A polynomial algorithm for partitioning a tree into single-center subtrees to minimize flat service costs

N. Apollonio[*] I. Lari[*] J. Puerto[†] F. Ricca[‡] B. Simeone[*]

In memory of Stefano Pallottino

Abstract

This paper deals with the following graph partitioning problem: given a graph with n nodes, p of which are prescribed to be centers (the remaining nodes are called units), the goal is to find a partition of the set of nodes into connected components containing only one center each, so as to minimize the total assignment cost of units to centers. This problem is known to be NP-hard in general graphs, and it is shown here to remain such even if the assignment cost is monotone and the graph is bipartite. Therefore, we specialize on tree graphs to derive polynomial time algorithms. For this class of graphs we provide several reformulations of the problem as integer linear programs. Moreover, we develop a dynamic programming algorithm, whose recursion is based on solving sequences of minimum weight closure problems, that solves the problem on trees in $O(n^2p)$.

1 Introduction

The problem of assigning units to centers arises in many applications where a facility must be located in different sites or centers of a territory in order to serve a set of customer points (units). In this paper we consider a set of units and a set of already fixed centers, and we study the problem of assigning each unit to exactly one center. Given a cost function which associates a service cost to each possible unit-center pair, our aim is to find an assignment of units

[*]Università di Roma "La Sapienza" Dip. Statistica, Probabilità e Statistiche Applicate

[†]Universidad de Sevilla Dep. Estadística e Investigación Operativa. The research of this authors is partially supported by Spanish research grants numbers: MTM2004:0909, HA2003:0121.

[‡]Università de L'Aquila Dip. Sistemi e Istituzioni per l'Economia

to centers which minimizes the total cost. Actually, this problem can be formulated as a graph partitioning problem. Consider a territory divided into n elementary units (small portions of land) and represent it by a connected n-vertex graph $G = (V, E)$, where each vertex corresponds to an elementary unit and an edge between two vertices exists if and only if the two corresponding units are neighboring. The graph G is called the *contiguity* graph. Let $S \subset V$ be the set of fixed centers and $U = V \setminus S$ be the set of units that must be served. Let n denote the total number of vertices of G and p the number of centers. Then, $|S| = p$ and $|U| = n - p$. A *connected partition* of G is a partition of V into nonempty subsets, called *components*, such that each component induces a connected subgraph of G. A *legal partition* of G is a connected partition of G where each component contains exactly one center. Each unit in the same component as s is said to be *served by* s. Let $c : U \times S \longrightarrow \mathbb{R}$ be a cost function which associates a cost c_{is} to each pair (i, s), $i \in U$, $s \in S$. In real applications c_{is} is a flat service cost due to the assignment of unit i to center s. Then, we define the cost of a legal partition of G as the sum of all the service costs c_{is}, $i \in U$, $s \in S$, such that in the partition i is assigned to s.
In this paper we deal with the following problem:

Minimum Cost Legal Partition Problem (MCLP) - Given a connected graph G, find a legal partition of G with minimum cost.

In the following, given the graph G, the set of centers S and the flat service cost function c, we will denote an instance of problem MCLP by (G, S, c).

Notice that, given a cost function c, a minimum cost legal partition with respect to c remains optimal when a positive constant M is added to the costs, since in each feasible solution the resulting increase of the total cost is $Mp(n-p)$, a constant. Thus, without loss of generality, in the following we will assume that the service costs are strictly positive.

Let F be a spanning forest of G, and let $\mathcal{F}(G)$ be the set of all the spanning forests of G. Let $T_s(F)$ denote the subtree of F which contains the center s. In the following, in order not to make the notation heavier, we shall identify the tree $T_s(F)$ with either its set of vertices or its set of edges, whichever is appropriate. This set will be denoted simply by T when additional specification is not necessary. Notice that, each spanning forest of G is a connected partition of G, and each connected partition of G can be represented as a spanning forest of G. Then, problem MCLP can be formulated as the problem of finding a spanning forest of G such that each tree in F contains exactly one center and the total service cost is minimized. We can formulate this problem as follows:

$$\begin{array}{ll} \min & \sum_{s\in S} \sum_{i\in T_s(F)} c_{is} \\ & F \in \mathcal{F}(G) \\ & |T \cap S| = 1 \qquad\qquad \forall T \in F \end{array} \tag{1}$$

Among the many possible applications of this problem, MCLP is strictly related to districting problems. In particular, we consider the political districting problem which consists of drawing a district map to be used in a political election. This problem is of particular importance especially when a majority voting rule is adopted because it interferes in the translation of votes into seats. It can be formulated as an MCLP if the territory involved in the political elections is represented by a graph-theoretic model.

The fact that one looks for a connected partition of the contiguity graph reflects the usual requirements of *integrity* (no unit should be split between any two or more districts) and *contiguity* (each district should consist of geographically contiguous units). Traditionally, population equality and compactness are the main and widely accepted criteria for political districting [1, 5]. In general, population equality can be taken into account by a constraint which forces the population of each district between fixed upper and/or lower bounds [7]. Let $\overline{p}$ be the average district population and α a given real in $[0,1]$. Then, one may require the population of each district to be at most equal to $(1+\alpha)\overline{p}$. Moreover, compactness can be considered in the objective function by minimizing a suitable measure of the dispersion of the units of each district around their center. If the distances between each unit and each center are available, the total inertia can be used to this purpose (see, for example, [4]). Let p_i denote the population of territorial unit i and let d_{is} be the distance between unit i and center s. If the territory is represented by the contiguity graph G, and a weight equal to p_i is associated to each vertex i, the districting problem can be formulated as an MLCP with a side constraint due to the population equality criterion as follows:

$$\begin{array}{lll} \min & \sum_{s\in S}\sum_{i\in T_s(F)} p_i d_{is}^2 & \\ & F \in \mathcal{F}(G) & \\ & |T\cap S| = 1 & \forall T \in F \\ & p(T) \leq (1+\alpha)\overline{p} & \forall T \in F \end{array} \tag{2}$$

where the weight $p(T)$ associated to the tree $T \in F$ is given by the sum of all p_i's for $i \in T$. Problem (2) differs from problem (1) only for the population equality constraints. If we include the population equality constraints into a lagrangean objective function we obtain problem (1) with $c_{is} = p_i(d_{is}^2 + \lambda_s)$, where the λ_s's are the non negative lagrangean multipliers of the population equality constraints.

In [2] it has been shown that problem MCLP is NP-hard on general graphs even in the case of two centers. The natural question that arises is whether the same problem is polynomially solvable in simpler graph topologies. For some well-known problems in location analysis, as the p-center or p-median [8, 9], this is the case when the model is restricted to tree networks. Thus, the main goal in this paper is to answer whether MCLP is polynomially solvable on trees.

Looking for an answer to our question, the reader may notice that the feasible solutions of MCLP correspond to the bases of a matroid on the set of edges of the input graph. Nevertheless, the greedy algorithm does not work in general.

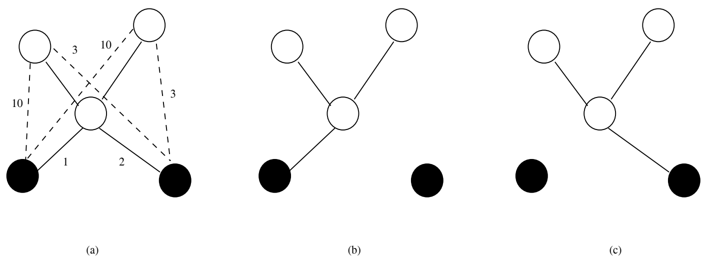

Figure 1: Consider the tree in (a), where black vertices are fixed as centers and the numbers refer to the costs c_{is} for each unit-center pair (i, s). In (b) the solution found by the greedy algorithm is shown, while (c) shows the optimal solution.

The trouble is that the costs are defined on unit-center pairs, rather than on the elements of the ground set, namely, the edges (see Fig. 1). This situation makes this problem even more challenging.

The main goal of this paper is to give an affirmative answer to the above question. We show that MCLP is solvable in polynomial time on trees, describing an $O(n^2p)$ algorithm for this case that combines dynamic programming techniques with algorithms for the maximum closure. Analyzing MCLP on trees we will also describe several interesting reformulations as binary integer programming programs, as well as some more advanced issues on the complexity of the problem.

In order to attain the above aims the paper is organized as follows. In Section 2 we provide two Binary Linear Programming models for problem MCLP on trees. Section 3 describes the polynomial time algorithm for trees, whereas Section 4 is devoted to give some further complexity results on the problem under consideration.

2 Two Binary Linear Programming models for trees

In this section we present two Binary Linear Programming formulations for problem MCLP on trees. Consider a tree $T = (V, E)$ and, as before, let $S \subset V$ be the set of centers and $U = V \setminus S$. Notice that in a tree, removing (cutting) $p - 1$ edges results into a connected partition of T into p components, and every connected partition can be obtained in this way.

In the first formulation we use the following binary variables:

$$y_{is} = \begin{cases} 1 & \text{if unit } i \text{ is assigned to center } s \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

In the following model we use order constraints, so that the partition is connected, and semiassignment constraints so that each unit is assigned to exactly one center. These constraints guarantee that the partition is legal. The Binary Linear Programming model based on the order constraints is the following:

$$\begin{array}{lll} \min & \sum_{i \in U} \sum_{s \in S} c_{is} y_{is} & \\ & y_{is} \leq y_{js} & \forall i \in U, j \in U, s \in S \text{ such that } j \text{ lies in } P_{is} \\ & \sum_{s \in S} y_{is} = 1 & \forall i \in U \\ & y_{is} \in \{0,1\} & \forall i \in U, s \in S \end{array} \tag{4}$$

where P_{is} is the unique path from i to s in the tree T. This model has $O(np)$ variables and $O(n^2p)$ constraints. Notice that if the semiassignment constraints are deleted, the constraint matrix of the resulting model is totally unimodular.

An alternative Binary Linear Programming model for problem MCLP on trees can be obtained on the basis of the notion of legal orientation, which provides a different, but useful, perspective on the problem. Given an instance (T, S, c) of MCLP, a *legal orientation* of T is an orientation of some of the edges of T such that:

(i) for each undirected edge ij, at most one of the two arcs (i, j) and (j, i) is present;

(ii) the outdegree of each unit is 1;

(iii) the outdegree of each center is 0.

Notice that some edges of T remain undirected.

Proposition 1 *In any legal orientation of T, the number of undirected edges is $|S| - 1$.*

Proof. The total number of vertices, centers, and units are equal to n, p, and $n - p$, respectively. After property (ii), the total number of arcs is $n - p$, so the number of undirected edges is $n - 1 - (n - p) = p - 1$. □

Proposition 2 *There is a one-to-one correspondence between legal partitions and legal orientations of T.*

Proof. With any legal partition π of T, we associate an orientation ω of some edges of T as follows. For each unit i, let s be the unique center serving i, and let e be the first edge along the path from i to s. Orient e out of unit i. All the edges that are cut in π remain undirected. Clearly, ω is a legal orientation, and one can get back π simply by deleting the undirected edges in ω. □

Proposition 3 *For any legal orientation and each unit i, there is a unique directed path from i to a center. Such center $s(i)$ is precisely the center that serves i in the legal partition associated with ω.*

Proof. Consider any legal orientation ω and apply the following simple procedure:

```
$j := i$;
while $j$ has a successor
    let $j'$ be the unique successor of $j$;
    $j := j'$;
endwhile
$s(i) := j$;
```

Since T is finite and acyclic, such a procedure stops after a finite number of iterations, returning a node $s(i)$, which must be a center after properties (ii) and (iii) of legal orientations. By the same properties, both $s(i)$ and the dipath from i to $s(i)$ must be unique. Finally, we show that, when the undirected edges are deleted, the center $s(i)$ is the same for all the units in the same connected component as i. Recall that these undirected edges are precisely those that are cut in the legal partition associated with ω. Suppose then that in the same connected component there are two units i' and i'' such that $s(i') \neq s(i'')$. Since i' and i'' are in the same connected component, there must be an elementary path P connecting i' and i''. All the edges along P must be directed, since all undirected edges have been deleted. But then at least one of the vertices along P (including i' and i'') must have outdegree 2, in contradiction with properties (ii) and (iii) of legal orientations. □

Let $s(i)$, $i \in U$, be the center that serves i in the legal orientation ω. Then, the cost of any legal orientation ω is given by

$$\sum_{i \in U} c_{is(i)}.$$

Notice that the cost of ω coincides with the cost of the legal partition π associated with ω. Then, consider the following problem:

Minimum Cost Legal Orientation Problem (MCLO) - Given a tree T, a set of centers S and a cost function c, find a legal orientation of T with minimum cost.

In view of Propositions 2 and 3, we obtain the following result.

Theorem 1 *In the case of trees, problems MCLP and MCLO are mutually reducible in polynomial time.*

Now let us formulate a Binary Linear Programming model for MCLO. Given a legal orientation ω, for each edge ij of T introduce binary variables x_{ij} and x_{ji}; for each unit i and each center s, introduce binary variables y_{is}, with the following meaning:

$$x_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is an arc in } \omega \\ 0 & \text{otherwise,} \end{cases}$$

$$y_{is} = \begin{cases} 1 & \text{if } s = s(i) \\ 0 & \text{otherwise} \end{cases}$$

Thus, the x_{ij}'s are decision variables that define the orientation ω, while the y_{is}'s are auxiliary variables which are needed in order to compute the cost of ω as a linear function.

As usual, denote by $N(i)$ the neighborhood of i, i.e., $N(i) = \{j | ij \in E\}$. The binary linear programming model is:

$$\begin{array}{llll} \min & \sum_{i \in U} \sum_{s \in S} c_{is} y_{is} & & \\ & x_{ij} + x_{ji} \leq 1 & \forall ij \in E & (a) \\ & \sum_{j \in N(i)} x_{ij} = 1 & \forall i \in U & (b) \\ & x_{sf} = 0 & \forall s \in S, fs \in E & (c) \\ & y_{fs} = x_{fs} & \forall s \in S, fs \in E & (d) \\ & y_{js} + x_{ij} - 1 \leq y_{is} & \forall s \in S, ij \in E & (e) \\ & x_{ij} \in \{0,1\} & \forall ij \in E & (f) \\ & y_{is} \in \{0,1\} & \forall i \in U, s \in S & (g) \end{array} \qquad (5)$$

Constraints (a), (b), and (c) enforce properties (i), (ii) and (iii) of legal orientations, respectively. In every optimal solution, constraints (d), the transitivity constraints (e), and the strict positivity of the costs c_{is} force y_{is} to be 1 iff there is a directed path from i to s. It follows that the objective function represents the cost of the legal orientation ω. The above model involves $O(np)$ variables and $O(np)$ constraints.

In the case of trees, it would be interesting to obtain a polynomial time algorithm for MCLP - or, equivalently, for MCLO - exploiting the special structure of the binary linear programs described in this section. In the next section, we shall derive an $O(n^2p)$ algorithm through a combination of dynamic programming and maximum weighted closure procedures for this problem. The very existence of such algorithm proves that the above two binary linear programs are solvable in polynomial time.

3 A polynomial algorithm for partitioning problem on a tree

In this section we describe a polynomial algorithm for partitioning a tree into single-center subtrees so as to minimize flat service costs. Given a tree $T = (V, E)$, $|V| = n$, as before we denote by U the set of all units and by S the set of all centers. Given a unit i and a center s, let us define a path from i to s to be *free* if it does not contain any center but s. For each unit i, we define the set $C_i = \{s : \text{the path from } i \text{ to } s \text{ is free}\}$. Notice that i may be served only by the centers in C_i . Moreover, we denote by $c(i) = (c_{is})_{s \in C_i}$ the cost vector restricted to the pairs (i, s), $s \in C_i$.

The following lemma shows that assuming that centers and leaves coincide causes no loss of generality in the case of trees.

Lemma 1 *[Leaf Property] Any instance (T, S, c) of MCLP, where T is a tree and S, c are arbitrary, can be reduced, preserving optimality, to a set of independent instances $(T_i, C_i, c(i))$, $i = 1, 2, ..., k$, $k \leq n - p$, where the T_i 's are subtrees of T such that: (1) the union of all the T_i's is equal to the whole tree T; (2) any pair of subtrees T_i and T_j, $i \neq j$ intersects in at most one node, this node being a center; (3) C_i is the set of leaves of T_i.*

Proof. We can always assume that in T there is no edge whose endnodes are both centers, since, in this case, such edge would be necessarily cut when assigning units to centers. Furthermore, without loss of generality, we may assume that every leaf l of T is a center. If not, let f be the unique node adjacent to l . If f is a center, then l is forced to be served by f and thus we can delete l from T. If f is a unit, then l and f must be assigned to the same center, so they can be condensed into a single node whose service cost is the sum of the service costs of l and f. It follows that we can always restrict ourselves to those instances of MCLP in which all the leaves of T are centers. Next, let us show that, conversely, we may always assume that every center is a leaf. So, assume that there exist centers that are not leaves. Let us declare two units i and j to be equivalent iff $C_i = C_j$. Then the set U of all units is accordingly partitioned into equivalence classes U_i. For each i, the subgraph T_i of T induced by $U_i \cup C_i$ is connected and hence it is a subtree of T. Notice that $T_i = T_j$ iff i and j are equivalent. On the other hand, if i and j are not equivalent, then T_i and T_j may intersect in at most one node, which must necessarily be a center. Furthermore, for each i the set of leaves of T_i coincides with C_i , since removing C_i leaves T_i connected and each unit in U_i separates at least two centers of C_i. Since, as noticed above, unit i may be served only by centers in C_i and, on the other hand, the total service cost is additive with respect to the units, the thesis follows. $\square$

From now on, we assume that the Leaf Property holds, i.e., the leaves and the centers coincide.

Before describing the algorithm, we introduce some definitions and notation. Let $T = (V, E)$ be an arbitrary rooted tree. As usual, we denote by (i, j) an edge directed from node i to node j. Node i is said to be the *predecessor* (or the *father*) of j and node j is a *successor* (or a *child*) of i. The two sets of predecessors and successors of node i are denoted by $\text{Pred}(i)$ and $\text{Succ}(i)$, respectively. If there is a directed path from node i to node j, then i is called an *ancestor* of j, and j a *descendant* of i. We regard i to be both an ancestor and a descendant of itself. The two sets of ancestors and descendants of node i are denoted by $\text{Anc}(i)$ and $\text{Desc}(i)$, respectively. For a given node i the *closure* of i is any subset of nodes that, whenever it contains node i, it contains all the predecessors (and hence all the ancestors) of i. If Z is any subset of nodes, the *cocycle* ∂Z of Z is the set of all edges with exactly one vertex not in Z. We call the *downtree* of T at v, and denote it by T_v , the subtree of T rooted at v induced by $\text{Desc}(v)$. Given an edge (u, v), the *partial downtree* T_{uv} is the subtree of T induced by $u \cup \text{Desc}(v)$. The subtree T_{uv} is rooted at u and it is obtained from T_v by the addition of the edge (u, v).

In the following we assume that T is rooted at one of its units, say r. We are going to describe a polynomial algorithm for solving MCLP which is based on a bottom-up dynamic programming recursion. In order to implement such a recursion, a sequence of minimum weight closure problems on trees are solved. Let i be a unit, T_i the downtree of T rooted at i, and l an arbitrary center (leaf) of T_i . We define z_{il} to be the minimum service cost of any legal partition of T_i, subject to the condition that i is served by l; we shall also define

$$z_i = \min\{z_{il} | l \text{ is a center of } T_i\}.$$

Thus, z_i is just the minimum service cost of any legal partition of T_i. Denote by Π_{is} the set of all legal partitions of T_i such that i is served by s. The dynamic programming algorithm starts from those nodes g that are predecessors of leaves l. Clearly,

$$z_{gl} = c_{gl}.$$

Now, consider an arbitrary unit i and assume that, for each child j of i and each center t in T_j, all values z_{jt} have been previously computed by the dynamic programming algorithm. Let s be any center in T_i. Then, in any cheapest legal partition of T_i where i is served by s, all nodes along the directed path P_{is}, from i to s, are also being served by s. Moreover, if T_v is an arbitrary downtree whose root v has its father f in P_{is}, some units of T_v may also be served by s (see Fig. 2). The set R of all such units must induce a subtree of T_v rooted at v. Notice that this subtree depends only on T_v and P_{is}, but is independent of other similar downtrees $T_{v'}$ (where the father of v' lies in P_{is}), no matter whether v' has the same father as v or not. Indeed, the fact that a certain unit h belongs to R is influenced only by the nodes along the path from h to s and by the descendants of h in T_v. Therefore, we can restrict our attention to a single such downtree T_v.

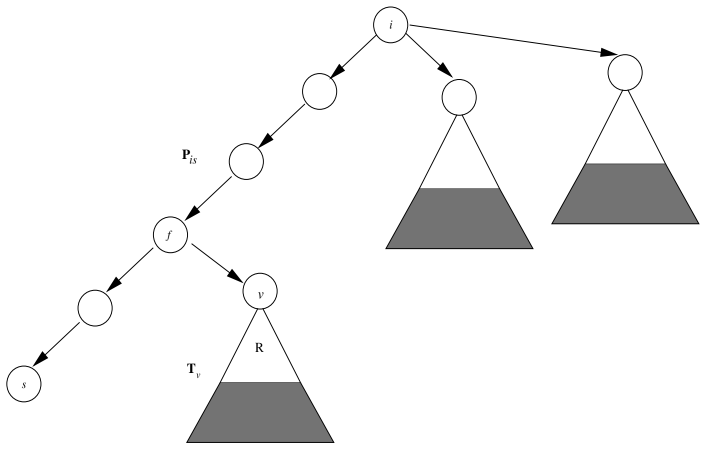

Figure 2: The white vertices are assigned to center s, while the vertices in grey are assigned to some center different from s.

Finding the above set R efficiently, we would be in position to compute the value z_{is} (and hence z_i) using the dynamic programming recursion. One should notice that finding R is itself a problem of type MCLP, where T is replaced by the partial downtree T_{fv} and f becomes a center with service costs $c_{hf} = c_{hs}$ for all the units h of T_{fv}. In order to find R, consider the subtree induced by all the units of T_v with root v and denote it by U_v. Associate with each node h of U_v an *upper weight*

$$u_h = c_{hs}, \tag{6}$$

a *lower weight*

$$l_h = z_h - \sum_{m \in \mathrm{Succ}(h)} z_m \tag{7}$$

and a *weight*

$$w_h = u_h - l_h. \tag{8}$$

Remark. In view of the above formulas (6), (7), (8), the weight w_h can be rewritten as

$$w_h = c_{hs} - (z_h - \sum_{m \in \mathrm{Succ}(h)} z_m). \tag{9}$$

The above expression has an interesting economic interpretation: at node h two options are possible: (i) h is served by s, or (ii) h is served by some leaf of the downtree T_h. Then the weight w_h is actually the marginal cost of option (i) w.r.t. option (ii).

Lemma 2 *For each node h of U_v, one has*

$$z_h = \sum_{k \in Desc(h)} l_k \tag{10}$$

Proof. By induction on the depth of h. If h is a leaf of U_v, then $\mathrm{Desc}(h) = \{h\}$, $\mathrm{Succ}(h) = \emptyset$ and both (7) and (8) amount to $l_h = z_h$. Assume that (8) holds for each node of depth $d-1$, and let h have depth d. One has from (7)

$$z_h = l_h + \sum_{m \in \mathrm{Succ}(h)} z_m,$$

and from the inductive hypothesis,

$$z_h = l_h + \sum_{m \in \mathrm{Succ}(h)} \sum_{k \in \mathrm{Desc}(m)} l_k.$$

□

Theorem 2 *Let i be any node of T, s a leaf of T_i and P_{is} the directed path from i to s in T. Let v be a unit outside P_{is} whose father f belongs to P_{is}. Finding the set S of those units of T_v that are served by s in some cheapest partition of Π_{is} is reducible to the minimum weight closure problem in the rooted tree U_v with node weights w_h.*

Proof. First of all, notice that a subset of nodes of a tree rooted at v is a closure if and only if it is either empty or it induces a subtree with root v. Let π be a cheapest partition in Π_{is}, and let R be the set of units of T_v that are served by s in π. Let ∂R be the cocycle of R in U_v. Since R is either empty or it induces a subtree of T_v with the same root v, R is a closure of U_v. Furthermore, the set R must be chosen so as to minimize the overall contribution of T_v to the service cost of π. Such contribution, in view of the decomposition followed in the algorithm, is equal to

$$\begin{aligned}
\gamma(R) &= \sum_{p \in R} c_{ps} + \sum_{(p,q) \in \partial R} z_q \\
&= \sum_{h \in R} u_h + \sum_{(p,q) \in \partial R} \sum_{k \in \mathrm{Desc}(q)} l_k && \text{by (10)} \\
&= \sum_{h \in R} u_h + \sum_{k \in V(U_v) \setminus R} l_k && \text{since } R \text{ is a closure in } U_v \\
&= \sum_{h \in R} w_h + \sum_{k \in V(U_v)} l_k.
\end{aligned}$$

Conversely, if R is a closure, then the above identities hold in reverse order. Therefore, $\gamma(R)$ and the weight of R differ by a constant, and the thesis follows.

□

3.1 Complexity analysis

For each unit i and for each center s in T_i, one has to compute z_{is}. Given the path P_{is} from i to s, for each edge uv such that u is in P_{is} and v is a child of u not lying in P_{is}, we need to solve a minimum weight closure problem on the downtree T_v. Each of these closure problems can be solved in time linear in the number of vertices of the downtree by an algorithm given in [6]. Moreover, having already computed by the recursion the values z_m for $m \in \mathrm{Succ}(h)$, finding the weights w_h for each h requires $O(|\mathrm{Succ}(h)|)$. Thus, in the dynamic programming procedure, computing each value z_{is} requires $O(n)$ time. The resulting overall time complexity is $O(n^2p)$.

The above algorithm gives better complexity bounds on some particular important classes of trees. The reader may easily check that for spiders (trees with at most one node of degree larger than two; such node is called the body of the spider) the implementation that considers the allocation of the body of the spider to the different centers, rather than considering all the possible unit-center pairs, results in an algorithm with complexity $O(np)$. It is also easy to verify that on paths just by traversing twice the units we get a linear time algorithm. (Notice that the naïve application of the general algorithm to the case of spiders will result in $O(n^2p)$ and on paths in $O(n^2)$).

4 Further Complexity Results

The aim of the present section is to give more insights on the hardness of MCLP on a general graph $G = (V, E)$. Any instance of MCLP takes the form (G, S, c). In [2] it has already been shown that bounding $|S|$ does not lead to easier solvable instances (clearly, if $|S| = 1$ the problem is trivial). Unfortunately, as shown below by Theorem 3.(a), enlarging the class of input graphs leading to polynomial time solvable classes of MCLP, looks hopeless, even if strong conditions are imposed on c. Actually, in view of Theorem 3.(b), the main source of difficulty in solving MCLP lies in the costs structure: requiring c to be metric makes the problem easy to solve essentially through a min cost bipartite assignment algorithm. So both MCLP on trees and MCLP with metric costs barely lie within the boundary, so as to speak, separating easy instances from hard ones. This is confirmed by 3.(c) where, even when G is a tree and c is metric, requiring some further conditions makes the problem NP-complete.

In order to proceed with the section we need some definitions. Let $w : E \to \mathbb{Z}_+$ be a weighting of the edges of a graph G and, for $u, v \in V$ denote by $d_w(u, v)$ the length of a shortest uv-path with respect to w and by $d(u, v)$ the geometric distance between u and v (the minimum number of edges of an uv-path). Given the service cost function c we say that it is *monotone*, if $d(u, s) \leq d(v, s) \Rightarrow c_{us} \leq c_{vs}$. The service cost function is said to be *metric*, if c is proportional to d_w for some w. Functions $b_1, b_2 : S \to \mathbb{Z}$, with $b_1 \leq b_2$ are also given. They are regarded as capacity functions. A *capacitated legal partition* (with respect to b_1 and b_2) is a legal partition $\pi = \{C_1 \ldots, C_p\}$ such that

$$b_1(t) \leq \sum_{v \in C_t} c_{vt} \leq b_2(t), \text{ where } t \text{ is the center in } C_t.$$

The following theorem collects the above mentioned results on the complexity of the problem.

Theorem 3 *Let G be a connected graph, S a set of p centers and c a cost function. Then,*

(a) *Problem MCLP is NP-complete even if c is monotone and the input graph is bipartite.*

(b) *Problem MCLP can be solved in strongly polynomial time if the input graph is arbitrary and c is metric (compare with our main result for trees).*

(c) *It is NP-complete to decide if a spider (hence a tree) whose legs have at most two vertices admits a feasible capacitated partition. Therefore, the capacitated version of MCLP is NP-hard even for trees and even for metric assignment functions.*

Proof. (a). Reduction from SAT. Let $C_1 \ldots, C_m$ be m clauses on the set of variables $\{u_1, \ldots, u_n\}$. Construct a bipartite graph as follows. For each clause C_i, $i = 1, \ldots, m$ there is a node v_i. For each variable u_j, $j = 1, \ldots, n$ there is a node z_j. There is an edge joining v_i to z_j if and only if clause C_i contains variable u_j. (The graph built so far is just the bipartite graph representing clause-variable incidence). For each node z_j take two more vertices s_j and t_j and connect them to z_j; s_j represents literal u_j while t_j represents literal $\overline{u}_j$. The resulting graph is bipartite with shores $\{v_1, \ldots, v_m\} \cup \{s_1, t_1\} \ldots \cup \{s_n, t_n\}$ and $\{z_1, \ldots, z_m\}$. Let $S = \{s_1, t_1\} \cup \ldots \cup \{s_n, t_n\}$ and define the assignment cost as follows:

- if variable u_j occurs in clause C_i as u_j set $c_{v_i s_j} = 0$;
- if variable u_j occurs in clause C_i as $\overline{u}_j$ set $c_{v_i t_j} = 0$;
- set $c_{v_i s_j} = c_{v_i t_j} = 0$, for $j = 1, \ldots, n$;
- set the assignment costs equal to 1 otherwise.

The function c is monotone. Moreover, there is a legal partition of cost zero if and only if the formula is satisfiable.

(b). The problem reduces to a min-cost assignment problem on a complete bipartite graph with shores $V - S$ and S. (Recall that in a bipartite graph with shores A and B an assignment of A into B is a set of edges having degree one on A). For $v \in V \setminus S$ and $s \in S$ the edge vs carries a weight equal to c_{vs}. Clearly each legal partition defines a feasible assignment. On the other hand, a minimum cost assignment defines a legal partition with the same cost. To see this consider any optimal assignment and observe that if v is assigned to s and u lies on the shortest vs-path then also u is assigned to s. Indeed, if u is assigned to s' we must have, by optimality, $c_{us'} \leq c_{us}$. Strict inequality cannot hold because otherwise $c_{vs'} < c_{vs}$ would hold as well, contradicting the optimality of the assignment. So $c_{us'} = c_{us}$ must apply and we can assign u to s affecting neither feasibility nor optimality. In particular if C_s is the set of vertices assigned to s then the subgraph induced by $C_s \cup \{s\}$ is connected and the proof follows.

(c). Reduction from SUBSET SUM [3]. Let $a_1 \ldots a_p$ be an instance of SUBSET SUM. Let $M = \sum_{i=1}^{p} a_i$. Let G be a star with $p+1$ leaves $v_0, v_1, \ldots, v_p$ and let q denote the unique non-leaf node in G. Set $S = \{v_0, v_1 \ldots, v_p\}$ and define $b_1, b_2 : S \to \mathbb{Z}$ as follows: $b_1(v_0) = b_2(v_0) = M/2$ and $b_1(v_i) = 0$, $b_2(v_i) = +1$, for $i = 1, \ldots, p$. Insert a new vertex u_i on each edge qv_i, $i = 1 \ldots p$. Define edge weights as follows: edges v_0q and u_iv_i, $i = 1 \ldots p$, have weight zero; edges qu_i, $i = 1 \ldots p$ have weight a_i. Denote by w this weight function and let $c_{vt} = d_w(v, t)$ the length of a shortest path between v and t, $v \notin S$, $t \in S$. Then every feasible capacitated legal partition defines a partition of $\{1, \ldots, p\}$ into sets A and B such that $\sum_{i \in A} a_i = \sum_{i \in B} a_i$ and conversely. $\square$

References

[1] B. Bozkaya, E. Erkut, G. Laporte. "A tabu search heuristic and adaptive memory procedure for political districting", European Journal of Operational Research 144: 12-26, 1983.

[2] R. Cordone (2001). "A Short Note on Graph Tree Partition Problems with Assignment or Communication Objective Functions", Internal Report DEI 2001, 7, Politecnico di Milano.

[3] M. R. Garey, D. S. Johnson (1999). "Computers and Intractability", W. H. Freeman and Company, New York.

[4] R. S. Garfinkel, G. L. Nemhauser. "Optimal Political Districting by Implicit Enumeration Techniques", Management Science, vol.16:495-508, 1970.

[5] P. Grilli di Cortona, C. Manzi, A. Pennisi, F. Ricca, B. Simeone (1999). "Evaluation and Optimization of Electoral Systems", SIAM Monographs on Discrete Mathematics and Applications, SIAM, Society for Industrial and Aplied Mathematics, Philadelphia.

[6] P.L. Hammer, B. Simeone. "Order relations of variables in 0-1 programming", in: S. Martello, G. Laporte, M. Minoux, C.C. Ribeiro (eds.) Surveys in Combinatorial Optimization, Annals of Discrete Mathematics 31: 83-112, 1987.

[7] S.W. Hess, J.B. Weaver, J.N. Siegfeldt, J.N. Whelan, P.A. Zhitlau. "Nonpartisan political districting by computer", Operations Research 13:998-1006, 1965.

[8] O. Kariv and S.L. Hakimi. “An algorithmic approach to network location problems I: The p-centers.” SIAM Journal of Applied Mathematics, 37:513–538, 1979.

[9] O. Kariv and S.L. Hakimi. “An algorithmic approach to network location problems II: The p-medians.” SIAM Journal of Applied Mathematics, 37:539–560, 1979.