Exact and Metaheuristic Algorithms for the Urban Transit Routing Problem

Bruno Coswig Fiss
Marcus Ritt
Universidade Federal do Rio Grande do Sul - UFRGS
Instituto de Informática
Departamento de Informática Teórica
Av. Bento Gonçalves, 9500 - Agronomia - 91501-970 - Porto Alegre - RS - Brasil
{bcfiss,marcus.ritt}@inf.ufrgs.br

RESUMO

O problema de roteamento de transporte urbano consiste em encontrar rotas satisfatórias para transporte público em uma cidade ou região. Cenários urbanos se tornam maiores e mais complexos com o passar do tempo, tornando o planejamento de rotas uma tarefa proibitivamente difícil, cujos resultados são frequentemente insatisfatórios, com altos custos e tempos de viagem. Nós propomos uma formulação MIP exata e um algoritmo genético multiobjetivo para a solução do problema com mais qualidade e eficiência do que técnicas atuais. Testamos nossas soluções com cenários reais e artificiais publicados anteriormente e obtemos resultados superiores para todos.

PALAVRAS CHAVES. Problema de Roteamento Urbano.
Área principal: L&T – Logística e transporte, MH – Metaheuristicas, OC – Otimização combinatória

ABSTRACT

The urban transit routing problem (UTRP) consists of finding satisfying routes for public transportation within a city or region. Urban scenarios get bigger and more complex every day, making the design of routes an overwhelming task whose results are often unsatisfactory, with high costs and travel times. We develop an exact MIP formulation and a multi-objective genetic algorithm to solve this problem with higher quality and more efficiently than with current techniques. We benchmark our solutions on generally available real and artificial test cases and achieve better results for all of them.

KEYWORDS. Urban Transit Routing Problem.
Main area: L&T – Logistics and transport, MH – Metaheuristics, OC – Combinatorial optimization

1 Introduction

The urban transit routing problem is of vital importance in public transport systems since it directly defines the routes taken by city dwellers on their daily tasks. Given the size of towns with high populations, their transportation demands get quickly out of manageable complexity for human planners. Concerns for the environment, and also the simple lack of space for more cars incentives the use of public transport. Thus, there is clear demand for cost and travel efficient public transit networks.

To deal with this demand, the urban transit network design problem (UTNDP) is defined. Its concerns are the definition of routes and schedules for the urban transport system. It is an NP-hard problem with multiple criteria, such as the satisfaction of the users, the cost for operating the network and others. Its research is timely in order to improve urban transit worldwide. The UTNDP is commonly divided into two components: the transit routing problem and the transit scheduling problem (Chakroborty (2004)). Given the problem complexity, it is usual to separate solutions to the UTNDP in these two components. Our concern here is solving the transit routing problem.

The initial task is formally defining the problem as simply as possible, but taking into consideration all of the aspects which are necessary to make the problem definition realistic. Based in previous works (Fan et al. (2009)), we choose a simple and adaptable definition that allows the comparison of multiple previous works. The advantage in choosing such a definition is the possibility to compare algorithms directly, and thus be able to select the best approaches to solve the UTRP, without stumbling on problem definition differences. This definition is given in Section 2.

We propose two solutions to the problem. The first one is based on Mixed Integer Programming, and solves the problem exactly. Exact solutions have been proposed and realized partially before, e.g. Israeli & Ceder (1989) takes a multi-step approach, where some steps are completed with a linear programming formulation, and others are decided by the traffic planner. Another example is the approach of Borndörfer et al. (2007), that generates feasible initial solutions using Linear Programming heavily, although not taking transfers into consideration. To the best of our knowledge, no complete exact solver for the UTRP has been published before. Our exact solver is described in Section 3.

The second solution is a genetic algorithm. Heuristics and metaheuristics are the common choice for solving the UTRP (Álvarez et al. (2010)), giving the problem complexity and high number of constraints. We base our solution on previous works which were able to achieve good results, and take advantage of key aspects that were not used before. Such aspects include a *simplification* operator, the use of minimum spanning tree based initial route sets, besides use of the linear relaxation of our MIP formulation for the generation of more feasible initial routes. This algorithm is described in Section 4.

Finally, to assess the quality of the heuristics, we compare our work with previous ones (Mandl (1980), Baaj & Mahmassani (1991), Kidwai (1998), Chakroborty (2004), Fan et al. (2009), Fan & Mumford (2010)) that use a common benchmark (Mandl's Swiss road network), since many new algorithms are not tested against common or openly available targets and thus cannot be compared directly (Agrawal & Mathew (2004), Fan & Machemehl (2008), Álvarez et al. (2010)). The results are presented and analyzed in Section 5. Our conclusions and final remarks are then presented in Section 6.

2 UTRP Definition

The urban transit routing problem consists of finding a set of traffic routes, given passengers and operator constraints, that achieves good average travel times, low number of transfers, low costs for the operator, or a combination of these goals. This and the next definition follow (Fan et al. (2009)).

A graph $G(V, E)$ represents the *traffic network*, where $V = \{v_1, \dots, v_n\}$ is the set of nodes representing bus stops, train stops, or more broadly, access points where the transport is able do pick up and drop off passengers, and where $E = \{e_1, \dots, e_n\} \subseteq V \times V$ represents the set of direct physical connections between nodes.

A route in the traffic network $G(V, E)$ is a path $r_a = (v_{i_1}, \dots, v_{i_q})$, where $i_k \in \{1, \dots, n\}$. A set of routes $R = \{r_a : 1 \le a \le N\}$, where N is the number of possible routes, defines a solution to the UTRP problem.

2.1 Objective function

To evaluate the quality of a route set, one must first define a *route network*, which contains only the edges of the respective route. We define as E_a the set of edges of the route r_a. Then, the *transit network* can be defined as a graph $H(V', E')$ in which $V' \subseteq R \times V$. The node $w_{ia} \in V'$ in the *transit network* is a pair that combines the route r_a and the node $v_i \in V$ from the *transport network*. Consequently, we define the edges in E' in two parts. E'_1 corresponds to the set of edges within individual routes, and E'_2 represents transfers:

$$E'_1 = \bigcup_{r_a \in R} \{(w_{ia}, w_{ja}) : (v_i, v_j) \in E_a\}, \quad E'_2 = \bigcup_{v_i \in V} \{(w_{ia}, w_{ib}) : v_i \in r_a \cap r_b\}.$$

Given the *transit network*, one can define two cost functions, one measuring the user cost and one the operator cost. The operator objective function is defined as:

$$C_O(R) = \sum_{r_a \in R} \sum_{e \in E_a} c(e),$$

where $c(e)$ is the cost of operating edge e.

To define the user cost, we use t_e, the time required to travel through edge e, and the travel penalty t_{pen}, which is the time it takes to make a transfer. The value t_{pen} is assumed to be constant since we do not deal with scheduling. The value t_{pen} also includes a time penalty regarding the inconvenience of having to make a transfer instead of staying on the same route. Given that, the edges in E'_2 are set to the length t_{pen}, while the edges $(w_{ia}, w_{ja}) \in E'_1$ to the length $t_{(v_i, v_j)}$. Now, the minimum journey time in transit network R from v_i to v_j, $\alpha_{ij}(R)$, is defined as the shortest path from a node in $\{w_{ia} : v_i \in r_a\}$ to a node in $\{w_{jb} : v_j \in r_b\}$.

Let d_{ij} denote the transit demand from node v_i to node v_j (defined as the number of passengers wishing to travel from v_i to v_j). Assuming the passengers will always choose to travel on the shortest paths, the user cost functions TTT (Total Travel Time) and ATT (Average Travel Time) can be defined as follows:

$$TTT = \sum_{(v_i, v_j) \in V \times V} d_{ij} \alpha_{ij}(R), \qquad ATT = TTT \left(\sum_{(v_i, v_j) \in V \times V} d_{ij} \right)^{-1}$$

Since there is more than one objective function to be optimized in this problem, solutions can be classified as *dominated* or *undominated*. In a set of solution candidates, a solution s is *undominated* if and only if no other candidate in the set is better than s on both quality measurements.

2.2 Restrictions

Besides the restriction requiring every route to be a path, further restrictions are often enforced in variants of the problem. It is common not to allow cycles or backtracks within routes, to limit the size of routes, and it is often assumed that routes are undirected, i.e. that the public transport travels in both directions of the route. Nevertheless, it is not uncommon there to be different paths depending on the direction of the route, and cycles are also known to occur. Therefore, we do not assume any of those restrictions as necessary, and all of our algorithms are able to deal with any subset of desired restrictions, within those mentioned above, and produce solutions that satisfy them.

Another common characteristic in the previous models of this problem is that they fix the number of desired routes. This may or may not be realistic, since on the one hand having more routes may reflect in more costs, but on the other hand, one already considers operator costs within the operator cost function. For this reason, a lower and upper bound on the number of routes can also be configured in our algorithms.

3 Mixed Integer Programing formulation

In this section we present a formulation that solves the UTRP using mixed integer programming.

3.1 Variables

Our formulation uses the following decision variables: let R be a set of routes. For every route $r \in R$, s_{rv} indicates if vertex v is the initial vertex of route r. Similarly, f_{rv} indicates if v is the final vertex of route r. x_{re} is 1 if and only if the edge $e \in E$ belongs to route r, and x_{re} is 0 if $e \notin E$. For every pair of vertices v and w, s^*_{vwr} indicates if the best path between v and w starts on route r. Likewise, f^*_{vwr} indicates if r is the final route of the best path between v and w. Finally, $x^*_{vwe'_{rnsm}}$ indicates if edge $e'_{rnsm} \in E'$ is on the best path between v and w, where E' is the set of edges of the *transit network*, as explained in Section 2. Additionally, the real variables p_{rv} indicate the position, starting from 0, of the vertex v on route r. To save space, we use Iverson's brackets $[P]$: the expression $[P]$ is 1 if the predicate P is true, and 0 otherwise.

3.2 Constraints

$$\sum_{v\in V} s_{rv} = 1 \qquad \forall r \in R \tag{1}$$

$$\sum_{v\in V} f_{rv} = 1 \qquad \forall r \in R \tag{2}$$

$$\sum_{e=(v,w)\in E} x_{re} + f_{rv} = \sum_{\bar{e}=(w,v)\in E} x_{r\bar{e}} + s_{rv} \qquad \forall r \in R \tag{3}$$

$$\sum_{e=(v,w)\in E} x_{re} + f_{rv} \leq 1 \qquad \forall r \in R, v \in V \tag{4}$$

$$p_{rv} \leq |V| - |V|s_{rv} \qquad \forall r \in R, v \in V \tag{5}$$

$$p_{rw} - |V| + |V|x_{re} \leq p_{rv} + 1 \qquad \forall r \in R, (v,w) \in E \tag{6}$$

$$p_{rv} + 1 \leq p_{rw} + |V| - |V|x_{re} \qquad \forall r \in R, (v,w) \in E \tag{7}$$

$$x^*_{vwe'_{rnrm}} \leq x_{re} + x_{r\bar{e}} \qquad \forall v, w \in V, r \in R, e = (n,m) \in E \tag{8}$$

$$x^*_{vwe'_{rnsm}} = 0 \qquad \forall v, w \in V, n \neq m \in V, r \neq s \in R \tag{9}$$

$$\sum_{r\in R} s^*_{vwr} = 1 \qquad \forall v, w \in V \tag{10}$$

$$\sum_{r\in R} f^*_{vwr} = 1 \qquad \forall v, w \in V \tag{11}$$

$$\sum_{\substack{s\in R,\\ m\in V}} x^*_{vwe'_{rnsm}} + [w=n] f^*_{vwr} = \sum_{\substack{s\in R,\\ m\in V}} x^*_{vwe'_{smrn}} + [v=n] s^*_{vwr} \qquad \forall v,w \in V, n \in V, r \in R \quad (12)$$

There are also constraints for improving performance, such as not allowing multiple transfers on the same node and path, not allowing cycles in the best paths, and ordering the routes such that shorter ones have smaller indices. These constraints only decrease search space without affecting the set of feasible solutions.

3.3 Objective functions

Given that $c(e)$ is the cost for operating edge e, t_e is the time for traversing edge e, d_{ij} is the demand between nodes i and j, and t_{pen} is the time penalty for making a transfer between routes, the following two objective functions are defined (as explained in Section 2.1):

$$\textbf{minimize} \quad \sum_{\substack{r\in R,\\ e\in E}} c(e)x_{re} \qquad \text{(operator cost)}$$

$$\textbf{minimize} \quad \sum_{\substack{r\in R,\\ v,w\in V,\\ e=(n,m)\in E}} t_e d_{vw} x^*_{vwe'_{rnrm}} + \sum_{\substack{r,s\in R,\\ v,w,n\in V}} t_{pen} d_{vw} x^*_{vwe'_{rnsn}} \qquad \text{(user travel time)}$$

3.4 Discussion

The formulation begins by fixing the number of routes $|R|$ (where $R = \{1, 2, \ldots, |R|\}$) in the solution route set. This implies in multiple optimizations when a range of number of routes is acceptable, one for each size.

Each route must then be defined. For each route $r \in R$ there are $2|V| + |E|$ binary variables that define it completely. Constraints (1) and (2) guarantee that there will be unique start and end nodes for each route. Constraints (3) and (4) assure that every node within a route will have an equal number of outgoing and incoming edges active in the route, and this number shall be smaller or equal to one, except for the start and end nodes, which shall have zero incoming or outgoing edges, respectively.

To remove closed loops which may be left in the previous steps, the p_{rv} variables and the Constraints (5)–(7). Constraint (5) sets the position of the first node in a route to 0. Constraints (6) and (7) set the position of a node that comes after another node in a route to one plus the position of the previous node, assuring sequential positioning. Therefore, the first node will have the smallest position. Since a closed loop has no start, no potential position can successfully be assigned to a node in it. It should be kept in mind that, if extra closed loop routes are acceptable in a solution (i.e. if the number of routes can be bigger than the given $|R|$), these constraints can be removed.

The rest of the formulation finds $|V|^2$ shortest paths, for each pair origin-destiny, based on the available routes, following a standard approach for shortest paths LP formulations. Constraint (8) assures that best paths will consist only of edges available in the chosen routes. Constraint (9) disallows moving between routes, except when this movement is from a node to itself, in which case it would characterize a transfer (and thus is allowed).

Constraints (10) and (11) guarantee that there will be unique start and end routes for each best path. Constraint (12) assures that every best path will actually be a path, by balancing the number of ingoing and outgoing edges of every node (which is, in this case, a pair in $R \times V$, as per Section 2.1).

The variables s^*_{vwr}, f^*_{vwr} and $x^*_{vwe'}$ are actually implemented as real variables between 0 and 1. If their values end up being non-integer on an optimal solution, the solution is still valid and can be interpreted as follows: the best path can be taken in several different manners, and each manner is used in proportion to the value of the corresponding variable. e.g. if s^*_{vwr} is 0.5, then 50% of the demand from v to w starts on route r. Furthermore, every single path has the same length, since otherwise the rest of the demand would also use the shortest path, and this would provide a solution which is better than the optimal, a contradiction. This also means that there is at least one integer solution which is equivalent to any optimal non-integer solution found.

Finally, using mixed integer programming, one can only optimize a single objective function. To handle this, we use two approaches: either summing both functions, each weighted by an importance factor, or setting one of the goals as a constraint, giving it a maximum value. This gives the traffic planner the possibility to find the best solution for one of the goals, respecting boundaries on the other, or to find the best solution given a certain *linear trade-off* between the two objective functions.

4 Metaheuristic approach

The metaheuristic used here is a genetic algorithm (Holland (1975)). Each solution consists of a set of routes, and each route is represented as a sequence of nodes. There are operations to add nodes to and delete nodes from routes, and add and remove routes from route sets. It follows the well-known structure of a genetic algorithm: an initial population is generated (as described in Section 4.1), and the constant sized population evolves until some stopping criterion, e.g. the number of generations, is satisfied. In each generation, each member of the population suffers a mutation, is evaluated, and in case it dominates an existing solution in the population, it will replace it. If more than one solution would be dominated by this new element, only the first solution found will be replaced (the search begins on its original self).

An important aspect of our approach is the representation, that is not as simple as commonly seen (Agrawal & Mathew (2004), Chakroborty (2004), Fan et al. (2009)). It consists of dynamic data structures such as sets and vectors instead of simple strings. This is justified because representing, moving, copying and modifying routes is not the bottleneck in this problem, but the *evaluation* of route sets. This means the structure is made more generic, easier to program and to adapt to new restrictions, without significant performance loss.

The next key aspect are the operators. The mutation operator is based on the *Make-Small-Change* procedure, from Fan et al. (2009). It applies very small changes to routes, and thus navigates through the neighborhood of solutions. This is important in order to find local minima, maximizing the potential of some route set, but must be complemented so that other areas of the solution space may also be explored. The only operations that are executed are the addition and removal of nodes at the start or end of a route. Only one node is added or removed at a time.

The last important characteristic is the maintenance of a base and an undominated solution list, apart from the population. The first one is used for the initialization procedure, and the second one maintains every undominated solution found. Every time a new solution is created by an operator, it may not dominate any solutions, but still be undominated. Since the population size is constant, this new undominated solution would be lost. Instead, it is kept in the undominated list, whose size is dynamic.

We attempt to achieve a more effective algorithm by carefully selecting initial solutions, applying new operators such as *simplification* and *crossover*, not letting a feasible solution be removed if it is *undominated* and allowing different route set sizes in the same population. These operations and generation of initial solutions will be explained next.

Since this is a multi-objective optimization problem, we only consider *domination* for the classification of solutions. Thus, there is no fitness level that rates and orders solutions, and all are considered equal as long as one does not dominate another. Therefore, when applying operators, every solution candidate has the same probability of being chosen, unless otherwise stated. Besides that, every element in the population suffers exactly one mutation per evolution step.

The evaluation, the most expensive step of the algorithm, is done using Dijkstra's algorithm. In this way, a solution can be evaluated in time $O(|R|^2|V|^2 \log(|R||V|))$, since there are $O(|V|)$ evaluations, one per node, and each one corresponds to an execution of Dijkstra's algorithm over a graph with $O(|R||V|)$ nodes and $O(|R|^2|V|)$ edges.

The overall running time of the algorithm would be, then, for a population of size P and T evolution steps, $O(PT|R|^2|V|^2 \log(|R||V|))$. An advantage of having the bottleneck on the evaluation of a solution is the fact that the overall running time can be parallelized up to a factor of P. This has not been taken advantage of yet, and will be discussed further in Section 6.

4.1 Generation of initial solutions

The first step of the generation of initial solutions is the creation of a base list of solutions. These are the minimum spanning trees of the *transport network* and the subgraph of the *transport network* containing only shortest paths given by the Dijkstra or Floyd-Warshall algorithm. All routes contained in one of those subgraphs are joined into a route set, and the route sets get simplified (as described in Sections 4.2 and 4.3). Another source of viable routes is the relaxed solution of the MIP formulation. One can create paths by randomly choosing a node and then random traveling through edges with probability equal to x_{re} and thus obtain new routes.

Then, the extraction of valid route sets begins. The extraction respects minimum and maximum number of nodes in route, existence of cycles and number of routes per route set. It is a random procedure, possibly extracting a different part of the route or route set at each run. The extractions are then performed several times (since they are random), and all of the resulting valid solutions initialize the base list. Then, a variable number of crossovers is executed between random valid solutions. At the end, every route gets simplified, and the base list is ready. Meanwhile, the undominated list is also fed with every undominated solution in the base list.

The next step is actually creating the initial population. This is done by taking one third of P solutions from the base list, one third of P solutions from the undominated list, and making crossovers to fill the remaining third.

The process of creating a new population can then be repeated after a number of evolution steps. This does not involve the creation of the base and undominated list anymore, and thus enables faster renewal of the whole population.

4.2 Route set simplification operation

When finding a shortest path between two nodes, one will travel exactly through the shortest paths between the intermediate nodes. This is a known characteristic of problems to which dynamic programming can be applied, and here it is taken advantage of in a different way.

As explained in Section 4.1, some route sets are determined and saved on a list of base route sets. This list will be used to build new initial route sets and candidate solutions. One of the source of routes for the base route sets is the shortest path between nodes. These are all added into the routes sets of the base list. But, as discussed above, there are many repetitions between these shortest routes.

Besides, when using a random decision based algorithm, changes to routes can naturally lead to the same situation: some routes can be *contained* in others. Even when not so, two routes may still be *joinable* without any disadvantage to users or to operators. This is formally defined as follows:

the route r can be *joined* with s if: $\exists i_{0 \leq i \leq |s|-1} : \forall j_{1 \leq j \leq min(|r|,|s|-i)} : r_j = s_{j+i}$. Two routes r and s are joinable if either one of them can be joined with the other.

These ideas lead to the development of an operation that attempts to combine every pair of routes belonging to a route set. This operation leads to big savings of operator costs for the initial route list, and improves the route set during the execution of the genetic algorithm.

4.3 Crossover

The route sets consist of many routes. The first crossover that has been implemented simply exchanges some of the routes, chosen randomly, between route sets, and simplifies the route set afterwards. This is useful to explore new solutions, but does not actually change routes.

In order to try to join two routes and perhaps keep qualities of both of them, a mix up operation was also defined. With it, a crossover not only exchanges routes, but also mixes them up. The operation works as follows: a cut point is randomly defined in both routes. From the beginning of the first route until its cut point, all nodes are present in the result. From there on, the nodes are copied from the cut point of the second route until its end. After that, all the currently applying restrictions are considered, such as prohibition of cycles and limits on the number of nodes on a route. Nodes are excluded from or included into the resulting route until it satisfies all restrictions.

5 Experimental Results

We tested our approaches on Mandl's Swiss road network (Mandl (1980)), with 15 nodes and 21 links, the only, to the best of our knowledge, commonly used benchmark for the UTRP, in order to compare our results with previous works (Baaj & Mahmassani (1991), Kidwai (1998), Chakroborty (2004), Fan et al. (2009)). We also experimented with larger test cases produced by Fan et al. (2009) to test the solution quality and scalability of our metaheuristic approach.

All results are evaluated using the following quantities, as in previous works: (a) the percentage d_i of the demand satisfied with i transfers, (b) the average travel time ATT (in minutes per passenger), including transfer penalties, and (c) the cost for the operator C_O, i.e., the total route length (in minutes, considering constant transport speed). The only quantity that should be as high as possible is d_0, which indicates how much of the demand can be satisfied without transfers. *All* other measures should be as small as possible.

When evaluating scenarios on which no tests have been previously made, it is clearly impossible to compare qualities with previous results. In this case, a good alternative measurement of solution quality is the gap between its Average Travel Time and the ATT lower bound for the instance. This lower limit can be obtained by calculating shortest paths between every demand pair, separately, in the *transport* network (as opposed to the *route* network).

The exact algorithm was implemented in GNU MathProg Modeling Language, and the metaheuristics using the C++ Programming Language. All execution times have been measured in seconds of real time. Results were obtained on a PC with an Intel Core 2 i5-460M 2.53 GHz processor and 4GB of RAM using Linux Ubuntu 12.04 OS.

5.1 Mixed Integer Programming approach

We obtained the best possible route sets on Mandl's network regarding user travel time for 2 and 3 routes. The *IBM ILOG CPLEX Optimizer 12.4* was used to solve our Mixed Integer Programming formulations. The quality, processing times and the actual routes of the solutions are given in Table 1.

Table 1: Best possible route sets found using the Mixed Integer formulation

Number of routes	2	3
d_0	84.90 %	93.67 %
d_1	14.00 %	5.43 %
d_2	1.10 %	0.90 %
ATT	11.33 min.	10.50 min.
C_O	98 min.	150 min.
Processing time (s)	1065	78992
Two Routes	6-14-7-5-2-1-4-3-11-10-9-13-12	
	0-1-3-5-7-9-6-14-8	
Three Routes	4-3-11-10-12-13-9-7-5-2-1-0	
	4-3-1-2-5-14-6-9-10-11	
	0-1-4-3-5-7-9-6-14-8	

It is clear that solving the UTRP, a NP-hard problem, using exact methods is not scalable nor feasible even for relatively small instances of the problem. Nevertheless, these experiments show the problem size to which it can still be applied, and also validate the correctness of the formulation.

Since the 2 routes case has 144 binary variables and the 3 routes scenario has 216, and given the processing times given in Table 1, one can estimate the processing time required for solving with 4 routes in about 77 days. These results were achieved using all available cores (the processing times can be lowered by further parallelization).

5.2 Metaheuristic approach

5.2.1 Mandl's network

The stopping criterion of the genetic algorithm when applied to Mandl's network was 200, 400 and 600 evolution steps when testing with limits of respectively 4, 6 and 8 routes. The population consisted of 1000 route sets. We chose these values proportional to the ones used in Fan et al. (2009), in order to allow a fair comparison, since the actual processing times were not available. Nevertheless, we decreased the number of iterations and total reruns in order to compensate for hardware advances and possible longer execution times per iteration in our approach.

We first compare our results with the multi-objective approach proposed by Fan et al. (2009), which is better suited for the UTRP since there are two major concerns when developing routes for urban transit: the quality for the passengers and the costs for the operators. The results are present in Table 2. We can see that, in comparison to previously published results, our solutions were always better, i.e. we achieve superior solutions with equal or smaller prices and better travel times. Moreover, they dominate the previous results, being better in all considered measures, or in only of them, but without affecting the others. The four route sets whose results are shown in Table 2 can be obtained with the correspondence author.

To make a broader comparison, the results in all previous works we found on Mandl's network (Mandl (1980), Baaj & Mahmassani (1991), Kidwai (1998), Chakroborty (2004)) were analyzed, and the best results are shown in Table 3, in comparison to our results. Here, the objective is only one: decrease passenger travel time, without taking transfer penalties into consideration. This was necessary to allow a comparison between heterogeneous penalty values. Not considering penalties favors results that were achieved using lower penalty values, given that these are closer to not having a penalty at all. Since we used five minutes penalty per transfer, and this is the highest amount applied in the cited publications, our results should not be favored.

To calculate the Average Travel Time without penalty (ATT_{wop}) the following formula is used (where T_{MAX} is the maximum number of transfers): $ATT_{wop} = ATT - \sum_{i \leq T_{MAX}} t_{pen} d_i i$.

It is important to keep in mind that the comparison made in this single-objective case is not as

Table 2: Comparison between best UTRP multi-objective solutions on Mandl's Network

Scenario	Q_p	BKV	Our results	Scenario	Q_p	BKV	Our results
Best for Passenger	d_0	94.54 %	98.84 %	Compromise Solution ($C_O \leq 126$)	d_0	90.88 %	91.23 %
	d_1	5.46 %	1.16 %		d_1	8.35 %	7.84 %
	d_2	0.00 %	0.00 %		d_2	0.77 %	0.93 %
	ATT	10.36 min.	**10.10** min.		ATT	10.65 min.	**10.59** min.
	C_O	283 min.	**259** min.		C_O	126 min.	126 min.
Compromise Solution ($C_O \leq 148$)	d_0	93.19 %	93.61 %	Best for Operator	d_0	66.09 %	77.78 %
	d_1	6.23 %	6.20 %		d_1	30.38 %	21.32 %
	d_2	0.58 %	0.19 %		d_2	3.53 %	0.90 %
	ATT	10.46 min.	**10.43** min.		ATT	13.34 min.	**12.97** min.
	C_O	148 min.	**147** min.		C_O	63 min.	63 min.

BKV: Best known value (Fan et al. (2009)).

Table 3: Comparison between best single-objective UTRP solutions on Mandl's Network

$\|R\|$	Best previous ATT_{wop} (Chakroborty (2004))	ATT_{wop} obtained by our approach
4	10.33 min.	**10.30** min.
6	10.43 min.	**10.11** min.
7	10.53 min.	**10.04** min.
8	11.22 min.	**10.05** min.

fair as in the multi-objective scenario. This is so because, when comparing results from various sources, slight differences in the definition of the problems occur, e.g. regarding the minimum and maximum number of nodes per route, allowance of cycles and others. Besides this, the differences in publishing dates, and therefore also in hardware used for running experiments, make it harder to compare the running time of the approaches. Nevertheless, our approach showed to be successful in the single-objective case as well.

5.2.2 British city based network

To assess the behavior of our metaheuristic approach when dealing with larger networks, we applied it to the network defined and used in Fan et al. (2009), Fan & Mumford (2010), which has 110 nodes and 275 links, with 3603360 journeys per day. The network's size and connectivity are based on a major British city. Two scenarios were defined, each one with its own minimum and maximum number of nodes per route, and total number of routes.

Since our approach can handle multiple route set sizes at the same time, the biggest difference in each scenario is the number of nodes per route. The minimum number is 2 and the maximum number is 29 in scenario I, being these limits derived from the transport network graph, its connectivity and number of nodes, as is stated in Fan et al. (2009). In scenario II, the minimum number is 10 and the maximum number is 22. The limits in scenario II are derived from the actual routes used in the major British city upon which the artificial network was based.

The running time informed in previous works averages between 13000 and 19000 seconds of processing, depending on the scenario. We used no more than 2500 seconds in each test case.

The full comparison between previous results and our outcome for this network is given in Table 4. This time we do not assess d_i for $i \geq 1$ because of space limitations. Nevertheless, one can calculate how much demand is satisfied with transfers by taking $1 - d_0$.

It can be noticed that we optimize the major objective of all scenarios better, with exception of ATT for Scenario I-Passenger, where we got slightly higher results.

The fact that our route sets were superior when not considering penalties shows that most pas-

Table 4: Comparison between best UTRP multi-objective solutions on artificial British city

Scenario	Q_p	BKV	Our results	Scenario	Q_p	BKV	Our results
I-Passenger	d_0	72.91 %	55.80 %	I-Operator	d_0	48.62 %	9.48 %
	ATT	**36.28** min.	36.35 min.		ATT	40.88 min.	55.08 min.
	ATT_{wop}	34.60 min.	**34.12** min.		ATT_{wop}	37.36 min.	45.66 min.
	C_O	2986 min.	8406 min.		C_O	1077 min.	**319** min.
II-Passenger	d_0	71.21 %	46.25 %	II-Operator	d_0	46.97 %	8.47 %
	ATT	37.52 min.	**36.61** min.		ATT	41.26 min.	55.48 min.
	ATT_{wop}	35.68 min.	**33.77** min.		ATT_{wop}	37.655 min.	47.90 min.
	C_O	2378 min.	5181 min.		C_O	1265 min.	**319** min.

BKV: Best known value (Fan et al. (2009)).

sengers travel through the fastest or almost fastest paths in the *transport* network. A reason for this is the frequent use of these fastest paths as candidate routes in our algorithm.

Another important remark is that our outcome had much cheaper (to operate) route sets in the operator-oriented scenarios. This shows that our metaheuristic approach was also successful when optimizing prices instead of travel times. This helps producing a much wider range of available compromise solutions to be chosen by a public transport network planner, thus improving overall network quality.

The possible drawback in our passenger-oriented route sets is the high number of people that need to make transfers. This is overcome, in most cases, by a faster arrival time, despite transfers (and considering transfer penalties). Improvements for this are discussed in Section 6.

6 Concluding Remarks and Future Work

We developed an exact MIP formulation for the UTRP. To the best of our knowledge, a full exact solution to this problem was never published before. With it, we obtained best possible solutions for small sized scenarios. A time estimative for exactly solving a certain instance of the problem was derived, and with it, limits on the problem size to which exact solutions are feasible can be derived. From the MIP formulation, we were also able to obtain route suggestions for our metaheuristic approach.

With a new approach, using a genetic algorithm, we achieved better results than every other published in literature, with the exception of one test case configuration. Most of our solutions also *dominated* previous solutions, and thus did not only improve the main goal, but all goals. This was done with a more flexible implementation in comparison to previous approaches, which is not bound to certain restrictions such as fixed number of routes, minimum and maximum number of nodes per route, allowance of cycles, directedness of routes and more. Our processing times were also significantly lower than in previous works. Thus, the algorithm achieves better results faster. A reason for this may be due to bigger speed of exploration of the solution space, and the avoidance of local minima by using operators which had not been used in previous algorithms, such as the *simplification* operator. Another characteristic of our algorithm is that it can be parallelized up to a factor equal to the population size. Actually implementing this remains as future work.

A possible drawback of our approach is the higher number of transfers on bigger networks when compared to previous works. This is most likely because we do not create routes attempting to maximize the covered demand, unlike previous approaches. Even though the main objectives are nevertheless better in our solutions, the use of a demand oriented procedure for the creation of routes is planned to be added to our algorithm to solve this.

To put the algorithm in real validation, we plan to use it on real networks for big capitals such as Porto Alegre, Brazil and Berlin, Germany. We could also apply our solution route sets to simulation software to validate and assess further. A known simulation package, which we plan to use, is MATSim (Balmer et al. (2009)). To handle greatly sized instances, the parallelization of the algorithm may be necessary. The last suggestion for taking the work on is the broadening of the problem definition as to take into consideration an already existing transit network, and the costs to change it in comparison to the benefits of doing so.

With this work, we hope to help improve the overall quality of available algorithms to solve the UTRP, which becomes more and more vital as public urban transit networks grow larger and more complicated.

Acknowledgements The support of this research by the Brazilian Conselho Nacional de Desenvolvimento Científico e Tecnológico, grant Labtrans 402450/2009-4, is gratefully acknowledged.

References

Agrawal, J. & Mathew, T. V. (2004). Transit route network design using parallel genetic algorithm. *J. Comput. Civ. Eng.*, 18:248.

Álvarez, A., Casado, S., González Velarde, J., & Pacheco, J. (2010). A computational tool for optimizing the urban public transport: A real application. *Journal of Computer and Systems Sciences International*, 49:244–252.

Baaj, M. & Mahmassani, H. (1991). An AI-based approach for transit route system planning and design. *J. Adv. Transp.*, 25(2):187–209.

Balmer, M., Rieser, M., Meister, K., Charypar, D., Lefebvre, N., & Nagel, K. (2009). *Multi-agent systems for traffic and transportation engineering*, chapter MATSim-T: Architecture and simulation times, pages 57–78. Information Science Reference.

Borndörfer, R., Grötschel, M., & Pfetsch, M. E. (2007). A column-generation approach to line planning in public transport. *Transp. Sci.*, 41(1):123–132.

Chakroborty, P. (2004). Optimal routing and scheduling in transportation: Using genetic algorithm to solve difficult optimization problems. *Directions*, 6(3):29–40.

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

Fan, L., Mumford, C. L., & Evans, D. (2009). A simple multi-objective optimization algorithm for the urban transit routing problem. In *Proc. 11th Conf. Evol. Comput.*, pages 1–7. IEEE Press.

Fan, W. & Machemehl, R. B. (2008). A tabu search based heuristic method for the transit route network design problem. In Hickman, M., Mirchandani, P., & Voss, S., editors, *Computer-aided Systems in Public Transport*, volume 600 of *Lect. Notes Econ. Math. Syst.*, pages 387–408.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Israeli, Y. & Ceder, A. (1989). Designing transit routes at the network level. In *Vehicle Navigation and Information Systems Conference, 1989*, pages 310 –316.

Kidwai, F. (1998). *Optimal design of bus transit network: a genetic algorithm based approach*. Doctoral Thesis, PhD. dissertation, Indian Institute of Technology, Kanpur, India.

Mandl, C. (1980). Evaluation and optimization of urban public transportation networks. *Eur. J. Oper. Res.*, 5(6):396–404.