7 problem instances is 49.77%. For *Initialization Procedure II*, the overall percent-

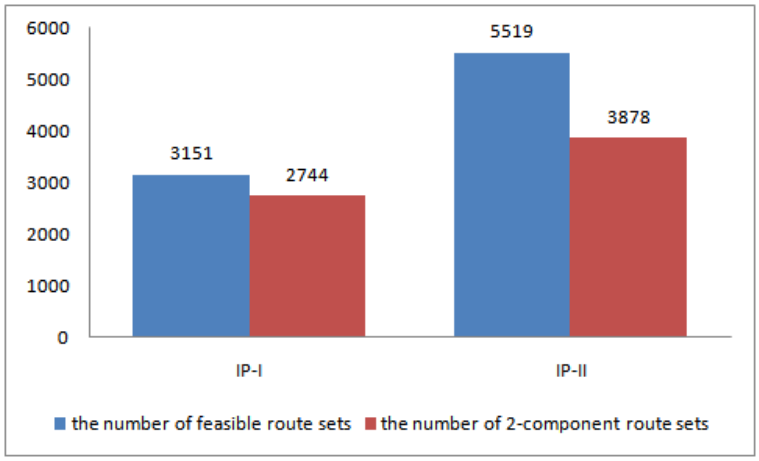

Figure 5.4. Mandl's Network Problem

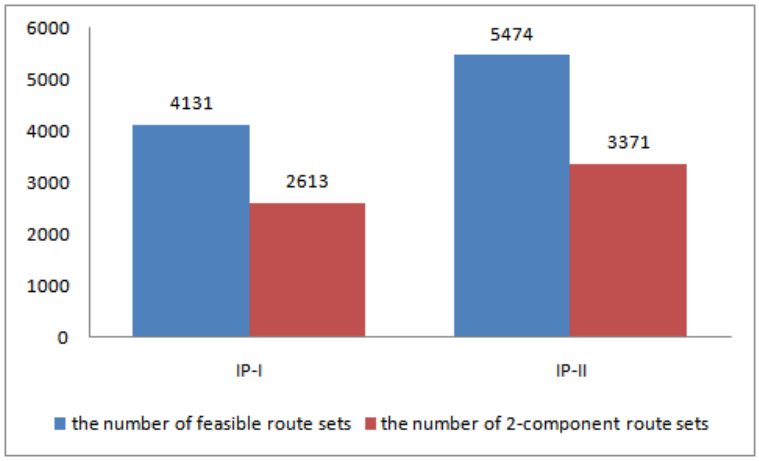

Figure 5.5. 70 Nodes and 175 Links Network Problem

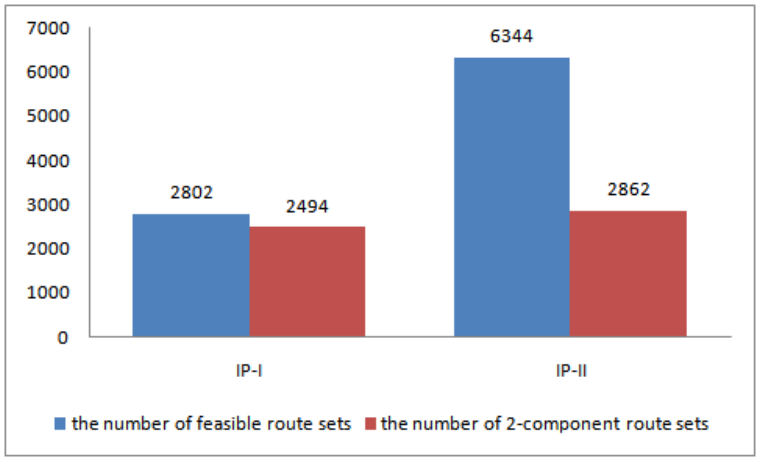

Figure 5.6. 70 Nodes and 245 Links Network Problem

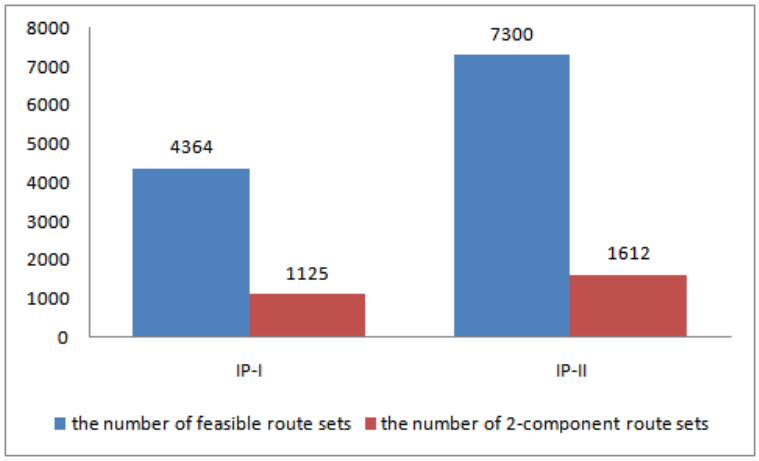

Figure 5.7. 110 Nodes and 275 Links Network Problem

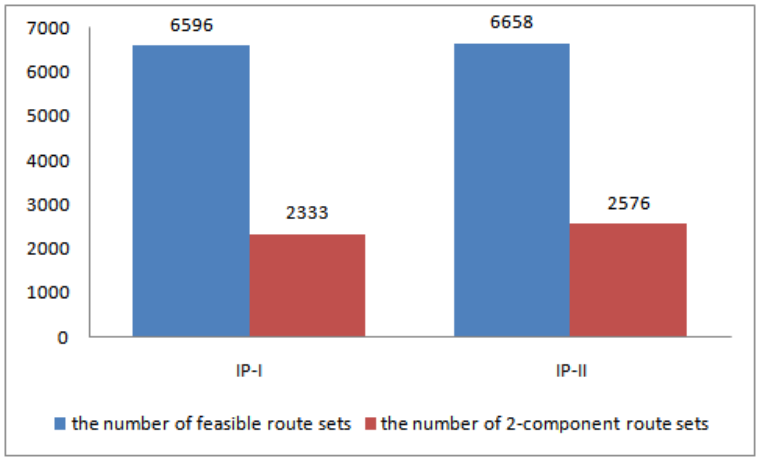

Figure 5.8. 110 Nodes and 385 Links Network Problem

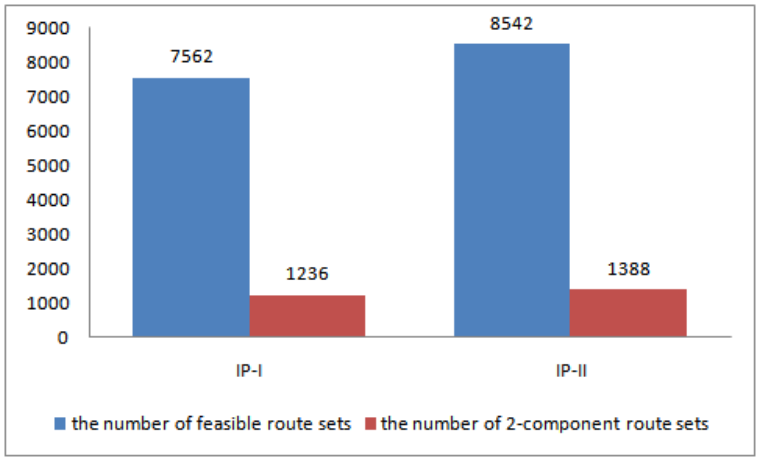

Figure 5.9. 130 Nodes and 325 Links Network Problem

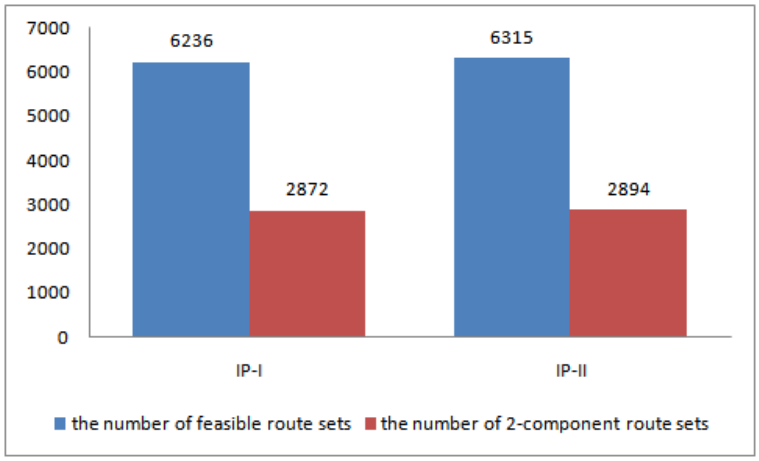

Figure 5.10. 130 Nodes and 455 Links Network Problem

age of good route sets over all 7 problem instances is 92.48% while the the overall percentage of feasible route sets over all 7 problem instances is 65.93%. The running time for generating 10000 route sets for each initialization procedure in different situations is also recoded (see Table 5.2). From Table 5.2 it would appear that the running time for *Initialization Procedure II* is slightly shorter than *Initialization Procedure I*. To sum up, *Initialization Procedure II* performs best of all: it can produce more feasible route sets with a shorter running time.

Table 5.2. Running Time Results

Problem	IP-I (millisecs)	IP-II (millisecs)
I	5412	5125
II	358844	274735
III	92782	87125
IV	2700943	2371100
V	1331257	1260163
VI	5065789	4181244
VII	2796938	2568843

In conclusion, we have shown that the improved route set initialization procedure can make a significant difference. Thus we will use the improved version for our multi-objective work in this chapter.

5.2 Introduction to Multi-Objective Optimization

Real-world problems often have multiple conflicting objectives. For example, for the UTNDP, the operators would ideally like to satisfy every passenger's requirements, but they also need to scrutinize the cost of providing the service. For the UTRP there is no single solution that is the best when measured on all objectives. Problems such as these are examples of a special class of optimization problem called multi-objective optimization problems. The question is, what is an optimal solution for a multi-objective problem? In general, it is called a *Pareto optimal solution* if there exists no other feasible solution which would decrease some objectives (assuming a minimization problem) without causing a simultaneous increase in at least one other objective [24].

With this definition of optimality, we usually find several trade-off solutions. These solutions are called the *Pareto optimal set* (after Vilfredo Pareto [83]), or the *Pareto optimal front* for the plot of the vectors corresponding to these solutions. *Pareto-optimal solutions* are *non-dominated solutions* in the sense that it is not possible to improve the value of any one of the objectives in such a solution, without simultaneously degrading the quality of one or more of the other objectives in the vector [28]. In that sense, the search for an optimal solution has fundamentally changed from what we see in the case of single-objective problems. The task of solving multi-objective optimization problems is called multi-objective optimization. Since users generally need only one solution from the set of optimal trade-off solutions, solving multi-objective optimization problems can be seen as a combination of both searching and decision-making [55].

Through the literature review, we have discovered very few research

papers on the use of multi-objective optimization techniques for the UTRP. Among those that exist, Ceder and Israeli [56] introduced a complex seven-stage approach, which includes several steps to create routes, identify transfers and calculate vehicle frequencies. A number of objectives such as travel time, waiting time, empty space and fleet size were then identified, and a set of multi-objective tradeoff solutions is presented to a human decision maker. Fan and Machemehl [37] also proposed a multi-objective decision-making approach to the UTNDP. Their basic idea is to experiment with different values for the weights of three objective functions, in order to obtain a range of non-dominated results from which a human decision maker can select a suitable compromise solution. Although this multi-objective optimization method is able to find some good solutions to the UTNDP, determining suitable values for the weights requires a large number of experiments, which can be very time consuming. Thus a generic and computationally efficient multi-objective optimization method to solve the UTNDP is desirable.

5.3 Our Two Key Objectives

The urban transit routing problem (UTRP) is a multi-objective problem which usually involves several objectives, for example the total travel time and the total transfer time from the passenger's point of view, and the number of routes and the total sum of all the bus route lengths from the operator's point of view. In reality, transport planners have to develop transit routes based on the practical requirements specified by the various stakeholders, for example bus companies and local government. This can involve the simultaneous consideration of multiple objectives as well as multiple constraints.

In this chapter we extend our simple model from Chapter 3, to cover operator costs and passenger costs simultaneously. As we have already established, passengers normally prefer to travel to their destination in the shortest possible time or along the shortest path. At the same time, passengers want the number of vehicle transfers kept to a minimum, since changing busses or trains involves extra waiting time and inconvenience.

From the operators' perspective, running costs will be an important consideration, which include the number of buses and drivers required and the total number of miles covered. In the real world, a certain level of service frequency will be required on each transit route to guarantee a reasonable quality of service for passengers (e.g., to ensure that passengers are not waiting excessive times for transfers). In order to satisfy the basic requirements, the operators should have sufficient vehicles and drivers to serve passengers on different transit routes. For example, if there are two transit routes, on which the required frequencies are the same but where one route is longer than the other, the operator will need to provide more buses and drivers on the longer route than the short route if the same level of service is to be maintained. At the same time, the fuel costs for the longer route will be higher than for the short route. Furthermore, it is clear that an important aspect of the operator's costs is related to the accumulated lengths of all the routes they operate. We will call this accumulated route length the "total-route-length", and use it as the operators' objective in our simple multi-objective model; passenger costs and operator costs will be traded off as dual objectives by our multi-objective evolutionary algorithm. At the same time, our problem constraints will ensure that all

the demand is met and a reasonable quality of service is maintained, so that only feasible solutions will be generated (avoiding the solution having zero passenger and operator costs for a service with no routes in the route set). The objective function for passenger costs, C_P in our multi-objective model is the same as that used in our single objective approach (see Section 3.1). The formulation for operators' costs is shown as follows:

- Let r denote the total number of routes in the route set.
- Let L_l denote the length of route l

$$Operators' costs : C_O = \sum_{l=1}^{r} L_l \tag{5.3.1}$$

5.4 A Simple Multi-Objective Optimization Algorithm for the UTRP

We present the *Simple Multi-Objective Optimization* (SMO) algorithm to solve our UTRP, shown as Algorithm 11. This scheme is based on the SEAMO algorithm [78, 97], but without the crossover operator.

Our SMO relies on the *Route Set Initialization Procedure II* to generate the initial population of route sets (introduced in Section 5.1.2), the *Make-Small-Change* procedure to modify an existing route set and the *Feasibility-Check* procedure to ensure that a feasible route set is obtained (introduced in Section 4.1).

In our algorithm, firstly the number of initial feasible route sets defined by the user is generated by the *Route Set Initialization Procedure II*. At the same time, best route sets for passenger and operator costs

Algorithm 11 *Simple Multi-Objective Optimization (SMO)*

Generate initial population of feasible route sets.
Calculate passenger and operator costs for each route set.
Record the best-route-set-so-far for both objectives.
repeat
  for each route set in the population
  Apply the *Make-Small-Change* procedure and *Feasibility-Check* procedure to produce a feasible offspring
  if offspring is a duplicate
  then delete offspring
  elseif offspring improves on either best-so-far
  then offspring replaces parent and best-so-far updated
  elseif offspring dominates parent
  then offspring replaces parent
  elseif offspring and parent are mutually non-dominated
  then find an individual in the population that is dominated by the offspring and replace it with the offspring.
  endif
  endfor
until the stopping condition is satisfied
print all non-dominated solutions in the final population.

are recorded. Secondly at each iteration of the loop, the *Make-Small-Change* and *Feasibility-Check* procedures are used to produce a feasible offspring. Next, the offspring is checked to see whether it is a duplicate (it has the same values for the two objective functions as an existing solution), and if so, then it is deleted. Next, if the offspring survives the duplicate test, its objective values are tested to see whether either improve on the "best-so-far" recorded for either passenger or operator costs. If a better value is found for either objective than has previously been discovered by the algorithm, the offspring will replace its parent regardless of the second objective. In this circumstance, the "best-so-far" is updated for the appropriate objective. Next, provided the offspring has not been deleted or replaced its parent, it will be tested to see whether it dominates its parent. If it does, it will replace the

parent. Finally, if the offspring has survived but has not passed any of the previous tests, the algorithm will check to see whether it and its parent are mutually non-dominated. If this is the case, an individual in the population dominated by the offspring will be found and replaced by the offspring. Offspring will be generated and tested in this way until the number of iterations defined by the user is achieved, then all non-dominated solutions will be returned as output.

5.5 Experimental Results

First we test the SMO algorithm on Mandl's Swiss transit network [73], then compare the results with those previously published in [36] (introduced in Section 4.3.3). Following this, we use the SMO algorithm and the metaheuristic approach published in [36] (introduced in Chapter 4) on these larger artificial instance of the UTRP (introduced in Section 4.3.4), where we rely on lower bound costs (see Table 4.7) to evaluate the performance of the SMO algorithm.

5.5.1 Experiments on Mandl's Network

In our experiments, similar to Chapter 4, we considered four separate scenarios for Mandl's network, namely route sets consisting of 4 routes, 6 routes, 7 routes and 8 routes, with a maximum of 8 nodes in each route. For each scenario, we recorded the results from 10 replicate runs (each seeded with different random numbers) using a population size of 200. The number of iterations of the multi-objective algorithm used in each experimental run for 4, 6, 7, 8 routes, was 1000, 3000, 4000, 5000 respectively. For each set of experimental runs, we accumulated the results and isolated the non-dominated solutions, trading off pas-

senger cost C_P against the operator cost C_O. Finally, we validated our solutions against our recently published results [36].

Table 5.3 shows the best route sets obtained for passengers and operators respectively, and compromise route sets (both objective values are "in the middle", which are not best for passengers or operators) for each of the 4 scenario on Mandl's network.

Note that many routes listed in for operators in Table 5.3 consist simply of source and destination nodes. Clearly, such short routes would probably prove uneconomic, in practice. Nevertheless, these 2 node routes are efficient in the context of our problem formulation, which uses a simplified objective function for the operators cost (the sum of the route lengths), and imposes a fixed number of routes in the route set (to comply with constraints used by other researchers). The present work can be viewed as a simple proof-of-concept study, and a more sophisticated formulation from the operators' viewpoint, would be needed to generate routes that comply with the practical requirements of bus operators.

As before, the following parameters [20] are used to evaluate our best route sets found by the multi-objective optimization algorithm from passengers' perspective:

- d_0 - Percentage demand satisfied without any transfers.
- d_1 - Percentage demand satisfied with one transfer.
- d_2 - Percentage demand satisfied with two transfers.
- ATT - Average travel time (minutes per passenger), including a penalty of 5 minutes per transfer.

Table 5.3. Routes Obtained Using the SMO Algorithm on Mandl's Network

Route No.	Best Routes for Passengers	Compromise Routes	Best Routes for Operators
4	13-12-10-9-7-5-3-4	4-3-5-14-6	1-2-5-7-14-6-9-10
	1-3-11-10-9-6-14-8	13-12-10-9-7-5-3-4	14-8
	10-9-7-5-2-1-0	0-1-2-5-7-9-10-11	13-12-10-11
	4-1-2-5-14-6-9	9-6-14-8	4-3-1-0
6	12-10-9-7-5-2-1-0	9-7-5-3-4	1-2-5-7-14-6-9
	6-14-5-2-1-3-4	9-13-12	8-14
	9-7-5-3-4	3-1-2	1-0
	12-13-9-10-11-3-1-0	1-2-5-14-6	13-12-10-9
	9-6-14-8	8-14-6-9	1-3-4
	11-10-12-13	0-1-2-5-7-9-10-11	10-11
7	6-14-7	9-7-5-2-1-0	8-14
	13-12-10-11-3-1-2	9-7-5-3-4-1-2	10-11
	11-10-9-6-14-8	11-10-9	3-4
	13-9-6-14-5-3-4	7-5-14-8	13-12-10
	9-7-5-3-4-1-2	2-1-3-11	1-3
	0-1-2-5-7-9-10-12	8-14-6-9-10-12-13	0-1-2-5-7-14-6-9
	3-1-0	2-5-14-6-9	10-9
8	1-3-11-10-12-13-9	9-13-12	3-1
	11-10-12-13-9-6-14-5	1-4	2-1
	4-1-2-5-7-14-8	8-14-7-5-2-1-3-4	1-0
	0-1-2-5-7-9-10-12	12-10-11	12-10
	11-10-12-13-9-7-5-3	14-6-9-10-11	3-4
	3-5-14-8	13-12-10-9-7-5-3-11	14-8
	4-3-5-7-9-10-12	6-14-7-5-2-1-0	11-10-9-6-14-7-5-2
	11-10-9-6-14-5-2-1	0-1-2-5-7-9-12	12-13

In addition, we also use the following parameter to evaluate our best route sets from operators' perspective:

- C_O - Operator cost function value (sum of the lengths of all the routes in the route set)

The values of these parameters for the route sets from Table 5.3 are presented in Table 5.4. To validate these results, we compare them against our previously published results [36] for the single objective situation (minimizing passenger costs). In addition, we also compare our results with the lower bound result of the average travel time (ATT) on Mandl's network (introduced in Section 3.3). The difference between the ATT of best route sets for passengers obtained and the "ideal" ATT (i.e., lower bound) is quoted as a percentage of the "ideal" quantity in our results called "ATT-error of best routes for passenger".

From Table 5.4, it is clear that a number of good route sets has been found from the passengers' point of view. The parameter values are very close to our previously published results for the single objective problem. However, the operator's costs in the "best for passenger" column, are consistently better than the corresponding costs obtained using our previous single objective approach. For the best route sets from the operator's perspective, it is reasonable that the lowest operator cost will correspond with the highest passenger's cost. In general, our multi-objective optimization algorithm can find good solutions to Mandl's network problem. In particular, assessment parameter values for the best route sets relative to the passenger's costs (such as d_0 and ATT), obtained by the SMO are close to our previously published results. We calculated the percentage difference of the SMO results relative to the previously published results, and found these values to

Table 5.4. The Best Results Obtained by Our SMO Algorithm on Mandl's Network (1) from the Passenger's Point of View and (2) from the Operator's Point of View and Compromise Routes

Routes No.	Parameters	Previously Published [36]	Best Routes for Passenger	Compromise Routes	Best Routes for Operator	ATT Error% of best routes for passenger
4	d_0	93.26	90.88	76.72	61.08	6.39
	d_1	6.74	8.35	10.34	36.61	
	d_2	0.00	0.77	12.94	2.31	
	ATT	11.37	10.65	12.55	13.88	
	C_O	147	126	101	63	
6	d_0	91.52	93.19	81.43	66.09	4.50
	d_1	8.48	6.23	11.38	30.38	
	d_2	0.00	0.58	7.19	3.53	
	ATT	10.48	10.46	11.62	13.34	
	C_O	215	148	112	63	
7	d_0	93.32	92.55	80.12	65.64	4.30
	d_1	6.36	6.68	10.83	26.20	
	d_2	0.32	0.77	9.05	8.16	
	ATT	10.42	10.44	11.89	13.54	
	C_O	231	166	121	63	
8	d_0	94.54	91.33	78.66	59.92	4.40
	d_1	5.46	8.67	9.45	21.97	
	d_2	0.00	0.00	11.89	18.11	
	ATT	10.36	10.45	12.15	13.57	
	C_O	283	245	143	63	

lie between 0.83% and 3.40% for d_0, and between 0.19% and 6.33% for ATT. On the other hand, for the best route sets relative to the operator's costs, the total length of these best route sets are the same, namely 63 minutes (evaluated by bus travel time). Non-dominated trade-off solutions from 10 runs for the 4-route scenario can be seen in Figure 5.11.

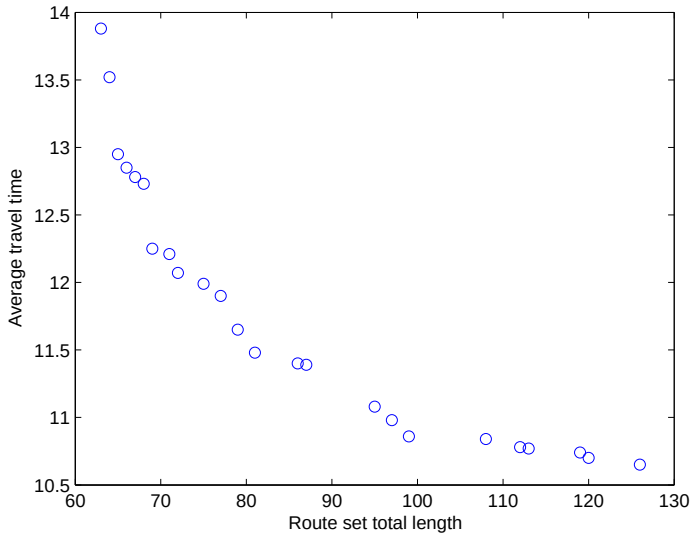

Figure 5.11. Non-dominated Solutions from 10 Runs for 4-route Scenario for Mandl's Network (route length and travel time both measured in minutes)

.

5.5.2 Scalability Experiments

In our scalability experiments, 6 larger networks (see Table 4.6) were tested by 12 experimental conditions (see Table 4.8). For each experiment, we recorded the results from 10 replicate runs (each seeded with different random numbers) using a population size of 100. The number of iterations of the multi-objective algorithm used in each experimental run was set to 100 and 200 respectively. We chose a smaller

number of iterations than Mandl's network, because of the long run times. The problem of run time for larger networks will be discussed in Chapter 6. Furthermore, we compare the results obtained using our multi-objective algorithm with those obtained using the single objective approach introduced in Chapter 4. We also assess the quality of our results obtained using our multi-objective algorithm by comparing the average travel time (ATT) per passenger with the lower bound on the ATT (see Table 4.7). The difference between the ATT and the lower bound is expressed as a percentage of the lower bound. These comparison results can be seen in Table 5.5, 5.6, 5.7 (SMO denotes the simple multi-objective optimization algorithm).

From these results, it is clear that our multi-objective optimization algorithm can also find good solutions for these larger network problems. For example, considering assessment parameter values (such as d_0 and ATT) for the best route sets relative to the passenger's costs for the 12 scenarios, the percentage difference between d_0 obtained by our SMO and d_0 obtained by the single objective method are between 0.88% and 2.93%. At the same time, the percentage difference between ATT obtained by the SMO and ATT obtained by the single objective method are between 0.59% and 5.62%. Note - percentage difference: $\|(parameter_{SMO} - parameter_{singleobjective})\| \times 100 / parameter_{singleobjective}$

Table 5.5. Comparison Results for 70 Nodes Network Problem

Network	Number of Routes	Number of Nodes in Route	Parameters	Best Single Objective	Best SMO Passenger	Best SMO Operator	Lower Bound on ATT	ATT Error%
70 nodes and 175 links	35 routes	2 - 17	d_0	53.70	52.46	36.19	29.72	12.15
			d_1	34.86	33.52	35.64		
			d_2	11.44	14.02	28.17		
			ATT	32.61	33.33	38.86		
			C_O	1828	1551	722		
70 nodes and 175 links	15 routes	10 - 30	d_0	43.26	42.18	35.27	29.72	14.37
			d_1	40.22	40.01	34.61		
			d_2	16.52	17.81	30.12		
			ATT	33.39	33.99	39.84		
			C_O	1329	1216	734		
70 nodes and 245 links	44 routes	2 - 13	d_0	53.68	52.41	36.12	28.98	12.22
			d_1	37.42	36.82	36.27		
			d_2	8.90	10.77	27.61		
			ATT	31.74	32.52	37.24		
			C_O	3476	3324	1675		
70 nodes and 245 links	15 routes	10 - 30	d_0	56.66	55.43	37.21	28.98	11.66
			d_1	33.02	32.14	32.65		
			d_2	10.32	12.43	30.14		
			ATT	31.66	32.36	37.88		
			C_O	2297	2184	1369		

Table 5.6. Comparison Results for 110 Nodes Network Problem

Network	Number of Routes	Number of Nodes in Route	Parameters	Best Single Objective	Best SMO Passenger	Best SMO Operator	Lower Bound on ATT	ATT Error%
110 nodes and 275 links	55 routes	2 - 29	d_0	72.91	71.26	48.62	33.84	4.28
			d_1	20.56	18.88	32.45		
			d_2	6.54	9.86	18.93		
			ATT	34.60	35.29	38.36		
			C_O	2986	2823	1077		
110 nodes and 275 links	56 routes	10 - 22	d_0	71.21	70.01	46.97	33.84	6.06
			d_1	20.71	19.21	31.84		
			d_2	8.08	10.78	21.19		
			ATT	35.68	35.89	38.55		
			C_O	2378	2257	1265		
110 nodes and 385 links	80 routes	2 - 19	d_0	74.82	72.63	50.44	30.76	9.62
			d_1	18.94	17.38	35.65		
			d_2	6.24	9.99	13.91		
			ATT	32.33	33.72	37.84		
			C_O	4399	4317	2921		
110 nodes and 385 links	56 routes	10 - 22	d_0	76.17	75.11	51.79	30.76	10.60
			d_1	18.77	16.98	33.64		
			d_2	5.06	7.91	14.57		
			ATT	32.21	34.02	37.15		
			C_O	4035	3966	3023		

Table 5.7. Comparison Results for 130 Nodes Network Problem

Network	Number of Routes	Number of Nodes in Route	Parameters	Best Single Objective	Best SMO Passenger	Best SMO Operator	Lower Bound on ATT	ATT Error%
130 nodes and 325 links	98 routes	2 - 21	d_0	78.35	77.66	52.54	28.46	6.89
			d_1	16.96	15.79	37.33		
			d_2	4.69	6.55	10.13		
			ATT	29.77	30.42	35.76		
			C_O	3642	3475	2282		
130 nodes and 325 links	60 routes	12 - 25	d_0	73.58	72.48	49.75	28.46	9.94
			d_1	22.52	21.11	35.41		
			d_2	3.90	6.41	14.84		
			ATT	29.97	31.29	36.16		
			C_O	3911	3802	2468		
130 nodes and 455 links	106 routes	2 - 20	d_0	75.13	74.24	50.36	26.41	8.56
			d_1	19.43	18.52	35.77		
			d_2	5.44	7.24	13.87		
			ATT	27.92	28.67	34.04		
			C_O	3876	3793	2315		
130 nodes and 455 links	60 routes	12 - 25	d_0	75.11	74.33	51.47	26.41	7.00
			d_1	20.96	18.94	14.02		
			d_2	3.93	6.73	35.51		
			ATT	27.85	28.26	33.85		
			C_O	4548	4518	3038		

5.6 Summary

In this chapter, we have presented an improved initialization procedure that produces a much higher proportion of feasible and "potentially feasible" route sets than the procedure used previously. Furthermore, an evolutionary multi-objective optimization algorithm has been presented to solve the UTRP, with the two key objectives, of minimizing both passenger costs and operator costs. Through the experiments on Mandl's benchmark data set, and some artificially generated larger data sets, we have demonstrated that our method is able to obtain efficient route sets which balance the requirements of passengers with those of the operator.

Chapter 6

CONCLUSION AND FUTURE RESEARCH

In this chapter, the main contributions of our research to the urban transit routing problem are summarized, and some ideas for future research are presented.

6.1 Conclusion

Through our research on the UTRP, we have learned much about the problem itself, particularly the difficulties involved when attempting to model the many constraints and practical difficulties that can arise in real-world situations. Nevertheless, we believe that we have made considerable progress towards establishing a generic model that incorporates key features of the problem, yet does not overcomplicate matters. In addition, we have created some basic techniques for solving the problem, and improved on previously published results. Finally, we have produced a problem generator that will create realistic data sets, and established some useful parameters and lower bounds on problem instances based on common-sense observations.

In more detail, the major contributions of our research on the UTRP

can be summarized as follows:

- We have devised a simple model of the UTRP, based on the generic model presented by some previous researchers but extending it with a highly effective objective function for the passenger cost. We believe that our objective function is a key factor in the success of our metaheuristic techniques.

- We have constructed a new and simple metaheuristic framework for solving the UTRP, consisting of the following:

  - a representation of candidate route sets,
  - an effective feasibility check procedure, to ensure route sets comply with given constraints,
  - our Make-Small-Change procedure - for (intelligent) neighbourhood moves and efficient repair of infeasible solutions,
  - simple hill-climbing and simulated annealing techniques to fit into the metaheuristic framework.

- We have written software to generate realistic artificial data sets to enable researchers to create test data which reflect their own requirements, such as the scale of the transit network, the range of the travel demand and the level of connectivity.

- We have established a lower bound on the passenger cost, to help assess the quality of route sets obtained for new (previously untested) data sets.

- We have established various constraints and other properties of transit networks and route sets.

- We have proposed and tested a prototype multi-objective optimization algorithm, which trades off operator's cost against passenger's cost.

6.2 Future Research

As mentioned previously (see Section 3.5), a larger network having 1200 nodes and 3600 links based on the Beijing bus network has also been investigated using our algorithms. Unfortunately, the run time required to obtain a suitable solution is much longer than that we had expected. Hence, besides considering the computer's configuration, the efficiency of our algorithm also needs to be reviewed.

In our algorithm, we find that the computational bottle-neck is the time required to find the number of transfers for each travel path to satisfy the demand at each node pair in the network. To do this the algorithm needs to determine which parts of each passenger travel path belongs to each transit route, which requires that every node in each path is checked, giving a run time complexity of $O(n^3)$ (where n is the number of nodes in the network) for each route set. In order to overcome the problem, we propose some improvements. We can either:

1. develop a new and more efficient method for counting the number of transfers using the existing model, or

2. we can use a more sophisticated model for the transit network and eliminate the need to compute the number of transfers from the objective function.

Method 2) would appear to be the easiest option, and for this we favour Mandl's model [73]. On the other hand, Method 1) would allow

us to maintain our current objective function (with components for travel time and number of transfers). For Method 2) the basic idea is to add extra nodes and links to the route network so that transfer times are included explicitly in the model, and the shortest path algorithm can take account of these, along with the travel times along the various transit links. For example, in Figure 6.1, two routes r_a and r_b make up the route network. There are two common nodes for each route in the route network. Therefore, if we add the transfer times, a new transit network can be established (see Figure 6.2). Let t_{ij} denote the time required to traverse the transport link between node i and node j, and τ denote the time required to transfer from route r_a to route r_b.

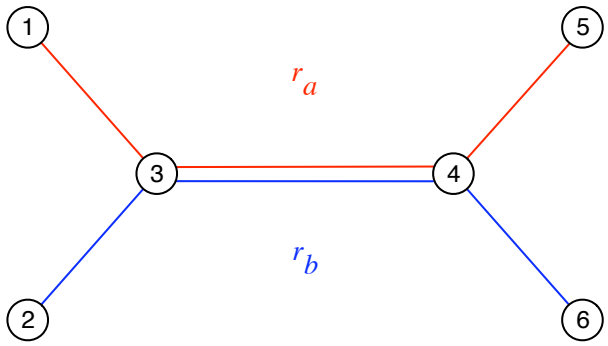

Figure 6.1. Route Network

Based on this model, we assume that passengers will always choose to travel on the shortest-time paths in the transit network. Therefore, our objective function can be revised with just the travel time (the number of transfers is not needed) as follows:

$$Minimize : Z = \sum_{i,j=1}^{N} d_{ij}\alpha_{ij} \tag{6.2.1}$$

Where d_{ij} denotes the transit demand from node i to node j (defined

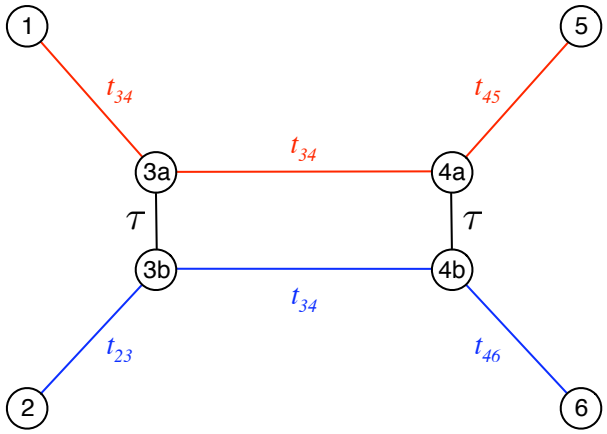

Figure 6.2. New Transit Network

in terms of the number of passengers wishing to travel between i and j) and α_{ij} is the shortest journey time from i to j (including transfer times). This is clearly more elegant than our current objective function, as it does not require (arbitrary) weighting of travel times versus the number of transfers. On the other hand, our current objective function has produced excellent results, beating some previously best published.

Other possibilities for future work to extend the present contributions include:

- Improve and enhance our metaheuristic framework. As well as making the software run faster, as discussed above, we could experiment with new types of neighbourhood move which provide a mechanism for creating or destroying routes, so that the number of routes will not need to be chosen prior to optimization.

- We could also try alternative metaheuristic optimization algorithms such as genetic algorithms [54], tabu search [48] or ant colony optimization [32]. Some parts of our current metaheuristic framework can be readily used in other metaheuristic algorithms,

for example, the *Feasibility Check* procedure can be used as an external procedure in genetic algorithms or the tabu search to ensure the feasibility of the route set. At the same time, the *Make-Small-Change* procedure can be regarded as a good "mutation" operator for genetic algorithms, as well as suitable neighbourhood moves for tabu search and other local search algorithms. However designing an efficient "crossover" operator for a genetic algorithm provides rather more of a challenge.

- Improve our prototype multi-objective optimization algorithm and experiment with the simultaneous optimization of more sophisticated objectives such as the number of buses on the routes or the utilization percentage (how full the busses are).

- Refine our simple model, in consultation with transport planners, and assess the applicability of our techniques to real-world problems. We could also explore how our problem-solving techniques could be integrated into commercial software toolkits such as VISUM.

A further direction of new research would be to build a simulator (to model a public transport system) for testing the route networks generated by our techniques. The simulator would allow the user (or our program) to input a route network and bus frequencies and mimic a given level of service. The simulator could be used to evaluate route sets generated by our route design program (or by any other means). We could then use these results to assess the validity of the static evaluation process employed by our route design program, which could help us improve the static evaluation process.

BIBLIOGRAPHY

[1] Brighton map. *http://www.buses.co.uk/travel/pdfmaps.aspx (Last accessed 30/07/2009).*

[2] Beijing map. *China Daily (newspaper)*, 01 January 2007.

[3] Public road network planning. *Department of Transportation, Yubei, Chongqing*, 2006.

[4] A complete guide to cardiff bus services. *Cardiff Bus Company*, 2008.

[5] A. Amberg, W. Domschke and S. Voss. Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. *European Journal of Operational Research*, 124:360–376, 2000.

[6] A. Hertz, E. Taillard and D. Werra. A tutorial on tabu search. *http://www.cs.colostate.edu/ whitley/CS640/hertz92tutorial.pdf (Last accessed 30/07/2009).*

[7] A.W. Alan. Urban transit systems : guidelines for examining options. *Word Bank Technical Paper*, WTP-52, 1986.

[8] ATKINS-SATURN. *http://www.saturnsoftware.co.uk/index.html (Last accessed 30/07/2009).*

[9] A.V. Aho, J.E. Hopcroft and J.D. Ullman. Data structures and algorithms. *Addison-Wesley*, 1983.

[10] B. Yu, Z.Z. Yang, C. Cheng and C. Liu. Optimization bus transit network with parallel ant colony algorithm. *Proceedings of the Eastern Asia Society for Transportation Studies*, 5:374–389, 2005.

[11] M.H. Baaj and H.S. Mahmassani. Trust: a lisp program for the analysis of transit route configurations. *Transportation Research Record*, 1283:125–135, 1990.

[12] M.H. Baaj and H.S. Mahmassani. An AI-based approach for transit route system planning and design. *Journal of Advance Transportation*, 25(2):187–210, 1991.

[13] M.H. Baaj and H.S. Mahmassani. Hybrid route generation heuristic algorithm for the design of transit networks. *Transportation Research*, 3(1):31–50, 1995.

[14] R. Balcombe. The demand for public transport: a practical guide. *TRL Report, TRL Limited, UK*, 2004.

[15] U. Blasum and W. Hochstattler. Application of the branch and cut method to the vehicle routing problem. *Zentrum fur Angewandte Informatik Koln Technical Report*, zpr:386–2000, 2000.

[16] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: overview and conceptual comparison. *Technical Report, IRIDIA, University Libre de Bruxelles, Belgium*, 13(2):23–45, 2001.

[17] P. Bolstad. GIS fundamentals: A first text on geographic information systems. *Second Edition. White Bear Lake, MN: Eider Press*, 2005.

[18] M.R. Bussieck. Optimal lines in public rail transport. *Ph.D. Thesis, TU Braunschweig*, 1998.

[19] A. Ceder and H.M. Wilson. Bus network design. *Transportation Research*, 20B(4):331–344, 1986.

[20] P. Chakroborty. Genetic algorithms for optimal urban transit network design. *Computer-Aided Civil and Infastructure Engineering*, 18:184–200, 2003.

[21] P. Chakroborty and T. Dwivedi. Optimal route network design for transit systems using genetic algorithms. *Engineering Optimization*, 34(1):83–100, 2002.

[22] Citilabs. *http://www.citilabs.com/cube-voyager.html (Last accessed 30/07/2009).*

[23] G. Clarke and J. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[24] C.A.C. Coello. Evolutionary multi-objective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, 2006.

[25] I. Constantin and M. Florian. Optimizing frequencies in a transit network: a nonlinear bi-level programming approach. *International Transactions in Operational Research*, 2(2):149–164, 1995.

[26] Z.J. Czech and P. Czarnas. Parallel simulated annealing for the vehicle routing problem with time windows. *10th Euromicro Workshop*

on Parallel, Distributed and Network-based Processing, Canary Islands, Spain, pages 376–383, 2002.

[27] A.A. Dayem. Optimizing bus transfer coordination. *Technique Report, International Institute for Geo-information Science and Earth Observation, Enschede, Netherlands*, 2005.

[28] K. Deb. Multi-objective optimization using evolutionary algorithms. *Chichester, New York: John Wiley and Sons*, 2001.

[29] Department for Transport, UK. *http://www.dft.gov.uk (Last accessed 30/07/2009)*.

[30] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numer.Math.*, 1:269–271, 1959.

[31] T. Domencich and D.L. McFadden. Urban travel demand: A behavioral analysis. *North-Holland Publishing Co.*, 1996.

[32] M. Dorigo. Optimization, learning and natural algorithms. *PhD thesis, Politecnico di Milano, Italy*, 1992.

[33] B. Emerson. Design and planning guidelines for public transport infrastructure-bus route planning and transit streets. *Public Transport Authority, USA*, 2003.

[34] F. Glover, J.P. Kelly and M. Laguna. Genetic algorithms and tabu search: Hybrids for optimization. *Computers and Operations Research*, 22(1):111–134, 1995.

[35] L. Fan and C.L. Mumford. A simplified model of the urban transit routing problem. *The 7th Metaheuristics International Conference, Montreal, Canada*, pages 16–17, 2007.

[36] L. Fan and C.L. Mumford. A metaheuristic approach to the urban transit routing problem. *Journal of Heuristics, ISSN 1381-1231 (Print) 1572-9397 (Online), published by Springer Online*, 2008.

[37] W. Fan and R.B. Machemehl. Optimal transit route network design problem: algorithms, implementations and numerical results. *Research Report, Centre for Transportation Research, University of Texas at Austin*, 2004.

[38] 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.

[39] W. Fan and R.B. Machemehl. Computer-aided systems in public transport. *Springer Berlin Heidelberg*, pages 387–408, 2008.

[40] M.L. Fisher. Optimal solution of vehicle routing problems using minimum k-trees. *Operations Research*, 42:626–642, 1994.

[41] M.L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124, 1981.

[42] R.W. Floyd. Algorithm 97: shortest path. *Commun. ACM*, 5(6):345, 1962.

[43] P.G. Furth and N.H.M. Wilson. Setting frequencies on bus routes: theory and practice. *Transportation Research Record*, 818:1–7, 1982.

[44] G. Fusco, S. Gori and M. Petrelli. A heuristic transit network design algorithm for medium size towns. *Proceedings of the 13th Mini-EURO Conference Handling Uncertainty in the Analysis of Traffic and Transportation Systems*, pages 652–656, 2002.

[45] M.R. Garey and D.S. Johnson. Computers and intractability: A guide to the theory of np-completeness. *W.H. Freeman*, pages 5–1045, 1979.

[46] B.E. Gillet and L.R. Miller. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 122:340–349, 1974.

[47] S. Glaister. UK bus deregulation: the reasons and the experience. *Investigaciones Economicas*, XV:285–308, 1991.

[48] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549, 1986.

[49] F. Glover and G. Kochenberger. Handbook of metaheuristics. *Kluwer Academic Publishers*, 2002.

[50] D.E. Goldberg. Genetic algorithm in search, optimization, and machine learning. *Addison-Wesley Publishing CO., Reading, Mass*, 1989.

[51] J.M. Guldmann. Urban transportation network design, tranffic allocation, and air quality control: an integrated optimization approach. *European Regional Science Association 36th Ruropean Congress, Switzerland*, pages 324–332, 1996.

[52] D. Hasselstrom. Public transportation planning-mathematical programming approach. *Department of Business Administration, University of Gothenburg, Gothenburg, Sweden*, 1981.

[53] H.N. Koutsopoulos, A. Odoni and N.H.M. Wilson. Determination of headways as function of time varying characteristics on a transit network. *North-Holland, Amsterdam*, pages 391–414, 1985.

[54] J.H. Holland. Adaptation in natural and artificial systems. *2nd ED., MIT Press, Cambridge, Mass*, 1992.

[55] J. Horn. "Multi-objective decision making" In handbook of evolutionary computation. *Institute of Physics Publishing*, pages 35–76, 1997.

[56] Y. Israeli and A. Ceder. Designing transit routes at the network level. *IEEE Vehicle Navigation and Information Systems Conference*, pages 310–316, 1989.

[57] J. Agrawal, T.V. Mathew and A.M. Asce. Transit route network design using parallel genetic algorithm. *Journal of Computing in Civil Engineering*, 18:248–256, 2004.

[58] J. Han, S. Lee and J. Kim. Meta-heuristic algorithms for a transit route design. *Advanced OR and AI Methods in Transportation*, pages 686–691, 2006.

[59] J. Koza, M. Keane, M. Streeter, W. Mydlowec, J. Yu and G. Lanza. Genetic Programming IV: Routine human-competitive machine intelligence. *Kluwer Academic Publishers*, 2003.

[60] W. Jih and J.Y. Hsu. Dynamic vehicle routing using hybrid genetic algorithms. *In Proceedings of the 1999 IEEE International Conference on Robotics and Automation. Detroit, Michigan*, pages 89–99, 1999.

[61] P. Judea. Heuristics. *Addison-Wesley Publication*, 1984.

[62] K. C. Tan, T. H. Lee, Y. H.Chew and L. H. Lee. A hybrid multiobjective evolutionary algorithm for solving truck and trailer vehicle

routing problems. *IEEE Congress on Evolutionary Computation*, pages 2134–2141, 2003.

[63] K.C. Tan, K. Ou and L.H. Lee. A messy genetic algorithm for the vehicle routing problem with time windows constraints. *IEEE Congress on Evolutionary Computation*, pages 679–686, 2001.

[64] F.A. Kidwai. Optimal design of bus transit network: a genetic algorithm based approach. *PhD.dissertation, Indian Institute of Technology, Kanpur, India*, 1998.

[65] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[66] L. Fan, C.L. Mumford and D. Evans. A simple multi-objective optimization algorithm for the urban transit routing problem. *IEEE Congress on Evolutionary Computation, Trondheim, Norway*, pages 1–7, 2009.

[67] L. Silman, Z. Barzily and U. Passy. Planning the route system for urban buses. *Comput. Ops. Res.*, 1:201–211, 1974.

[68] W. Lampkin and P.D. Saalmans. The design of routes, service frequencies and schedules for a municipal bus undertaking: a case study. *OR Quarterly*, 18:375–397, 1967.

[69] L.E. Fernandez, C.J. Cea and R.H. Malbran. Demand responsive urban public transport system design: methodology and application. *Transportation Research Part A: Policy and Practice*, 42(7):951–972, 2005.

[70] M. Bielli, M. Caramia and P. Carotenuto. Genetic algorithms in bus network optimization. *Transportion Research Part C*, 10:19–34, 2002.

[71] M. Friedrich, T. Haupt and K. Noekel. Planning and analyzing transit networks: an integrated approach regarding requirements of passengers and operators. *2nd GIS in Transit Conference, Tampa, Florida*, pages 19–39, 1999.

[72] R.L. Mackett and M. Edwards. Guidelines for planning a new urban public transport system. *Proceedings of the Institution of Civil Engineers. Transport*, 117:193–201, 1996.

[73] C.E. Mandl. Applied Network Optimization. *Academic, London*, 1979.

[74] C.E. Mandl. Evaluation and optimization of urban public transport networks. *In: Third Congress on Operations Research, Amsterdam, Netherlands*, pages 396–404, 1979.

[75] C.E. Mandl. Evaluation and optimization of urban public transport networks. *European Journal of Operational Research*, 5:396–404, 1980.

[76] VISUM 10.0 User Manual. *PTV Vision*, 2007.

[77] M.C. Shih, H.S. Mahmassani and M.H. Baaj. A planning and design model for transit route networks with coordinated operations. *The 77rd Annual Meeting of the Transportation Research Board, Washington,D.C.*, pages 16–23, 1998.

[78] C.L. Mumford. Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm. *Genetic and Evo-*

lutionary Computation Conference (GECCO), Seattle, Washington, USA, pages 1389–1400, 2004.

[79] O. Arbelaitz, C. Rodriguez and I. Zamakola. Low cost parallel solutions for the vrptw optimization problem. *International Conference on Parallel Processing Workshops, IEEE Computer Society, Valencia, Spain*, pages 176–181, 2001.

[80] NCHRP Synthesis of Highway Practice. Bus route and schedule planning guidelines. *Transportation Research Board, National Research Council, Washington,D.C.*, 1980.

[81] P. Chakroborty, K. Deb and B. Srinivas. Network-wide optimal scheduling of transit systems using genetic algorithms. *Computer-Aided Civil and Infrastructure Engineering*, 13:363–376, 1998.

[82] P. Chakroborty, K. Deb and R.K. Sharma. Network-wide optimal scheduling of urban transit networks using genetic algorithms. *Transportation Planning and Technology*, 24(3):209–226, 2001.

[83] V. Pareto. Cours d'e conomie politique professe a l'universite de lausanne. *F.Rouge, Laussanne*, 1(2), 1896.

[84] R.C. Prim. Shortest connection networks and some generalisations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[85] H.N. Psaraftis. Dynamic vehicle routing problems. *Vehicle Routing: Methods and Studies*, 16:223–248, 1988.

[86] S.J. Russell and P. Norvig. Artificial intelligence: a modern approach. *Upper Saddle River, NJ: Prentice Hall*, pages 111–114, 2003.

[87] P. Saadah and B. Paechter. Improving vehicle routing using a customer waiting time colony. *EvoCOP, Springer-Verlag*, pages 188–198, 2004.

[88] S.B. Pattnaik, S. Mohan and V.M. Tom. Urban bus transit route network design using genetic algorithm. *Journal of Transportation Engineering*, 124:368–375, 1998.

[89] S. Scheele. A supply model for public transit services. *Transportation Research*, B(14):133–146, 1980.

[90] M.C. Shih and H.S. Mahmassani. A vehicle sizing model for bus trasnit systems. *The 73rd Annual Meeting of the Transportation Research Board, Washington,D.C.*, pages 35–41, 1994.

[91] S.Kirkptrick, C.D. Gelatt and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–679, 1983.

[92] S. Stephen and Z. Liu. China's urban transport development strategy: proceedings of a symposium in Beijing,. *Washington, D.C: World Bank*, pages 352–855, 1996.

[93] Welcome to Emme 3. *INRO*, 2007.

[94] V.M. Tom and S. Mohan. Transit route network design using frequency coded genetic algorithm. *Journal of Transportation Engineering*, 129:186–195, 2003.

[95] P. Toth. The vehicle routing problem. *Philadelphia: Siam*, 2001.

[96] P. Toth and D. Vigo. The granular tabu search and its application to the vehicle routing problem. *Technical Report, University of Bologna*, 1998.

[97] C.L. Valenzuela. A simple evolutionary algorithm for multi-objective optimization (SEAMO). *Congress on Evolutionary Computation (CEC), Hawaii*, pages 717–722, 2002.

[98] D. Van Vliet. Improved shortest-path algorithms for transport networks. *Transportation Res.*, 12:7–20, 1978.

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

[100] J.L. Wang. The public transportation optimum route algorithm based on the least transfer. *Economic Geography*, 25(5):673–676, 2005.

[101] L. Wang and W.Q. Li. Best-routing algorithm for public transportation systems. *Journal of Southest University*, 34(2):264–267, 2004.

[102] P. White. Public transport: Its planning, management and operation. *4th Edition, Spon Press*, 2002.

[103] C.M. Williams. Transportation planning. *Department of Planning and Community Development, Stafford, Virginia*, 1996.

[104] J.Y. Yen. An algorithm for finding shortest routes from all source nodes to a given destination in general network. *Quart. Appl. Math.*, 27:526–530, 1970.

[105] Q.F. Zeng and K.C. Mouskos. Heuristic search strategies to solve transportation network design problems. *Final Report, New Jersey Department of Transportation*, 1997.

[106] F. Zhao. Large-scale transit network optimization by minimizing

user cost and transfer. *Journal of Public Transportation*, 9(2):107–129, 2006.

[107] F. Zhao and A. Gan. Optimization of transit network to minimize transfers. *Final Report, Research Centre Florida Department of Transportation*, 2003.

[108] F. Zhao and I. Ubaka. Transit network optimization-minimizaing transfers and optimizing route directness. *Journal of Public Transportation*, 7(1):63–82, 2004.