Location Routing Problems on Trees

Julián Aráoz [*1], Elena Fernández [†1,2], and Salvador Rueda [‡1]

[1] *Department of Statistics and Operation Research, Universitat Politècnica de Catalunya-BcnTech, Spain*
[2] *Barcelona Graduate School of Mathematics (BGSMath)*

Abstract

This paper addresses combined location/routing problems defined on trees, where not all vertices have to be necessarily visited. A mathematical programming formulation is presented, which has the integrality property. The formulation models a directed forest where each connected component hosts at least one open facility, which becomes the root of the component. The problems considered can also be optimally solved with ah-hoc solution algorithms. Greedy type optimal algorithms are presented for the cases when all vertices have to be visited and facilities have no setup costs. Facilities set-up costs can be handled with low-order interchanges, whose optimality check is a re-statement of the complementary slackness conditions of the proposed formulation. The general problems where not all vertices have to be necessarily visited can also be optimally solved with low-order optimal algorithms based on recursions.

1 Introduction

Location and routing problems are among core problems in combinatorial optimization. Both types of problems have received the attention of a large number of researchers and have multiple applications fields, for instance in logistics and telecommunications, where difficult optimization problems arise. It is well-known that location and routing decisions are most often closely interrelated. Indeed there are a number of location applications in which the selected locations become the depots for the routes that will serve the demand of a given set of customers. However, location problems frequently ignore the tactical or operational routing decisions and focus on the strategic location/allocation decisions. On the other hand, in routing problems it is typically assumed that the depots for the routes are set in advance, despite the enormous influence that the location of such depots may have in the design of efficient service routes. Therefore, because of the impact that joint location/routing decisions may have on the overall costs, a joint location/routing perspective is fully justified regardless of the increase in the difficulty of the resulting problem. Different types of formulations and solution techniques have been proposed for various location/routing problems (LRPs) under different modeling assumptions [2, 4, 6, 13]. The interested reader is addressed to [1, 9, 10, 11, 14] for overviews and surveys on the topic.

LRPs are usually stated on generic graphs where both location and routing decisions are associated with NP-hard optimization problems. To the best of our knowledge, the topology of the graph where problem is defined has not been exploited so far for LRPs. In contrast, the topology of the network has been exploited quite extensively to derive polynomial time solution algorithms for a number of location problems, which are NP-hard in the general case. This is particularly true with tree networks for location/allocation problems [7, 8, 18, 19, 16, 17] and for some location/covering problems as well [5, 15].

We work on an undirected connected graph that we assume is a tree, with a cost function on the edge set and a given set of users with service demand, which can be located at both the vertices and

[*] julian.araoz@upc.edu
[†] e.fernandez@upc.edu
[‡] Deceased

the edges of the graph. We asume there are no capacity constraints, neither for the demand that can be allocated to each open facility, nor for the demand that can be served by a route. We study several types of LPRs, which mainly differ from each other on the characteristics of the set of demand customers. Another possible difference among problems is whether or not open facilities incur set-up costs. In all cases, the LRPs that we study consists of (i) selecting the best location for a set of facilities at vertices of the graph, (ii) allocating each demand customer to one selected facility, and (iii) designing a set of closed routes through the given set of customers, each of them using as depot one of the selected facilities, of minimum total cost. Once the location of the facilities and the allocation of customers to the selected facilities are determined, the optimal routing problem can be decomposed into a number of smaller subproblems, one associated with each facility, which becomes the depot for its allocated customers. In all cases all optimal routes are bipaths where the edges that are used are traversed exactly twice.

In the first part of this paper we present a mathematical programming formulation for general LRPs defined on a tree. The formulation models a directed forest where each connected component hosts at least one open facility, which becomes the root of the component. We show that the formulation has the integrality property, so LRPs defined on a tree can be solved in polynomial time.

In the second part of the paper we focus on ah-hoc solution algorithms for optimally solving LRPs. We first consider simple particular cases and progressively move to the general case. Broadly speaking, when all the vertices of the tree have to be visited, LRPs reduce to finding a suitable partition of the original vertex set. In the absence of facilities set-up costs, such a problem can be solved with a simple greedy algorithm, both for demand at the vertices or the edges of the tree. Facilities set-up costs can be handled with low-order interchanges, whose optimality check is a re-statement of the complementary slackness conditions of the proposed formulation.

We also present optimal solution algorithms for general LRPs on tress where not all vertices have demand or when the graph induced by demand edges does not span the complete vertex set. Now the interpretation of a solution as a partition of the vertex set does not hold and such LRPs can no longer be solved with a greedy algorithm. While in general graphs, non-demand vertices can be *eliminated* by defining edges associated with shortest paths, this is no longer possible in the case of a tree without breaking the tree-structure, unless all non-demand vertices have degree two. From an algorithmic point of view, the fact that it is not known a priori what non-demand vertices will be *visited* in an optimal solution, makes these LRPs substantially more difficult than when all vertices must be necessarily visited. As we will see, a dynamic programming algorithm can optimally solve LRPs of this type in low polynomial time.

The paper is structured as follows. Section 2 introduces the notation, formally defines the problem, and discusses some of its properties. Section (3) presents polyhedral representation for LRPs and proves that it has the integrality property. In Section (4) we present ad-hoc exact algorithms for several particular cases, as well as for the general case. In particular Section 4.1 deals with LRPs without set-up costs, in which all vertices have to bee visited, both for the case of demand at the vertices or the edges of the tree. The optimality conditions for the cases with set-up cost are given in Section 4.2. Section 4.3 studies LRPs in which all the vertices of the original graph do not necessarily have to be visited, and gives a polynomial time dynamic programming algorithm for these problems. The paper ends in Section 5 with some conclusions and avenues for research.

2 Preliminaries

We use standard notation and definitions for graphs. Next we recall some basic notation and give the definition of the problems that we study and some of their properties.

Let $G = (V, E)$ denote an undirected tree with vertex set V, $|V| = n$, and edge set E, $|E| = m$.

- For any subgraph H of G we denote by $V(H)$ the vertex set of H and by $E(H)$ the edge set of H, in particular, $V = V(G),\ E = E(G)$.

- For a given vertex set $S \subset V$, we denote by $E(S) = \{e \in E | e = (u, v), u, v \in S\}$, the set of edges with both vertices in S. For any vertex set $S \subseteq V$ we denote by $G[S] = (S, E(S))$ the induced graph.
- For any edge set $F \subseteq E$, the set of vertices incident to edges in F is denoted by $V(F)$. The subgraph induced by F is $G(F) = (V(F), F)$.
- For any nonempty subset of vertices $S \subset V$, $\delta(S) = \{e \in E | e = (u, v), u \in S, v \in V \setminus S\} = \delta(V \setminus S)$, denotes the set of edges in the cut between S and $V \setminus S$.
  For a singleton $\{v\} \subset V$ we do not use the brackets and simply write $\delta(v) \equiv \delta(\{v\})$ to denote the set of edges $(v, u), v \neq u$, incident to v.

We use the standard compact notation $f(A) \equiv \sum_{e \in A} f_e$ when $A \subseteq E$, and f is a vector or a function defined on E.

In the reminder of this paper we assume that G is a tree that we denote by T so $|E| = n - 1$.

2.1 Location-Routing Problem on a tree T

Consider a tree $T = (V, E)$ with a cost function on the edges $c : E \to \mathbb{R}_+$. For a reason that will become evident next, we assume that c_e represents the cost for traversing twice edge $e \in E$. Let also $D \subseteq V$ and $R \subseteq E$ be two given *demand sets* of vertices and edges, respectively, and $L = D \cup V(R)$ the set of *potential locations* for facilities with a setup cost function $f : L \to \mathbb{R}_+ \cup \{0\}$. Finally, let $p > 0$ be a given integer number.
A feasible solution to the p-Location-Routing Problem on T with vertex demand D and edge demand R, consists of a set of p open facilities, $O \subseteq L$, $|O| = p$, together with a set of routes, each of them rooted at some selected facility, visiting all the demand vertices and traversing all the demand edges. Since T is a tree, in any feasible solution a route consists of a bipath replicating all the edges that it traverses. Taking into account the above definition of the costs of the edges, the cost of a solution is thus the sum of the setup costs of the open facilities, plus the sum of the costs of the edges of the routes.
Any solution is associated with a forest $T^k = (V^k, E^k)$, $k \in K$, $|K| \leq p$, with $V^k \subseteq V$, $|V^k| \geq 1$, and $E^k = E(V^k)$, where each subtree T^k hosts at least an open facility. The subtrees T^k, $k \in K$ will also be called *S-components*. For $k \in K$, $O(k) \subseteq V^k \cap L$ denotes the set of open facilities in S-component T^k, and $o(k)$ an arbitrarily selected lowest setup cost vertex in $O(k)$, i.e. $o(k) \in \arg\min\{f_i \mid i \in O(k)\}$. We will indistinctively use $S = \{T^k\}_{k \in K}$ and $S = (O^S, F^S)$, where $O^S = \bigcup_{k \in K} O(k)$ is the set of open facilities, and $F^S = \bigcup_{k \in K} E^k$ the set of edges traversed in the routes (note that $|F^S| = n - |K|$). When the solution S is clear from the context will drop the superindex.

A solution is feasible if, in addition, it satisfies the following conditions:

- Each demand vertex $v \in D$ is a vertex of one of the S-components, i.e. $v \in V^k$ for some $k \in K$.
- Each demand edge $e \in R$ is an edge of one of the S-components, i.e. $e \in E^k$ for some $k \in K$.

Definition 1 The p-Location-Routing Problem on T with vertex demand D and edge demand R, denoted by (T, f, c, p, D, R), is to find a feasible solution of minimum total cost.

The version of the problem in which at most p facilities must be opened is denoted by $(T, f, c, p^{\leq}, D, R)$. The more general version in which the maximum number of facilities is not set is, thus, $(T, f, c, n^{\leq}, D, R)$. Since an optimal solution to $(T, f, c, p^{\leq}, D, R)$ can be obtained by identifying the best solution among the optimal solutions to problems (T, f, c, r, D, R), with $1 \leq r \leq p$, an exact algorithm for (T, f, c, p, D, R) also gives an exact algorithm for $(T, f, c, p^{\leq}, D, R)$, whose complexity increases by a factor of p with respect to the complexity of the algorithm for (T, f, c, p, D, R).

In the following we focus on the p-Location-Routing Problem on T, (T, f, c, p, D, R).

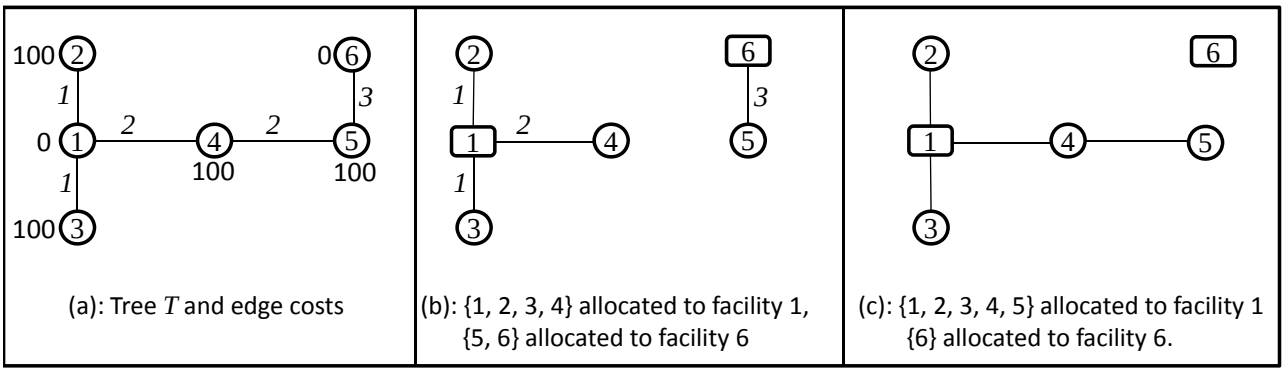


Figure 1: Optimal location/allocation and location/routing on a tree

2.2 Properties and particular cases

An interesting particular case of (T, f, c, p, D, R) arises when demand is located only at the vertices of the tree and all vertices have to be visited, i.e. $D = V$, $R = \emptyset$ and $L = V$. Then, an optimal solution to $(T, f, c, p, V, \emptyset)$ exists with exactly p S-components. Indeed, if the forest associated with a feasible solution has less than p components, removing any edge from the solution produces a feasible solution with a smaller cost. Hence, optimal solutions to these problems have $|K| = p$ or, equivalently, $n - p$ edges. Since $|K| = p$ implies that $|O(k)| = 1$ for all $k \in K$, in optimal solutions to $(T, f, c, p, V, \emptyset)$ each V^k contains exactly the customers allocated to facility $o(k)$. Note that it is possible that $E^k = \emptyset$ for some $k \in K$; in this case, the associated V^k is a singleton.

In contrast to location/allocation problems on trees, in an optimal solution to $(T, f, c, p, V, \emptyset)$, a vertex is not necessarily allocated to its closest open facility. Figure 1 illustrates this observation. Figure 1(a) depicts the original tree; values next to the edges indicate their costs, and the values next to the vertices the facilities set-up costs. In Figures 1(b)-(c) we assume that $p = 2$. It is easy to see that the optimal location decision for both the location/allocation and the location/routing problems is to locate the two facilities at vertices 1 and 6, respectively. As depicted in Figure 1(b), the optimal allocation in the location/allocation problem assigns {1, 2, 3, 4} to the component facility 1, and {5, 6} to the component of facility 6. The location/allocation objective function value of this solution is (1+1+2)+3=7, which coincides with the objective function value for the location/routing problem. However, the optimal location/routing decision, which is depicted in Figure 1(c), allocates {1, 2, 3, 4, 5} to the component of facility 1, and {6} to the component of facility 6, with an overall routing cost of $(1 + 1 + 2 + 2) = 6$. The (non-optimal) location/allocation objective function value of this solution is 1+1+2+4=8 (since the allocation cost of vertex 5 to facility 1 is 2+2).

When demand is located at the edges, i.e., $R \neq \emptyset$, the number of trees in optimal solutions to (T, f, c, p, D, R) is not known in advance, even if $D = V$. This is illustrated in the example of Figure 7, with two demand components induced by demand edges and $p = 2$, whose optimal solution, depicted in Figure 7.(c), consists of a unique S-component $T^1 = (V^1, E^1)$ with $V^1 = V$, $E^1 = E$ and $O(1) = \{5, 6\}$.

Hence, it is possible that $|K| < p$ in some optimal solution. In that case, since exactly p facilities have to be located, $|O(k)| > 1$ for some $k \in K$. Taking into account the minimization objective function, once the S-components are known, the sets $O(k)$ indicating the optimal locations for the facilities can be found in a simple way: for each $k \in K$, one facility is first located at $o(k)$, a lowest setup cost potential facility of S-component k. The remaining $p - |K|$ facilities are then located in a greedy fashion by increasing values of their setup costs.

The main difficulty of (T, f, c, p, D, R) arises when not all vertices have to be necessarily visited, i.e., $V \setminus L \neq \emptyset$. In the case of general graphs, non-demand vertices can be eliminated by defining edges associated with shortest paths if the intermediate nodes of such paths use some non-demand vertex.

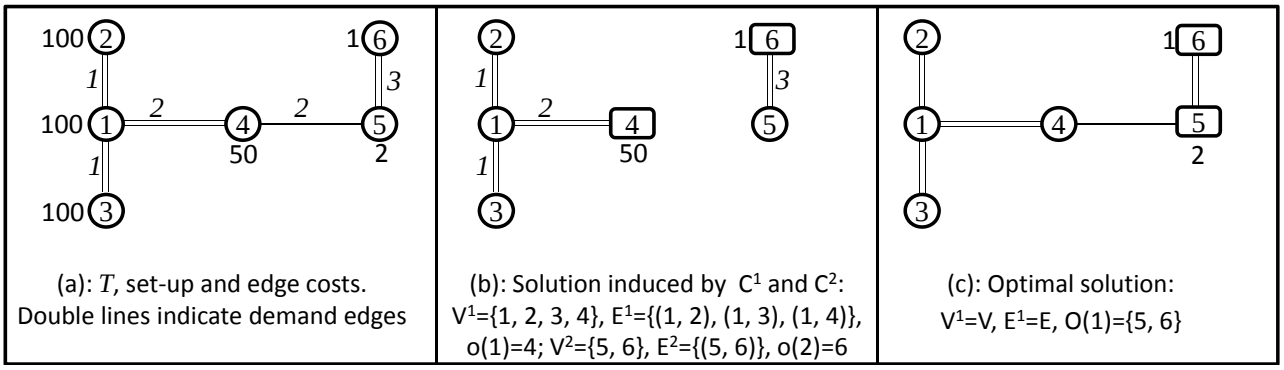


Figure 2: p-ELRP example with $r = p$ and optimal solution with edge in $E \setminus R$

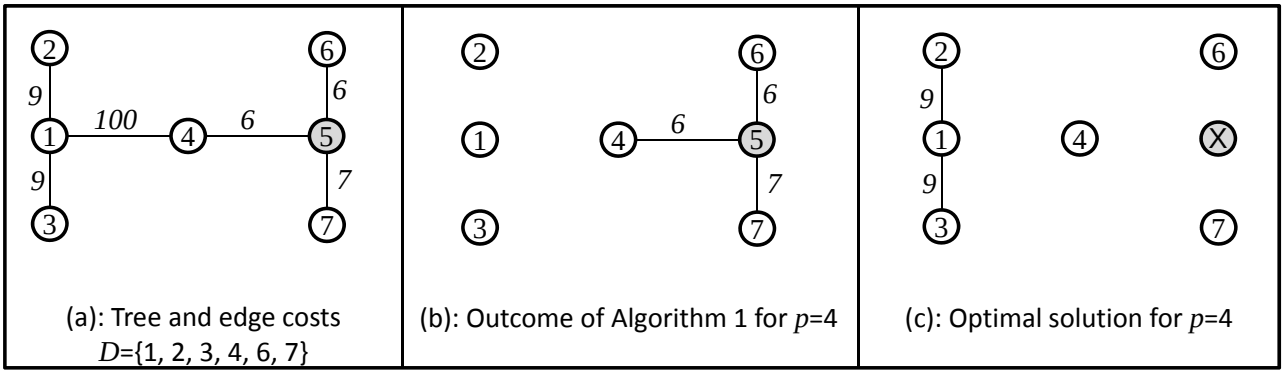


Figure 3: Example of instance with non-demand vertices

This approach is however not valid in the case of a tree without breaking the tree-structure of the original graph, except for non-demand vertices with degree two.

In the case of trees, allowing for vertices that do not necessarily have to be covered in feasible solutions increases notably the difficulty of the associated location routing problem. The main reason is that a priori one does not know the vertices in $V \setminus L$ that will be covered in an optimal solution. In other words, one does not know in advance, what edges incident to non-demand vertices should be considered as potential candidates for an optimal solution. This is illustrated by the example of Figure 8 with $V = \{1, ..., 7\}$, $D = V \setminus \{5\}$ and $R = \emptyset$, so vertex 5 does not necessarily have to be visited. Figure 8(b) shows the best solution (of value 19) that visits all the vertices, for $p = 4$, which is non-optimal. The optimal solution to this instance is depicted in Figure 8(c); it has a value of 18 and does not cover vertex 5. Observe, however, that if in this example we modify slightly the edge costs to $c_{45} = c_{56} = 5$ and $c_{57} = 6$, all other costs remaining unchanged, then the solution depicted in Figure 8 (b), is optimal. The value of this solution for the instance with modified costs is 16, and it covers the non-demand vertex 5. Note also that all distances between pairs of demand vertices in the S-component {4, 5, 6, 7}, are greater than 9, which is the length of two tree edges connecting pairs of demand vertices, that do not make part of this optimal solution. This may seem counterintuitive at first sight.

The fact that when $V \setminus L \neq \emptyset$ the set of vertices that must be visited is not known in advance, justifies that we exclude such vertices as potential locations for the facilities. Otherwise, without additional constraints, feasible solutions could be meaningless and have some S-component consisting just of one non-demand vertex but no edges, i.e. $V^k\{u\} \subset V \setminus L$ and $E^k = \emptyset$.

In Section 4 we will give low order polynomial-time solution algorithms of increasing difficulty for several particular cases of (T, f, c, p, D, R) as well as for the general case. First, however, we study the

polyhedral representation for the general case of (T, f, c, p, D, R).

3 An integral polyhedral representation for (T, f, c, p, D, R)

In this section we present a mathematical programming formulation for (T, f, c, p, D, R), which has the integrality property. The formulation builds a directed forest where p facilities are located at vertices of $L = D \cup V(R)$, and each connected component hosts at least one open facility. The arcs of the forest are directed towards the vertices where facilities are not located. The reason for using a directed formulation instead of an undirected one is that, as we will see later on, the undirected version does not have the integrality property.
Let A denote the set containing two arcs associated with each edge of E, one in each direction. That is, $A = \{(u,v),(v,u) \mid (u,v) \in E\}$. The in- and out- *dicuts* associated with a vertex subset $S \subset V$, are denoted as $\delta^+(S) = \{(u,v) \in A \mid u \in S, v \in V \setminus S\}$ and $\delta^-(S) = \{(u,v) \in A \mid u \in V \setminus S, v \in S\}$, respectively.

The formulation uses two sets of decision variables, one to represent the vertices where facilities are located and another one to represent the arcs of the forest. For $i \in L$ let y_i a binary variable, which takes the value one if and only if a facility is located at vertex i. In addition, each arc $(u,v) \in A$ is associated with a binary decision variable x_{uv}, which takes the value 1 if and only if no facility is located at v and the solution contains a dipath from an open facility to v that traverses arc (u,v). The formulation is as follows:

$$LR: \quad \min \sum_{i \in L} f_i y_i + \sum_{(u,v) \in A} c_{uv} x_{uv} \tag{1a}$$

$$s.t. \quad \sum_{i \in L} y_i = p \tag{1b}$$

$$y_v + \sum_{(u,v) \in \delta^-(v)} x_{uv} = 1 \qquad v \in D \setminus V(R) \tag{1c}$$

$$y_v + \sum_{(u,v) \in \delta^-(v)} x_{uv} \geq 1 \qquad v \in V(R) \tag{1d}$$

$$\sum_{(u,v) \in \delta^-(v)} x_{uv} \geq x_{vv'} \qquad v \in V \setminus L, (v,v') \in \delta^+(v) \tag{1e}$$

$$x_{uv} + x_{vu} \leq 1 \qquad (u,v) \in E \setminus R \tag{1f}$$

$$x_{uv} + x_{vu} = 1 \qquad (u,v) \in R \tag{1g}$$

$$x_{uv} \in \{0,1\} \qquad (u,v) \in A \tag{1h}$$

$$y_i \in \{0,1\} \qquad i \in D \tag{1i}$$

Constraints (1b) impose that exactly p facilities are selected. Constraints (1c)-(1e) guarantee that the connected components induced by the vertices that must be visited (i.e. vertices in L) contain at least one open facility. For this, they impose that at least one arc points towards each vertex of L containing no facility. By Constraints (1c), one single arc will point towards each vertex of $D \setminus V(R)$ where no facility is located when a connected component hosts one single open facility. However, as shown before (see the example of Figure 7) when $R \neq \emptyset$ there may be optimal solutions where a connected component contains more than one open facility. In such cases when assigning directions to the arcs as indicated in the definition of the decision variables, and because of constraints (1g), there can be several arcs pointing towards an end-vertex of some demand arc, as indicated by (1d). Otherwise, for singletons defined by demand vertices not involved in demand arcs, these *connectivity* constraints must hold as equalities, as indicated in (1c). Furthermore, Constraints (1e) prevent connected components visiting non-demand vertices in $V \setminus L$ and containing no facility, by imposing that if some arc (v,v) emanating from a vertex $v \in V \setminus L$ is activated, then some arc pointing towards the non-demand vertex v must also be used. Finally, Constraints (1f)-(1g) ensure that no edge is traversed in both

directions and that all demand edges are traversed. The objective function computes the setup costs of the open facilities plus the routing costs of the used arcs.

Remark 1 *As mentioned, the undirected version of formulation* (1a)-(1i) *in which variable* x_{uv}*, takes the value 1 if and only if edge* $(u, v) \in E$ *is used does not have the integrality property. A simple counterexample for a case with* $D = V$ *and* $R = \emptyset$ *is shown in Figure 4.*

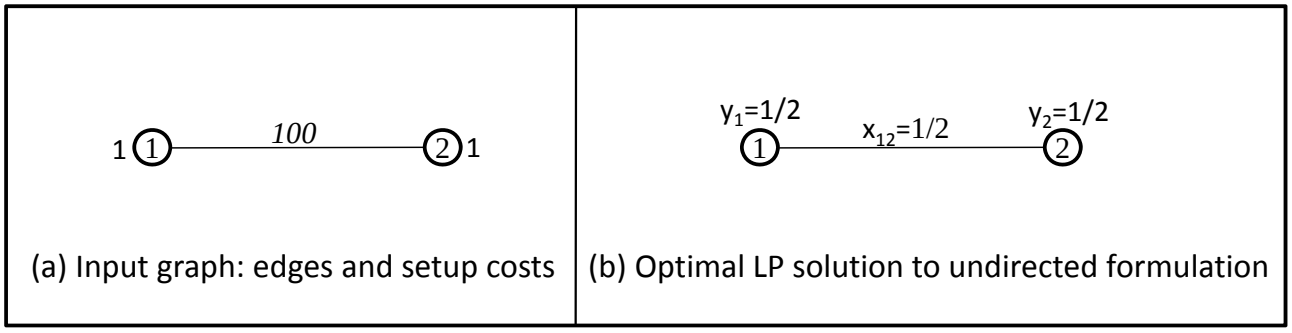


Figure 4: Example that undirected formulation has no integrality property

As we will see, the coefficients matrix of constraints (1b)-(1g) is totally unimodular (TU). First we prove that when all vertices must be visited, i.e. $V = L$, the resulting coefficients matrix is TU.

Theorem 1 *The coefficients matrix of constraints* (1b)-(1d) *and* (1f)-(1g) *is TU.*

Proof: We can partition the coefficients matrix of constraints (1b)-(1d) and (1f)-(1g) in two blocks of rows, one with constraints (1b), (1f), and (1g), and another one with constraints (1c) and (1d). It is clear that the only non-zero coefficients of each variable, y_i or x_{uv}, are a "+1" in each of the two blocks. It thus follows that the coefficients matrix is TU (see, for instance, Proposition 3.2, p. 39 in [20].)

It is easy to see that the sufficient condition for TU matrices used in Theorem 1 no longer applies to the general case where the coefficient matrix will have rows associated with constraints (1e), since some columns may have more than two non-zero entries. Note that an essential difference between constraints (1e) and constraints (1c)-(1d) and is that each non-demand vertex $v \in V \setminus L$ generates $|\delta^+(v)|$ constraints (1e), one for each arc $(v, v') \in \delta^+(v)$ emanating from v, instead of just one single constraint, as it happens with the constraints (1c)-(1d) associated with demand vertices $v \in L$. Moreover, the constraints (1e) associated with a given $v \in V \setminus L$ differ from each other in only one variable, namely the one associated with the selected arc $(v, v') \in \delta^+(v)$, but the term $y_v + \sum_{(u,v)\in\delta^-(v)} x_{uv}$ remains the same for all the constraints of the group.
Hence, for the general case we will use the following characterization of TU matrices (see, for instance, Theorem 2.7, p. 39 in [12]):
A matrix $A = (a_{ij})_{i\in I, j\in J}$ is TU if and only if for every subset of rows $Q \subseteq I$ there exists a partition Q_1 and Q_2 of Q such that for each column $j \in J$ of it holds that

$$\left| \sum_{i\in Q_1} a_{ij} - \sum_{i\in Q_2} a_{ij} \right| \leq 1. \tag{2}$$

In order to facilitate the required analysis, we first classify variables in the following types and identify the non-zero coefficients of each type of the variables:

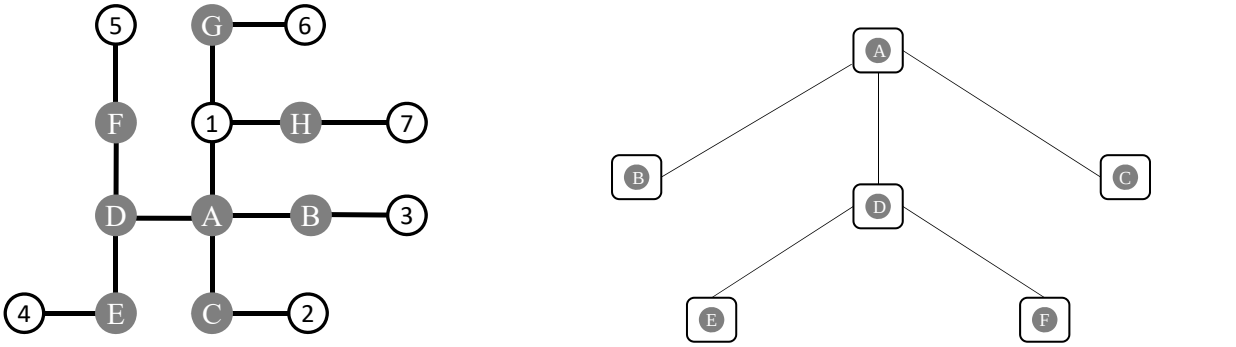
(*i*) Variables, y_i, $i \in L$ have a non-zero coefficient ("+1") in constraint (1b) and another "+1" in the corresponding constraint (1c)-(1d).

(*ii*) Variables x_{uv} associated with arcs incident with demand vertices, i.e. $(u, v) \in E(L)$. The only non-zero coefficients of x_{uv} are a "+1" in the constraint (1c)-(1d) associated with v and another "+1" in the constraint (1f)-(1g) associated with edge (u, v). This type of arcs, and their associated variables, will be referred to as *demand/demand* or simply DD.

(*iii*) Variables x_{uv} associated with arcs $(u, v) \in \delta^-(L)$ with $u \in V \setminus L$. These variables also have a coefficient "+1" in the constraint (1c)-(1d) associated with v and another "+1" in the constraint (1f)-(1g) associated with edge (u, v). Furthermore, since $u \in V \setminus L$, variable x_{uv} has a coefficient "-1" in the constraint (1e) associated with non-demand vertex u and arc $(u, v) \in \delta^+(u)$. This type of arcs, and their associated variables, will be referred to as *non-demand/demand* or simply ND.

(*iv*) Variables x_{uv} associated with arcs $(u, v) \in \delta^-(V \setminus L)$. These variables have the "+1" coefficient in the constraint (1f)-(1g) associated with edge (u, v), but do not intervene in constraints (1c)-(1d). Instead, they have a "+1" in the $|\delta^+(v)|$ constraints (1e) associated with v (for the different arcs $(v, v') \in \delta^+(v)$). Furthermore, when $u \in V \setminus L$ variable x_{uv} has a coefficient "-1" in the constraint (1e) associated with u and arc $(u, v) \in \delta^+(u)$. Arcs of this type, and their associated variables, will be referred to as *demand/non-demand* (or just DN) when $u \in L$, and as *non-demand/non-demand* or just NN when $u \in V \setminus L$.

Table 1 illustrates the structure of the coefficients matrix for the small example of Figure 5a, where we emphasize the role of variables related to non-demand vertices. In Figure 5a demand vertices are labeled with numbers and non-demand grey vertices are labeled with letters.



(a) Input graph (b) Graph of pairings for labeling of constraints (1e)

Figure 5: Input graph and pairings for labeling Example 1

Below we prove that the coefficients matrix of (1b)-(1g) remains TU even if there are non-demand vertices and the matrix contains a block of rows associated with constraints (1e).

Theorem 2 *The coefficients matrix of constraints* (1b)-(1g) *is totally unimodular.*

Proof:

Let us assume that $V \setminus L \neq \emptyset$, since otherwise the statement reduces to Theorem 1.

First we prove that (2) holds for the full set of row indices, i.e. for $Q = I$.
In order to define the partition of the full set of rows of (1b)-(1g) we apply a two-coloring: each row will be labeled *black* (B) or *white* (W). Row (1b) as well as all rows (1f)-(1g) are labeled B. All rows (1c) and (1d) are labeled W. Indeed, this labeling already guarantees that condition (2) holds for all the columns associated with DD arcs.

	v	(v,v')	A1	G1	H1	C2	B3	E4	F5	G6	H7	1A	BA	CA	DA	3B	AB	2C	AC	AD	ED	FD	4E	DE	5F	DF	1G	6G	1H	7H	1	2	3	4	5	6	7
(1b)																															1	1	1	1	1	1	1
	1		1	1	1																										1						
	2					1																										1					
	3						1																									1					
(1c)	4							1																								1				
	5								1																							1				
	6									1																						1			
	7										1																						1		
	A	(A,1)	-1									1	1	1	1																				
	A	(A,B)										1	1	1	1		-1																		
	A	(A,C)										1	1	1	1				-1																
	v	(A,D)										1	1	1	1					-1															
	B	(B,3)					-1									1	1																		
	B	(B,A)											-1			1	1																		
	C	(C,2)				-1												1	1																
	C	(C,A)												-1				1	1																
	D	(D,A)													-1					1	1	1													
(1e)	D	(D,E)																		1	1	1		-1											
	D	(D,F)																		1	1	1				-1									
	E	(E,4)						-1																1	1										
	E	(E,D)																				-1		1	1										
	F	(F,5)							-1																	1	1								
	F	(F,D)																					-1			1	1								
	G	(G,1)		-1																								1	1						
	G	(G,6)								-1																		1	1						
	H	(H,1)			-1						-1																			1	1				
	H	(H,7)										-1																		1	1				
			1									1																							
				1																								1							
					1																									1					
						1											1																		
							1									1																			
								1																1											
									1																	1									
(1g)										1																	1								
											1																			1					
															1			1																	
														1					1																
																1					1														
																				1				1											
																							1			1									

Table 1: Coefficient matrix for the graph of Figure 5a

Moreover, the above labeling also guarantees that condition (2) holds for the columns associated with arcs (u, v) of type ND since, as mentioned, such columns consist of a +1 in the row (1c)-(1d) corresponding to v (labeled W), another one in the row (1f)-(1g) associated with edge (u, v) (labeled B), plus a "-1" in the row (1e) corresponding to non-demand vertex u. Thus, condition (2) holds, independently of the color label assigned to the involved row (1e).

Hence, the *difficult* columns are the ones associated with arcs of types DN and NN, entering non-demand vertices, since such variables will have non-zero coefficients in more that one of the (1e) rows. Note that in both cases the corresponding columns have one single non-zero element outside the set of rows (1e), and in both cases the row of the additional non-zero coefficient is labeled B. Hence what we do is to split the rows (1e) in two parts of *approximately* the same size, and assign one of the two labels to all the rows in one part and the other label to the rows of the other part, according to certain rules, since otherwise condition (2) will not hold.

Since each vertex $v \in V \setminus L$ defines a block of $|\delta^+(v)|$ constraints (1e) and all the rows of the block contain a +1 in all the columns associated with arcs $(v', v) \in \delta^+(v)$, we need that, for each *difficult* column, the number of B and W labels in the rows of its non-zero entries is either the same, or there is exactly one more W label than B labels. For DN arcs the above guideline would be enough to satisfy condition (2), since the only (1e) rows where variable $x_{vv'}$ has a non-zero coefficient (a "+1") are the ones associated with $v \in V \setminus L$. However, for NN arcs additional precautions must be taken, since $x_{vv'}$ will also have a non-zero coefficient (a "-1") in the (1e) row associated with arc (v', v), which is also NN. Hence, in order to satisfy condition (2) we assign the same color label to the two involved rows (the one associated with vertex v and arc (v, v'), and the one associated with vertex v' and arc (v', v)).
Observe that such a *pairing* is indeed possible. On the one hand, each row (1e) has to be *paired* with at most a different (1e) row. On the other hand, the fact that the input graph is a tree guarantees that no *conflict* will arise in this pairing process.

Once the pairings have been set, it only remains to assign color labels within the block of rows

label	v	(v,v')	A1	G1	H1	C2	B3	E4	F5	G6	H7	1A	BA	CA	DA	3B	AB	2C	AC	AD	ED	FD	4E	DE	5F	DF	1G	6G	1H	7H
W	A	(A,1)	-1									1	1	1	1															
B	A	(A,B)										1	1	1	1		-1													
B	A	(A,C)										1	1	1	1				-1											
W	A	(A,D)										1	1	1	1					-1										
W	B	(B,3)					-1									1	1													
B	B	(B,A)											-1			1	1													
W	C	(C,2)				-1												1	1											
B	C	(C,A)												-1				1	1											
W	D	(D,A)													-1					1	1	1								
W	D	(D,E)																		1	1	1		-1						
B	D	(D,F)																		1	1	1			-1					
B	E	(E,4)						-1															1	1						
W	E	(E,D)																			-1		1	1						
W	F	(F,5)							-1																1	1				
B	F	(F,D)																				-1			1	1				
B	G	(G,1)		-1																							1	1		
W	G	(G,6)								-1																	1	1		
B	H	(H,1)			-1						-1																		1	1
W	H	(H,7)										-1																	1	1

Table 2: A possible labeling of rows (1e) for the graph of Figure 5a

(1e) associated with each non demand vertex v. When $|\delta^+(v)|$ is even, half of the (1e) rows are labeled B and the other half W, ensuring that the *pairing* rules are followed. Hence, for each *difficult* column, the difference between the number of B and W labels in the rows of its non-zero entries in the complete set system (1b)-(1g) will be exactly one (corresponding to the B label of the involved row (1f)-(1g)). When $|\delta^+(v)|$ is odd we proceed quite similarly. Now we cannot balance exactly the number of B and W labels within the block of rows (1e), so we assign one more W label than B labels. This compensates with the B labels of the rows (1f)-(1g).

Indeed a similar coloring process can be applied to any subset of rows of the full set of rows of (1b)-(1g). Hence, condition (2) also holds for any subset of rows $Q \subseteq I$, so the result follows.

Example 1 *Figure 5b shows the tree of pairings for the labeling of rows 1e of matrix 1 for the input graph of Figure 5a. In Table* (2) *we show a possible labeling satisfying condition 2 for this example. Indeed such a labeling need not be unique.*

4 Solution algorithms

In this section we present optimal solution algorithms for the (T, f, c, p, D, R). We start with simple greedy algorithms for some particular cases when all the vertices have to be visited, i.e., $D = V$, and progressively increase the difficulty and generality of the considered problems and proposed algorithms. We conclude with a dynamic programming algorithm to solve the general case case of (T, f, c, p, D, R) when $L = D \cup V(R) \subseteq V$, but both sets do not necessarily coincide.

4.1 Location/Routing on Trees without Set-up Costs

In this section we address $(T, 0, c, p, V, R)$ and we further assume there are no setup costs, i.e. $f_i = 0$ for all $i \in V$. Because exactly p facilities are located, $(T, 0, c, p, V, R)$ also solves the case when all facilities have the same set-up cost. We first address the *Vertex Location/Routing* case in which demand is located only at the vertices of the graph (i.e. $D = V$, $R = \emptyset$). For this problem we give a simple greedy algorithm and prove its optimality. Then we see that the *Edge Location/Routing* case when demand is located only at the edges (i.e. $D = \emptyset$, $V(R) = V$) can be reduced to *Vertex Location/Routing* and finally we indicate how to handle the general case with both $D \neq \emptyset$ and $R \neq \emptyset$. The more general problem in which all vertices do not have to be visited necessarily, i.e. $L \subset V$, will be studied in Section 4.3.

4.1.1 Vertex Location/Routing: p-VLRP

As mentioned, when $D = V$ and $R = \emptyset$ an optimal solution to $(T, 0, c, p, V, \emptyset)$ exists with exactly p S-components, where $|O(k)| = 1$ for all $k \in K$ and V^k contains exactly the customers allocated

to facility $o(k)$. Hence, we look for solutions with $|K| = p$ or, equivalently, with $n - p$ edges, i.e. $|F| = n - p$.
By analogy with location/allocation problems, this particular case will be referred to as p-VLRP.

Since any subset $F \subset E$ with $|F| = n - p$ induces a solution, feasible solutions to p-VLRP can be obtained by simply removing $p - 1$ edges from E. Moreover, since there are no setup costs, if the removed edges are the $p - 1$ largest ones, the resulting solution will be optimal. This is the idea of Algorithm 1, which produces a solution by removing edges in a greedy fashion by decreasing values of their costs.

Algorithm 1

0. $F^0 \leftarrow E$; $t \leftarrow 1$
1. $e^t \leftarrow \arg\max\{c_e \mid e \in F^{t-1}\}$ (with ties arbitrarily broken).
2. $F^t \leftarrow F^{t-1} \setminus \{e^t\}$.
3. $t \leftarrow t + 1$ if $(t < p)$ goto 1.
4. end

It is clear that the solution produced by the above algorithm is optimal to p-VLRP. The complexity of Algorithm 1 is $\mathcal{O}(pn)$. An alternative optimal algorithm for the p-VLRP is to start with an empty solution and to iteratively incorporate $n-p$ edges greedily selected by increasing values of their costs. Our preference for Algorithm 1 is justified, as the complexity of such a constructive algorithm is $\mathcal{O}((n - p)n)$, which, in practice, is expected to be larger than $\mathcal{O}(pn)$. Indeed, if initially we sorted edges by non-increasing costs, the complexity of Algorithm 1 would be $\mathcal{O}(nlogn)$, which, in practice, is expected to be larger than $\mathcal{O}(pn)$.

4.1.2 Edge Location Routing on trees

Below we study the $(T, 0, c, p, \emptyset, R)$ under the assumption that $V(R) = V$. This problem is referred to as p-ELRP. Observe that now any feasible solution must traverse all edges of R. Thus $c(R)$ is a fixed minimum traveling cost that will be incurred by all solutions. Let R^i, $i = 1, \ldots, r$, denote the connected components induced by R. Since there are no setup costs, when $r \leq p$ the solution where each R^i, $i \in \{1, \ldots, r\}$ defines an S-component, i.e., $T^k = G(R^k)$, $k \in \{1, \ldots, r\}$, is optimal. When $r < p$ the number of S-components defining this solution is smaller than p, so some T^k will host more than one facility. The location of the facilities can be arbitrarily selected, provided that $|O(k)| \geq 1$ for all $k \in K$, as all setup costs are zero.

In order to obtain a feasible solution when $r > p$, some R^is have to be connected with each other until the number of S-components is reduced to p. For this, $r - p$ edges from $E \setminus R$ must be added to the solution. Consider the graph $T^R = (V^R, E^R)$ where V^R contains a vertex associated with each R^i, $i \in \{1, \ldots, r\}$, and $E^R = E \setminus R$ are the non-demand edges of the tree T. Note that T^R is a tree with $|V^R| = r$ and $|E^R| = n - 1 - |R|$. Consider also the cost vector c^R, where each non-demand edge $e \in E^R$ inherits its cost from T ($c^R_e = c_e$). It is clear that any optimal solution to the p-VLR $(T^R, 0, c^R, p, V^R, \emptyset)$ is also an optimal solution to $(T, 0, c, p, \emptyset, R)$. Therefore, Algorithm 2, which is the adaptation of Algorithm 1 to $(T^R, 0, c^R, p, V^R, \emptyset)$ is an optimal algorithm for the Edge Location/Routing problem.

Algorithm 2

0. $F^0 \leftarrow E \setminus R$; $t \leftarrow 1$
1. $e^t \leftarrow \arg\max\{c_e \mid e \in F^{t-1}\}$ (with ties arbitrarily broken).
2. $F^t \leftarrow F^{t-1} \setminus \{e^t\}$.
3. $t \leftarrow t + 1$ if $(t < p)$ goto 1.

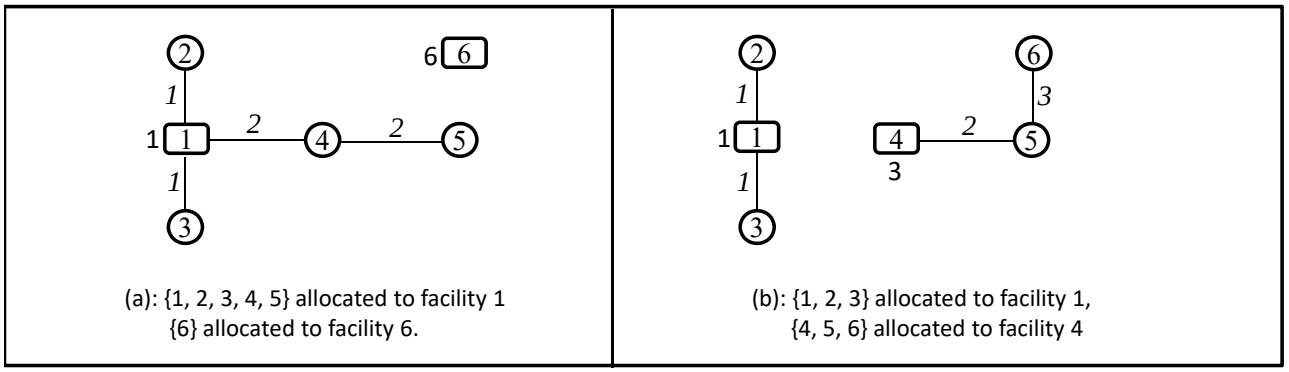


Figure 6: Outcome of Algorithm 1 not optimal for $(T, f, c, p, V, \emptyset)$

4. $F^t \leftarrow F^{t-1} \cup R$.

5. end

The complexity of Algorithm 2 is $\mathcal{O}(pn)$. Depending on the number of components induced by the demand edges (r), $r - p + 1$ can be smaller than $p - 1$. In this case, the constructive version of the greedy algorithm would be preferred to Algorithm 2 for $(T^R, 0, c^R, p, V^R, \emptyset)$, since it would request a smaller number of iterations. In this case the initial set of edges will consist of all demand edges, i.e. $E^0 \leftarrow R$.

The case $(T, 0, c, p, D, R)$ with $D \neq \emptyset$ and $R \neq \emptyset$ and $D \cup V(R) = V$ can be solved as follows. If $D \subseteq V(R)$, then $(T, 0, c, p, D, R)$ reduces to $(T, 0, c, p, \emptyset, R)$. Thus, let us assume that $V \setminus V(R) \subseteq D$ and let $s = |V \setminus V(R)|$. Consider the tree $T^R = (V^R, E^R)$ where now V^R contains a vertex associated with each vertex $i \in V \setminus V(R)$ and a vertex associated with each R^i, $i \in \{1, \ldots, r\}$, and $E^R = E \setminus R$ are the non-demand edges of the tree T. Now $|V^R| = s + r$ and $|E^R| = (n - 1) - |R|$. Consider as before the cost vector c^R, where each non-demand edge $e \in E^R$ inherits its cost from T ($c^R_e = c_e$). Since any optimal solution to the p-VLR $(T^R, 0, c^R, p, V^R, \emptyset)$ is also optimal to $(T, 0, c, p, D, R)$, the general $(T, 0, c, p, D, R)$ can also be solved with Algorithm 2. The following result thus follows:

Corollary 1 *$(T, 0, c, p, D, R)$ can be solved in polynomial time.*

4.2 Location/Routing on Trees with Set-up Costs

4.2.1 p-VLRP with setup costs

Next we address the case $(T, f, c, p, V, \emptyset)$ with setup costs f, which are not necessarily all the same. Indeed Algorithm 1 produces a feasible solution also for this case. Now the outcome of Algorithm 1 may fail to produce an optimal solution, even if a facility is placed at a lowest setup cost vertex in each S-component, i.e. $o(k) \in \arg\min\{f_i \mid i \in V^k\}$ for all $k \in K$. This is illustrated in Figure 6, which uses the same input tree as Figure 1, but now we assume that $f_1 = 1$, $f_4 = 3$ and $f_6 = 6$, and $f_i = 100$ for all $i \neq 1, 4, 6$. Figure 6(a) depicts the outcome of Algorithm 1, whereas Figure 6(b) depicts an optimal solution. As can be seen, the outcome of Algorithm 1 is $E^1 = \{(1, 2), (1, 3), (1, 4), (4, 5)\}$, $E^2 = \emptyset$, which induces the vertex set partition $V^1 = \{1, 2, 3, 4, 5\}$, $V^2 = \{6\}$, with set of open facilities $O = \{o(1), o(2)\}$ with $o(1) = 1$ and $o(2) = 6$. The value of this solution is $(1+6)+(1+1+2+2) = 13$. The solution depicted in Figure 6(b) is associated with the edge sets $E^1 = \{(1, 2), (1, 3)\}$, $E^2 = \{(4, 5), (5, 6)\}$, with vertex set partition $V^1 = \{1, 2, 3\}$, $V^2 = \{4, 5, 6\}$, and set of open facilities $O = \{1, 4\}$ ($o(1) = 1$ and $o(2) = 4$). The total cost of this solution is $(1 + 3) + (1 + 1) + (2 + 3) = 11$. Note that the solution of Figure 6(b) can be obtained from the solution of Figure 6(a) by interchanging facilities 4 and 6. Opening facility 4, implies removing edge (1,4); in its turn, closing facility 6, requires to add edge (5,6).

The above example suggests that improvements to a given solution $S = (O, F)$ may be obtained by closing one open facility in O and opening one non-open facility in $V \setminus O$. The effect of such an interchange is twofold. On the one hand the S-component k of the newly open facility $i \in V^k$, $i \neq o(k)$, splits in two subtrees. For this one edge must be removed from the path connecting i and $o(k)$, $P_{i,o(k)}$. On the other hand, the component k' of the facility $j = o(k')$ that is closed must be merged with some other component. For this one edge in $\delta(V^{k'})$ must be added. If the opening and closing facilities are given, i and j, respectively, the best possible move, in terms of the objective function value, is clear: select as leaving edge $e_S^+(i)$, the one with maximum cost in $P_{i,o(k)}$, and select as entering edge, $e_S^-(j)$, the one of minimum cost in $\delta(V^{k'})$. That is $e_S^+(i) \in \arg\max\{c_e : e \in P_{i,o(k)}\}$ and $e_S^-(j) \in \arg\min\{c_e : e \in \delta(V^{k'})\}$. We denote by $k_S^+(i)$ the index of the newly created S-component after opening facility i, and by $k_S^-(j) \in K$ the index of the S-component to which component k' is merged after closing facility j, i.e. $e_S^-(j) = \delta(V^{k'}) \cap \delta(V^{k_S^-(j)})$.

The variation in the objective function implied by opening facility $i \notin O$ is thus, $\Delta_S^+(i) = f_i - c_{e_S^+(i)}$, whereas the *saving* due to closing facility $j = o(k') \in O$ is $\Delta_S^-(j) = \left[f_j - c_{e_S^-(j)}\right]$. Hence, interchanging facilities $i \in V \setminus O$ and $j \in O$ in solution S yields an improved solution if and only if $\Delta_S^+(i) - \Delta_S^-(j) < 0$. Indeed, the *best* facilities to be interchanged are, respectively, $i \in \arg\min\{\Delta_S^+(l) \mid l \in V \setminus O\}$ and $j \in \arg\max\{\Delta_S^-(l) \mid l \in O\}$.

Definition 2 *A solution $S = (O, F)$ is of* minimum cost for O, *if $c(F') \geq c(F)$ for any solution $S' = (O, F')$.*

Remark 2

(a) Let $S = (O, F)$ be the solution to $(T, f, c, p, V, \emptyset)$ obtained with Algorithm 1. Then S is of minimum cost for O.

(b) Let $S = (O, F)$ be of minimum cost for O. For $k_1, k_2 \in K$ given, let $\overline{e} \in E$ the only edge of $P_{o(k_1),o(k_2)}$ that does not belong to F. Then $c_{\overline{e}} \geq c_e$, for all $e \in P_{o(k_1),o(k_2)}$.

Theorem 3 [*Facility interchange optimality*] *A solution to $(T, f, c, p, V, \emptyset)$, $S = (O, F)$ of minimum cost for O, is optimal if and only if $\Delta_S^+(i) - \Delta_S^-(j) \geq 0$ for all $i \in V \setminus O$, $j \in O$.*

Proof:
$\Rightarrow$: *From the above analysis it is clear that if there exist $i \in V \setminus O$, $j \in O$, such that $\Delta_S^+(i) - \Delta_S^-(j) < 0$, then the current solution is not optimal.*

$\Leftarrow$: *For the reverse implication we give a proof based on the complementary slackness conditions of a primal-dual pair.*

Let us recall that (1b)-(1d), (1f) *gives a valid formulation for the case when all vertices have to be visited, and that its coefficient matrix is TU. Consider the LP relaxation of* (1b)-(1d), (1f), *expressed in standard form, that we denote by LR_{LP}^1:*

$$LR_{LP}^1: \quad \min \sum_{i \in V} f_i y_i + \sum_{(u,v) \in A} c_{uv} x_{uv} \tag{3a}$$

$$s.t. \quad \sum_{i \in D \cup V(R)} y_i = p \tag{3b}$$

$$y_v + \sum_{(u,v) \in \delta^-(v)} x_{uv} = 1 \qquad v \in V \tag{3c}$$

$$x_{uv} + x_{vu} + h_{uv} = 1 \qquad (u,v) \in E \tag{3d}$$

$$x_{uv}, x_{uv}, h_e \geq 0, \qquad e = (u,v) \in E \tag{3e}$$

$$y_i \geq 0 \qquad i \in D, \tag{3f}$$

and its dual:

$$LR_D^1: \quad \max \ p\lambda + \sum_{v\in V} \mu_v + \sum_{e=(u,v)\in E} \rho_e \tag{4a}$$

$$s.t. \quad \lambda + \mu_v \le f_v \qquad v \in D \tag{4b}$$

$$\left.\begin{array}{l} \mu_u + \rho_e \le c_e \\ \mu_v + \rho_e \le c_e \end{array}\right\} \qquad e = (u,v) \in E \tag{4c}$$

$$\rho_e \le 0 \qquad e \in E \tag{4d}$$

Let $S = (O, F)$ be a feasible solution $(T, f, c, p, V, \emptyset)$ of minimum cost for O such that $\Delta_S^+(i) - \Delta_S^-(j) \ge 0$ for all $i \in V \setminus O$, $j \in O$. Let $(\overline{y}, \overline{x}, \overline{h})$ denote the values of the decision variables, representing its associated branching with p roots located at the vertices of O.

Consider the following sets of variables:

$$Y^1 = \{v \in V : \overline{y}_v = 1\} = \{v \in V : v = o(k) \text{ for some } k \in K\}. \tag{5}$$

$$X^1 = \{e = (u,v) \in E : \overline{x}_{uv} + \overline{x}_{vu} = 1\} = \{e = (u,v) \in E : \text{ edge } e \text{ is in the solution}\}. \tag{6}$$

$$H^1 = \{e = (u,v) \in E : \overline{h}_e = 1\} = \{e = (u,v) \in E : \text{ edge } e \text{ is not in the solution}\}. \tag{7}$$

Consider now the following solution to the dual LR_D^1:

- $\lambda = \max_{k\in K} \left[f_{o(k)} - \min_{e\in\delta(V^k)} c_e \right]$.
- $\mu_v = f_{o(k)} - \lambda$, *for all* $v \in V^k$, $k \in K$.
- *For all* $e = (u,v) \in E$,

$$\rho_e = \begin{cases} c_{uv} - \mu_u, & \text{if edge } e = (u,v) \in X^1 \text{ (i.e. edge } e \text{ is used in the solution)}, \\ 0, & \text{otherwise (i.e. } h_e \in H^1\text{)}. \end{cases}$$

Indeed, the above solution (λ, μ, ρ) is feasible to LR_D^1. The proof is given in Lemma 1 of the Appendix. Furthermore, it is straight forward to check that the above sets of primal and dual variables jointly satisfy the complementary slackness conditions. Thus, the result follows.

Below we present an algorithm, which produces a solution that satisfies the optimality condition of Proposition 3. We denote by $S^t = (O^t, F^t)$ the partial solution at iteration t with S-components (V^k, E^k), $k \in K^t = \{0, \dots t\}$, and open facilities $o(k) \in \arg\min\{f_i : i \in V^k\}$, $k \in K^t$. For any edge $e \in E^k$, $k \in K^t$, we denote by $V(e) \subset V^k$ the vertex set of the new component that would be created if edge e were removed from the partial solution and by $i(e) \in \arg\min\{f_i : i \in V(e)\}$ the node that would become the open facility of the new S-component in such a case.

Algorithm 3

0. $V^0 \leftarrow V$; $F^0 \leftarrow E$; $K^0 = \{0\}$; $o(0) \in \arg\min\{j_i \mid i \in V\}$;
   $t \leftarrow 1$.
1. $e^t \leftarrow \arg\max\{c_e - f_{i(e)} \mid e \in E^k, k \in K^{t-1}\}$ (with ties arbitrarily broken)
   $k(t) \in K^t$ such that $e^t \in E^{k(t)}$ (the index of the S-component of e^t).
2. $V^{k(t)} \leftarrow V^{k(t)} \setminus V(e^t)$; $V^{t+1} \leftarrow V(e^t)$; $o(t+1) = i(e^t)$; $F^t \leftarrow F^{t-1} \setminus \{e^t\}$.
3. $K^{t+1} \leftarrow K^t \cup \{t\}$
   $t \leftarrow t+1$; if $(t < p)$ goto 1.

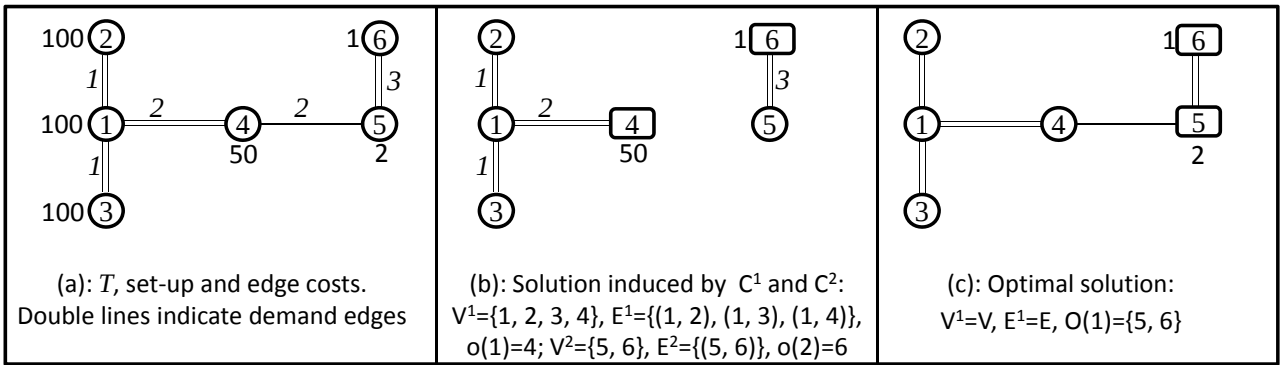


Figure 7: p-ELRP example with $r = p$ and optimal solution with edge in $E \setminus R$

4. end

Proposition 1 Algorithm 3 produces a solution to $(T, f, c, p, V, \emptyset)$ that satisfies the optimality condition of Proposition 3.

Proof: *It is easy to see that the criterion used to select the leaving edge at each step guarantees that the resulting solution guarantees the optimality condition (for a fixed number of components, $t + 1$)*
■

In iteration t there are $\mathcal{O}(n - t)$ potential edges to leave as well as $\mathcal{O}(n - t)$ potential facilities to open in Step 2. Thus each iteration in the while loop is dominated by $\mathcal{O}(n^2)$. Therefore,

Proposition 2 The complexity of Algorithm 3 is $\mathcal{O}(pn^2)$.

4.2.2 p-ELRP with setup costs

Next we address the case $(T, f, c, p, \emptyset, R)$ with $V(R) = V$ and setup costs f, which are not necessarily all the same. For similar reasons to the p-VLRP, the outcome of the greedy heuristic of Algorithm 2 may fail to produce an optimal solution, and interchanging closed and open facilities may produce improved p-ELRP solutions. Additional difficulties may arise now, since the edges involved in such interchanges must be restricted to non-demand edges. As the example of Figure 7.(b) shows it is now possible that no feasible facility interchange producing an improved solution exists even if the current solution is not optimal. Since the solution contains only demand edges, none of the edges in the solution may leave it, so no feasible facility interchange exists. Still the solution is not optimal. This situation is related to the fact that, in contrast to the cases studied so far, the number of trees in optimal p-ELRP solutions is not known in advance. This is illustrated in the example of Figure 7, with two demand components ($r = 2$) and $p = 2$, whose optimal solution, depicted in Figure 7.(c), consists of a unique S-component $T^1 = (V^1, E^1)$ with $V^1 = V$, $E^1 = E$ and $O(1) = \{5, 6\}$.

We can thus conclude that, when the number of subtrees produced by Algorithm 2 (which is always p) does not coincide with the number of subtrees in an optimal solution, an adaptation of Algorithm 3 to avoid removing demand edges, will terminate with a non-optimal solution, as the algorithm produces solutions with a fixed number of S-components.

A natural way overcome this situation is to allow modifications that involve a change in the number of subtrees induced by the solutions. Next we consider the change in a solution due to adding one edge. Consider a feasible solution $S = (O, F)$, where we assume that: (i) the facilities of O have been chosen as indicated in Section 2.2 (one facility is first located a lowest setup cost potential facility of each S-component, and the remaining $p - |K|$ facilities are selected by increasing values of their setup costs); and, (ii) S is of minimum cost for O. Let $e \in E \setminus F$ be a non-demand edge connecting S-components T^k and $T^{k'}$ where, without loss of generality, we assume that $f_{o(k')} \geq f_{o(k)}$. The effect of adding edge e to the solution S without removing any edge from the solution is the following. The set of edges in the solution is extended to $F \cup \{e\}$ and S-components T^k and $T^{k'}$ merge, i.e.

$K \leftarrow K \setminus \{k'\}$. Then, if $|O(k')| > 1$ the set of open facilities will remain unchanged. (This is a consequence of (i) above, since $|O(k')| > 1$ means that all open facilities in this S-component obey to the greedy criterion.) Thus, $O(k) \leftarrow O(k) \cup O(k')$ and the variation in objective function value due to adding edge e to solution S is $\Theta_S[e] = c_e \geq 0$. However, when $|O(k')| = 1$, facility $o(k')$ does not necessarily satisfy the greedy criterion, so it will close if $f_{o(k')} > f_{i^+}$ where i^+ is the index of the cheapest vertex among the ones where no facility sits in solution S, i.e. $i^+ \in \arg\min\{f_i \mid i \in V \setminus O\}$. In this case, the set of open facilities of the S-component containing vertex i^+, say $\bar{k} \in K$, is updated to $O(\bar{k}) \cup \{i^+\}$. Now, the variation in objective function value is $\Theta_S[e] = c_e + f_{i^+} - f_{o(k')}$, which can be negative. Hence, adding edge $e \in E \setminus F$ to solution S yields an improved solution if and only if $\Theta_S[e] < 0$. We thus have the following result in which the proof omitted, as it is largely based on that of Proposition 3 and the above analysis.

Proposition 3 *A solution $S = (O^S, F^S)$ to $(T, f, c, p, \emptyset, R)$ is optimal if and only it holds that:*

- $\Delta_S^+(i) - \Delta_S^-(j) \geq 0$ *for all* $i \in V \setminus O$, $j \in O$
- $\Theta_S[e] \geq 0$, $e \in E \setminus F$

4.3 The case with non-demand vertices

In this section we consider again the case when demand is located at the vertices, but we now assume that not all vertices must be necessarily visited, i.e. $L = D \cup V(R) \subset V$. We study the case without setup costs and with only vertex demand $(T, 0, c, p, D, \emptyset)$. Like in previous sections the case with demand at edges can be reduced to this case by compacting each demand component into one single vertex.

In the case of general graphs, non-demand vertices can be eliminated by defining edges associated with shortest paths if the intermediate nodes of such paths use some non-demand vertex. This approach is however not valid in the case of a tree without breaking the tree-structure of the original graph, except for non-demand vertices with degree two.

In the case of trees, allowing for vertices that do not necessarily have to be covered in feasible solutions increases notably the difficulty of the associated location routing problem. The main reason is that a priori one does not know the vertices in $V \setminus D$ that will be covered in an optimal solution. In other words, one does not know in advance, what edges incident with non-demand vertices should be considered as potential candidates for an optimal solution. This is illustrated by the example of Figure 8 where vertex 5 has not demand. For $p = 4$ Algorithm 1 produces the non-optimal solution of value 19 Figure 8(b). The optimal solution to this instance is depicted in Figure 8(c); it has a value of 18 and does not cover vertex 5. Observe, however, that if in this example we modify slightly the edge costs to $c_{45} = c_{56} = 5$ and $c_{57} = 6$, all other costs remaining unchanged, then the outcome of Algorithm 1, which is the solution depicted in Figure 8 (b), is optimal. The value of this solution for the instance with modified costs is 16, and it covers the non-demand vertex 5. Note also that all distances between pairs of demand vertices in the S-component {4, 5, 6, 7}, are greater than 9, which is the length of two tree edges connecting pairs of demand vertices, that do not make part of this optimal solution. This may seem counterintuitive at first sight.

Below we present the recursion of an algorithm that optimally solves $(T, 0, c, p, D, \emptyset)$. We arbitrarily root the tree T at some demand vertex $i_0 \in D$. For each $i \in V$, we say that j is a *successor* of i if i is on the unique path connecting j to i_0. If j is a successor of i then we say that i is a *predecessor* of j. A *direct descendant* (*children*) of i is a descendant connected to i via an edge of T. Let $C(j) = \{i_1(j), i_2(j), \ldots, i_{n_j}(j)\}$ denote the index set of the children of j. When clear from the context we just write $C(j) = \{i_1, i_2, \ldots, i_{n_j}\}$. Without loss of generality we assume that the elements of $C(j)$ are ordered by nondecreasing distance to j, i.e. $c_{j,i_1} \leq c_{j,i_2} \cdots \leq c_{j,i_{n_j}}$. If $C(j)$ is empty, j is called a *leaf* of T. (Note that all leaves have degree one in the original tree T, and the only possible non-leaf with degree one could be the root.)

Throughout the section we make the following assumptions:

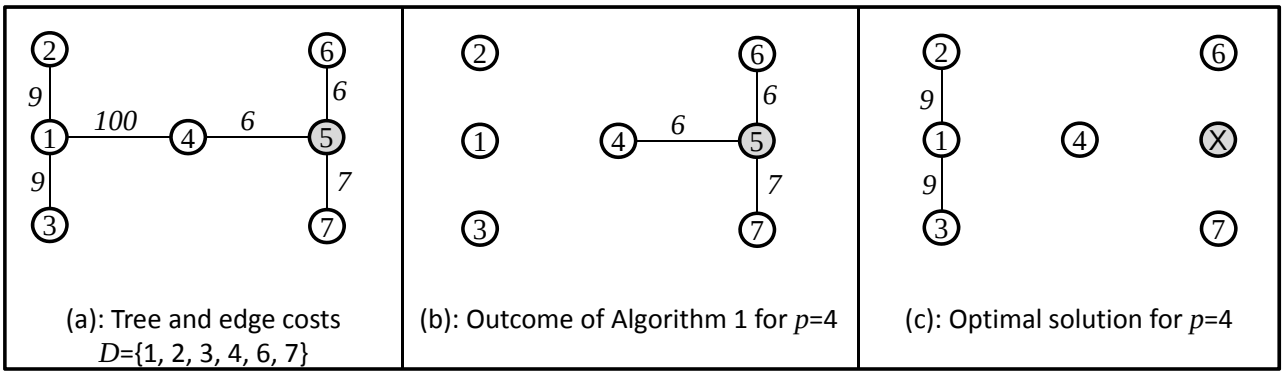


Figure 8: Example of instance with non-demand vertices

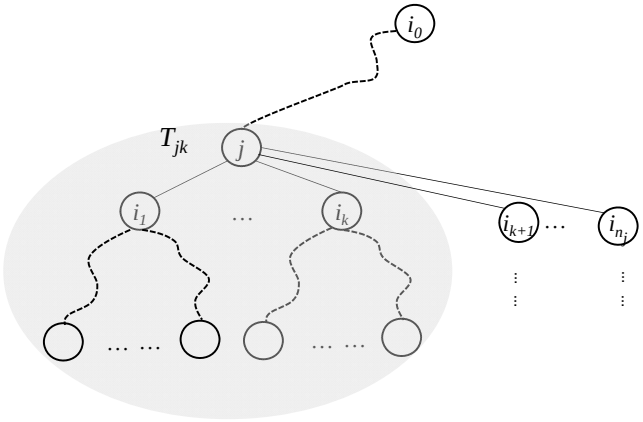


Figure 9: Rooted tree for the recursion of Algorithm 4

(*i*) All the leaves of T are demand vertices. Otherwise, they can be eliminated from the tree, as they will not be covered in any optimal solution.

(*ii*) All non-demand vertices have degree at least three. (All non-demand vertices with degree two have been eliminated).

(*iii*) No optimal solution contains an S-component with no demand vertex. (Such an S-component can be eliminated)

Since there are no set-up costs, similarly to Section 4.1.1, we focus on solutions where each S-component contains exactly one open facility, thus associating the number of open facilities with the number of S-components.

To solve $(T, 0, c, p, D, \emptyset)$, we recursively solve a sequence of problems defined on certain subtrees of T, starting from the leaves. To define these subtrees, consider a vertex $j \in V$. For any $k = 1, \ldots, n_j$, let $T_{j,k}$ denote the subtree induced by the vertices in $\{j\} \cup C(j_1) \cup \cdots \cup C(j_k)$. We also denote by $D_{j,k}$ the set of demand vertices in $T_{j,k}$, i.e. $D_{j,k} = D \cap V(T_{j,k})$, and by $q_{jk} = \min\{|D_{j,k}|, p\}$. Note that q_{jk} is an upper bound on the number of facilities that can be located in the subtree $T_{j,k}$ in any feasible solution to $(T, 0, c, p, D, \emptyset)$. Figure 9 depicts an example of these definitions.

In the recursion, for all $j \in V$, $z(j, k, q)$ denotes the optimal value to $(T_{j,k}, 0, c, q, D_{j,k}, \emptyset)$, where $k \leq n_j$ and $q \leq q_{jk}$. We initialize $z(j, k, 0) = \infty$ for all $j \in V$, $k \leq n_j$, indicating that in any feasible solution any subtree containing no facility must be connected to some vertex outside the subtree. When j is a leaf, $z(j, 0, 1)$ is initialized to 0, indicating the cost of an S-component containing one facility located at the singleton $\{j\}$. For all other vertices the values of k are restricted to be greater

than or equal to 1. When j is not a leaf, the values $z(j,k,q)$ are only computed after the values $z(i,r,s)$ have been computed for all its children $i \in C(j)$ and possible values of r and s. For ease of presentation and without loss of generality we asume that vertices indices are ordered in such a way that $i_k(j) > j$ for all $k = 1 \dots n_j$. Then, selecting the indices j by decreasing values, ensures that the recursions $z(j,k,q)$ are computed in a correct order. The optimal value to $(T, 0, c, p, D, \emptyset)$ will be given by $z(i_0, n_{i_0}, p)$.

Only for clarity purposes, we first present the case where all vertices have demand, $D = V$. In this case, for any subtree $T_{i,k}$, $D_{j,k} = V(T_{j,k})$. The general case where $D \subseteq V$ will follow.

Proposition 4 Suppose $D = V$. Let $j \in V$ be a non-leaf vertex and suppose the values $z(i,r,s)$ have been computed for all $i \in V$, $i > j$ and possible values of r and s. Then,

(*i*) $z(j,1,q) = \min\{z(i_1, n_{i_1}, q-1), z(i_1, n_{i_1}, q) + c_{j,i_1}\}$, for all $1 \le q \le q_{jk}$

(*ii*) For all $k \in \{2, \dots, n_j\}$, $1 \le q \le q_{jk}$

$$z(j,k,q) = \min \begin{cases} \min\limits_{\substack{q_1+q_2=q \\ 1\le q_1 \le q_{j,k-1} \\ 1 \le q_2 \le q_{i_k,n_{i_k}}}} [z(j,k-1,q_1) + z(i_k, n_{i_k}, q_2)] & (8) \\ c_{j,i_k} + \min\limits_{\substack{q_1+q_2=q+1 \\ 1\le q_1 \le q_{j,k-1} \\ 1 \le q_2 \le q_{i_k,n_{i_k}}}} [z(j,k-1,q_1) + z(i_k, n_{i_k}, q_2)] & (9) \end{cases}$$

Proof:

(*i*) In any optimal solution to $(T_{j,1}, 0, c, q, V(T_{j,1}), \emptyset)$, either vertex j defines one S-component on its own (and $T_{i_1,n_{i_1}}$ contains $q-1$ facilities), or $T_{i_1,n_{i_1}}$ contains q facilities and vertex j is connected to some S-component of $(T_{i_1,n_{i_1}}, 0, c, q, V(T_{i_1,n_{i_1}}), \emptyset)$ via edge (j, i_1). In the former case, the optimal value is given by $z(i_1, n_{i_1}, q-1)$. In the second case, the optimal value is given by $z(i_1, n_{i_1}, q) + c_{j,i_1}$.

(*ii*) When $k > 1$, in any optimal solution to $(T_{j,k}, 0, c, q, V(T_{j,k}), \emptyset)$, either the solution contains edge (j, i_k) or it does not. In the first case, since $c_{j,i_1} \le c_{j,i_2} \cdots \le c_{j,i_{k-1}} \le c_{j,i_k}$, vertex j must also be connected to some other vertex in $T_{j,k-1}$. Otherwise the solution could be improved (or would not change, if ties exist) by interchanging edge (j, i_k) with edge (j, i_1). Thus, if the first case holds, the optimal value of $(T_{j,k}, 0, c, q, V(T_{j,k}), \emptyset)$ can be obtained by computing the best value $z(j, k-1, q_1) + z(i_k, n_{i_k}, q_2)$ among all possible combinations of q_1 facilities in $T_{j,k-1}$, $1 \le q_1 \le q_{j,k-1}$ and q_2 facilities in $T_{i_k,n_{i_k}}$, $1 \le q_2 \le q_{i_k,n_{i_k}}$ with $q_1 + q_2 = q$. In the second case, edge (j, i_k) would *connect* a partial solution with with q_1 facilities (components) in $T_{j,k-1}$ and a partial solution with with q_2 facilities (components) in $T_{i_k,n_{i_k}}$. That is, the solution containing edge (j, i_k) and the two partial solutions will have $q_1 + q_2 - 1$ components. Hence, when the second case holds, the optimal value of $(T_{j,k}, 0, c, q, V(T_{j,k}), \emptyset)$ can be obtained by computing the best value $c_{j,i_k} + z(j, k-1, q_1) + z(i_k, n_{i_k}, q_2)$ among all possible combinations of q_1 facilities in $T_{j,k-1}$, $1 \le q_1 \le q_{j,k-1}$ and q_2 facilities in $T_{i_k,n_{i_k}}$, $1 \le q_2 \le q_{i_k,n_{i_k}}$ with $q_1 + q_2 - 1 = q$.

Proposition 4 gives rise to the following algorithm, which solves $(T, 0, c, q, V, \emptyset)$ for all $q \in \{1, \dots, |V|\}$.

Algorithm 4

Initialization:

$z(j,0,1) = 0$ for all $j \in V$ leaf

$z(j,k,0) = \infty$ for all $j \in V$, $1 \le k \le n_j$

for ($j \in V$ not leaf) do

for ($k \in \{1, \dots, n_j\}$, $1 \le q \le q_{jk}$) do

if($k = 1$)

$$z(j,1,q) = \min\{z(i_1, n_{i_1}, q-1), z(i_1, n_{i_1}, q) + c_{j,i_1}\}$$

else

$$z(j,k,q) = \min \begin{cases} \min\limits_{\substack{q_1+q_2=q \\ 1\le q_1\le q_{j,k-1} \\ 1\le q_2\le q_{i_k,n_{i_k}}}} [z(j,k-1,q_1) + z(i_k, n_{i_k}, q_2)] \\ \\ c_{j,i_k} + \min\limits_{\substack{q_1+q_2=q+1 \\ 1\le q_1\le q_{j,k-1} \\ 1\le q_2\le q_{i_k,n_{i_k}}}} [z(j,k-1,q_1) + z(i_k, n_{i_k}, q_2)] \end{cases}$$

Since $\sum_{j\in V} n_j = n-1$, the initialization step of Algorithm 4 as well as the number of possible pairs (j,k) in the main loop are $\mathcal{O}(n)$. For a given pair (j,k) at most p possible values of q will be considered. Computing $z(j,k,q)$ with j,k,q fixed is $\mathcal{O}(q) \le \mathcal{O}(p)$, as there are at most $q-1$ possible combinations $q_1 + q_2 = q$, with $q_1, q_2 \ge 1$. Therefore:

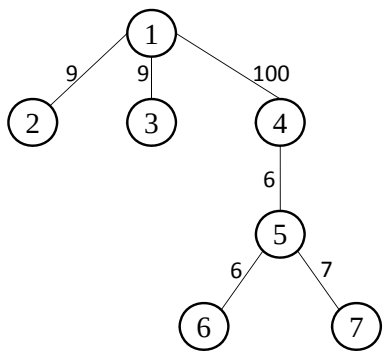
Proposition 5 *The complexity of Algorithm 4 is $\mathcal{O}(np^2)$.*

Because in the rooted tree used in Algorithm 4 the children of each node j are ordered by nondecreasing costs, $c_{j,i_1} \le c_{j,i_2} \cdots \le c_{j,i_{n_j}}$, we can assume that if vertex j is connected to some S-component in one of its subtrees, this would be a component in its first subtree $T_{j,1}$. Introducing facilities set-up costs would make this assumption false. Finding the best subtree for connecting any explored node j in this case would increase the complexity of the algorithm in one order of magnitude.

Example 2 *We illustrate Algorithm 4 on the graph of Figure 8 for p=4, assuming that all vertices (including vertex 5) have demand, i.e., $D = \{1,2,3,4,5\}$. Figure 10a depicts the rooted tree. Values next to each edge are their costs.*

In the initialization we set $z(2,0,1) = z(3,0,1) = z(6,0,1) = z(7,0,1) = 0$ and $z(1,1,0) = z(1,2,0) = z(1,3,0) = z(4,1,0) = z(5,1,0) = z(5,2,0) = \infty$. Figure 10b gives the $z(j,k,q)$ values computed via the recursion. The optimal value is $z(1,3,4) = 19$.

Algorithm 4 produces solutions in which all vertices belong to some S-component. Thus it is not valid for the case when $D \subset V$ and there is an optimal solution where some non-demand vertex does not belong to any S-component. Below we present the extension to the general case $(T, 0, c, p, D, \emptyset)$ with $D \subseteq V$. Using the same notation as before, $D_{i,k} = D \cap V(T_{i,k})$ denotes again the set of demand vertices in $T_{i,k}$, although it is now possible that $V(T_{i,k}) \setminus D_{i,k} \neq \emptyset$. Again, for all $j \in V$, $z(j,k,q)$



(a) Rooted tree

j	k	$\lvert D_{jk}\rvert$	q	$z(j,k,q)$
5	1	2	1	$c_{56} + z(6,0,1) = 6$
			2	$z(6,0,1) = 0$
	2	3	1	$c_{57} + z(5,1,1) + z(7,0,1) = 13$
			2	$z(5,1,1) + z(7,0,1) = 6$
			3	$z(5,1,2) + z(7,0,1) = 0$
4	1	4	1	$c_{45} + z(5,2,1) = 19$
			2	$c_{45} + z(5,2,2) = 12$
			3	$z(5,2,2) = 6$
			4	$z(5,2,3) = 0$
1	1	2	1	$c_{12} + z(2,0,1) = 9$
			2	$c(2,0,1) = 0$
	2	3	1	$c_{13} + z(1,1,1) + z(3,0,1) = 18$
			2	$z(1,1,1) + z(3,0,1) = 9$
			3	$z(1,1,2) + z(3,0,1) = 0$
	3	7	1	$c_{14} + z(1,2,1) + z(4,1,1) = 137$
			2	$z(1,2,1) + z(4,1,1) = 37$
			3	$z(1,2,2) + z(4,1,1) = 28$
			4	$z(1,2,3) + z(4,1,1) = 19$

(b) Values of $z(j,k,q)$

Figure 10: Illustration of Algorithm 4 for Example 2

denotes the optimal value of $(T_{j,k}, 0, c, q, D_{j,k}, \emptyset)$, with $k \leq n_j$, and $1 \leq q \leq q_{jk}$. As before, the optimal value to $(T, 0, c, p, D, \emptyset)$ will be given by $z(i_0, n_{i_0}, p)$.

Now in the solution producing $z(j,k,q)$, vertex j will not be covered unless $j \in D$, or $j \notin D$ but it is used to *connect* S-components in two of its subtrees, $T_{o(r),n_{o(r)}}$ and $T_{o(s),n_{o(s)}}$, with $r, s \leq k$. The same may happen to some of its non-demand descendants. Nevertheless, even it vertex j is not covered in the solution producing $z(j,k,q)$, it is possible that vertex j is covered in the solution to some subtree of some predecessor of j. Hence, at a later step the recursion may consider solutions that connect j (or some of its uncovered descendants) with some predecessor of j. For computing correctly the value of such solutions we need additional information. In particular, we use an auxiliary function $g(j,k,q)$, which records the distance between j and the vertex covered in the solution producing $z(j,k,q)$, which is closest to j. Indeed, if $j \in D$, then $g(j,k,q) = 0$. Moreover, $g(j,k,q) = 0$ even if $j \notin D$ but it is covered by the partial solution of value $z(j,k,q)$. Therefore, $g(j,k,q) > 0$ only when j is not covered by the solution associated with $z(j,k,q)$. In this case, the value of $g(j,k,q)$ is given by the shortest distance between j and any of the vertices covered by the partial solution associated with $z(j,k,q)$. Because, such distance may correspond to a path of G, rather than to an edge, it has to be updated recursively.

To illustrate the above comments, consider again the rooted tree of Figure 10a, but assume now that $D = \{1,2,3,4,6,7\}$ and vertex 5 does not have demand. Now $z(5,1,1) = 0$ corresponds to the S-component $\{6\}$, in which vertex $j = 5$ is not covered, and $g(5,1,1) = 6$ is the distance from vertex 5 to the S-component, which is given by $c_{5,6} = 6$. The solution which gives $z(4,1,2)$ consists of the S-components $\{4,\ 5,\ 6\}$ and $\{7\}$. The value $z(4,1,2)$ is now given by $(c_{45} + z(5,1,1) + g(5,1,1)) + z(7,0,1) = 6 + 0 + 6 + 0 = 12$.

Taking into account the definition of the function $g(j,k,q)$, the analog to Proposition 4 is now:

Proposition 6 *Suppose $D \subseteq V$. Let $j \in D$ be a non-leaf vertex and suppose the values $z(i,r,s)$ and $g(i,r,s)$ have been computed for all $i \in V$, $i > j$ and possible values of r and s. Then, $g(j,k,q) = 0$, for all $k \in \{1,\ldots,n_j\}$, $1 \leq q \leq q_{jk}$. Furthermore,*

(i) $z(j,1,q) = \min\{z(i_1, n_{i_1}, q-1), z(i_1, n_{i_1}, q) + c_{ji_1} + g(i_1, n_{i_1}, q)\}$ *for all* $1 \leq q \leq |D_{j,k}|$

(ii) For all $k \in \{2,\ldots,n_j\}$, $1 \leq q \leq q_{jk}$

$$z(j,k,q) = min \begin{cases} \min\limits_{\substack{q_1+q_2=q \\ 1\leq q_1 \leq q_{j,k-1} \\ 1 \leq q_2 \leq q_{i_k,n_{i_k}}}} \{z(j,k-1,q_1) + z(i_k, n_{i_k}, q_2)\} & (10) \\ \min\limits_{\substack{q_1+q_2=q+1 \\ 1\leq q_1 \leq q_{j,k-1} \\ 1 \leq q_2 \leq q_{i_k,n_{i_k}}}} \{c_{j,i_k} + g(i_k, n_{i_k}, q_2) + z(j,k-1,q_1) + z(i_k, n_{i_k}, q_2)\} & (11) \end{cases}$$

The recursion for non-demand vertices is slightly more complicated, but follows the rationale explained above. It is summarized below:

Proposition 7 *Suppose $D \subseteq V$. Let $j \notin D$ be a non-leaf non-demand vertex and suppose the values $z(i,r,s)$ and $g(i,r,s)$ have been computed for all $i \in V$, $i > j$ and possible values of r and s. Then,*

(i) $z(j,1,q) = z(i_1, n_{i_1}, q)$, *for all* $1 \leq q \leq q_{jk}$
$g(j,1,q) = g(i_1, n_{i_1}, q) + c_{j,i_1}$, *for all* $1 \leq q \leq q_{jk}$

(ii) For all $k \in \{2,\ldots,n_j\}$, $1 \leq q \leq q_{jk}$

$$z(j,k,q) = min \begin{cases} \min\limits_{\substack{q_1+q_2=q \\ 2\leq q_1 \leq q_{j,k-1} \\ 2 \leq q_2 \leq q_{i_k,n_{i_k}}}} \{z(j,k-1,q_1) + z(i_k, n_{i_k}, q_2)\} & (12) \\ \min\limits_{\substack{q_1+q_2=q+1 \\ 2\leq q_1 \leq q_{j,k-1} \\ 2 \leq q_2 \leq q_{i_k,n_{i_k}}}} \{g(j,k-1,q_1) + c_{j,i_k} + g(i_k, n_{i_k}, q_2) + z(j,k-1,q_1) + z(i_k, n_{i_k}, q_2)\} & (13) \end{cases}$$

(iii) For $k \in \{1, \ldots, n_j\}$, $1 \leq q \leq q_{jk}$

$$g(j,k,q) = \begin{cases} \min\{g(j,k-1,\bar{q}_1), c_{j,i_k} + g(i_k, n_{i_k}, \bar{q}_2)\} & \text{if value of } z(j,k,q) \text{ given by (12)} \\ & (\bar{q}_1, \bar{q}_2 \text{ arguments which give } z(j,k,q)) \\ 0 & \text{if value of } z(j,k,q) \text{ given by (13).} \end{cases}$$

Propositions 6 and 7 give rise to an algorithm with the same structure as Algorithm 4 which uses the expressions above for computing all possible values of $z(j,k,q)$ and $g(j,k,q)$, depending on whether or not $j \in D$. Broadly speaking it performs in the same number of iterations, so its complexity is again $\mathcal{O}(np^2)$.

5 Conclusions

In this paper we have studied combined location/routing problems (LRPs) defined on simple graphs, namely trees. Several problems have been studied, which consider demand both at the vertices and the edges of the input tree. An integral polyhedral representation of LRPs han been presented, which has the integrality property. The formulation models a directed forest where each connected component hosts at least one open facility, which becomes the root of the component. Greedy type optimal algorithms have been presented for the cases when all vertices have to be visited and facilities have no set-up costs. Facilities set-up costs can be handled with low order-interchanges producing solutions of proven optimality. Such methods solve no longer are no longer valid for LRPs where some of the vertices do not have to be necessarily visited. For the general case, a low order optimal algorithm based on recursions has been presented.

All presented algorithms can be extended with slight modification to cacti, graphs in which each edge may belong to at most one cycle. It is possible to deal with each such cycle by considering two possible cases for an optimal solution: either all the edges of the cycle are traversed once, or at least one edge of the cycle is not traversed. The former case can be handled by compacting the cycle into one single vertex. In the later case the cost of largest cost in the cycle can be removed. (See [3] for further details in the case of prize collecting problems on cacti.) Indeed, such extensions imply an increase of one order of magnitude in the complexity of the presented algorithms.

All considered problems consider no capacity constraints. A promising avenue for future research is the study of LRPs with the simplest type of capacity constraints, i.e. cardinality constraints on the number of vertices in each route. From a different perspective,it can be interesting to study the usefulness of the proposed algorithms in heuristic methods for LRPs in general graphs.

6 Acknowledgements

This research has been partially supported by the Spanish Ministry of Economy and Competitiveness through MINECO/FEDER grants MTM2015-63779-R and MDM-2014-044. This support is gratefully acknowledged.

References

[1] M. Albareda-Sambola. Location-routing and location–arc routing. In *Location Science*. G. Laporte, S. Nickel and F. Saldanha da Gama(eds), Springer, 2015.

[2] M. Albareda-Sambola, J.A. Díaz, and E. Fernández. A compact model and tight bounds for a combined location-routing problem. *Computers & Operations Research*, 32:407–428, 2005.

[3] J. Aráoz, E. Fernández, and C. Zoltan. Privatized rural postman problems. *Computers & Operations Research*, 33:3432–3449, 2006.

[4] J. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R. Wolfler Calvo. A branch-and-cut method for the capacitated location-routing problem. *Computers & Operations Research*, 38:931–941, 2011.

[5] M. Conforti, G. Cornuéjols, and K. Vušković. Balanced matrices. *Discrete Mathematics*, 306:2411–2437, 2006.

[6] B. Golden, T. Magnanti, and H. Nguyen. Implementing vehicle routing algorithms. *Networks*, 7:113–148, 1977.

[7] O. Kariv and L. Hakimi. An algorithmic approach to network location problems i: The p-centers. *SIAM Journal on Applied Mathematics*, 37:513–538, 1979.

[8] O. Kariv and L. Hakimi. An algorithmic approach to network location problems i: The p-medians. *SIAM Journal on Applied Mathematics*, 37:539–560, 1979.

[9] G. Laporte. Location-routing problems. In *Vehicle Routing: Methods and Studies*, pages 163–198. B.L. Golden, A.A. Assad(eds), North-Holland, Amsterdam., 1988.

[10] R.B. Lopes, C. Ferreira, B. S. Santos, and S. Barreto. A taxonomical analysis, current methods, and objectives on location-routing problems. *International Transactions of Operations Research*, 20:795–822, 2013.

[11] G. Nagy and S. Salhi. Location-routing: Issues, models and methods. *European Journal on Operational Research*, 177:649–672, 2007.

[12] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley and Sons, 1988.

[13] J. Perl and M. Daskin. A warehouse location-routing problem. *Transportation Research B*, 19:381–396, 1985.

[14] C. Prodhon and C. Prins. A survey of recent research on location-routing problems. *European Journal on Operational Research*, 238:1–17, 2014.

[15] A. Tamir. A class of balanced matrices arising from location problems. *SIAM J. Algebraic Discrete Methods*, 4:363–370, 1983.

[16] A. Tamir. An $o(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19:59–64, 1996.

[17] A. Tamir and T. Lowe. The generalized p-forest problem on a tree network. *Networks*, 22:217–230, 1992.

[18] B.C. Tansel, R.L. Francis, and T.J. Lowe. Location on networks: A survey. part i: The p-center and p-median problems. *Management Science*, 29:482–497, 1983.

[19] B.C. Tansel, R.L. Francis, and T.J. Lowe. Location on networks: A survey. part ii: Exploiting tree network structure. *Management Science*, 29:498–511, 1983.

[20] L.A. Wolsey. *Integer Programming*. Wiley-Interscience series in discrete mathematics and optimization. John Wiley and Sons, New York, 1998.

7 Appendix:

Lemma 1 *Let $S = (O, F)$ be a feasible solution to to $(T, f, c, p, V, \emptyset)$ of minimum cost for O such that $\Delta_S^+(i) - \Delta_S^-(j) \geq 0$ for all $i \in V \setminus O$, $j \in O$. Let $(\overline{y}, \overline{x}, \overline{h})$ denote the values of the decision variables in formulation LR_{LP}^1, (3a)-(3f), representing its associated branching with p roots located at the vertices of O. Let also, Y^1, X^1, H^1 denote the sets of variables defined in (5), (6), and (7), respectively.*

Then, the solution (λ, μ, ρ) as defined below is feasible to the dual of LR_{LP}^1.

- $\lambda = \max_{k \in K} \left[f_{o(k)} - \min_{e \in \delta(V^k)} c_e \right]$.
- $\mu_v = f_{o(k)} - \lambda$, *for all* $v \in V^k$, $k \in K$.
- *For all* $e = (u, v) \in E$,

$$\rho_e = \begin{cases} c_{uv} - \mu_u, & \text{if edge } e = (u,v) \in X^1 \text{ (i.e. edge } e \text{ is used in the solution)}, \\ 0, & \text{otherwise (i.e. } h_e \in H^1\text{)}. \end{cases}$$

Proof:

Recall that the dual of LR_{LP}^1 is given by:

$$LR_D^1: \quad \max \; p\lambda + \sum_{v \in V} \mu_v + \sum_{e=(u,v) \in E} \rho_e \tag{14a}$$

$$s.t. \quad \lambda + \mu_v \leq f_v \qquad v \in D \tag{14b}$$

$$\left.\begin{array}{l} \mu_u + \rho_e \leq c_e \\ \mu_v + \rho_e \leq c_e \end{array}\right\} \qquad e = (u,v) \in E \tag{14c}$$

$$\rho_e \leq 0 \qquad e \in E \tag{14d}$$

Next we see that the solution (λ, μ, ρ) is feasible to LR_D^1:

(*i*) *Constraints* (4b) *are satisfied. For all* $v \in V^k$, $\lambda + \mu_v = \lambda + f_{o(k)} - \lambda = f_{o(k)} \leq f_v$, *since* $o(k) \in \arg\min\{f_v : v \in V^k\}$.

(*ii*) *constraints* (4c) *are satisfied. For all* $e = (u, v) \in E$, *consider the following subcases:*

(*a*) $(u, v) \in X^1$. *Then,* $u, v \in V^k$ *for some* $k \in K$, $\mu_u = \mu_v = f_{o(k)} - \lambda$, *and* $\rho_e = c_{uv} - \mu_u$. *Hence, the corresponding constraints constraints* (4c) *hold since* $\mu_u + \rho_e = \mu_v + \rho_e = \mu_v + c_{uv} - \mu_v = c_e$.

(*b*) $(u, v) \in H^1$ *with* $u \in V^k$ *and* $v \in V^{k'}$, $k \neq k'$. *Then,* $\rho_e = 0$. *Hence,*

- $\mu_u + \rho_e = \mu_v = f_{o(k)} - \lambda \leq c_e$ *if and only if* $f_{o(k)} - c_e \leq \lambda$.
- $\mu_v + \rho_e = \mu_v = f_{o(k')} - \lambda \leq c_e$ *if and only if* $f_{o(k')} - c_e \leq \lambda$.

The value of λ guarantees that this conditions holds.

(*iii*) *Since* $\rho_e = 0$ *for all* $h_e \in H^1$, *to see that constraints* (4d) *are satisfied, it is enough to consider the case when* $\rho_e \neq 0$, *i.e.* $(u, v) \in X^1$. *Then,* $u, v \in V^k$ *for some* $k \in K$, *and* $\rho_e = c_{uv} - \mu_u$ *with* $\mu_u = \mu_v = f_{o(k)} - \lambda$. *Thus,* $\rho_e \leq 0$ *if and only if* $c_{uv} \leq f_{o(k)} - \lambda$.
Hence, constraints (4d) *are satisfied if and only if*

$$\lambda \leq \min_{k \in K} \{ f_{o(k) - \max_{e \in E^k} c_e} \}. \tag{15}$$

To prove that inequality (15) *holds, let us denote by* $k^\lambda \in K$ *the index of the S-component that gives the value of* λ. *That is* $k^\lambda \in \arg\max_{k \in K} \{ f_{o(k)} - \min_{e \in \delta(V^k)} c_e \}$. *Let also* $\hat{k} \in K$

denote the index of the S-component producing the smallest value of $f_{o(k)-\max_{e\in E^k} c_e}$. *That is* $\hat{k} \in \arg\min_{k\in K}\{f_{o(k)-\max_{e\in E^k} c_e}\}$. *Then, for all* $i \in V^{k^\lambda}$ *we have:*

$$\lambda = f_{o(k^\lambda)} - \min_{e\in\delta(V^{k^\lambda})} c_e \leq f_i - \min_{e\in\delta(V^{k^\lambda})} c_e \leq f_i - c_{e_S^+(i)} = \Delta_S^+(i),$$

where the first inequality follows from the criterion used to select $o(k^\lambda)$, *and the second one because* $c_{e'} \geq c_e$ *for all* $e \in E^{k^\lambda}$, $e' \in \delta(V^{k^\lambda})$.

On the other hand, by the optimality condition of Theorem 3, for all $i \in V^{k^\lambda}$

$$\Delta_S^+(i) \leq \Delta_S^-(o(\hat{k})) = f_{o(\hat{k})} - c_{e_S^-(o(\hat{k}))} \leq f_{o(\hat{k})} - \max_{e\in E^{\hat{k}}} c_e,$$

where the final inequality now follows since $e_S^-(o(\hat{k})) \in \delta(V^{\hat{k}})$ *and* $c_{e'} \geq c_e$ *for all* $e \in E^{\hat{k}}$, $e' \in \delta(V^{\hat{k}})$.