A Metaheuristic Approach to the Urban Transit Routing Problem

Lang Fan and Christine Mumford
Cardiff University,
School of Computer Science,
Queen's Buildings,
5 The Parade, Roath,
Cardiff CF24 3AA, UK
Contact: C.L.Mumford@cs.cardiff.ac.uk

Preprint of journal article

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 depend 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*. Our paper concentrates on these three issues, and presents a framework which can be used as a starting point for solving this problem. We devise a simple model of the UTRP to evaluate candidate route sets. Finally, our approach is validated using simple hill-climbing and simulated annealing algorithms. Our simple method beats published results for Mandl's benchmark problem. In addition, the potential for solving larger problem instances has been explored.

Keywords

Urban transit routing, hill-climbing, simulated annealing

1 Introduction

With the development of modern cities, the increase in population and concerns about the environment and traffic congestion, efficient urban public transport systems are needed throughout the world. The urban transit network design problem (UTNDP) is concerned with the determination of a set of routes with corresponding schedules for such an urban public transport system. This complex NP-Hard problem must optimize many criteria in order to efficiently meet the needs of the users, while at the same time minimizing the cost to the service provider. From the passengers' point of view, for example, an ideal service will provide rapid transit between source and destination, with a minimum of transfers between vehicles on the way. Operators, on the other hand, aim to minimize their cost, yet a low cost option may provide a poor service to the customer. In addition, there are other stake-holders involved: typically national and local government as well as taxpayers and local business. 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 a slightly different definition of what efficiency means. From the point of view of the scientific research, the urban transit network design problem provides an enormous challenge.

The UTNDP can be subdivided into two major components, namely the transit routing problem and the transit scheduling problem [8]. Generally, the urban transit routing problem (UTRP) involves the development of efficient transits routes (e.g., bus routes) on an existing road network, with predefined pickup/dropoff 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 (or iteratively), with the routes determined in advance of the schedules.

In this paper we concentrate on the urban transit routing problem, and present a basic metaheuristic framework for solving it, consisting of: a *representation scheme*, an *initialization procedure* and a set of simple *neighbourhood moves*. We demonstrate the effectiveness of our scheme, by embedding some simple search mechanisms into our metaheuristic framework and comparing our results with previously published results on a benchmark instance. Furthermore, we explore the scalability of our approach by testing it on some larger instances, generated by ourselves. In addition, we introduce a simplified model of the UTRP, which allows us to concentrate on the key issues of minimizing travel time and the number of transfers simultaneously.

Section 2 surveys relevant literature on the UTRP and the methods used to solve it, and Section 3 highlights the main contributions of our work. In Section 4 we introduce key features of the UTRP and define our model, then in Section 5 we explain the representation we use for route sets and also the main routines for generating, evaluating and improving the route sets. Section 6 describes our metaheuristic framework and we present our results in Section 7. Finally, we summarize our work and discuss some future plans in Section 8

2 Relevant Literature

Historically, transport planners have devised reasonable bus route networks and schedules, without the aid of computer programs, by relying on past experience, following simple guidelines and utilizing local knowledge. However, for a large urban area where the number of bus routes may be over a hundred and the number of bus stops in the thousands, past experience and simple guidelines may not be enough to produce an efficient transit route network configuration and bus schedule [24].

Whether relying on a manual or a computer-aided system for solving the UTNDP however, it is important to realize that the resulting system will not be useful in practice if it is designed on the back of poor quality information - such as inaccurate travel demand estimation, for example. Travel demand between various sources and destinations can be collected or forecasted 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 [4]. In addition, design guidelines are determined by many additional factors such as the street environment in the local area and the transport and management policies of the local government. These researches have been carried out by White [23] and Emerson [9].

We shall now survey some key papers covering the historical development of research on automatic methods for solving the UTNDP, focussing our attention primarily on the UTRP.

2.1 Pioneering Work

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, [16] and [21]. An exception is the pioneering work of Christoph Mandl [17, 18, 19] who tackled the problem in a rather more generic form. Indeed, his common-sense account of the UTNDP in [17] 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 [6] or Floyd's algorithm [13], 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: (i) obtaining new routes by exchanging parts of routes at an intersection node; (ii) Including a node that is close to a route, if travel demand between this node and the nodes on the route is high; and (iii) 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 as constant values, according to specified vehicle frequencies.

2.2 Heuristic Developments

In the following decade, Ceder and Wilson [5] in 1986 and Israeli and Ceder [14] 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 [5] 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 [14] 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 multiobjective tradeoff solutions to be presented to a human decision maker.

More recently Baaj and Mahmassani in 1995 [3] described and implemented an 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.

2.3 Metaheuristic Approaches

The last two decades have seen a rapid growth in computing power and, as computers have become faster, metaheuristic techniques have become ever more popular for solving hard combinatorial problems. Methods such as genetic algorithms (GAs), tabu search (TS) and simulated annealing (SA) have all played important roles in recent research on the UTNDP. GAs are particularly popular, and several researchers have used them to determine simultaneously the route network and the associated vehicle frequencies. Pattnaik, Mohan and Tom [20], Tom and Mohan [22], and Agrawal and Mathew [1] all used a binary encoding scheme to identify candidate routes. In this way candidate routes can be pre-determined and stored in a list, and it is the job of the GA 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 is biased towards saving the better solutions over the poorer ones. In this approach 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 are also encoded as part of the chromosome in [1]and [22].

On the other hand, in 2002 Chakroborty and Dwivedi [7] took a different approach to encoding a GA: listing the nodes explicitly, rather than binary coding a route as an entity. This work was taken further by Chakroborty in

2003 [8] to cover scheduling (UTSP) as well as routing (UTRP). For other metaheuristic methods, examples can be seen in Fan and Machemehl's 2004 [11] and 2006 [12] papers. They utilized their solution methodology with tabu search and simulated annealing to solve specialized UTNDP problems.

3 Our Research Characteristics

We believe that metaheuristic approaches are highly suitable for the UTNDP, yet the success of such methods depend 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*. Yet, to the best of our knowledge little attention has been paid to these issues in the UTNDP literature. In this paper we concentrate on the urban transit routing problem, and present a basic metaheuristic framework for solving it, consisting of: a representation scheme, an initialization procedure and a set of simple neighbourhood moves. Furthermore, we demonstrate the effectiveness of our scheme, as best we can, given the lack of "standard models" for the problem, and shortage of benchmark data. To do this we experiment with two simple algorithms: hill-climbing and simulated annealing, embedding their simple search mechanisms into our metaheuristic framework. In addition, we introduce a simplified model of the UTRP, along the lines of Mandl [17] for his Swiss network.

Given the practical importance of the UTNDP, it is perhaps rather surprising that so little work has been done on extracting generic features, formulating simplified models and devising benchmark data sets, such that comparative studies are facilitated to identify which algorithms work best. This is certainly not the case for other combinatoric optimization problems, for example: the travelling salesman, capacitated vehicle routing, examination timetabling, job-shop scheduling, bin packing etc. all have well-known benchmark instances, and researchers developing new algorithms to solve these problems are expected to validate their approaches by beating other people's results on standard benchmarks. Perhaps the lack of fundamental research can be explained by the enormous complexity of the UTNDP. It may be difficult for researchers to agree which aspects of the problem are most important, and thus decide which should extracted as "generic" to formulate a simplified model. Nevertheless, we attempt to do exactly this for the UTRP as part of our research. Guided by our study of the literature, we focus on the potential user of a public transit system, and identify two key objectives to be minimized: 1) total travel time (in-vehicle plus waiting time) and 2) the number of transfers. In practice we recognize that there are many other vital issues to be considered for practical purposes. These considerations include service frequency, capacity, and operating costs. However, these features greatly complicate the model. Our main interest in the present study is to validate our initialization procedure, our representation scheme and our neighbourhood operators.

4 The Urban Transit Routing Problem

In the literature different models of the UTRP are characterized by different criteria to optimize and special constraints. However, the following criteria of a good route set have been generally accepted by most researchers [8]:

- 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.

At the same time, in the real world some basic constraints have to be satisfied, for example:

- a maximum and minimum length for each route - that means a limited number of stops in a single route. Normally the planner will set constraints on the route length based on consideration of issues such as the difficulty in maintaining bus schedule adherence and bus driver fatigue [24];
- a connected route set - which is an essential requirement for the UTRP. Basically, this kind of route set can cover the whole network in a city and ensure that all customers can get to their destinations in that city;
- a fixed number of routes in the route set - generally the bus company will decide the number of routes in advance, constrained by funding;
- normally no cycles or backtracks are allowed in individual routes - although this is not always the case, we will make this assumption here.

4.1 A Simple Model of the Urban Transit Routing Problem

¿From the basic criteria of a good route set mentioned above, an efficient bus route set could reasonably be expected to minimize both the travel distance or time and the number of transfers. Further, for any given "bus route set" we can create a corresponding "bus route network" simply by fusing together all the routes in that route set. A particular bus route network will differ from the original road network from which it is derived, provided some links present in the road network are absent from the bus 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 bus route network. We will assume that each traveller chooses the shortest path (in the bus 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 (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 [10]):

1. To store the basic problem information we need:

    - An undirected graph, $G(V, A)$, consisting of n vertices (or nodes), $V = \{v_1, v_2, v_3, \ldots, v_n\}$, and m arcs, $A = \{a_1, a_2, a_3, \ldots, a_m\}$. this will store the road network.
    - A demand matrix, D, where d_{ij} = travel demand between nodes i and j.
    - 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. To find the simple objective function:

$$Minimize : Z = A \sum_{i=1}^{i=n-1} \sum_{j=i+1}^{j=n} d_{ij} p_{ij} + B \sum_{i=1}^{i=n-1} \sum_{j=i+1}^{j=n} d_{ij} t_{ij} \quad (1)$$

    where:
    p_{ij} is length of the shortest path between i and j for the current route

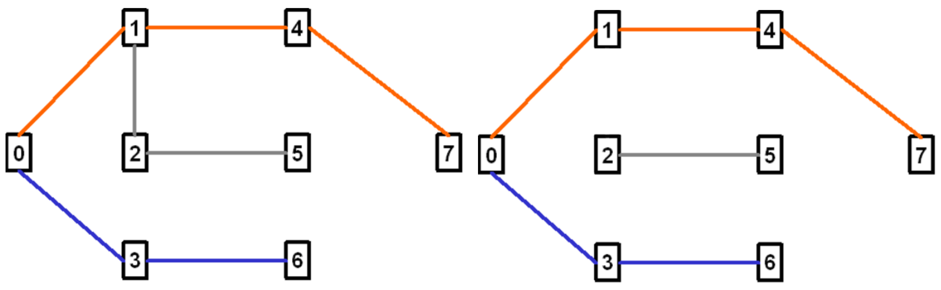


Figure 1: A connected and an unconnected 8 node network

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. (A and B are chosen to ensure the two parts of the objective function are of similar magnitude).
Our current objective function is subject to the following constraints:

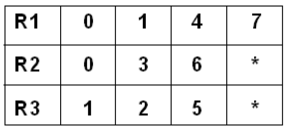
- *each route in a given route set is free of cycles and backtracks.* This is easily checked when generating or modifying a route, simply by checking that there are no repeated nodes. (see details in Sections 5.2 and 5.4)
- *the route set is connected* (see Figure 1). The connectivity of the route set is checked as part of the Feasibility Check Procedure (see details in Section 5.3).
- *there are exactly r routes in the route set* (usually 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.*

5 Methods of Representing and Improving Route Sets

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*.

5.1 Representation

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 Figure 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 2.



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

Figure 2: Two dimensional array. The * occupies the blank space in the array.

5.2 Initialization

The purpose of the Initialization procedure is to construct an initial route set at random according to the constraints listed in Section 4.1 and some user-defined parameters. In the initial route set, each route is a connected path containing no cycles or backtracks.

5.3 Feasibility Check

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 road 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 1 is a connected route set, but the route consisting of nodes 2 and 5 in the second graph is not linked to the rest of the route network. Hence this route set is an unconnected route set. 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 1

Algorithm 1 Feasibility Check Procedure

```
Input the route set, S
Input N, the the number of nodes in the road 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
```

5.4 Make-Small-Change

The Make-Small-Change procedure is responsible for making local (intelligent) 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 road 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 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.

The Make-Small-Change procedure proceeds as follows. First of all, it 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 route is equal to maximum number of nodes. (III) the length of a route is equal to 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 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.

6 Framework of Implementation

Simple hill-climbing (HC) and simulated annealing (SA) algorithms have been implemented in order to test our procedures. Generally the two algorithms have similar structure. The framework is summarized in Algorithm 2.

Algorithm 2 *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, $S$
***Main loop***
**repeat**
  ***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 $\sum_{i,j=1}^{i,j=n} d_{ij}p_{ij}$, $\sum_{i,j=1}^{i,j=n} d_{ij}t_{ij}$, and the objective function, $Z$
  ***Selection:***
  Select either $S$ or $S'$ as new focus of search following rules of Hill-Climbing or Simulated Annealing
**until** the stopping condition is satisfied
**Output** Best route set and $Z$ for the best route set
```

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 and stores it following 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 1. We consider the route network obtained by fusing all the routes from a given route set, as explained in Section 4.1. (Recall that a *route* network is a subset of the specified *road* 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 for making a transfer). All required shortest paths are calculated from the route network using Dijkstra's algorithm, and the first component of Equation 1, $\sum_{i=1}^{i=n-1}\sum_{j=i+1}^{j=n} d_{ij}p_{ij}$, is calculated. This gives the total travel distance (or time), for the route network, summed over all passengers. Note: if there is more than one contender for the shortest path between an origin and destination, the path with the highest demand is selected.

The total number of vehicle transfers summed over the entire demand (i.e., the second term in Equation 1) also needs to be calculated. This is done by checking every part of each shortest travel path, to identify which route it belongs to. 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, $\sum_{i=1}^{i=n-1}\sum_{j=i+1}^{j=n} d_{ij}t_{ij}$.
Selection: In the two algorithms the selection rules are different. 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 output. 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, similar to hill-climbing. If the neighbourhood route set is "worse" than the current route set, however, 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, however, the temperature cools and poor solutions tend to be accepted considerably less frequently. As is the case with hill-climbing, the best route set found during the entire run of the algorithm will be output when the algorithm terminates.

Values for the acceptance probability - *prob* - for a minimization problem, are evaluated using Equation (2) and (3). 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 temperature, which is determined by the cooling schedule.

$$\begin{aligned} \Delta &= C(S') - C(S) & (2) \\ prob &= min(1, e^{-\Delta/T}) & (3) \end{aligned}$$

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 N cooling steps to correspond to the number of iterations, so that the temperature is decreased slightly between each iteration. Thus, knowing N and setting initial and final acceptance probabilities, P_0 and P_N, as well as an additional parameter, M, that signifies an initial number of random trials, the starting temperature T_0, the final temperature T_N, and the cooling factor α can be calculated, as indicated below.

$$\Delta_i = C(S') - C(S) \tag{4}$$

$$\Delta_{ave} = \frac{\sum_{i=1}^{i=M} \mid \Delta_i \mid}{M} \tag{5}$$

$$T_0 = -\frac{\Delta_{ave}}{\log P_0} \tag{6}$$

$$T_n = -\frac{\Delta_{ave}}{\log P_N} \tag{7}$$

$$\alpha = \exp^{\frac{\log T_N - \log T_0}{N}} \tag{8}$$

Please note that Δ_{ave} (Equation (5)) 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" (omitted from the Framework above for simplicity), 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. In our study, $P_0 = 0.999$, $P_N = 0.001$, $M = N = 1,000$, and $L = 100$.

7 Experimental Results

There is only one popular benchmark instance - Mandl's (see Figure 3). We use this to compare our results against those of other researchers in our first set of experiments. Although our objective function is different from those used by other researchers, we are able, nevertheless, to make direct comparisons on the basis of common criteria.

In the second set of experiments we test our techniques on three further instances, to examine the scalability of our approach. Unfortunately, comparisons with other work is not possible for these instances. However, it is an easy matter to obtain quite good lower bounds, making it possible to assess our solutions in terms of percentage error. In addition, we use all four instances to examine the efficiency of the Make-Small-Change routine, with respect to problem size, number of nodes per route, and number of routes per route set. Our concern here is to ensure that our algorithms do not spend a disproportionate time generating and evaluating infeasible solutions as the problem size and/or difficulty increases.

7.1 A Note on Assessment Parameters

As mentioned above, we use Mandl's network to compare our approach with others. The following parameters are used to compare the quality of our route sets with those obtained in [18, 2, 15, 7].

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.

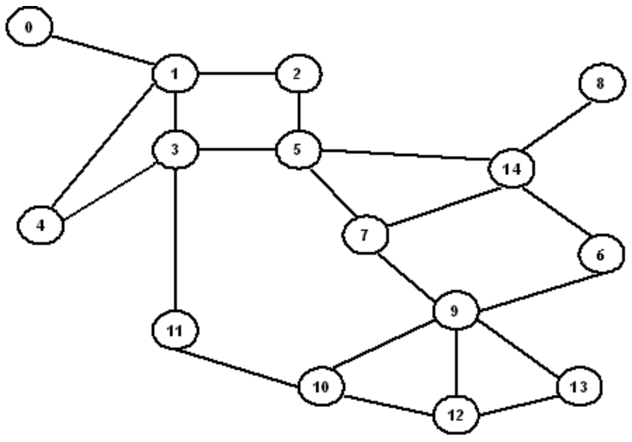


Figure 3: Mandl's Swiss Road Network

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 algorithm. 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. 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 path belongs to which route in the best route set. Hence we can discover the number of transfers which passengers needs 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 [18]. The routes (from the network shown in Figure 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 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 1. Method 2 produces a smaller value for ATT and a larger percentage of travellers reach their destinations with zero transfers.

7.2 Results for Mandl's Swiss Road Network

In order to establish the viability of our approach we first compare the results obtained by running our algorithms with those previously published by

Table 1: Parameters for Mandl's best route set [18] by our Method 1 and Method 2

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

Mandl[18], Baaj and Mahmassani[2], Kidwai[15] and Chakroborty [7]. For consistency with the other work, the route sets were developed for Mandl's network (Figure 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 for each time a transfer is made, as discussed in Section 7.1. We also set a maximum eight nodes in each route. We carried out 20 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, for the simulated annealing.

The results in Table 2 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 [7] for the 6 and 7 route cases. Although average solutions for SA are slightly better than those obtained using HC, t-tests on ATT and d_0 values show that some of these differences are significant and some are not at the 5 percent level. The actual routes produced for our best solutions for each situation are presented in Table 3.

Table 4 gives the average run times for our hill-climbing and simulated annealing algorithm. Note: our computer platform is Windows XP with Inter(R) Pentium(R) D CPU 3.00GHz and 1GB of RAM. Clearly, the SA is faster than the HC.

7.3 Scalability Experiments

Because of a lack of published benchmarks, it was necessary to create our own data to establish whether the techniques would scale to larger instances.

Table 2: Results for Mandl's Network

Number of Routes	Parameters	Mandl [18]	Baaj [2]	Kidwai [15]	Chakroborty [7]	Our Best	HC Average	SA Average	No. Runs
4	d_0	69.94	N	72.95	86.86	***93.26***	91.83	92.48	20
	d_1	29.93	N	26.92	12.00	***6.74***	8.17	7.52	
	d_2	0.13	N	0.13	1.14	***0.00***	***0.00***	***0.00***	
	d_{un}	***0.00***	N	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	
	ATT	12.90	N	12.72	11.90	***11.37***	11.69	11.55	
6	d_0	N	78.61	77.92	86.04	***91.52***	90.23	90.87	20
	d_1	N	21.39	19.68	13.96	***8.48***	9.26	8.74	
	d_2	N	***0.00***	2.40	***0.00***	***0.00***	0.51	0.39	
	d_{un}	N	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	
	ATT	N	11.86	11.87	***10.30***	10.48	10.78	10.65	
7	d_0	N	80.99	***93.91***	89.15	93.32	92.21	92.47	20
	d_1	N	19.01	***6.09***	10.85	6.36	7.13	6.95	
	d_2	N	***0.00***	***0.00***	***0.00***	0.32	0.66	0.58	
	d_{un}	N	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	
	ATT	N	12.50	10.69	***10.15***	10.42	10.74	10.62	
8	d_0	N	79.96	84.73	90.38	***94.54***	93.23	93.65	20
	d_1	N	20.04	15.27	9.62	***5.46***	6.18	5.88	
	d_2	N	***0.00***	***0.00***	***0.00***	***0.00***	0.59	0.47	
	d_{un}	N	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	***0.00***	
	ATT	N	11.86	11.22	10.46	***10.36***	10.69	10.58	

Table 3: 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

For these tests we use Mandl's network (which consists of 15 nodes and 20 links) plus 3 additional data sets: I - a small instance obtained from [20] consisting of 8 nodes and 9 links; II - one we devised ourselves, based on a small Chinese town, consisting of 20 nodes and 24 links; and III) a 50 node and 65 link network obtained by joining three Mandl's networks together.

A simple method to assess the quality of our results on these instances is to compare the average travel time (ATT) per passenger with the lower-bound result, which assumes that every passenger travels on the shortest path on the *road* network (as opposed to the *route* network) between source and destination without any transfers. The difference between the actual ATT and the "ideal" ATT is quoted as a percentage of the "ideal" quantity in our results. Table 5 presents the best results for average travel time (ATT)

Table 4: Average run times for the HC and SA algorithms.

Number of Routes	HC Time (secs)	SA Time (secs)
4	254	98
6	244	89
7	232	81
8	221	74

for 20 replicate runs of hill-climbing on our four networks. In addition, we give the ATT-error, as mentioned above. We include our best results for Mandl's network (8 routes and maximum 8 nodes), so that we can observe whether on not the quality of our solutions, with respect to the lower bound, will decline with increasing problem size. Each hill-climbing run consisted of 100,000 iterations. From Table 5 we can see that the percentage errors are similar for all the instances, with no observable deterioration for larger instance.

Table 5: Comparing best solutions with "ideal" solutions. ATT—Average Travel-Time; Rs—Routes; Ns—Nodes

Network	Ideal ATT	Test Situation	Best ATT	ATT-error%
Mandl's	10.01	8 Rs, max 8 Ns	10.36	3.50
I	2.36	3 Rs, max 4 Ns	2.57	8.90
II	8.59	6 Rs, max 9 Ns	8.95	4.19
III	15.48	9 Rs, max 10 Ns	16.52	6.72

In the final set of experiments we examine 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 route set is produced. Clearly, the efficiency of the Make-Small-Change procedure can be assessed by counting the number of iterations required before a 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 all four of our problem instances. The results are presented in Table 6. Column four of the Table records the average run time required, per solution, for the Make- small-Change routine to produce a feasible solution.

Table 6: Make-Small-Change Procedure Tests

Network Type	Route Set Condition	Feasible Fraction	Time (secs)
I	3 routes, max 4 nodes	1 / 14	0.000123
	3 routes, max 5 nodes	1 / 6	0.000061
	4 routes, max 4 nodes	1 / 5	0.000055
	4 routes, max 5 nodes	1 / 3	0.000016
Mandl's	4 routes, max 8 nodes	1 / 582	0.001735
	6 routes, max 8 nodes	1 / 126	0.000454
	7 routes, max 9 nodes	1 / 88	0.000378
	8 routes, max 7 nodes	1 / 57	0.000158
II	8 routes, max 8 nodes	1 / 786	0.002365
	8 routes, max 9 nodes	1 / 689	0.001013
	9 routes, max 10 nodes	1 / 175	0.000684
	10 routes, max 9 nodes	1 / 162	0.000598
III	10 routes, max 10 nodes	1 / 1340	0.003465
	11 routes, max 10 nodes	1 / 967	0.002692
	12 routes, max 12 nodes	1 / 577	0.001487
	12 routes, max 16 nodes	1 / 365	0.000856

¿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 number of nodes allowed per route is increased.

8 Conclusions and Future Work

In our paper we have presented a framework for solving the UTRP, consisting of the following components: 1) a *representation* for the problem, 2) an *initialization procedures* to construct feasible route sets at random, and 3) 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. Finally, we have introduced a simplified model of the UTRP, which evaluates routes according to the average in-vehicle travel time and the number of transfers between vehicles. Nevertheless, even this relatively simple model involves complicated calculations, and infeasible route sets are all too easily produced by random procedures that form the basis of metaheuristic techniques.

In future work we plan to extend our study to even larger problems, and incorporate operator costs into our model, by making some simple assumptions regarding service frequencies. In addition, we propose to incorporate more realism into route choice, recognizing the diversity of preferences amongst the travelling public. In doing this, our priority will be to develop a more sophisticated and realistic model, yet maintain simplicity, as far as possible, so that other researchers will be able to replicate our results. In addition, we will experiment further with various heuristic and metaheuristic algorithms, in an attempt to improve performance.

References

[1] Jitendra Agrawal and Tom V.Mathew, Transit Route Network Design Using Parallel Genetic Algorithm, Journal of Computing in Civil Engineering, July (2004) 248-256.

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

[3] M. Hadj Baaj and Hani S.Mahmassani, Hybrid Route Generation Heuristic Algorithm for The Design of Transit Networks, Transportation Research 3(1) (1995) 31-50.

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

[5] Avishai Ceder and H.M.Wilson, Bus Network Design, Transportation Research -B 20B(4) (1986) 331-344.

[6] E.W. Dijkstra, A Note on Two Problems in Connexion with Graphs, Numerische Mathematik (1959) Vol.1, 269-271.

[7] Partha Chakroborty and Tathagat Dwivedi, Optimal Route Network Design For Transit Systems Using Genetic Algorithms, Engineering Optimization (2002) Vol.34(1), 83-100.

[8] Partha Chakroborty, Genetic Algorithms for Optimal Urban Transit Network Design, Computer-Aided Civil and Infrastructure Engineering 18 (2003) 184-200.

[9] B.Emerson, Design and Planning Guidelines for Public Transport Infrastructure-Bus Route Planning and Transit Streets, Public Transport Authority, 2003.

[10] Lang Fan and Christine Mumford, A Simplified Model of the Urban Transit Routing Problem, the 7th Metaheuristics International Conference, Montreal, Canada, 2007.

[11] Wei Fan and Randy B.Machemehl, A Tabu Search Based Heuristic Method for the Transit Route Network Design Problem, the 9th International Conference on Computer-Aided Scheduling of Public Transport, San Diego, California, 2004.

[12] Wei Fan and Randy B.Machemehl, Using a Simulated Annealing Algorithm to Solve the Transit Route Network Design Problem, Journal of Transportation Engineering, February (2006) 122-132.

[13] R. W. Floyd, Algorithm 97: Shortest Path, Communications of the ACM (1962), Vol. 5 (6), 345.

[14] Yechezkel Israeli and Avishai Ceder, Designing Transit Routes at the Network Level, IEEE Vehicle Navigation and Information Systems Conference, (1989), 310-316.

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

[16] 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 (1967), 375-397.

[17] Christoph E.Mandl, Applied Network Optimization, Academic Press, London, 1979.

[18] Christoph E.Mandl, Evaluation and Optimization of Urban Public Transport Networks, Third Congress on Operations Research, Amsterdam, Netherlands (1979).

[19] Christoph E.Mandl, Evaluation and Optimization of Urban Public Transport Networks, European Journal of Operational Research 5 (1980) 396-404.

[20] S.B. Pattnaik, S. Mohan and V.M. Tom, Urban Bus Transit Route Network Design Using Genetic Algorithm, Journal of Transportation Engineering July/August (1998) 368-375.

[21] L. Simman, Z. Barzily and U. Passy, Planning the Route System for Urban Buses, Comput. Ops. Res. 1 (1974), 201-211.

[22] V.M.Tom and S.Mohan, Transit Route Network Design Using Frequency Coded Genetic Algorithm, Journal of Transportation Engineering, March/April (2003) 186-195.

[23] Peter White, Public Transport: Its Planning, Management and Operation, 4th Edition, Spon Press, 2002.

[24] Fang Zhao and Albert Gan, Optimization of Transit Network to Minimize Transfers, Final Report, Lehman Center for Transportation Research, Florida International University, 2003.