Computers & Operations Research 36 (2009) 2440 – 2449

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

A route set construction algorithm for the transit network design problem

Antonio Mauttone*, María E. Urquhart

Operations Research Department, Universidad de la República, J. Herrera y Reissig 565 Piso 5, Montevideo, Uruguay

ARTICLE INFO

Available online 17 October 2008

Keywords:
Transit network design problem
Constructive algorithm
Heuristic
Demand covering constraints

ABSTRACT

The transit network design problem (TNDP) aims to determine a set of bus routes for a public transportation system, which must be convenient from the viewpoints of both users (people who use public transportation) and operators (companies who own the resources to give the service). This article presents a constructive algorithm for the TNDP. It is specially designed to produce a set of routes that fulfils demand covering constraints, while taking into account the interests of both users and operators. Its general structure is inspired in the Route Generation Algorithm (RGA) of Baaj and Mahmassani, where its original expansion of routes by inserting individual vertices is replaced by a strategy of insertion of pairs of vertices. The algorithm proposed, called Pair Insertion Algorithm (PIA) can be used to generate initial solutions for a local improvement or evolutionary algorithm, as well as to complete an unfeasible solution with respect to demand covering constraints. Numerical results comparing PIA with RGA over a real test case show that both algorithms produce solutions with similar quality from the users viewpoint (in terms of in-vehicle travel time), while the former produces better solutions from the operators viewpoint (in terms of number of routes and total route duration) and requires a higher execution time. Since the TNDP arises in a context of strategic planning, a solution that reduces the operation cost of the system is highly desirable, even though it takes more time to be computed. The experimental study of the proposed algorithm also shows its ability to produce diverse solutions in both decision and objective spaces; this is a useful property when looking at the use of PIA as a subroutine in the context of another algorithm such as metaheuristics, in particular for a multi-objective problem like TNDP.

1. Introduction

Public transportation plays an important role in the dynamic of many cities. It is widely recognized as a means to reduce traffic congestion, to improve urban environmental conditions and to contribute to the social inclusion of the inhabitants.

A key component in the overall planning of a public transportation system is the network design, where a set of routes is defined over the street network. According to Ceder and Wilson [1], network design is the first of the five stages of a systematic decision sequence, followed by frequency setting, timetable development, bus and driver scheduling. Decisions at network design level are usually taken for a long time horizon, in the context of strategic planning [2]. The transit network directly determines characteristics of the public transportation system with respect to the users interest such as geographical accessibility and travel time; alongside with the frequencies, it also defines an important component of the cost for the operators. Once the transit network is defined, all the subsequent decisions about timetable development, and bus and driver scheduling are conditioned to it, so the overall cost of the public transportation system highly depends on the transit network [1].

The transit network design problem (TNDP) aims to find a set of routes with their corresponding frequencies, optimizing the objectives of users and operators [3]. Main problem data are the street network and the demand of trips between different points of the city. Constraints refer usually to demand covering, required level of service and resource availability. Frequencies are included in the TNDP as decision variables because they also have a direct influence in the cost structure of both users (determining the waiting time) and operators (defining the required fleet size).

Several formulations have been proposed to model the TNDP [3–6]. Other aspects of the problem have been modelled, such as elastic demand [7–9], multi-modality [10,11], transfers and route directness [12].

The TNDP is a hard to solve combinatorial problem [5], given the discrete nature of some of its variables (those that represent routes). It is also difficult to formulate with a mathematical programming approach [13]. For all these reasons, most existing approaches to solve it rely on approximative methods, i.e. heuristics and metaheuristics [4,14–17]. Most of these methods use an specific purpose algorithm

* Corresponding author. Tel.: +598 2 711 42 44; fax: +598 2 711 04 69.
E-mail address: mauttone@fing.edu.uy (A. Mauttone).

0305-0548/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2008.09.014

to explicitly construct a set of routes, which is not always feasible with respect to demand covering constraints; a solution is said to be feasible with respect to demand covering constraints if all the demand given by an origin–destination matrix can be transported by the routes of the solution. These constraints are very important to fulfil.

A few works exist in the literature about heuristic algorithms to construct a set of routes for the TNDP, while ensuring demand covering feasibility [5,6,14]. Only the Route Generation Algorithm (RGA) of Baaj and Mahmassani [14] generates starting from an empty solution, a set of routes that covers the whole demand, while considering some aspects of interest from both users and operators points of view. RGA was designed taking into account several desirable properties and design principles (some of them taken from the practitioners); in this sense, it can be considered as a heuristic that incorporates a deep knowledge of the problem from the application viewpoint. It includes a mechanism to explicitly cover the demand (with or without transfers). However, the cost for the operators (represented by number of routes and total route mileage) of the resulting set of routes produced by RGA remains high when the requirement of demand covering is increased [14].

In this work we take some ideas from RGA and we add new ones to propose an algorithm that produce solutions that are more convenient for the operators in comparison with RGA, while maintaining almost the same cost for the users (in terms of travel time). Those solutions are highly desirable, since they reduce the operation cost (both fixed and variable) of the transit system, allowing to operate a more sustainable system while not degrading its quality from the users viewpoint. The algorithm proposed address the problem of the fulfilment of demand covering constraints, while taking care in producing solutions that are convenient for both users (low in-vehicle travel time) and operators (low number of routes and total route mileage). A solution constructed by the proposed algorithm can be used as a starting point for a local improvement method or as an initial solution for an evolutionary algorithm, in order to improve its quality. The algorithm can be also used to complete an unfeasible solution with respect to demand covering constraints. We have performed computational experiments which are based on a real test case for which a description of its construction is given; the advantage of using a real case is that it constitutes a reference to compare the results produced by the algorithm.

The article is organized as follows. In Section 2 we give some definitions and notation used in the following sections. A brief literature review specially focused on approximative methods and route set construction algorithms for the TNDP is given in Section 3, as well as the motivation of this work. A detailed description of the proposed route set construction algorithm is presented in Section 4. Numerical results of an experimental study and their analysis are shown in Section 5 while conclusions and future work are given in Section 6. In Appendix A, a summarized explanation of the procedures followed in order to construct the real test case is given.

2. Definitions and notation

We model a network that enables the definition of routes as an undirected graph $G=(N,E)$, where N is the set of vertices ($|N|=n$) and E is the set of edges representing connections between vertices. The cost c_e of an edge $e=(i,j)\in E$ models the in-vehicle travel time, i.e. the time spent by vehicles to travel between vertices i and j. An origin–destination matrix $D=\{d_{ij}, i,j\in[1\dots n]\}$ is given, which characterizes the demand; d_{ij} denotes the demand from vertex i to vertex j, expressed in trips per time unit in a given time horizon. A route is a sequence of adjacent vertices in G. A solution S for the TNDP is a pair (R,F) where $R=\{r_1,\dots,r_r\}$ is the set of routes and $F=\{f_1,\dots,f_r\}$ is the set of frequencies; each f_k is a real value that represents the inverse of the time between subsequent vehicles on route r_k. We only work with the component R in this article.

Table 1
Demand covering for three different sets of routes

R	$D_0(R)$	$D_0(R)\geqslant D_0^{min}$	$D_{01}(R)$	$D_{01}(R)\geqslant D_{01}^{min}$
$\{(1,2)\}$	0.10	×	0.10	×
$\{(1,2);(2,3)\}$	0.20	×	1.00	✓
$\{(1,2,3)\}$	1.00	✓	1.00	✓

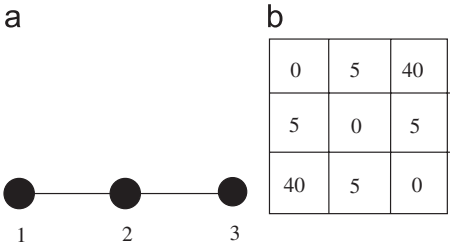

Fig. 1. Illustrative example.

The interests of users and operators are represented by functions Z_1 and Z_2, respectively. They are defined in terms of the component R of a solution $S=(R,F)$, since we only consider routes. The former is defined as

$$Z_1(R)=\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}\Delta_{ij}(R), \tag{1}$$

where $\Delta_{ij}(R)=t_{ij}(R)/t_{ij}^{*}$ represents (for passengers travelling from i to j) a deviation ratio of the minimum in-vehicle travel time using routes of R from the minimum possible value (independent of any set of routes). According to this, $t_{ij}(R)$ is calculated by using an all-or-nothing assignment approach [18], and t_{ij}^{*} is calculated as the cost of the shortest path in G between i and j. In the remaining part of this article, when we refer to shortest paths in G and cost of a route r, it is always with respect to the values of in-vehicle travel time represented by c_e for every edge $e\in E$; in this way $cost(r)=\sum_{e\in r} c_e$.

Operators interests are represented by

$$Z_2(R)=\sum_{r_k\in R} t_k, \tag{2}$$

which is the total duration of routes in R (equivalent to the total route mileage), where $t_k=2\sum_{e\in r_k} c_e$ is the duration (round-trip time) of route r_k. Low values in both functions (1) and (2) are considered in this work as a desirable property for a set of routes R which is intended to be part of a good solution $S=(R,F)$ for the TNDP.

For a given set of routes R, $D_0(R)\in[0,1]$ is the proportion of the total demand $D_{tot}=\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}$ covered by routes in R directly (without transfers). Analogous, $D_{01}(R)$ is the proportion of D_{tot} covered by routes in R directly or indirectly (one transfer, at most). D_0^{min} and D_{01}^{min} are constant values, which constrain $D_0(R)$ and $D_{01}(R)$, respectively, as

$$D_0(R)\geqslant D_0^{min}, \tag{3}$$

$$D_{01}(R)\geqslant D_{01}^{min}. \tag{4}$$

We denominate constraints (3) and (4) as *demand covering constraints*. Table 1 shows for the illustrative case of Fig. 1 and for three different sets of routes R, the corresponding values of $D_0(R)$ and $D_{01}(R)$ and results of checking the fulfilment of demand covering constraints when $D_0^{min}=0.75$ and $D_{01}^{min}=1.00$.

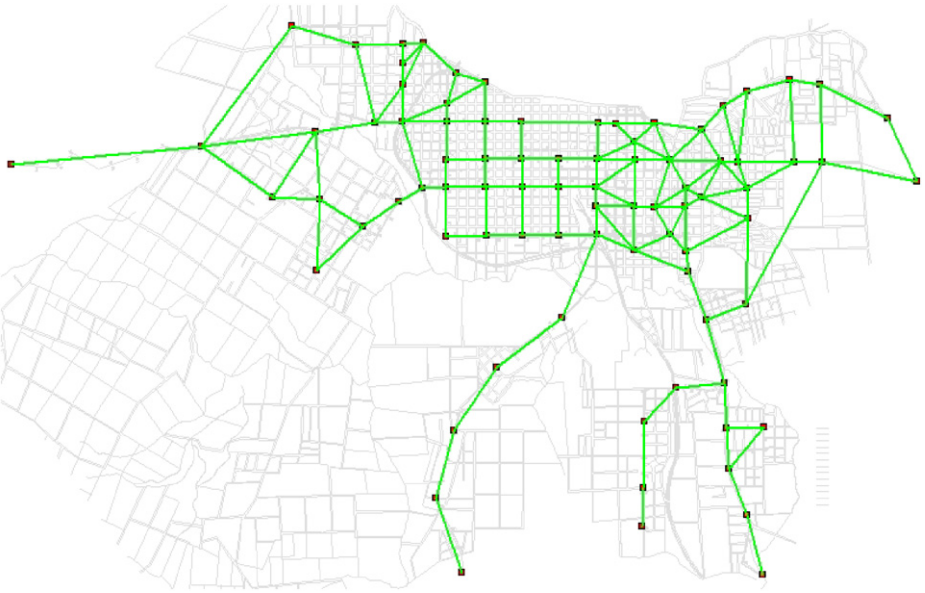

Fig. 2. Graph G for the test case of Rivera.

Some comments about the model and its application. Vertices of G represent zones of the city, created according to a given aggregation criterion; edges are logical connections between vertices, whose travel time must be estimated. Routes are sequences of zones. When transferring the results of the algorithm to the reality, forward and backward directions of a given route may be composed by different real streets, therefore their durations may differ slightly. Observe that an appropriate level of aggregation must be used to define the set of vertices N (zones), in order to construct a model whose results (routes R) are useful. Fig. 2 shows a graphical representation of the graph G corresponding to the test case used on this work.

3. TNDP and route construction

The TNDP is a hard to solve optimization problem. Its exact resolution has several difficulties [3,13], namely, high combinatorial complexity [5], the requirement of an assignment submodel [2] and a multi-objective nature [15].

It has been treated almost exclusively with approximative methods. Its combinatorial complexity prohibits the exhaustive enumeration of all feasible solutions. Mathematical programming formulations face the difficulty of modelling the assignment component; existing work using this approach has made simplifications to it [10,19].

Existing approximative methods for the TNDP can be classified in two groups [20,21]:

1. Heuristics. Classical approximative methods, either constructive or local improvement procedures, as well as combinations of them [3,14,16]. Another kind of heuristics for the TNDP consists of selecting the best possible set of routes from a previously generated pool of candidates [5].
2. Metaheuristics. Modern approximative methods that implement efficient mechanisms to explore the search space. The application of metaheuristics to the TNDP has been concentrated in using Genetic Algorithms [6,13,17,22,23], but some works also explore the use of Tabu Search [4], GRASP [15] and simulated annealing [24].

When solving the TNDP with some approximative methods, routes have to be explicitly designed by using a route construction algorithm. Several routes must be generated, which are then grouped to form a set of routes that fulfils the constraints of the problem. An important type of these constraints are the *demand covering constraints*.

A few works exist in the literature about heuristic algorithms to construct a set of routes for the TNDP, while ensuring demand covering feasibility. Baaj and Mahmassani [14] propose a greedy constructive algorithm that generates a set of routes from scratch. Israeli and Ceder [5] use a non-linear set covering formulation to select a subset of routes from a previously generated pool of candidate routes. Fan and Machemehl [4] use an optimization model that includes the uncovered demand in the objective function. The algorithms of Ngamchai and Lovell [6] and Pattnaik et al. [22] do not guarantee demand covering at the end of the execution.

The RGA of Baaj and Mahmassani [14] is the only constructive algorithm that generates from scratch a set of routes, ensuring the fulfilment of demand covering constraints; it also takes care of the interests of users and operators in the produced results. RGA also allows to specify a predetermined set of routes as an initial partial solution. It proceeds by iteratively adding routes to the solution under construction. Routes are generated by using the shortest path in G between vertices with high demand. Additional vertices are then inserted in these routes (expansion of routes) according to a prespecified criterion. The algorithm ends when the set of routes under construction fulfils demand covering constraints (3) and (4). Fig. 3 shows a pseudo-code of RGA.

A key step in RGA is the expansion of routes (procedure **ExpandRoute**), where the algorithm takes advantage of a previously existing route to cover the demand between vertices which are close to it and vertices which are already included into the route. Thus, the expansion of a route considers the insertion of vertices on it, taken from a set of candidate vertices. A feasible candidate is a vertex v which is at distance 1 (measured in number of edges) from the route r and which fulfils the following constraints (as explained by Baaj and Mahmassani in [14]):

1. It does not already belong to r.
2. It still has a high percentage of its total originating demand left uncovered after previous insertions in other routes.
3. The resulting route (after insertion of v in r) does not become circuitous.

```
procedure RGA(in D_0^min, D_01^min, out R);
  R ← ∅; D_0(R) ← 0; D_01(R) ← 0;
  l ← List of pairs of vertices (i, j) of G with d_ij ≠ 0;
  while D_0(R) < D_0^min or D_01(R) < D_01^min do
    (u, v) ← Select (i, j) with maximum d_ij in l;
    r ← Create a route with the shortest path between u and v in G;
    ExpandRoute(r);
    R ← R ∪ {r};
    Delete from l pairs of vertices whose demand is covered directly by r;
    Update D_0(R) and D_01(R);
  end while;
  return R;
end RGA;
```

Fig. 3. RGA, general structure.

4. The ratio of the contributed incremental demand covered to the insertion cost (in-vehicle travel time) exceeds a minimum value.
5. The required frequency of service on the resulting route does not exceed a maximum operationally implementable value.
6. The round-trip time of the resulting route does not exceed a maximum allowable value.

A route is expanded until the set of feasible candidates to be inserted is empty. Other elements of RGA (the initial number of routes, the order of expansion of routes and the use of *k*-shortest paths) were left aside on this simplified description; the essence of the algorithm is given by the strategy of the expansion of routes.

The computational experiments performed with RGA in [14] show that when demand covering requirements are increased, values of number of routes and total route mileage are significantly increased. For example, a change in D_{01}^{min} from 0.90 to 1.00 causes an increase of about 100% in the number of routes and 60% in the total route mileage. Despite the fact that the algorithm can produce good solutions from the users viewpoint, if these solutions have a high cost for the operators, they could result in a system that either is very expensive for the users (in terms of fares) or it may need a great amount of subsidies to cover the operation costs.

This issue motivated the research concerning this article, where a new strategy of insertion of vertices on existing routes is proposed, inspired in the general structure of the RGA.

4. Pair Insertion Algorithm

The Pair Insertion Algorithm (PIA) is based on the observation that the expansion of routes is a key component in the overall design of RGA, and therefore it determines the quality of the solutions produced. Since only the insertion of individual vertices on existing routes is considered by RGA, the inter-zonal nature of the demand is not taken into account. This inter-zonal nature, which is given by the origin–destination matrix, is addressed on this work by considering the insertion of *pairs of vertices* on existing routes, therefore, covering directly the demand associated to them. In this way, vertices which are at distances higher than 1 from the route will be potentially inserted on it.

The basic principle of PIA is to connect pairs of vertices with high values of demand. The connection is made either by creating a new route based in the shortest path in *G* between those vertices or by inserting both vertices on an existing route. Fig. 4 outlines the main structure of the algorithm. It starts with an empty set of routes *R*, and iteratively seeks to cover the demand given by origin–destination

```
procedure PIA(in D_0^min, D_01^min, in ρ_max, t_max, out R);
  R ← ∅; D_0(R) ← 0; D_01(R) ← 0;
  l ← List of pairs of vertices (i, j) of G with d_ij ≠ 0;
  while D_0(R) < D_0^min or D_01(R) < D_01^min do
    (u, v) ← Select (i, j) with maximum d_ij in l;
    r ← Create a route with the shortest path between u and v in G;
    r' ← Create a route by inserting u and v in the most suitable
         positions in the most convenient route r'' in R, by calling
         Candidate(u, v, R, r');
    if cost(r) < cost(r') − cost(r'') then
      R ← R ∪ {r};
      Delete from l pairs of vertices whose demand is covered directly by r;
    else
      R ← R ∪ {r'} − {r''};
      Delete from l pairs of vertices whose demand is covered directly by r';
    end if;
    Update D_0(R) and D_01(R);
  end while;
  Filter routes in R;
  return R;
end PIA;
```

Fig. 4. PIA, general structure.

matrix *D*. A list *l* of pairs of vertices whose demand is still not covered directly is maintained. At each iteration step, the pair of vertices (u, v) with the highest demand d_{uv} in *l* is selected and that demand is covered according to one of the two following possibilities:

1. Creating a new route, using the shortest path between u and v in *G*.
2. Inserting vertices u and v in suitable positions of a convenient route of *R*. It evaluates the cost of insertion of both u and v between all pair of consecutive vertices in routes of *R*. The most convenient route and the most suitable positions for insertion of vertices u and v on it are those which minimize the cost increase in the solution under construction.

The lowest cost increase due to insertion of vertices u and v in a route of *R* according to case 2 is compared with the cost of the shortest path between u and v according to case 1; the best (less costly) case is selected and the algorithm proceeds. It ends when constraints of demand covering imposed by parameters D_0^{min} and D_{01}^{min} are fulfilled. Observe that the structure of the main loop of PIA is the same as that one of RGA. The difference is that where RGA always create a new route to cover the demand associated to the first element of *l*, PIA evaluates if that demand can be covered by expanding an existing route, thus trying not to add an additional route.

Fig. 5 explains the computation of the most convenient candidate route built by insertion of a pair of vertices; we discriminate the cases when no vertex belongs to the route and when one vertex belongs to the route. A route r with $|r|$ vertices has $|r|+1$ possible positions for insertion of a single vertex; 1 denotes the position before the first vertex of the route and $|r|+1$ denotes the position after the last vertex. When insertion of vertex v is performed between two consecutive vertices v_i and v_{i+1} in r, it is connected to them by using the shortest paths in *G* between v_i and v, and between v and v_{i+1}, respectively. If the resulting route after the insertion contains a loop, it is discarded as candidate.

```
procedure Candidate(in u, v, in R, out r');
  r' ← ∅; cost(r') ← ∞;
  for each r ∈ R do
    if u ∈ r then
      for each p ∈ [1..|r| + 1] do
        rAux ← Insert v in position p in r;
        if cost(rAux) < cost(r') then r' ← rAux; end if;
      end for;
    else if v ∈ r then
      for each p ∈ [1..|r| + 1] do
        rAux ← Insert u in position p in r;
        if cost(rAux) < cost(r') then r' ← rAux; end if;
      end for;
    else
      for each p1, p2 ∈ [1..|r| + 1] do
        r' ← Insert u and v in positions p1 and p2 respectively in r;
        if cost(rAux) < cost(r') then r' ← rAux; end if;
      end for;
    end if;
  end for;
  return r';
end Candidate;
```

Fig. 5. Computation of the most convenient route.

In procedure **Candidate**, when trying to insert vertices into routes, two constraints are imposed: maximum duration (round-trip time) and maximum circuity factor, represented by parameters t_{max} and ρ_{max}, respectively. The circuity factor ρ of a route r with extreme vertices u and v, is defined in [14] as the ratio between the in-vehicle travel time between u and v using r, and the cost of the shortest path between u and v in G (independent of any route), i.e. $\rho(r) = cost(r)/t^*_{uv}$. These constraints are imposed to limit the growth of the route, when several vertices have been inserted on it. Other constraints such as route capacity constraints can be easily incorporated at the model and implemented in the proposed algorithm.

Since the insertion of pairs of vertices on a route r may imply the insertion of a whole path P on r, it may be possible that there is an already existing route $r' \in R$ included on P. For this reason, there can be at the end of the main loop of PIA, routes that are completely included in other ones. PIA has a filter procedure that eliminates these included routes, because our objective is to minimize the number of routes and total route duration.

4.1. Rationale of the algorithm

The design of PIA is based on the following line of thought.

Probably the main objective in transit network design is to cover the demand in the best possible way, given a restriction in the available resources (however, other objectives can be stated when designing a transit network [25]). Since the demand has an inter-zonal nature, expressed in the form of an origin–destination matrix, it has to be covered for *pairs of vertices*. For this reason, we consider the insertion of pairs of vertices on routes as a key idea of our algorithm.

The ideal solution from the users viewpoint is one that covers every non-null element of demand d_{ij} with a route that includes the shortest path between i and j in G. It is desirable that this condition can be fulfilled for a high number of elements in D. Almost every work related to the TNDP agrees that there must exist a route including the shortest path in G between pairs of vertices with high demand. Based on these observations, the algorithm proposed considers the elements of D in decreasing order of demand and generates routes using the shortest path between them.

Since the ideal solution from the users viewpoint is not convenient from the operators viewpoint, the number of routes has to be restricted. For this reason, when we consider the next pair of vertices (u, v) whose demand d_{uv} has to be covered, we test the possibility of including these two vertices on an existing route in the solution under construction. Doing this, we take advantage of the existence of a route, and we modify it so it can serve the demand associated to a pair of vertices which are close to it. Since it is not desirable to extend so much an existing route by inserting vertices on it (because travel time will be increased for demand already served by the route) we impose constraints of maximum route duration and circuity factor to candidate routes resulting from the insertion.

Even if a pair of vertices (u, v) can be inserted on an existing route, we compare the cost of extending this route with the cost of the shortest path in G between u and v. We always try to minimize the increase of the overall duration of routes in the system. This criterion represents somehow the interest of the operators, since the sum of the durations of all routes in the system is directly proportional to the fleet size (which is an important component in the costs for the operators).

4.2. Implementation variants

The PIA constructive algorithm as presented in Fig. 4 admits some variants on its implementation. The following is a list of modifications that can be considered with different purposes:

1. Concerning the decision of the next element of list l to be considered to cover its corresponding demand, the original strategy (as presented in Fig. 4) is deterministic, i.e. always the pair of vertices with maximum demand is selected. One possible alternative is to consider a sublist l' of elements with highest demand in l and then select one element from it, in a systematic way. A particular case of this strategy is the greedy randomized construction [26], where a random selection of an element of l' is performed, with a given distribution of probabilities.
2. The shortest path in G is always used when PIA creates a new route to be added to the solution under construction. Baaj and Mahmassani [14] observe that by using alternative paths, there are chances to cover more demand with a slight increase in the route length; k-shortest paths [27] have been used in [4,7,14] in the context of the resolution of the TNDP and this feature can be easily incorporated to the original version of PIA.
3. The proposed algorithm starts with an empty set of routes, but it can be fed with an initial set of given routes R; if so, $D_0(R)$, $D_{01}(R)$ and list l have to be appropriately initialized with information given by R. This characteristic of incremental construction of the algorithm allows for example to consider a set of initial fixed routes given by the decision maker. Another use of the algorithm can be made when there is a need to complete an unfeasible solution with respect to demand covering constraints, since some algorithms (for example some local search algorithms [20]) manipulate unfeasible solutions at intermediate steps, and may end with no feasible solution.
4. Though PIA uses an undirected graph as underlying model, it can be adapted to work with a directed graph as input network. This constitutes a more realistic modelling since routes may not have the same duration in both ways. Both the structure of routes and some parts of the algorithm (specially those that check and

update demand covering) must be modified in order to model this characteristic.
5. Demand covering with more than one transfer can be considered in the model and implemented in the algorithm. Though it adds more complex subroutines to the algorithm, it may decrease its overall execution time, since under this scenario there is no need to create or modify routes when a given percentage of the whole demand is already covered with a given number of transfers (therefore the algorithm will stop earlier).

5. Experimental study

Both PIA and RGA were tested using a real case related to the city of Rivera, Uruguay, which is described in Appendix A. The real case gives a reference for comparison, specially in terms of the required number of routes as a function of demand covering requirements. Experiments were designed to produce results in the following two categories:

1. Comparison between PIA and RGA. This set of experiments was made to compare the behavior of both algorithms, in terms of their sensibility under changes in demand covering requirements, and in terms of values of Z_1 and Z_2 of the solutions produced, as well as execution times.
2. Analysis of diversity. This second set of experiments was designed to investigate the ability of PIA to produce different solutions, by changing values of some of its parameters. This is made in order to evaluate the usefulness of PIA as a subroutine of a more structured algorithm, which may require a set of diverse solutions.

Implementations were made in C++; programs were run on a PC Pentium 4, with a 1.6 GHz processor and 512 MB of RAM. Values of Z_1 are calculated in terms of the elements of the origin–destination matrix, which are expressed in trips per minute; Z_2 is expressed in minutes.

5.1. Comparison between PIA and RGA

The implementations of both algorithms use the same data structures and subroutines, thus trying to make the comparison as fair as possible. Both algorithms rely on the availability of pre-computed shortest paths (and their cost) between all pairs of vertices in G. For the implementation of RGA, the criterion of maximum demand per minimum time insertion was implemented [14]. According to this, the vertex which maximizes d_v/t_v is selected (from the set of feasible candidates) for insertion on route r, where d_v is the demand of elements in l between vertex v and vertices on r, and t_v is the cost increase of the route resulting after the insertion of v. The implemented procedure **ExpandRoute** applies three out of the sixf constraints of the original version of RGA, namely, loop avoiding, maximum route duration and maximum circuity factor.

5.1.1. Sensibility for different levels of demand covering

As mentioned in [14], increasing the imposed levels of demand covering causes an increase in the number of routes and total route mileage in the solutions produced by RGA. This tendency was also expected in PIA, but to a lower extent.

In this experiment we run both algorithms for different combinations of D_0^{min} and D_{01}^{min}, and we investigate its effect in values of deviation from the shortest path Z_1, total route duration Z_2 and number of routes $|R|$.

Table 2 shows that parameter D_{01}^{min} is the main factor that rules the increase of Z_2 and $|R|$ for both algorithms; values of Z_1 do not necessarily increase because they are summations of ratios that can decrease, in particular for higher levels of D_0^{min}. While increasing D_0^{min} for a fixed value of D_{01}^{min} does not impact so much (values along the same column), increasing D_{01}^{min} has a significant impact (values along the same row).

Table 2
Sensibility under changes in levels of required demand covering

D_{01}^{min}	0.50	0.75	0.95	0.99	1.00
(a) PIA					
D_0^{min}	Z_1				
0.50	11.35	12.61	15.90	16.20	16.09
0.75	–	14.41	15.90	16.20	16.09
0.95	–	–	16.24	16.20	16.09
0.99	–	–	–	16.14	16.09
1.00	–	–	–	–	16.09
D_0^{min}	Z_2				
0.50	207.48	306.63	713.24	831.40	1093.04
0.75	–	359.68	713.24	831.40	1093.04
0.95	–	–	763.09	831.40	1093.04
0.99	–	–	–	1042.43	1093.04
1.00	–	–	–	–	1117.98
D_0^{min}	$\|R\|$				
0.50	3	4	10	11	17
0.75	–	5	10	11	17
0.95	–	–	10	11	17
0.99	–	–	–	15	17
1.00	–	–	–	–	18
(b) RGA					
D_0^{min}	Z_1				
0.50	12.20	13.46	16.17	16.26	15.97
0.75	–	15.16	16.17	16.26	15.97
0.95	–	–	16.01	16.26	15.97
0.99	–	–	–	16.17	15.97
1.00	–	–	–	–	15.97
D_0^{min}	Z_2				
0.50	280.14	368.47	1106.24	1602.67	2319.86
0.75	–	717.63	1106.24	1602.67	2319.86
0.95	–	–	1486.28	1602.67	2319.86
0.99	–	–	–	1991.08	2319.86
1.00	–	–	–	–	2402.13
D_0^{min}	$\|R\|$				
0.50	3	4	12	18	26
0.75	–	8	12	18	26
0.95	–	–	16	18	26
0.99	–	–	–	22	26
1.00	–	–	–	–	27

When comparing PIA with RGA, we can observe that while values of Z_1 vary in a similar way, values of Z_2 and $|R|$ increase to a high extent for RGA. For example, for a fixed value of $D_0^{min} = 0.50$, an increase of D_{01}^{min} from 0.50 to 1.00 causes an increase in Z_2 of 427% for PIA and 728% for RGA; the increase in $|R|$ is 467% for PIA and 767% for RGA. This shows that when we increase the amount of demand covered to that level (which is desirable from the users viewpoint), the cost increase for the operators (represented by number of routes and total route duration) when using RGA is around 65% higher than the cost increase using PIA. The cost increase for the users (represented by travel time) is almost the same for both algorithms.

5.1.2. Objective values of the produced solutions

In this experiment we compare the results produced by both algorithms in terms of functions Z_1 and Z_2, which are intended to be minimized when solving the TNDP. We also compare values of number of routes and execution time.

In order to compare, we implemented the greedy randomized construction variant explained in Section 4.2 (implementation variant number 1), in both PIA and RGA. The list l' is constructed by selecting the $\alpha|l|$ elements with highest demand of l, where $\alpha \in [0,1]$ is a parameter. The random selection of an element (u,v) from l' is made by using a biased probability distribution [28], where

Table 3
Results of 10 independent executions

Execution	Z_1	Z_2	$\|R\|$	*over*	T
PIA					
1	16.20	1179.83	19	660.45	15.20
2	16.32	1198.66	19	665.79	15.16
3	16.75	1098.18	19	584.33	13.39
4	16.09	1212.77	19	662.04	13.77
5	15.90	1190.65	19	647.47	13.11
6	16.46	1225.81	16	697.35	15.23
7	16.14	1074.17	19	560.99	11.56
8	16.65	1128.35	15	645.15	12.23
9	15.65	1218.93	20	673.14	15.42
10	17.04	1081.83	18	578.34	11.25
Average	16.32	1160.92	18	637.51	13.63
RGA					
1	16.30	2114.59	23	1503.43	0.48
2	16.10	1851.23	20	1250.00	0.39
3	15.89	1960.08	21	1338.90	0.38
4	16.49	1997.97	22	1392.33	0.52
5	16.16	1818.06	20	1210.66	0.39
6	16.13	2307.79	24	1681.13	0.44
7	16.05	2040.36	22	1395.03	0.34
8	16.10	1966.98	21	1358.40	0.39
9	15.71	2144.98	23	1527.34	0.41
10	15.92	2036.59	21	1403.43	0.33
Average	16.09	2023.86	22	1406.07	0.41

$bias(u,v) = d_{uv}$, and the corresponding probability is

$$\text{Prob}(u,v) = \frac{bias(u,v)}{\sum_{(i,j)\in l'} bias(i,j)}. \quad (5)$$

Demand covering related parameters were set as $D_0^{min} = D_{01}^{min} = 1.00$; we impose this strong requirement of demand covering for two reasons: (i) to compare results under extreme conditions of required demand covering, (ii) to obtain solutions which are comparable to the public transportation system of Rivera, where all the demand is served without transfers. Values of parameters over routes were $\rho_{\max} = 1.5$ and $t_{\max} = 120$ (minutes). Parameter α for the greedy randomized construction was set to 0.2.

Table 3 shows for each set of routes R produced by each algorithm in 10 independent executions, values of functions Z_1 and Z_2, number of routes $|R|$ and execution time T (in seconds). We also calculate a measure of the overlapping of routes in a solution, defined as

$$over(R) = \sum_{e\in E} c_e \max\{0, R_e - 1\},$$

where R_e is the number of routes in R which use edge e.

We can observe that while both algorithms produce results with similar values in Z_1, when comparing averages, RGA produces higher values than PIA in number of routes (22% higher) and total route duration (74% higher). This difference is accompanied by an overlapping of routes which is more than 100% higher for RGA, suggesting that there are many routes serving the same demand in solutions produced by this algorithm. On the other hand, execution time is highly favorable for RGA, which is in average 27 times lower than the execution time for PIA. This significant difference can be explained by (i) the quadratic computational complexity of the **Candidate** subroutine in PIA and (ii) the computational burden resulting from the handling of paths in the insertion of pairs of vertices in PIA.

Table 4 shows results from 1000 independent executions of both algorithms, summarized in minimum, average and maximum values. We can observe that averages remain approximately the same as those in Table 3. Minima in Z_1 are very similar between PIA and RGA, and relatively close to its lower (theoretical) possible value, that is attained when $\Delta_{ij}(R) = 1$ for every (i,j), in this case, 13.94. One remarkable fact is that PIA has produced a solution with 12 routes, less than the number of routes in the real system of Rivera, which is 13; on the other hand, the smallest set of routes produced by RGA has 15 elements.

Table 4
Summarized results of 1000 independent executions

	Z_1	Z_2	$\|R\|$	*over*	T
PIA					
Minimum	15.26	903.60	12	401.74	6.91
Average	16.30	1146.08	18	615.23	12.82
Maximum	17.67	1441.05	24	860.55	21.36
RGA					
Minimum	15.49	1429.40	15	880.49	0.27
Average	16.18	1998.44	22	1377.50	0.47
Maximum	17.47	2610.48	28	1950.24	1.13

5.2. Analysis of diversity

Since PIA can be used as a subroutine of another algorithm (for example, a metaheuristic working with populations or with a multi-start strategy), it can be useful to obtain different (diverse) solutions; this type of diversity is considered with respect to decision variables. On the other hand, given the multi-objective nature of the TNDP, it may be desirable to obtain solutions with different trade-off levels between the conflicting objectives of users and operators; this type of diversity is considered with respect to objective space.

In this section we study the diversity in decision space by computing a measure of similarity among solutions, which takes into account the structure of their routes (in terms of the sequences of vertices). Diversity in objective space is grasped graphically, by plotting the results in a two-dimensional space defined by the objectives.

5.2.1. Diversity in decision space

When using PIA as a subroutine of an approximative algorithm for the TNDP, a possible requirement imposed to it, is the ability to generate a diverse set of solutions (sets of routes). Some metaheuristics require a constructive algorithm capable of producing several solutions, being each one different from the other ones (see for example GRASP, Genetic Algorithms and scatter search in [29]). This difference (or diversity) is considered with respect to decision variables, in this case the structure of routes. Several existing metaheuristic based algorithms for the TNDP have this requirement on the constructive algorithm [6,15,17,23].

In this work, we propose a diversity measure *diver* over a set of solutions $\Re = \{R_1, \ldots, R_m\}$, which is defined as

$$diver(\Re) = 1 - \frac{\sum_{(R_i,R_j)\in\Re, i<j} sim(R_i,R_j)}{|\Re|(|\Re|-1)/2},$$

where *sim* is a measure of similarity between two sets of routes calculated as

$$sim(R_i,R_j) = \frac{\sum_{r\in R_i} sim(r,R_j) + \sum_{r\in R_j} sim(r,R_i)}{|R_i| + |R_j|}.$$

Similarity between sets of routes is calculated in terms of the similarity of a route with respect to a set of routes, it is defined as

$$sim(r,R) = \max_{r_i\in R}\{sim(r,r_i)\},$$

where $sim(r,r_i)$ is the proportion of the cost of the arcs in r which are already included in r_i with respect to its cost, i.e. $sim(r,r_i) = \sum_{e\in r\wedge e\in r_i} c_e / \sum_{e\in r} c_e$. Thus, the diversity measure is based on a similarity measure that takes into account the structure of routes in

detail. Note that $diver(\mathfrak{R}) \in [0,1]$, where 0 states that all solutions in $\mathfrak{R}$ are identical (meaning that $\mathfrak{R}$ is not a diverse set) while 1 indicates the opposite situation.

In order to obtain diverse solutions in decision space by using PIA, we change its α parameter and we use the implementation variant number 2 (explained in Section 4.2), where the new route to be considered for addition to the solution under construction is generated by using one of the k-shortest paths (k is a parameter); the selection is made randomly with a uniform probability distribution in the discrete interval $[1 \ldots k]$. The k-shortest paths are generated in a previous step using the Yen algorithm [27].

In this experiment we vary α and k, and for each combination of these parameters we perform 10 independent executions of PIA, thus generating sets of 10 solutions over which diversity values are computed and shown in Table 5. Observe that *diver* is not so sensible to parameter α; any value of $\alpha > 0$ will cause PIA to produce a diverse set of solutions (since $\alpha = 0$ represents the deterministic version of the algorithm). Parameter k shows a higher influence in diversity, with a monotonically increasing tendency; diversity increases even for high values of k; however, we must take into account that respective solutions are not necessarily good in terms of Z_1 and Z_2.

We observe that diversity is always far from 1.00. This fact suggests that still it may be possible to improve the diversity of the results, by introducing other mechanisms in the algorithm; however, we must consider that the fulfilment of demand covering constraints (specially for higher values of D_0^{min}) keeps bounded the maximum reachable diversity.

Table 5
Diversity in decision space

α/k	1	2	5	10	20	50
Deterministic	0.00	0.00	0.00	0.00	0.00	0.00
0.2	0.31	0.37	0.39	0.44	0.44	0.45
0.4	0.30	0.37	0.42	0.43	0.44	0.46
0.6	0.31	0.38	0.41	0.42	0.46	0.47
0.8	0.32	0.37	0.41	0.43	0.45	0.47
1.0	0.34	0.36	0.41	0.44	0.44	0.46

5.2.2. Diversity in objective space

Since the TNDP is a multi-objective problem, where objectives of users and operators are conflictive [5,15], a possible use of PIA consists in producing solutions with different trade-off levels between these objectives (in our case represented by functions Z_1 and Z_2, respectively). These different solutions can be obtained simply by taking advantage of the stochastic nature of the greedy randomized version of the algorithm. However, by changing the parameter of maximum duration of routes t_{max}, a wider range of trade-off levels can be obtained. In this experiment we made 10 executions of the greedy randomized version of PIA with the same parameter configuration as that used in Section 5.1.2; at each execution the value of t_{max} is randomly selected (with uniform probability) from the real interval $I = [40, 120]$ (minutes).

Several measures have been proposed to evaluate the diversity in objective space of a set of solutions for a multi-objective problem [30]. However, all of them must be applied to a non-dominated set of solutions. Because we are interested in getting a sound of the diversity in objective space of all obtained solutions, we use a graphical method consisting in plotting the solutions on the two-dimensional space defined by Z_1 and Z_2.

Fig. 6 shows the 10 solutions obtained by the greedy randomized version of PIA (set $\mathfrak{R}_1$, already shown in Table 3) as well as the 10 solutions obtained by the same version of the algorithm also including the variation of t_{max} as explained before (set $\mathfrak{R}_2$). We can see that there is a region of the objective space (corresponding to low values of Z_1 and high ones of Z_2) that is covered by $\mathfrak{R}_2$ and not covered by $\mathfrak{R}_1$. This shows that by changing the parameter t_{max}, the algorithm is able to produce solutions with very low cost for the users (and very high for the operators), which cannot be obtained by the greedy randomized version. Despite these solutions may represent an extreme trade-off level between the conflicting objectives (and therefore they may not be practicable in the real system), they may be useful in the context of an algorithm that is designed to produce a Pareto front as solution of the multi-objective TNDP. The highest value reached by PIA for Z_2 in 1000 independent executions using only the greedy randomized version is 1441.05 (see Table 4), which is significantly lower than the maximum reached in $\mathfrak{R}_2$ (which is greater than 2100.00, with a low value in Z_1, suggesting that it is close to the optimal Pareto front [30]). Thus, the variation of parameter

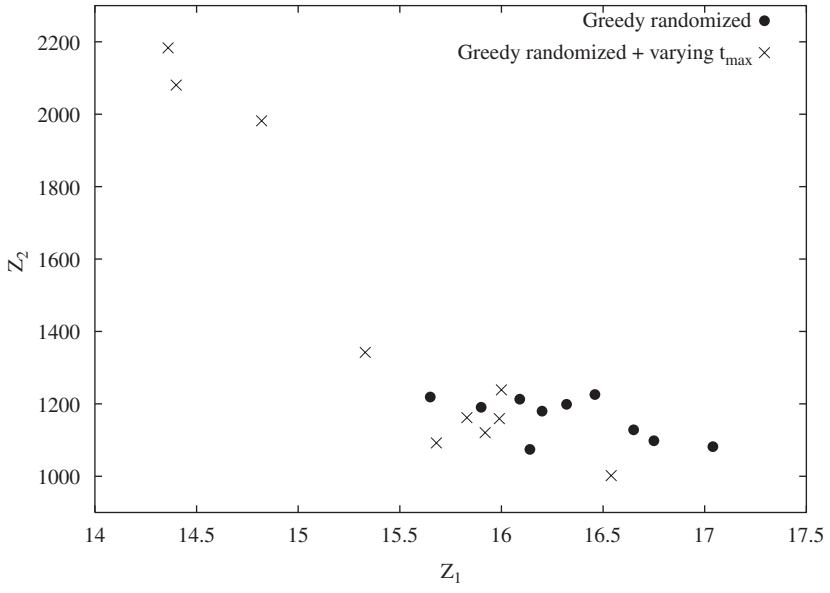

Fig. 6. Diversity in objective space.

t_{max} allows the algorithm to produce solutions in a wide range of trade-off levels between functions Z_1 and Z_2. It is interesting to note that diversity in objective space does not imply necessarily diversity in decision space for solutions in $\mathfrak{R}_2$; we observe that $diver(\mathfrak{R}_2) = 0.27$, which is lower than the minimum diversity already shown in Table 5.

6. Conclusions and future work

The PIA proposed modifies the RGA of Baaj and Mahmassani by using a new strategy of insertion of pairs of vertices, instead of the original expansion of routes by inserting single vertices.

When compared with RGA, PIA produces solutions with similar values of in-vehicle travel time, and significantly better in terms of number of routes and total route duration. On the other hand, execution time is significantly higher; this fact is explained mainly by the quadratic computational complexity of the subroutine of insertion of pairs of vertices, which is intensively used by PIA. Further investigation looking at possible strategies to reduce the complexity of the algorithm are required to improve execution times. However, it is worth mentioning that execution time is not the main concern in the context of strategic planning, where TNDP takes place. Though execution times are highly favorable to RGA, the cost of the solutions produced by PIA from the operators viewpoint are much lower (while the cost for the users is almost the same). In terms of the real transit system, this reduction of operation costs may imply a reduction on fares and/or subsidies, while maintaining the revenues of the operators and the level of service for the users.

The algorithm has shown to be flexible to be used as a subroutine in other algorithms such as metaheuristics; it is capable to produce diverse solutions in both decision and objective spaces. It is used in [15] to generate solutions with different trade-off levels between the objectives of users and operators, in the context of a metaheuristic based algorithm; it is also used in [31] to complete unfeasible solutions with respect to demand covering constraints.

A real medium-small size test case was used in this work. The algorithm has shown to be capable of producing solutions which are comparable (in terms of number of routes) to real solutions. For cases of much bigger sizes, the applicability of the algorithm has to be tested, specially with regard to execution time. Demand covering plays an important role on those cases: high imposed levels of demand covering without transfers may impact on the performance of the algorithm, by increasing its execution time. Also the possibility of covering the demand with more than one transfer has to be considered in both model and algorithm for big cases, since the budget of the operators will not allow direct connections as users might wish; this impacts on the complexity of the implementation of the algorithm but not necessarily it degrades its computational performance (as explained in Section 4.2).

It would be also interesting to study how close are the solutions produced by PIA to Pareto optimal solutions according to objectives Z_1 and Z_2.

Acknowledgments

We would like to thank Héctor Cancela for his comments on previous versions of this article, to the PDT-DINACYT for the financial support through the project S/C/OP/48/02 and to PEDECIBA Computer Science. This work is partially supported by Programme ALFA II-0457-FA-FCD-FI-FC. We thank anonymous referee for his comments, which helped to improve this paper.

Appendix A. Real test case

A real case was used for testing the algorithms on this work. It is related to a medium-small size city of 65,000 inhabitants in Uruguay, the city of Rivera. By using a real case, we have a graph G and an origin–destination matrix D of real characteristics, which is not common in fictitious cases. This provides a basis for comparison and (in some sense) validation of the algorithm.

When data gathering for this work was accomplished on August 2004, the public transportation system of Rivera operated 13 bus lines, with an average route length of 13.6 km, and an imposed duration (round-trip time) of 60 min each. The inter-zonal demand has a radial pattern, being the city center the main attractor of trips. In a regular mid-week day, an average of 13,360 trips are performed using public transportation in Rivera.

Data to construct a real test case for the TNDP are not easy to gather. Graph G can be generated from a representation of the street network; a Geographic Information System can be helpful for this processing. However, to construct the origin–destination matrix D, a considerable amount of information has to be compiled [18], to express the needs of public transportation between different points of the city.

The activities related to the construction of the test case of Rivera were part of a related project. The web-site http://www.fing.edu.uy/~mauttone/tndp contains a description of the project as well as the data of the case. Details of the procedures are given in [32]. In this section we give a brief overview of its three main components: zone system, graph and demand.

Zone system: The city was divided into zones. Each zone comprises approximately 4×4 blocks of 100 m each. This size is intended to apply a criterion of geographical accessibility of the people to the public transportation system. We consider that 400 m represent a maximum reasonable walk distance to access to a bus line passing through the zone. The demand produced (attracted) by a given zone is considered as covered when a line passes by any place in the street network inside the zone and inside the destination (origin) zone.

Graph: The graph G is an abstraction of the real street network. We do not use the real street network because final decisions about the routing of buses have to take into account the location of special places, such as schools or hospitals, and other considerations which are difficult to include in the graph model. There is a vertex in G representing each zone of the zone system; it is located over the intersection of streets that is nearest to the barycenter of the corresponding zone. An edge exists between two vertices in G if their corresponding zones are adjacent; its value of in-vehicle travel time is calculated from the distance of the shortest path in the street network between its extreme vertices and a bus commercial speed estimated in 13.6 km per hour. The resulting graph of Rivera, constructed according to the explained procedure, has 84 vertices and 143 edges (Fig. 2).

Demand: Demand data were collected by means of a survey made on-board of lines operating on the public transportation system of Rivera. The methodology of the survey is based on the one proposed by Stopher et al. [33]. A sample of 13 out of the 23 bus runs performed per hour by the lines of the system was selected; a run is a trip of a bus going over the route at a given time. For each one of these runs, origin and destination stops were recorded for every person using the bus. Data were collected on a time period of 12 h. For each line, origin–destination counts were expanded according to the sample size. Line origin–destination matrices from the 13 lines were consolidated in a single system origin–destination matrix. It contains average values in the 12 h time horizon. This matrix is intended to represent the need for public transportation across all the operation period of the system. Given the size of the considered time horizon, significant peaks of demand for some pairs of vertices in some time periods may be smoothed in the average; for this reason this matrix is not suitable for estimation of passengers flow, and therefore not suitable for the calculation of required frequencies on the bus lines. A final step in this data processing consist in transforming the origin–destination matrix from the level of bus stops to the level of

zones. The resulting matrix constructed according to the explained procedure has 5% of non-null elements. This matrix is based on observed values which depend on the particular lines operating on the system when data were collected. However, we consider that for the city of Rivera, this matrix of observed trips is a good approximation to the matrix of desired trips, mainly due to (i) the highly captive characteristic of the users of the public transportation system of Rivera and (ii) the high spatial coverage of the city by the existing lines.

References

[1] Ceder A, Wilson N. Bus network design. Transportation Research B 1986;20(4):331–44.
[2] Desaulniers G, Hickman M. Public transit. In: Barnhart C, Laporte G, editors. Hand books in operations research and management science volume 14, Transportation. Amsterdam: North-Holland; 2007. p. 69–128.
[3] Baaj MH, Mahmassani H. An AI-based approach for transit route system planning and design. Journal of Advanced Transportation 1991;25(2):187–210.
[4] Fan W, Machemehl R. A Tabu search based heuristic method for the transit route network design problem. In: 9th international conference on computer aided scheduling of public transport, San Diego, United States, 2004.
[5] Israeli Y, Ceder A. Transit route design using scheduling and multiobjective programming techniques. In: Daduna J, Branco I, Pinto J, editors. Proceedings of the sixth international workshop on computer aided scheduling of public transport. Berlin: Springer; 1993. p. 56–75.
[6] Ngamchai S, Lovell D. Optimal time transfer in bus transit route network design using a genetic algorithm. Journal of Transportation Engineering 2003;129(5):510–21.
[7] Fan W, Machemehl R. Optimal transit route network design problem: algorithms, implementations, and numerical results. Technical Report 167244-1, University of Texas at Austin, United States; 2004.
[8] Hasselström D. Public transportation planning—a mathematical programming approach. Doctoral dissertation, University of Göteborg, Sweden; 1981.
[9] Lee YJ, Vuchic V. Transit network design with variable demand. Journal of Transportation Engineering 2005;131(1):1–10.
[10] Borndörfer R, Grötschel M, Pfetsch M. A column-generation approach to line planning in public transport. Transportation Science 2007;41(1):123–32.
[11] Shih MC, Mahmassani H, Baaj MH. Planning and design model for transit route networks with coordinated operations. Transportation Research Record 1998;1623:16–23.
[12] Zhao F, Ubaka I. Optimization of transit network to minimize transfers and optimize route directness. Journal of Public Transportation 2004;7(1):63–82.
[13] Chakroborty P. Genetic Algorithms for optimal urban transit network design. Computer-Aided Civil and Infrastructure Engineering 2003;18(3):184–200.
[14] Baaj MH, Mahmassani H. Hybrid route generation heuristic algorithm for the design of transit networks. Transportation Research C 1995;3(1):31–50.
[15] Mauttone A, Urquhart ME. A multi-objective metaheuristic approach for the transit network design problem. In: 10th international conference on computer aided scheduling of public transport, Leeds, United Kingdom, 2006.
[16] Silman L, Barzily Z, Passy U. Planning the route system for urban buses. Computers & Operations Research 1974;1(2):201–11.
[17] Tom VM, Mohan S. Transit route network design using frequency coded Genetic Algorithm. Journal of Transportation Engineering 2003;129(2):186–95.
[18] Ortúzar J de D, Willumsen L. Modelling transport. New York: Wiley; 1996.
[19] Wan Q, Lo H. A mixed integer formulation for multiple-route transit network design. Journal of Mathematical Modelling and Algorithms 2003;2(4):299–308.
[20] Glover F, Laguna M. Tabu search. Dordrecht: Kluwer Academic Publishers; 1997.
[21] Reeves C. Modern heuristic techniques for combinatorial problems. New York: McGraw-Hill; 1995.
[22] Pattnaik SB, Mohan S, Tom VM. Urban bus transit route network design using Genetic Algorithm. Journal of Transportation Engineering 1998;124(4):368–75.
[23] Krishna Rao KV, Muralidhar S, Dhingra SL. Public transport routing and scheduling using Genetic Algorithms. In: 8th international conference on computer aided scheduling of public transport, Berlin, Germany, 2000.
[24] Fan W, Machemehl R. Using a simulated annealing algorithm to solve the transit route network design problem. Journal of Transportation Engineering 2006;132(2):122–32.
[25] van Nes R. Optimal stop and line spacing for urban public transport networks. Delft: Delft University Press; 2000.
[26] Feo T, Resende M. Greedy randomized adaptive search procedures. Journal of Global Optimization 1995;6:109–33.
[27] Yen JY. Finding the *k* shortest loopless paths in a network. Management Science 1971;17:712–6.
[28] Resende M, Ribeiro C. Greedy randomized adaptive search procedures. In: Glover F, Kochenberger G, editors. Handbook of metaheuristics. Dordrecht: Kluwer Academic Publishers; 2003. p. 219–49.
[29] Glover F, Kochenberger G. Handbook of metaheuristics. Berlin: Springer; 2003.
[30] Deb K. Multi-objective optimization using evolutionary algorithms. New York: Wiley; 2001.
[31] Mauttone A, Urquhart ME. Una heurística basada en memoria para el problema del diseño de recorridos en transporte público urbano. In: XIII Congreso Latino Iberoamericano de Investigación Operativa, Montevideo, Uruguay, 2006.
[32] Mauttone A. Optimización de recorridos y frecuencias en sistemas de transporte público urbano colectivo. Master thesis in computer science, Universidad de la República, Uruguay; 2005.
[33] Stopher PR, Shillito L, Grober DT, Stopher HMA. On-board bus surveys: no questions asked. Transportation Research Record 1986;1085:50–7.