RAIRO-Oper. Res. 45 (2011) 315–338
DOI: 10.1051/ro/2011116

RAIRO Operations Research
www.rairo-ro.org

ON DESIGNING CONNECTED RAPID TRANSIT NETWORKS REDUCING THE NUMBER OF TRANSFERS

Laureano Fernando Escudero[1] and Susana Muñoz[2]

Abstract. In this paper we introduce some improvements on an approach that we described elsewhere for solving a modification of the well-known extended rapid transit network design problem. Firstly, we propose an integer programming model for selecting the stations to be constructed and the links between them, in such a way that a connected rapid transit network is obtained. Secondly, we consider a linear 0-1 programming model for determining a route of minimum length in the rapid transit network between certain pairs of locations, and present a greedy heuristic procedure which attempts to minimize an estimation of the total number of transfers that should be made by the users to arrive at their destinations. We also report several computational experiments that show that this procedure can significantly reduce the estimated total number of transfers required for the solutions obtained using our previous approach.

Keywords. Station and link location, line designing, degree of a node, transfer, greedy heuristic procedure.

Mathematics Subject Classification. 90B06, 90B80, 90C10, 90C35.

1. Introduction

Given the crucial role that transportation plays in society, the problems of developing and improving urban public transportation networks have been widely studied in the literature (see *e.g.* [3,9,11] for good surveys on the subject). These types of problems are so complex that, in order to obtain acceptable solutions in

Received October 25, 2010. Accepted December 1, 2011.

[1] Departamento de Estadística e Investigación Operativa, Universidad Rey Juan Carlos, 28933 Móstoles (Madrid), Spain. laureano.escudero@urjc.es

[2] Corresponding author. Departamento de Estadística e Investigación Operativa, Facultad de Ciencias Matemáticas, Universidad Complutense de Madrid, 28040 Madrid, Spain. smunoz@estad.ucm.es

Article published by EDP Sciences

a reasonable amount of time, it is necessary to resort to simplifications and/or heuristic methods.

The extended rapid transit network design problem was stated in [14]. Given a set of potential station locations and a set of potential links between them, this problem basically consists in selecting which stations and links to construct without exceeding budget, and determining an upper bounded number of noncircular lines from them, to maximize the total expected number of trips. A linear 0-1 programming model for solving this problem is presented in [14], where, in order to compute the total expected number of trips, it is assumed that the demand and a private transportation cost are known for each origin-destination pair of locations. It is also implicitly assumed that the users obey the well-known Wardrop's first principle (see [17]), and, consequently, they seek to minimize their expected travel costs (in [14], the travel cost is interpreted as the travel distance; another common interpretation of the travel cost is the travel time).

A two-stage approach for solving a modification of the extended rapid transit network design problem to allow the definition of circular lines is provided in [5], and it is shown that it outperforms the solving of a modification of the model given in [14] to adapt it to this new problem. In the first stage of the proposed approach, a linear integer programming model is solved for selecting the stations and links to be constructed without exceeding budget, so that the total expected number of trips on the rapid transit network is maximized (without loss of generality, it is assumed that whichever two locations are linked by one line at most). In the second stage, the line design problem is solved by assigning each selected link to exactly one line, in such a way that the number of lines that go to each selected location is as small as possible; the required computational effort is not appreciable for all the instances under consideration.

The models presented in [5, 14] have some disadvantages, namely, that they do not guarantee either a connected rapid transit network or recommended routes of minimum length for the users of each origin-destination pair of locations, and they do not take into consideration the transfers that should be made by the users to arrive at their destinations. The model proposed in [10] for designing robust rapid transit networks neither takes transfers into consideration.

Several indicative papers dealing with transfers in transit networks are [1,2,7,8, 13]. Some of the assumptions and simplifications that they consider are as follows:

In [13] it is assumed that there are no capacity constraints on the lines of the transportation network. It proposes a heuristic algorithm that starts from a feasible set of lines and iteratively searches for better solutions. The main disadvantages of this approach are that the output solution will depend to a great extent on the initial one, and that it only involves three different ways of attempting to reach better solutions.

In [1] it is assumed that the users take trips which involve the fewest possible number of transfers. This assumption may contradict Wardrop's first principle.

In [2], the lines for the transportation network can only be selected from a predefined set of potential lines (obviously, this restriction could rule out solutions which are even better than the considered feasible solutions). The goal is to maximize the number of users that require no transfer during their trips. As mentioned in [8], this optimization criterion could lead to unsatisfactory solutions due to its partial measure of service quality.

In [8], as in [2], the lines can only be selected from a predefined set of potential lines, and it is assumed that the station locations and the links between them are already known. The waiting time for the users and the effect of passenger crowding are not considered.

In [7] it is assumed that the number of lines for the transportation network is upper bounded, and that the length of each line and the total length of the lines are lower and upper bounded. Moreover, the endpoints of each line are predefined.

The aim of the present work is to improve the approach proposed in [5]. We provide a two-fold improvement on this. Firstly, we introduce several modifications in the model considered in the first stage to obtain a connected rapid transit network. Secondly, we explicitly determine the shortest routes for those origin-destination pairs of locations whose users are expected to utilize the rapid transit network, and present a greedy heuristic procedure which is a modification of the algorithm proposed for solving the line design problem of the second stage that attempts to minimize an estimation of the total number of transfers made by the users, without increasing the number of lines going to each location. We shall assume that the users obey Wardrop's first principle of route choice, where, as in [5, 14], the travel cost is interpreted as the travel distance.

We do not take into consideration the capacities of the lines, since they directly depend on the headways, which in turn will depend on the demands for the origin-destination pairs of locations. Notwithstanding the fact that we are dealing with static demands, actually the demands will vary over time. Therefore, it does not seem appropriate to determine the headways for the lines at this early stage of the problem. Instead, we propose to determine them during subsequent stages by taking into account the trade-off between operating costs and service quality. To this end, it may be advisable to develop transit assignment models for predicting as accurately as possible the way in which the users choose the routes to take their trips. Some works exclusively devoted to the development of such models are [4,16]; both of them take into account the expected waiting times for the users, among many other factors.

The remainder of the paper is structured as follows: Section 2 provides a nonlinear integer programming model for selecting the stations and links to be constructed. Section 3 shows two methods for solving the line design problem. Specifically, Section 3.1 reproduces the algorithm proposed in [5], which does not consider transfers. Section 3.2 states a linear 0-1 programming model for determining a minimum-length route between two locations, and presents a greedy heuristic procedure for solving the line design problem attempting to minimize an estimated number of required transfers, where it is assumed that the users choose routes of minimum length to take their trips. Section 4 proposes a linearization of the model stated in Section 2 and reports some computational experience on several instances from an extension of a network considered in [14]; the results show that the greedy procedure presented in Section 3.2 can significantly reduce the estimated number of transfers obtained by the procedure given in Section 3.1. Finally, Section 5 draws some conclusions from this work.

2. Station and link location

We consider the following notation and assumptions (see [5] for more details):

Let $V = \{1, \ldots, n\}$ be the set of potential locations for the stations, and let E be the set of (nonordered) pairs of locations that can be linked, *i.e.*, $E = \{\{i, j\} \in V \times V \mid i \neq j \text{ and it is possible to link } i \text{ and } j\}$. Without loss of generality, whenever we refer to an edge $\{i, j\} \in E$ it will be assumed that $i < j$.

Let us consider the simple graph $G = (V, E)$. For each $i \in V$, let $\Gamma(i)$ be the set of locations that can be linked to i (notice that $\Gamma(i)$ is the set of nodes adjacent to i in G and $|\Gamma(i)|$ is the degree of i in G).

Let W be the set of origin-destination pairs of locations in demand, and let us denote $w = (o_w, d_w) \quad \forall w \in W$, where o_w and d_w are the origin and the destination of pair w, respectively, and $o_w \neq d_w$. Let $\overline{w} = (d_w, o_w) \quad \forall w \in W$ such that $o_w > d_w$, and let $s(w) = \begin{cases} w \text{ if } o_w < d_w \\ \overline{w} \text{ if } o_w > d_w \end{cases} \quad \forall w \in W$ and $W' = \{s(w) \mid w \in W\}$.

Throughout the paper, we shall consider $W = \{(i, j) \in V \times V \mid i \neq j\}$, hence $W' = \{(i, j) \in V \times V \mid i < j\}$. We shall store only the values of $\{o_w\}_{w \in W'}$ and $\{d_w\}_{w \in W'}$.

Let a_i denote the cost of constructing a station at location i, c_{ij} the cost of linking locations i and j, b the available budget for constructing the rapid transit network, d_{ij} the distance between locations i and j, and g_w the demand for pair w, *i.e.*, the number of potential trips from o_w to d_w in a given time period. (We are assuming that $c_{ji} = c_{ij}$ and $d_{ji} = d_{ij} \quad \forall \{i, j\} \in E$.)

If there are λ lines going to a location i, then the associated construction cost will be λa_i, since it is assumed that we construct as many stations at i as the number of lines that go to it. Nevertheless, depending on the transportation company's construction policy, it could be more appropriate to compute this cost in some other way.

From now on it will be assumed that whichever two locations are linked at most by one line.

We define the following variables:

$$x_{ij} = \begin{cases} 1 \text{ if } i \text{ and } j \text{ are linked} \\ 0 \text{ otherwise} \end{cases} \quad \forall \{i, j\} \in E$$

$$p_w = \begin{cases} 1 \text{ if the users of pair } w \text{ will utilize the rapid transit network} \\ 0 \text{ otherwise} \end{cases} \quad \forall w \in W$$

$$f_{ij}^w = \begin{cases} 1 \text{ if the users of pair } w \text{ are recommended to utilize edge } \{i, j\} \\ 0 \text{ otherwise} \end{cases}$$

$$\forall w \in W', \forall \{i, j\} \in E$$

$$\varepsilon_i^w = \begin{cases} 1 \text{ if the users of pair } w \text{ are recommended to pass through } i \\ 0 \text{ otherwise} \end{cases} \qquad \forall w \in W', \forall i \in V \setminus \{o_w, d_w\}$$

$$y_i = \begin{cases} 1 \text{ if at least one station is constructed at } i \\ 0 \text{ otherwise} \end{cases} \qquad \forall i \in V$$

$$\gamma_i = \begin{cases} 1 \text{ if } \sum_{j \in \Gamma(i), j>i} x_{ij} + \sum_{j \in \Gamma(i), j<i} x_{ji} \text{ is odd} \\ 0 \text{ otherwise} \end{cases} \qquad \forall i \in V$$

$$\Delta_i \in \{0, \dots, r(i)\} \quad \forall i \in V,$$

where $r(i) = \begin{cases} \frac{|\Gamma(i)|}{2} & \text{if } |\Gamma(i)| \text{ is even} \\ \frac{|\Gamma(i)|-1}{2} & \text{if } |\Gamma(i)| \text{ is odd} \end{cases}$

and $\Delta_i = \frac{\sum_{j \in \Gamma(i), j>i} x_{ij} + \sum_{j \in \Gamma(i), j<i} x_{ji} - \gamma_i}{2} \quad \forall i \in V$.

The reason for considering the variables $\{\gamma_i\}_{i \in V}$ and $\{\Delta_i\}_{i \in V}$ is that, in order to compute the cost of constructing a station at each location, we need to know the number of lines that go to that location. The two methods presented in Section 3 for solving the line design problem will define the lines in such a way that the number of lines that go to each selected location is as small as possible. Notice that, for each $i \in V$, the value of $\sum_{j \in \Gamma(i), j>i} x_{ij} + \sum_{j \in \Gamma(i), j<i} x_{ji}$ is the number of selected links with an endpoint at i, and, according to the definition of Δ_i, this number equals $2\Delta_i + \gamma_i$. Thus, if $2\Delta_i + \gamma_i$ is even (which is equivalent to having $\gamma_i = 0$), then the number of lines going to i will be Δ_i, whereas if $2\Delta_i + \gamma_i$ is odd (which is equivalent to having $\gamma_i = 1$), then the number of lines going to i will be $\Delta_i + 1$. Therefore, in both cases the number of lines going to i will be $\Delta_i + \gamma_i$.

It is assumed that the users of each pair $w \in W$ will utilize the rapid transit network if and only if $y_{o_w} = 1$, $y_{d_w} = 1$ and $\sum_{\{i,j\} \in E} d_{ij} f_{ij}^{s(w)} \leq \mu\, u_w^{\text{pri}}$, where u_w^{pri} is the so-called generalized cost of satisfying the demand of pair w through an existing private network and μ is a so-called congestion factor verifying that $\mu\, u_w^{\text{pri}}$ is the distance covered by the users of pair w through the private network (see [14] for more details). Since we are interested in maximizing the total expected number of trips on the rapid transit network, if some route from o_w to d_w with length less or equal to $\mu\, u_w^{\text{pri}}$ is found, then the variables $f_{ij}^{s(w)}$ such that $\{i,j\} \in E$ and $f_{ij}^{s(w)} = 1$ will define one of such routes as the recommended route for the users of pair w, but it is worth noting that it may not be of minimum length.

Constraints (14) from Model 2 in [5] allow nonconnected rapid transit networks, since they are redundant if $p_w = 0$ (for the sake of completeness, this model is reproduced in Appendix A). The model below contains the modifications that we propose to introduce in Model 2 for selecting the stations and links to be constructed without exceeding budget, so that the resulting rapid transit network is connected and the total expected number of trips is maximized (notice that we have taken $W' = \{(i,j) \in V \times V \mid i < j\}$, and, therefore, any feasible solution to this model will give rise to a connected rapid transit network); see [5] for more details.

$$\text{Maximize} \quad z = \sum_{w \in W} g_w p_w$$

subject to:

$$\sum_{j \in \Gamma(i), j>i} x_{ij} + \sum_{j \in \Gamma(i), j<i} x_{ji} = 2\Delta_i + \gamma_i \quad \forall i \in V \tag{2.1}$$

$$\sum_{i \in V} a_i \left(\Delta_i + \gamma_i\right) + \sum_{\{i,j\} \in E} c_{ij} x_{ij} \leq b \tag{2.2}$$

$$y_i \leq \Delta_i + \gamma_i \quad \forall i \in V \tag{2.3}$$

$$(r(i)+1) y_i \geq \Delta_i + \gamma_i \quad \forall i \in V \tag{2.4}$$

$$f_{ij}^w \leq x_{ij} \quad \forall w \in W', \forall \{i,j\} \in E \tag{2.5}$$

$$\varepsilon_i^w \leq y_{o_w} y_{d_w} \quad \forall w \in W', \forall i \in V \setminus \{o_w, d_w\} \tag{2.6}$$

$$\sum_{j \in \Gamma(i), j>i} f_{ij}^w + \sum_{j \in \Gamma(i), j<i} f_{ji}^w = \begin{cases} y_{o_w} y_{d_w} & \text{if } i \in \{o_w, d_w\} \\ 2\varepsilon_i^w & \text{otherwise} \end{cases} \quad \forall w \in W', \forall i \in V \tag{2.7}$$

$$p_w \leq y_{o_{s(w)}} y_{d_{s(w)}} \quad \forall w \in W \tag{2.8}$$

$$\sum_{\{i,j\} \in E} d_{ij} f_{ij}^{s(w)} - \mu\, u_w^{\text{pri}} - M_w(1-p_w) \leq 0 \quad \forall w \in W \tag{2.9}$$

$$\begin{gathered} x_{ij} \in \{0,1\} \quad \forall \{i,j\} \in E \\ p_w \in \{0,1\} \quad \forall w \in W \\ f_{ij}^w \in \{0,1\} \quad \forall w \in W', \forall \{i,j\} \in E \\ \varepsilon_i^w \in \{0,1\} \quad \forall w \in W', \forall i \in V \setminus \{o_w, d_w\} \\ y_i \in \{0,1\} \quad \forall i \in V \\ \gamma_i \in \{0,1\} \quad \forall i \in V \\ \Delta_i \in \{0, \ldots, r(i)\} \quad \forall i \in V, \end{gathered}$$

where $M_w = \sum_{\{i,j\} \in E} d_{ij} - \mu\, u_w^{\text{pri}} \quad \forall w \in W$ (notice that M_w is an upper bound for the value of $\sum_{\{i,j\} \in E} d_{ij} f_{ij}^{s(w)} - \mu\, u_w^{\text{pri}}$).

The objective function is the same as in Model 2 from [5], which computes the total number of expected trips on the rapid transit network (this optimization criterion was also considered in [14]). Constraints (2.1) and (2.2) are the constraints (10) and (11) from Model 2, respectively, which impose the budget constraint. Constraints (2.3) and (2.4) impose that, for each $i \in V$, $y_i = 0$ if and only if no station is constructed at i (*i.e.*, if $\Delta_i + \gamma_i = 0$). Constraints (2.5) are a modification of constraints (14) from Model 2 to impose that, for each $\{i,j\} \in E$, if i and j are not linked, then the users of no pair $w \in W'$ are recommended to utilize edge $\{i, j\}$. Constraints (2.6) impose that, for each $w \in W'$, if no station is constructed at an endpoint of pair w, then its users are not recommended to pass through any location $i \in V \setminus \{o_w, d_w\}$. Constraints (2.7) are a modification of constraints (12) from Model 2 to allow that, for each $w \in W'$, if no station is constructed at an endpoint of pair w, then its users are not recommended to utilize any edge $\{i, j\} \in E$. Constraints (2.8) impose that, for each $w \in W$, if no station is constructed at an endpoint of pair w, then its users will not utilize the rapid transit network. Constraints (2.9) are the constraints (13) from Model 2, which, jointly with the optimization criterion and the constraints (2.8), impose that, for each $w \in W$, $p_w = 1$ if and only if $y_{o_w} = 1$, $y_{d_w} = 1$ and $\sum_{\{i,j\} \in E} d_{ij} f_{ij}^{s(w)} \leq \mu u_w^{\text{pri}}$.

For the particular case where at least one station should be constructed at each location, it would suffice to replace constraints (14) from Model 2 in [5] with constraints (2.5) above; thus, a linear integer programming model could be considered.

3. Line designing from a given set of links to be constructed

Let $(\overline{x}_{ij})_{\{i,j\} \in E}$, $(\overline{p}_w)_{w \in W}$, $(\overline{f}_{ij}^w)_{w \in W', \{i,j\} \in E}$, $(\overline{\varepsilon}_i^w)_{w \in W', i \in V \setminus \{o_w, d_w\}}$, $(\overline{y}_i)_{i \in V}$, $(\overline{\gamma}_i)_{i \in V}$, $(\overline{\Delta}_i)_{i \in V}$ be an optimal solution to the model stated in Section 2 (or an incumbent solution if the model has not been solved to optimality), and let $\overline{V} = \{i \in V \mid \overline{y}_i = 1\}$ and $\overline{E} = \{\{i, j\} \in E \mid \overline{x}_{ij} = 1\}$.

Let us consider the partial subgraph $\overline{G} = (\overline{V}, \overline{E})$ of G. For each $i \in \overline{V}$, let $\overline{\Gamma}(i)$ be the set of nodes adjacent to i in $\overline{G}$ (notice that $|\overline{\Gamma}(i)| = \sum_{j \in \Gamma(i), j > i} \overline{x}_{ij} + \sum_{j \in \Gamma(i), j < i} \overline{x}_{ji}$).

In Section 3.1 we reproduce the method proposed in [5] for solving the line design problem for $\overline{G}$. In Section 3.2 we present a new method which is a modification of the previous one and attempts to minimize an estimation of the total number of transfers that should be made by the users to arrive at their destinations; in order to make this estimation, it is assumed that the users choose routes of minimum length in $\overline{G}$ to take their trips, and a linear 0-1 programming model is provided for determining such routes.

It is worth noting that both methods would be valid for any partial subgraph $\overline{G}$ of G, not necessarily defined from an optimal or an incumbent solution to the model stated in Section 2; it would suffice to have an estimation $\overline{g}_w$ for the number of trips on the rapid transit network from o_w to d_w in the given time period for each $w \in W$, and define the set $\overline{W}$ considered in Section 3.2 as $\{w \in W \mid \overline{g}_w > 0\}$ (for example, [12, 15] make use of the Logit function to do these estimations). Thus, these methods could also be employed for redesigning the lines of existing rapid transit networks.

In both methods, the aim of assigning each selected link to exactly one line so that the number of lines that go to each selected location is as small as possible is accomplished by imposing that each node with odd degree in $\overline{G}$ is an endpoint of exactly one line, whereas each node with even degree in $\overline{G}$ is an endpoint of no line.

3.1. Line designing without considering transfers

For the sake of completeness, below we reproduce the algorithm for solving the line design problem for $\overline{G}$ proposed in [5]. It is based on the following idea: Starting from a node with odd degree, or, in its absence, with positive even degree, other nodes are reached sequentially through edges in $\overline{E}$, until we reach a node which either has previously been visited or it has no incident edges (once an edge has been considered, it is eliminated from $\overline{E}$). In the first case, a circular line is defined, and the above procedure is carried on from the last node reached which is an endpoint of an edge that has been eliminated from $\overline{E}$ but has not yet been assigned to a line, if such a node exists. In the second case, a noncircular line is defined. This approach is repeated until we get $\overline{E} = \emptyset$.

Proceeding in this way, the number of lines going to each location $i \in \overline{V}$ will be $\frac{|\overline{\Gamma}(i)|}{2}$ if $|\overline{\Gamma}(i)|$ is even, or $\frac{|\overline{\Gamma}(i)|+1}{2}$ if $|\overline{\Gamma}(i)|$ is odd. In both cases, this number equals $\overline{\Delta}_i + \overline{\gamma}_i$, *i.e.*, the smallest possible number of lines that can go to i (see Sect. 2). (Notice that it is necessary to firstly consider as starting nodes those with odd degree, since, otherwise, it may occur that the number of lines going to some node with even degree is not as small as possible).

In order to store the sequence of nodes chosen at each iteration, a nonnegative integer value $p(i)$ is associated to each node $i \in \overline{V}$, in such a way that a positive value $p(i)$ means that node i has been reached from node $p(i)$ (for the starting node i_0 we define $p(i_0) = i_0$). A counter l for the number of lines that are being defined is also considered. Each one of these lines is denoted by $L(l)$, and it is expressed as the union of its edges, considering the nodes in reverse order as they have been reached. When the last node j of the current iteration sequence is reached, the current line $L(l)$ starts to be defined from j. The last node to be included in $L(l)$ is denoted by j_0 and defined as j if $L(l)$ is going to be a circular line, or as i_0 if $L(l)$ is going to be a noncircular line.

Step 1 of Algorithm 1 initializes the values of $\{p(i)\}_{i\in\overline{V}}$ and l to zero. Step 2 performs the stopping criterion, *i.e.*, it checks whether $\overline{E} = \emptyset$. Step 3 is the beginning of a new iteration; it chooses the starting node i_0, increases by one unit the value of l, initializes line $L(l)$ to the empty set, sets the value of $p(i_0)$ and initializes the value of i to i_0 (i denotes the current node of the sequence). Step 4 chooses the next node j of the sequence, eliminates edge $\{i, j\}$ from $\overline{E}$ and, if j belongs to the current sequence of nodes, sets the value of j_0 and goes to Step 6 to start defining the circular line $L(l)$. If j currently has some adjacent node, Step 5 sets the value of $p(j)$, updates the value of i and goes back to Step 4; otherwise, it sets the value of j_0 and goes to Step 6 to start defining the noncircular line $L(l)$. Step 6 keeps on adding edges to $L(l)$ and eliminating their endpoints from the current sequence of nodes until reaching j_0. If $j_0 = i_0$, it means that all of the edges so far considered at the current iteration have been assigned to a line; in this case, Step 7 sets $p(i_0)$ to zero and goes back to Step 2. If $j_0 \neq i_0$, it means that the edges defining the sequence of nodes from i_0 to i have not yet been assigned to a line (notice that $i = j_0$ at this point); in this case, Step 8 increases by one unit the value of l, initializes line $L(l)$ to the empty set and attempts to continue adding nodes to the sequence from i (if it is not possible, it updates the values of j, i and $p(j)$, sets the value of j_0 and goes back to Step 6 to start defining the noncircular line $L(l)$).

Algorithm 1.

Step 1. *Set* $p(i) = 0 \quad \forall i \in \overline{V}$ *and* $l = 0$.
Step 2. *If* $|\overline{\Gamma}(i)| = 0 \quad \forall i \in \overline{V}$, *STOP.*
Step 3. *If* $|\overline{\Gamma}(i)|$ *is even* $\forall i \in \overline{V}$, *choose* $i_0 \in \overline{V}$ *such that* $|\overline{\Gamma}(i_0)| > 0$; *otherwise, choose* $i_0 \in \overline{V}$ *such that* $|\overline{\Gamma}(i_0)|$ *is odd. Set* $l = l+1$, $L(l) = \emptyset$, $p(i_0) = i_0$ *and* $i = i_0$.
Step 4. *Choose* $j \in \overline{\Gamma}(i)$ *and set* $\overline{\Gamma}(i) = \overline{\Gamma}(i) \setminus \{j\}$ *and* $\overline{\Gamma}(j) = \overline{\Gamma}(j) \setminus \{i\}$. *If* $p(j) > 0$, *set* $j_0 = j$ *and go to Step 6.*
Step 5. *If* $|\overline{\Gamma}(j)| > 0$, *set* $p(j) = i$, $i = j$ *and go to Step 4; otherwise, set* $j_0 = i_0$.
Step 6. *Set* $L(l) = L(l) \cup \{\{j, i\}\}$. *If* $i \neq j_0$, *set* $j = i$, $i = p(i)$, $p(j) = 0$ *and repeat Step 6.*
Step 7. *If* $j_0 = i_0$, *set* $p(i_0) = 0$ *and go to Step 2.*
Step 8. *Set* $l = l+1$ *and* $L(l) = \emptyset$. *If* $|\overline{\Gamma}(i)| > 0$, *go to Step 4; otherwise, set* $j = i$, $i = p(i)$, $p(j) = 0$, $j_0 = i_0$ *and go to Step 6.*

Remark 3.1. If $|\overline{\Gamma}(i_0)|$ is even, then it will always be $|\overline{\Gamma}(j)| > 0$ in Step 5 and $|\overline{\Gamma}(i)| > 0$ in Step 8.

The flowchart for Algorithm 1 is given in Appendix B.

Example 3.2. Consider the graph $\overline{G} = (\overline{V}, \overline{E})$, where $\overline{V} = \{1, 2, 3, 4, 5\}$ and $\overline{E} = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 4\}, \{3, 4\}, \{3, 5\}, \{4, 5\}\}$ (see Fig. 1). Then $\overline{\Gamma}(1) = \{2, 3, 4\}$, $\overline{\Gamma}(2) = \{1, 4\}$, $\overline{\Gamma}(3) = \{1, 4, 5\}$, $\overline{\Gamma}(4) = \{1, 2, 3, 5\}$ and $\overline{\Gamma}(5) = \{3, 4\}$.

Algorithm 1 proceeds as follows:

Step 1. $p(1) = p(2) = p(3) = p(4) = p(5) = 0$, $l = 0$
Step 3. $i_0 = 1$, $l = 1$, $L(1) = \emptyset$, $p(1) = 1$, $i = 1$
Step 4. $j = 2$, $\overline{\Gamma}(1) = \{3, 4\}$, $\overline{\Gamma}(2) = \{4\}$
Step 5. $p(2) = 1$, $i = 2$
Step 4. $j = 4$, $\overline{\Gamma}(2) = \emptyset$, $\overline{\Gamma}(4) = \{1, 3, 5\}$
Step 5. $p(4) = 2$, $i = 4$
Step 4. $j = 1$, $\overline{\Gamma}(4) = \{3, 5\}$, $\overline{\Gamma}(1) = \{3\}$, $j_0 = 1$
Step 6. $L(1) = \{\{1, 4\}\}$, $j = 4$, $i = 2$, $p(4) = 0$
Step 6. $L(1) = \{\{1, 4\}, \{4, 2\}\}$, $j = 2$, $i = 1$, $p(2) = 0$
Step 6. $L(1) = \{\{1, 4\}, \{4, 2\}, \{2, 1\}\}$
Step 7. $p(1) = 0$

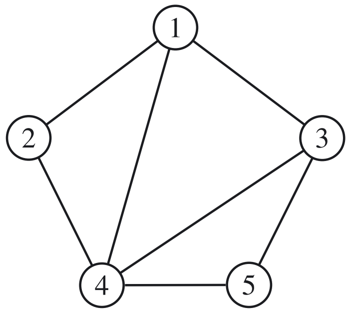


FIGURE 1. Graphic representation of $\overline{G} = (\overline{V}, \overline{E})$.

Step 3. $i_0 = 1,\ l = 2,\ L(2) = \emptyset,\ p(1) = 1,\ i = 1$

Step 4. $j = 3,\ \overline{\Gamma}(1) = \emptyset,\ \overline{\Gamma}(3) = \{4, 5\}$

Step 5. $p(3) = 1,\ i = 3$

Step 4. $j = 4,\ \overline{\Gamma}(3) = \{5\},\ \overline{\Gamma}(4) = \{5\}$

Step 5. $p(4) = 3,\ i = 4$

Step 4. $j = 5,\ \overline{\Gamma}(4) = \emptyset,\ \overline{\Gamma}(5) = \{3\}$

Step 5. $p(5) = 4,\ i = 5$

Step 4. $j = 3,\ \overline{\Gamma}(5) = \emptyset,\ \overline{\Gamma}(3) = \emptyset,\ j_0 = 3$

Step 6. $L(2) = \{\{3, 5\}\},\ j = 5,\ i = 4,\ p(5) = 0$

Step 6. $L(2) = \{\{3, 5\}, \{5, 4\}\},\ j = 4,\ i = 3,\ p(4) = 0$

Step 6. $L(2) = \{\{3, 5\}, \{5, 4\}, \{4, 3\}\}$

Step 8. $l = 3,\ L(3) = \emptyset,\ j = 3,\ i = 1,\ p(3) = 0,\ j_0 = 1$

Step 6. $L(3) = \{\{3, 1\}\}$

Step 7. $p(1) = 0$

Consequently, two circular lines $L(1) = \{\{1, 4\}, \{4, 2\}, \{2, 1\}\}$ and $L(2) = \{\{3, 5\}, \{5, 4\}, \{4, 3\}\}$, and one noncircular line $L(3) = \{\{3, 1\}\}$ have been defined.

3.2. LINE DESIGNING CONSIDERING TRANSFERS

Although Algorithm 1 is computationally very efficient, it has the disadvantage of not taking into consideration the transfers that should be made by the users to arrive at their destinations. These transfers could be taken into account simply by imposing certain rules for choosing the node j in Step 4. In order to establish appropriate rules, it is necessary to estimate the number of transfers that should be made at each location; for this purpose, it will be assumed that the users always choose routes of minimum length to take their trips.

Let $\overline{W} = \{w \in W \mid \overline{p}_w = 1\}$ and $\overline{W}' = \{s(w) \mid w \in \overline{W}\}$ (notice that $\overline{W}$ is the set of origin-destination pairs of locations whose users are expected to utilize the rapid transit network, and $\overline{W}' \subseteq W'$). As pointed out in Section 2, the values of $\{\overline{f}_{ij}^{w}\}_{w\in\overline{W}',\,\{i,j\}\in\overline{E}}$ define recommended routes such that $\sum_{\{i,j\}\in\overline{E}} d_{ij}\overline{f}_{ij}^{s(w)} \le \mu\, u_w^{\mathrm{pri}} \quad \forall w\in\overline{W}$, but they are not necessarily routes of minimum length in $\overline{G}$ from o_w to d_w for each $w\in\overline{W}$. Obviously, one way for determining such routes is to solve the following problem $(P_w) \quad \forall w\in\overline{W}'$ (notice that $(\overline{f}_{ij}^{w})_{\{i,j\}\in\overline{E}}$, $(\overline{\varepsilon}_i^{w})_{i\in\overline{V}\setminus\{o_w,d_w\}}$ is a feasible solution to (P_w), and, consequently, it could be taken as a starting solution for solving this):

$$\text{Minimize} \quad z_w = \sum_{\{i,j\}\in\overline{E}} d_{ij} f_{ij}^{w}$$

subject to:

$$\sum_{j\in\overline{\Gamma}(i),j>i} f_{ij}^{w} + \sum_{j\in\overline{\Gamma}(i),j<i} f_{ji}^{w} = \begin{cases} 1 & \text{if } i\in\{o_w,d_w\} \\ 2\varepsilon_i^{w} & \text{otherwise} \end{cases} \quad \forall i\in\overline{V} \qquad (P_w)$$

$$f_{ij}^{w}\in\{0,1\} \quad \forall\{i,j\}\in\overline{E}$$

$$\varepsilon_i^{w}\in\{0,1\} \quad \forall i\in\overline{V}\setminus\{o_w,d_w\}.$$

For each $w\in\overline{W}'$, let $(\hat{f}_{ij}^{w})_{\{i,j\}\in\overline{E}}$, $(\hat{\varepsilon}_i^{w})_{i\in\overline{V}\setminus\{o_w,d_w\}}$ be an optimal solution to problem (P_w).

From now on it will be assumed that the users follow the routes defined by the values of $\{\hat{f}_{ij}^{w}\}_{w\in\overline{W}',\,\{i,j\}\in\overline{E}}$. For simplicity of notation, we define $\hat{f}_{ji}^{w}=\hat{f}_{ij}^{w} \quad \forall w\in\overline{W}', \forall\{i,j\}\in\overline{E}$.

Let $\overline{W}_i = \{w\in\overline{W} \mid i\notin\{o_w,d_w\}, \hat{\varepsilon}_i^{s(w)}=1\} \ \forall i\in\overline{V}$, let $t_j(i) = \sum_{w\in\overline{W}_i,\hat{f}_{ij}^{s(w)}=1} g_w \ \forall i\in\overline{V}, \forall j\in\overline{\Gamma}(i)$, and let $t_k(i,j) = \sum_{w\in\overline{W}_j,\hat{f}_{ij}^{s(w)}+\hat{f}_{jk}^{s(w)}=1} g_w \ \forall i\in\overline{V}, \forall j\in\overline{\Gamma}(i), \forall k\in\overline{\Gamma}(j)\setminus\{i\}$ (notice that $\overline{W}_i$ is the set of origin-destination pairs of locations whose users pass through location i, $t_j(i)$ is the number of transfers at location i made by the users that utilize the link joining i with j, provided that i is an endpoint of the line that links i and j, and $t_k(i,j)$ is the number of transfers at location j made by the users that utilize one and only one of the links joining i with j, and j with k, provided that these links belong to the same line).

The algorithm below is a modification of Algorithm 1 that incorporates a greedy rule for reaching the nodes, thus resulting in a greedy heuristic procedure that attempts to minimize the estimated total number of required transfers. This greedy rule is applied in Step 3 for choosing the second node of the sequence, as well as in Step 6 for choosing the subsequent nodes of the sequence. (Notice that we are imposing that each node with odd degree in $\overline{G}$ is an endpoint of exactly one line, and each node with even degree in $\overline{G}$ is an endpoint of no line; see the beginning of Sect. 3).

Once the starting node i_0 has been chosen in Step 3, the criterion for selecting the next node j of the sequence depends on whether $|\overline{\Gamma}(i_0)|$ is even or odd. If $|\overline{\Gamma}(i_0)|$ is even, then i_0 cannot be an endpoint of any of the lines that will be defined later on; therefore, we can choose any node $j\in\overline{\Gamma}(i_0)$. If $|\overline{\Gamma}(i_0)|$ is odd, then i_0 has to be an endpoint of exactly one of the lines that will be defined later on, hence, in order to attempt to reduce the number of transfers made at i_0, we choose a node $j \in \overline{\Gamma}(i_0)$ with minimum value of $t_j(i_0)$.

Given the two last nodes i and j that have been added to the sequence so far, in Step 6 we have to decide whether to continue or stop adding nodes to the current iteration sequence, and, if we decide to continue, then we have to choose the next node k. For this purpose, and in order to attempt to reduce the number of transfers made at j, we determine a node $k \in \overline{\Gamma}(j)$ with minimum value of $t_k(i,j)$. If $|\overline{\Gamma}(j)|$ is odd, then j cannot be an endpoint of any of the lines that will be defined later on; thus, we have to continue adding nodes to the sequence, and we choose k as the next one. If $|\overline{\Gamma}(j)|$ is even, then j has to be an endpoint of exactly one of the lines that will be defined later on, hence we can either continue or stop adding nodes to the sequence. In order to make this decision, we compare the values of $t_k(i,j)$ and $t_i(j)$. If $t_k(i,j) > t_i(j)$, it can be expected that the number of transfers made at j if we continue adding nodes will be greater than if we stop; consequently, we decide to stop adding nodes, *i.e.*, we set the value of j_0 and go to Step 7 to start defining the noncircular line $L(l)$. Otherwise, we continue adding nodes to the sequence, and we choose k as the next one.

Algorithm 2.

Step 1. *Set* $p(i) = 0 \quad \forall i \in \overline{V}$ *and* $l = 0$.
Step 2. *If* $|\overline{\Gamma}(i)| = 0 \quad \forall i \in \overline{V}$, *STOP.*
Step 3. *If* $|\overline{\Gamma}(i)|$ *is even* $\forall i \in \overline{V}$, *choose* $i_0 \in \overline{V}$ *such that* $|\overline{\Gamma}(i_0)| > 0$ *and* $j \in \overline{\Gamma}(i_0)$; *otherwise, choose* $i_0 \in \overline{V}$ *such that* $|\overline{\Gamma}(i_0)|$ *is odd and set* $j = \arg\min\{t_{j'}(i_0) \mid j' \in \overline{\Gamma}(i_0)\}$. *Set* $l = l + 1$, $L(l) = \emptyset$, $p(i_0) = i_0$ *and* $i = i_0$.
Step 4. *Set* $\overline{\Gamma}(i) = \overline{\Gamma}(i) \setminus \{j\}$ *and* $\overline{\Gamma}(j) = \overline{\Gamma}(j) \setminus \{i\}$. *If* $p(j) > 0$, *set* $j_0 = j$ *and go to Step 7.*
Step 5. *If* $|\overline{\Gamma}(j)| = 0$, *set* $j_0 = i_0$ *and go to Step 7.*
Step 6. *Set* $k = \arg\min\{t_{k'}(i,j) \mid k' \in \overline{\Gamma}(j)\}$. *If* $|\overline{\Gamma}(j)|$ *is even and* $t_k(i,j) > t_i(j)$, *set* $j_0 = i_0$; *otherwise, set* $p(j) = i$, $i = j$, $j = k$ *and go to Step 4.*
Step 7. *Set* $L(l) = L(l) \cup \{\{j,i\}\}$. *If* $i \neq j_0$, *set* $j = i$, $i = p(i)$, $p(j) = 0$ *and repeat Step 7.*
Step 8. *If* $j_0 = i_0$, *set* $p(i_0) = 0$ *and go to Step 2.*
Step 9. *Set* $l = l + 1$, $L(l) = \emptyset$, $j = i$, $i = p(i)$, $p(j) = 0$ *and go to Step 5.*

Remark 3.3. If $|\overline{\Gamma}(i_0)|$ is even, then Step 5 can be skipped, since it will always be $|\overline{\Gamma}(j)| > 0$.

The flowchart for Algorithm 2 is given in Appendix C.

Example 3.4. Consider the graph $\overline{G} = (\overline{V}, \overline{E})$ from Example 3.2, and let us assume that $\overline{W} = W$ and that the route of minimum length between each origin-destination pair of locations and the demands $\{g_w\}_{w \in \overline{W}}$ are those given in Table 1.

Therefore, we get that $\overline{W}_1 = \emptyset$, $\overline{W}_2 = \emptyset$, $\overline{W}_3 = \{(1,5),(5,1)\}$, $\overline{W}_4 = \{(2,3),(2,5),(3,2),(5,2)\}$ and $\overline{W}_5 = \emptyset$.

TABLE 1. Shortest routes and demands for the pairs $w \in \overline{W}$.

o_w	d_w	Shortest route between o_w and d_w	$g_{(o_w,d_w)}$	$g_{(d_w,o_w)}$
1	2	$\{\{1,2\}\}$	1	1
1	3	$\{\{1,3\}\}$	1	1
1	4	$\{\{1,4\}\}$	1	1
1	5	$\{\{1,3\},\{3,5\}\}$	2	2
2	3	$\{\{2,4\},\{4,3\}\}$	3	3
2	4	$\{\{2,4\}\}$	1	1
2	5	$\{\{2,4\},\{4,5\}\}$	2	2
3	4	$\{\{3,4\}\}$	1	1
3	5	$\{\{3,5\}\}$	1	1
4	5	$\{\{4,5\}\}$	1	1

Algorithm 2 proceeds as follows:

Step 1. $p(1) = p(2) = p(3) = p(4) = p(5) = 0,\ l = 0$

Step 3. $i_0 = 1, t_2(1) = 0, t_3(1) = 0, t_4(1) = 0, j = 2, l = 1, L(1) = \emptyset, p(1) = 1, i = 1$

Step 4. $\overline{\Gamma}(1) = \{3,4\},\ \overline{\Gamma}(2) = \{4\}$

Step 6. $t_4(1,2) = 0,\ k = 4,\ p(2) = 1,\ i = 2,\ j = 4$

Step 4. $\overline{\Gamma}(2) = \emptyset,\ \overline{\Gamma}(4) = \{1,3,5\}$

Step 6. $t_1(2,4) = 10,\ t_3(2,4) = 4,\ t_5(2,4) = 6,\ k = 3,\ p(4) = 2,\ i = 4,\ j = 3$

Step 4. $\overline{\Gamma}(4) = \{1,5\},\ \overline{\Gamma}(3) = \{1,5\}$

Step 6. $t_1(4,3) = 4,\ t_5(4,3) = 4,\ k = 1,\ t_4(3) = 0,\ j_0 = 1$

Step 7. $L(1) = \{\{3,4\}\},\ j = 4,\ i = 2,\ p(4) = 0$

Step 7. $L(1) = \{\{3,4\},\{4,2\}\},\ j = 2,\ i = 1,\ p(2) = 0$

Step 7. $L(1) = \{\{3,4\},\{4,2\},\{2,1\}\}$

Step 8. $p(1) = 0$

Step 3. $i_0 = 1,\ j = 3,\ l = 2,\ L(2) = \emptyset,\ p(1) = 1,\ i = 1$

Step 4. $\overline{\Gamma}(1) = \{4\},\ \overline{\Gamma}(3) = \{5\}$

Step 6. $t_5(1,3) = 0,\ k = 5,\ p(3) = 1,\ i = 3,\ j = 5$

Step 4. $\overline{\Gamma}(3) = \emptyset,\ \overline{\Gamma}(5) = \{4\}$

Step 6. $t_4(3,5) = 0,\ k = 4,\ p(5) = 3,\ i = 5,\ j = 4$

Step 4. $\overline{\Gamma}(5) = \emptyset,\ \overline{\Gamma}(4) = \{1\}$

Step 6. $t_1(5,4) = 4,\ k = 1,\ p(4) = 5,\ i = 4,\ j = 1$

Step 4. $\overline{\Gamma}(4) = \emptyset,\ \overline{\Gamma}(1) = \emptyset,\ j_0 = 1$

Step 7. $L(2) = \{\{1,4\}\},\ j = 4,\ i = 5,\ p(4) = 0$

Step 7. $L(2) = \{\{1,4\},\{4,5\}\},\ j = 5,\ i = 3,\ p(5) = 0$

Step 7. $L(2) = \{\{1,4\},\{4,5\},\{5,3\}\},\ j = 3,\ i = 1,\ p(3) = 0$

Step 7. $L(2) = \{\{1,4\},\{4,5\},\{5,3\},\{3,1\}\}$

Step 8. $p(1) = 0$

Consequently, one noncircular line $L(1) = \{\{3,4\},\{4,2\},\{2,1\}\}$, *and one circular line* $L(2) = \{\{1,4\},\{4,5\},\{5,3\},\{3,1\}\}$ *have been defined.*

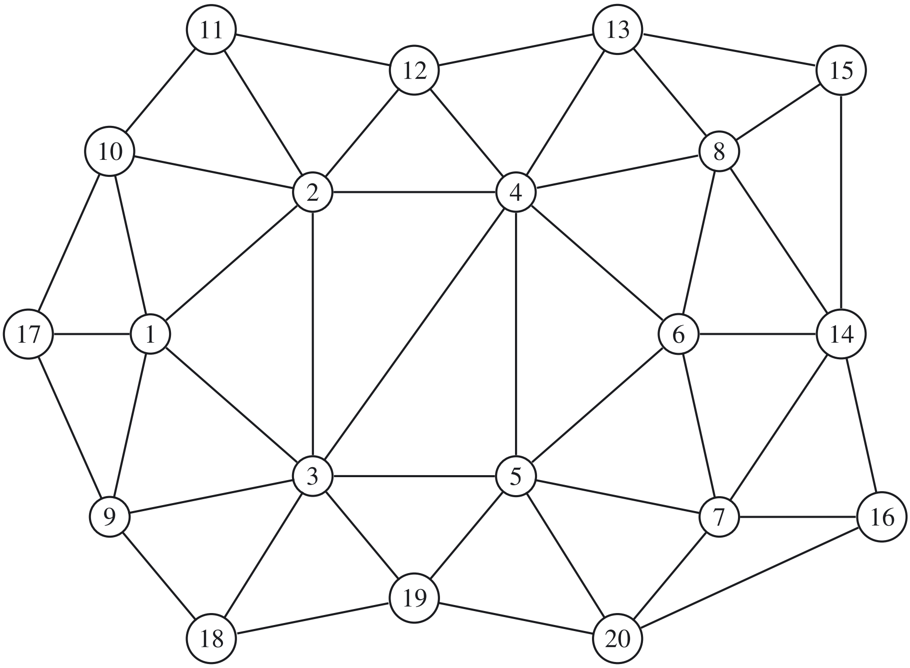


FIGURE 2. Graphic representation of $G = (V, E)$.

It can easily be verified that the estimated number of transfers required for the line designs obtained by Algorithms 1 and 2 (see Ex. 3.2) is 14 and 4, respectively.

4. Computational experience

We consider an extension of network $R2$ from [14] ($R2$ was also considered in [5]), consisting of twenty nodes and forty-five edges. Figure 2 shows its underlying graph $G = (V, E)$.

The station construction costs $\{a_i\}_{i \in V}$, the linking construction costs $\{c_{ij}\}_{\{i,j\} \in E}$, the distances $\{d_{ij}\}_{\{i,j\} \in E}$, the demands $\{g_{(i,j)}\}_{i,j \in V,\, i \neq j}$ and the generalized costs $\{u^{\text{pri}}_{(i,j)}\}_{i,j \in V,\, i \neq j}$ can be found in [6].

The implementation platform has been Microsoft Visual C++ 2005, CPLEX v11.2 and Pentium 4, 3.00 GHz, 1.00 Gb RAM.

In order to solve the model stated in Section 2 by using the optimization engine CPLEX, it is necessary to convert it into a linear integer programming model. Given that the unique nonlinear expressions in the model are of the form $y_{o_w} y_{d_w}$, where $w \in W'$, we have defined the variables

$$\delta_{ij} = \begin{cases} 1 \text{ if } y_i = 1 \text{ and } y_j = 1 \\ 0 \text{ otherwise} \end{cases} \qquad \forall i, j \in V \text{ such that } i < j$$

(notice that we are considering $W' = \{(i,j) \in V \times V \mid i < j\}$), we have appended to the model the following constraints, which impose that $\delta_{ij} = y_i\, y_j \quad \forall i,j \in V$ such that $i < j$:

$$\delta_{ij} \le y_i \quad \forall i,j \in V \text{ such that } i < j$$

$$\delta_{ij} \le y_j \quad \forall i,j \in V \text{ such that } i < j$$

$$\delta_{ij} \ge y_i + y_j - 1 \quad \forall i,j \in V \text{ such that } i < j$$

$$\delta_{ij} \in \{0,1\} \quad \forall i,j \in V \text{ such that } i < j,$$

and we have replaced constraints (2.6)–(2.8) with constraints (4.1)–(4.3) below, respectively.

$$\varepsilon_i^w \le \delta_{o_w,d_w} \quad \forall w \in W', \forall i \in V \setminus \{o_w, d_w\} \tag{4.1}$$

$$\sum_{j \in \Gamma(i), j>i} f_{ij}^w + \sum_{j \in \Gamma(i), j<i} f_{ji}^w = \begin{cases} \delta_{o_w,d_w} & \text{if } i \in \{o_w, d_w\} \\ 2\varepsilon_i^w & \text{otherwise} \end{cases} \quad \forall w \in W', \forall i \in V \tag{4.2}$$

$$p_w \le \delta_{o_{s(w)},d_{s(w)}} \quad \forall w \in W. \tag{4.3}$$

We have run the CPLEX mixed integer optimizer by using the default rules, except that the relative and absolute optimality tolerances have been set to zero, a time limit of one hour has been imposed and, in the branching process, the priorities for the variables $\{\Delta_i\}_{i \in V}$, $\{f_{ij}^w\}_{w \in W', \{i,j\} \in E}$, $\{x_{ij}\}_{\{i,j\} \in E}$ and $\{p_w\}_{w \in W}$ have been set to 1, 2, 3 and 4, respectively. The reason for imposing this time limit is that the main goal of our computational experience is to compare the performance of Algorithms 1 and 2; the tightening of the model proposed for the first stage and the development of a more efficient method to solve this are areas of future research.

Table 2 shows the computational results obtained by considering several values for the available budget b.

The columns headed "$\overline{z}$", "*Nodes*" and "*Time 1*" give, respectively, the value of the objective function at the optimal or incumbent solution, the number of branch-and-cut nodes evaluated and the CPU time expressed in seconds related to the solving of the linearization of the model stated in Section 2 taking $\mu = 1$.

The column headed "*Time 2*" gives the CPU time expressed in seconds required for solving the problems $\{(P_w)\}_{w \in \overline{W}'}$ by taking $(\overline{f}_{ij}^w)_{\{i,j\} \in \overline{E}}$, $(\overline{\varepsilon}_i^w)_{i \in \overline{V} \setminus \{o_w, d_w\}}$ as the starting feasible solution for each $w \in \overline{W}'$.

We have followed two different strategies for choosing the nodes i_0 and j in Steps 3 and 4 of Algorithm 1, and the nodes i_0, j and k in Steps 3 and 6 of Algorithm 2. The first strategy is to choose each one of these nodes as the minimum of its possible values. The second strategy is to choose each one of them randomly and uniformly distributed over the set of all of its possible values.

The columns headed "*Trans. 1*" give the estimated total number of transfers required for the line design obtained by applying once Algorithms 1 and 2 following the first strategy above (the CPU times have been inappreciable for all the instances).

TABLE 2. Computational results.

					Algorithm 1			Algorithm 2		
b	$\overline{z}$	Nodes	Time 1	Time 2	Trans. 1	Trans. 2	Line design	Trans. 1	Trans. 2	Line design
150	4613	2284	>3600	0.43	6078	3975	L_1: 8-13-12-2-1-17	4547	4013	L_1: 3-9-18-3
							L_2: 6-7-14-6			L_2: 2-3-4-2
							L_3: 5-7-20-5			L_3: 5-7-20-5
							L_4: 2-3-4-2			L_4: 6-7-14-6
							L_5: 3-9-18-3			L_5: 8-13
							L_6: 2-11-10-1-3-5-6-8-15-13			L_6: 2-1-10-11-12-4-6
							L_7: 6-4-12-11			L_7: 11-2-12-13-15-8-6-5-3-1-17
160	4119	2392	>3600	0.39	5802	2930	L_1: 1-3-5-6-8-15-13-12-2-10-17-1	3848	3603	L_1: 3-4-13-8
							L_2: 6-4-12-11-2-3-9			L_2: 9-3-2-11
							L_3: 6-7-14-6			L_3: 1-3-5-4-2-10-1
							L_4: 5-7-20-5			L_4: 5-20-7-6
							L_5: 5-4-13-8			L_5: 1-2-12-11-10-17-1
							L_6: 3-4-2-1-10-11			L_6: 5-6-14-7-5
										L_7: 4-6-8-15-13-12-4
170	4177	3194	>3600	0.45	5283	3026	L_1: 1-3-5-6-8-15-13-12-11-10-17-1	3082	2881	L_1: 3-4-13-8-14-6
							L_2: 5-7-20-5			L_2: 11-2-3-18
							L_3: 1-2-10-1			L_3: 5-7-20-5
							L_4: 15-14-7-6-4-12-2-3-18			L_4: 1-2-4-12-11-10-17-1
							L_5: 4-8-13-4			L_5: 8-4-6-7-14-15
							L_6: 6-14-8			L_6: 1-3-5-6-8-15-13-12-2-10-1
							L_7: 3-4-2-11			
180	3684	3689	>3600	0.52	4799	2714	L_1: 3-18-19-3	3761	2607	L_1: 7-14-16-20-19-18-9
							L_2: 1-9-3-2-10-1			L_2: 4-5-20-7-6-4
							L_3: 5-7-16-20-19-5			L_3: 16-7-5-19-3-18
							L_4: 4-5-20-7-6-8-4			L_4: 1-9-3-2-10-1
							L_5: 10-17-1-3-5-6-14-16			L_5: 1-2-4-8-6
							L_6: 9-18			L_6: 10-17-1-3-5-6-14
							L_7: 1-2-4-6			
							L_8: 7-14			

TABLE 2. Continued.

b	$\overline{z}$	Nodes	Time 1	Time 2	Algorithm 1			Algorithm 2		
					Trans. 1	Trans. 2	Line design	Trans. 1	Trans. 2	Line design
190	5746	5926	>3600	0.57	8808	4971	L_1: 1-9-18-19-20-16-7-5-4-2-11-10-1	5195	4611	L_1: 8-13
							L_2: 8-13-12-2-10			L_2: 5-7-16-20-5
							L_3: 6-14			L_3: 1-9-18-19-20-7-6-4-12-11-10-1
							L_4: 9-3-19-5-20-7-6-4-12-11			L_4: 1-2-4-5-19-3-9
							L_5: 13-15-8-6-5-3-1-17			L_5: 11-2-3-18
							L_6: 1-2-3-18			L_6: 10-2-12-13-15-8-6-5-3-1-17
										L_7: 6-14
200	5788	2825	916.13	0.76	8709	5290	L_1: 4-5-19-20-7-6-4	5272	4649	L_1: 8-13
							L_2: 2-10-11-2			L_2: 10-2-4-5-19-3-18
							L_3: 11-12-4-2-1-17			L_3: 6-14-7
							L_4: 5-7-16-20-5			L_4: 5-7-16-20-5
							L_5: 10-1-3-5-6-8-15-13			L_5: 2-3-9-18-19-20-7-6-4-12-11-2
							L_6: 15-14-16			L_6: 16-14-15-8-6-5-3-1-17
							L_7: 6-14-7			L_7: 11-10-1-2-12-13-15
							L_8: 3-9-18-3			
							L_9: 8-13-12-2-3-19-18			
210	5796	1781	350.22	0.73	8423	5205	L_1: 5-7-16-20-5	5176	4562	L_1: 1-2-12-13-15-14-7
							L_2: 2-4-6-14-15-13-12-11-10-2			L_2: 8-13
							L_3: 7-14-16			L_3: 9-3-19-5-4-2-10
							L_4: 3-9-18-3			L_4: 6-14-16
							L_5: 6-7-20-19-5-4-12-2-1-9			L_5: 15-8-6-5-3-1-17
							L_6: 11-2-3-19-18			L_6: 11-2-3-18
							L_7: 15-8-6-5-3-1-17			L_7: 1-9-18-19-20-7-6-4-12-11-10-1
							L_8: 8-13			L_8: 5-7-16-20-5
							L_9: 1-10			

TABLE 2. Continued.

					Algorithm 1			Algorithm 2		
b	$\overline{z}$	Nodes	Time 1	Time 2	Trans. 1	Trans. 2	Line design	Trans. 1	Trans. 2	Line design
220	5803	590	108.81	0.83	8173	4910	L_1: 1-2-10-17-1	5604	4264	L_1: 1-9-18-19-20-16-7-5-4-2-10-1
							L_2: 5-7-20-5			L_2: 3-18
							L_3: 2-3-18-19-20-16-7-6-4-12-11-2			L_3: 6-14-7
							L_4: 11-10-1-3-5-6-8-15			L_4: 1-2-12-13-15-14-16
							L_5: 8-13-15-14-16			L_5: 2-3-19-5-20-7-6-4-12-11-2
							L_6: 6-14-7			L_6: 8-13-4-3-9
							L_7: 1-9-18			L_7: 11-10-17-1-3-5-6-8-15
							L_8: 2-4-13-12-2			
							L_9: 3-4-5-19-3			
							L_{10}: 3-9			
230	5803	682	65.31	0.94	8229	4957	L_1: 4-6-14-8-4	5124	4230	L_1: 6-14
							L_2: 5-7-16-20-19-5			L_2: 1-2-12-13-15-14-7
							L_3: 2-3-9-17-10-11-2			L_3: 4-6-7-20-19-18-9-17-10-11-12-4
							L_4: 15-8-6-5-3-1-17			L_4: 4-5-20-16-14-8-13-4
							L_5: 3-18-19-3			L_5: 8-4-2-10-1-9-3-19-5-7-16
							L_6: 1-2-10-1			L_6: 15-8-6-5-3-1-17
							L_7: 2-4-13-12-2			L_7: 11-2-3-18
							L_8: 6-7-20-5-4-12-11			
							L_9: 7-14-16			
							L_{10}: 8-13-15-14			
							L_{11}: 1-9-18			
240	5803	300	33.16	0.81	7337	5159	L_1: 4-8-15-13-12-4	4839	4054	L_1: 4-5-7-16
							L_2: 14-15			L_2: 3-4-13-8-14-7
							L_3: 1-2-11-10-17-1			L_3: 1-9-3-19-5-20-7-6-4-12-11-10-1
							L_4: 5-7-20-5			L_4: 8-4-2-10-17-9-18-19-20-16-14
							L_5: 1-3-5-6-8-13-4-2-10-1			L_5: 1-2-12-13-15-14-6
							L_6: 1-9-3-4-6-7-16-14-8			L_6: 15-8-6-5-3-1-17
							L_7: 11-12-2-3-18-9-17			L_7: 11-2-3-18
							L_8: 6-14-7			
							L_9: 16-20-19-18			
							L_{10}: 3-19-5-4			

The columns headed "*Trans. 2*" and "*Line design*" give, respectively, the minimum estimated total number of required transfers and the associated line design obtained by applying Algorithms 1 and 2 for 15 seconds each, following the second strategy above (each lth line is denoted as L_l and defined by the sequence of locations to which it goes). We have also increased the time limit to 10 minutes, but the results have not varied.

Given a line design obtained by Algorithm 1 or Algorithm 2, for each $e \in \overline{E}$ let $\hat{l}(e)$ denote the unique value of l verifying that edge e has been assigned to line $L(l)$. Moreover, for each $i \in \overline{V}$ with $|\overline{\Gamma}(i)| \geq 3$ and each $w \in \overline{W}_i$, let $j_1(i,w)$ and $j_2(i,w)$ denote the two unique nodes in $\overline{\Gamma}(i)$ such that $\hat{f}^{s(w)}_{i,j_1(i,w)} = 1$ and $\hat{f}^{s(w)}_{i,j_2(i,w)} = 1$ (see problem (P_w) in Sect. 3.2).

The estimation for the total number of required transfers that has been considered for computing the values for "*Trans. 1*" and "*Trans. 2*" is given by $\sum_{i\in\overline{V},|\overline{\Gamma}(i)|\geq 3}\sum_{w\in\widehat{W}_i} g_w$, where $\widehat{W}_i = \{w \in \overline{W}_i \mid \hat{l}(e_1(i,w)) \neq \hat{l}(e_2(i,w))\}$ and

$$e_k(i,w) = \begin{cases} \{i, j_k(i,w)\} \text{ if } i < j_k(i,w) \\ \{j_k(i,w), i\} \text{ if } i > j_k(i,w) \end{cases} \quad \forall k \in \{1,2\}.$$

In addition to the values for b considered in Table 2, we have performed the computational experiments for $b \in \{10, 20, \ldots, 140\}$. However, for $b \in \{10, 20, 30\}$ it is possible to define only one line in the resulting graph $\overline{G}$, hence these instances are not significant. For $b \in \{40, 50, \ldots, 140\}$, the values for "*Trans. 1*" obtained by Algorithm 1 are greater or equal to the ones obtained by Algorithm 2 (the equality holds for $b \in \{40, 90, 110\}$), whereas both algorithms achieve the same values for "*Trans. 2*" in each instance. Thus, we are not presenting here the related results.

We can observe from Table 2 that the values for "*Trans. 1*" obtained by Algorithm 1 are much greater than those obtained by Algorithm 2. With regard to the values for "*Trans. 2*", Algorithm 2 obtains smaller values than Algorithm 1, except for $b \in \{150, 160\}$, where the opposite occurs. Therefore, given the little computational effort required by these algorithms, we propose to apply both of them for a certain time period following the second strategy, and choose a line design with the smallest estimated total number of required transfers.

Given that for some of the values of b considered in Table 2 we are dealing with an incumbent (non-optimal) solution to the linearization of the model stated in Section 2, we have not performed a sensitivity analysis of the parameters.

Figures 3 and 4 depict the best line design obtained for $b = 150$ and $b = 240$, respectively.

5. Conclusions

In this paper we have presented some improvements on a two-stage approach that we described elsewhere for solving a modification of the extended rapid transit network design problem to allow the definition of circular lines. We have introduced several modifications in the model considered in the first stage for selecting the stations and links to be constructed to guarantee that the resulting rapid tran-

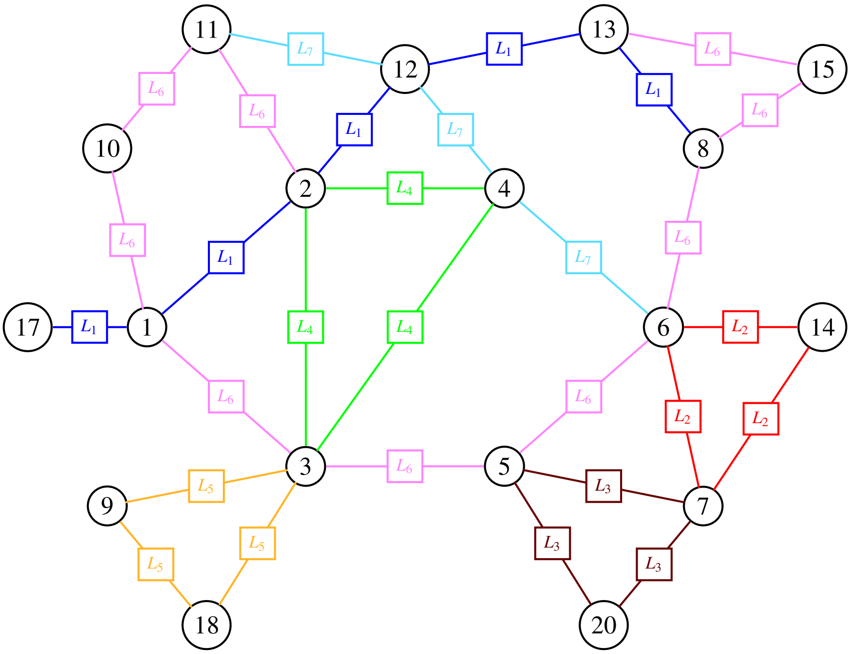


FIGURE 3. Graphic representation of the best line design obtained for $b = 150$.

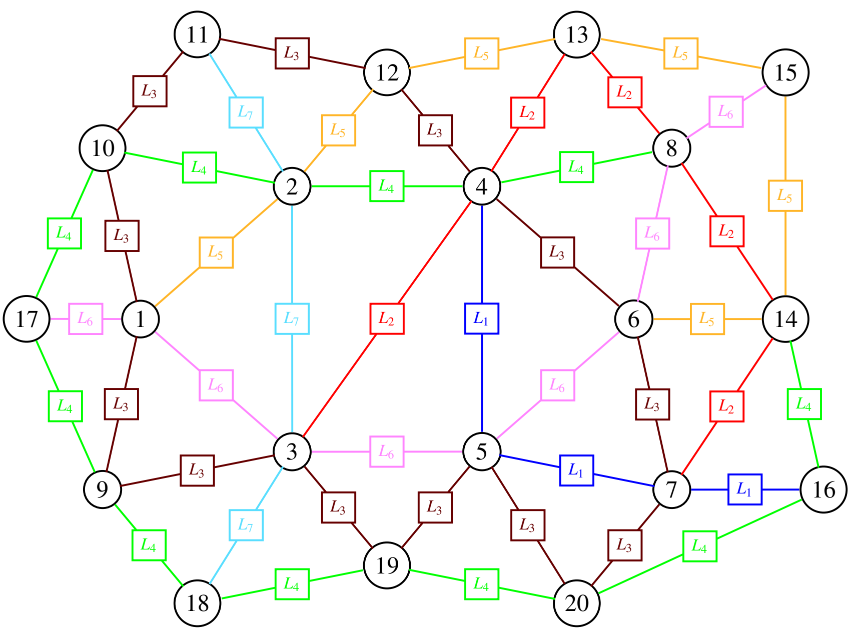


FIGURE 4. Graphic representation of the best line design obtained for $b = 240$.

sit network will be connected. Furthermore, we have proposed a greedy heuristic procedure for solving the line design problem of the second stage that attempts to minimize an estimation of the total number of transfers that should be made by the users of the rapid transit network. This procedure and the one that we described elsewhere are valid for any set of stations and links to be constructed, not necessarily obtained by solving the model considered in the first stage, hence they could also be used for redesigning the lines of existing rapid transit networks. The reported comparative computational experience between both procedures shows that, for the considered instances with a bigger number of links to be constructed, our greedy procedure significantly reduces the estimated number of transfers, but this performance does not remain true for the rest of the instances. Consequently, it is likely that the greedy procedure will achieve better line designs for large-size instances. Nevertheless, given the computational efficiency of these procedures, it will be possible to apply both of them and choose the best line design obtained.

Acknowledgements. This research was supported in part by the grants MTM2009-14087-C04-01 and TIN2009-07901 from Ministerio de Ciencia e Innovación, Spain. The authors are grateful to two anonymous reviewers for their helpful comments on the first version of this paper.

Appendix A. Model 2 in [5]

The Model 2 proposed in [5] is as follows:

$$\text{Maximise} \quad z = \sum_{w \in W} g_w p_w$$

subject to:

$$\sum_{j \in \Gamma(i), j>i} x_{ij} + \sum_{j \in \Gamma(i), j<i} x_{ji} = 2\Delta_i + \gamma_i \quad \forall i \in V \tag{10}$$

$$\sum_{i \in V} a_i \left(\Delta_i + \gamma_i\right) + \sum_{\{i,j\} \in E} c_{ij} x_{ij} \leq b \tag{11}$$

$$\sum_{j \in \Gamma(i), j>i} f_{ij}^w + \sum_{j \in \Gamma(i), j<i} f_{ji}^w = \begin{cases} 1 & \text{if } i \in \{o_w, d_w\} \\ 2\varepsilon_i^w & \text{otherwise} \end{cases} \quad \forall w \in W', \forall i \in V \tag{12}$$

$$\sum_{\{i,j\} \in E} d_{ij} f_{ij}^{s(w)} - \mu\, u_w^{\text{pri}} - M_w(1 - p_w) \leq 0 \quad \forall w \in W \tag{13}$$

$$f_{ij}^{s(w)} + p_w - 1 \leq x_{ij} \quad \forall w \in W, \forall \{i,j\} \in E \tag{14}$$

$$x_{ij} \in \{0,1\} \quad \forall \{i,j\} \in E$$

$$f_{ij}^w \in \{0,1\} \quad \forall w \in W', \forall \{i,j\} \in E$$

$$p_w \in \{0,1\} \quad \forall w \in W$$

$$\varepsilon_i^w \in \{0,1\} \quad \forall w \in W', \forall i \in V \setminus \{o_w, d_w\}$$

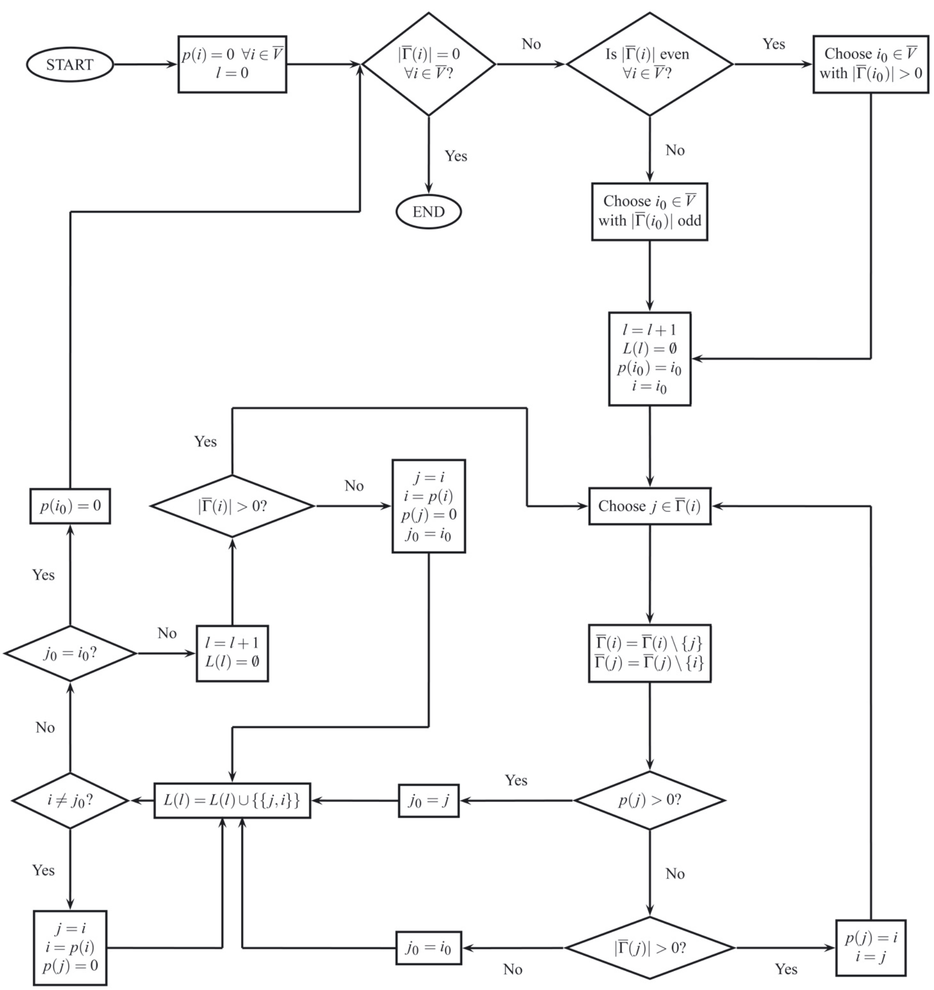
$$\gamma_i \in \{0,1\} \quad \forall i \in V$$

$$\Delta_i \in \{0,\dots,r(i)\} \quad \forall i \in V,$$

where $M_w = \sum_{\{i,j\}\in E} d_{ij} - \mu\, u_w^{\mathrm{pri}} \quad \forall w \in W$.

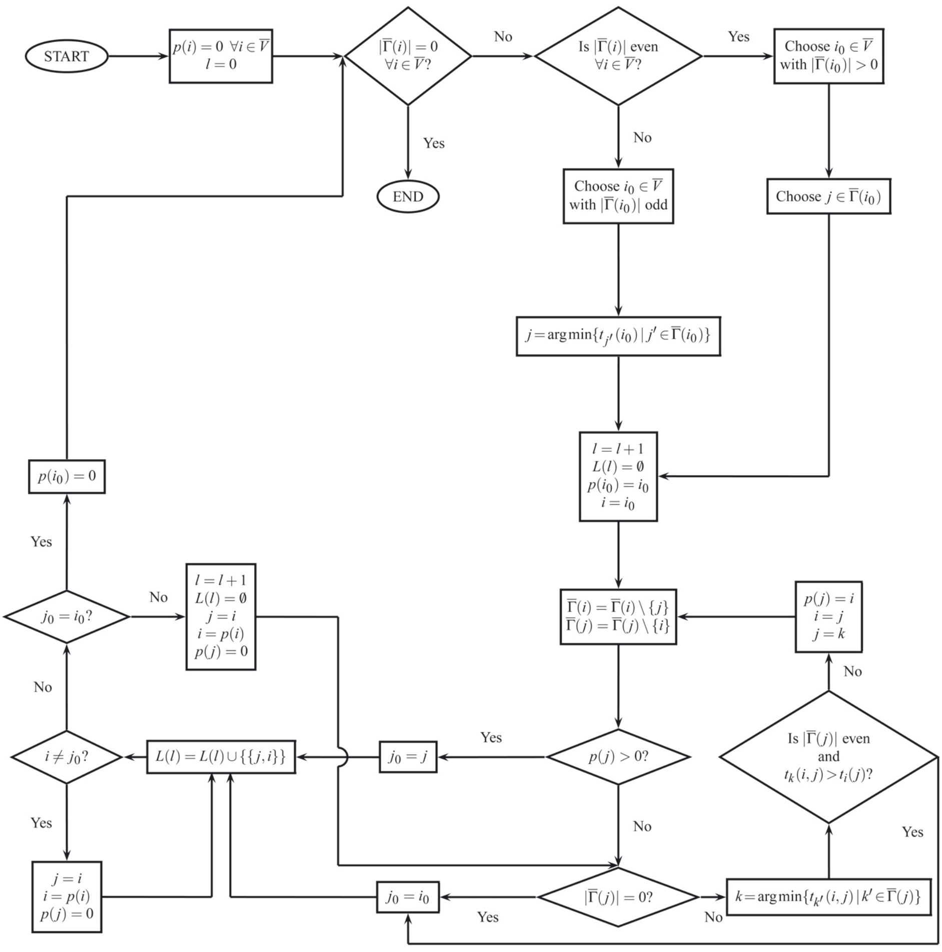
Appendix B. Flowchart for Algorithm 1

The flowchart for Algorithm 1 in Section 3.1 is as follows:



Appendix C. Flowchart for Algorithm 2

The flowchart for Algorithm 2 in Section 3.2 is as follows:



References

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

[2] M.R. Bussieck, P. Kreuzer and U.T. Zimmermann, Optimal lines for railway systems. *Eur. J. Oper. Res.* **96** (1997) 54–63.

[3] M.R. Bussieck, T. Winter and U.T. Zimmermann, Discrete optimization in public rail transport. *Math. Program.* **79** (1997) 415–444.

[4] M. Cepeda, R. Cominetti and M. Florian, A frequency-based assignment model for congested transit networks with strict capacity constraints: characterization and computation of equilibria. *Transp. Res. Part B* **40** (2006) 437–459.

[5] L.F. Escudero and S. Muñoz, An approach for solving a modification of the extended rapid transit network design problem. *Top* **17** (2009) 320–334.

[6] L.F. Escudero and S. Muñoz, *An approach for designing connected rapid transit networks considering transfers*. Technical Reports on Statistics and Decision Sciences TR09/01, Universidad Rey Juan Carlos, Móstoles, Madrid, Spain (2009).

[7] R. García, A. Garzón-Astolfi, Á. Marín, J.A. Mesa and F.A. Ortega, Analysis of the parameters of transfers in rapid transit network design, in *5th Workshop on Algorithmic Methods and Models for Optimization of Railways*, edited by L.G. Kroon and R.H. Möhring, Oasics **2**. Saarbrücken (2006).

[8] J.F. Guan, H. Yang and S.C. Wirasinghe, Simultaneous optimization of transit line configuration and passenger line assignment. *Transp. Res. Part B* **40** (2006) 885-902.

[9] V. Guihaire and J.K. Hao, Transit network design and scheduling: A global review. *Transp. Res. Part A* **42** (2008) 1251–1273.

[10] G. Laporte, Á. Marín, J.A. Mesa and F. Perea, Designing robust rapid transit networks with alternative routes. *J. Adv. Transp.* **45** (2011) 54–65.

[11] G. Laporte, J.A. Mesa, F.A. Ortega and F. Perea, Planning rapid transit networks. *Socio-Econ. Plan. Sci.* **45** (2011) 95–104.

[12] G. Laporte, J.A. Mesa, F.A. Ortega and I. Sevillano, Maximizing trip coverage in the location of a single rapid transit alignment. *Ann. Oper. Res.* **136** (2005) 49–63.

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

[14] Á. Marín, An extension to rapid transit network design problem. *Top* **15** (2007) 231–241.

[15] Á. Marín and R. García-Ródenas, Location of infrastructure in urban railway networks. *Comput. Oper. Res.* **36** (2009) 1461–1477.

[16] H. Spiess and M. Florian, Optimal strategies: A new assignment model for transit networks. *Transp. Res. Part B* **23** (1989) 83–102.

[17] J.G. Wardrop, Some theoretical aspects of road traffic research. *ICE Proceeding Engineering Divisions* **1** (1952) 325–362.