Discrete Optimization ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Uniform and most uniform partitions of trees☆

Isabella Lari[a], Justo Puerto[b], Federica Ricca[a], Andrea Scozzari[c,*]

[a] *Sapienza Università di Roma, Italy*
[b] *IMUS, Spain*
[c] *Università degli Studi Niccolò Cusano - Roma, Italy*

ARTICLE INFO

Article history:
Received 6 March 2017
Received in revised form 22 February 2018
Accepted 8 June 2018
Available online xxxx

Keywords:
Graph partitioning
Most uniform partition problems
Centered partitions
Equipartition problems

ABSTRACT

This paper addresses centered and non centered equipartition tree problems into p connected components (p-partitions). In the former case, each partition must contain exactly one special vertex called center, whereas in the latter, partitions are not required to fulfill this condition. Among the different equipartition problems considered in the literature, we focus on: (1) Most Uniform Partition (MUP) and (2) Uniform Partition (UP). Both criteria are defined either w.r.t. weights assigned to each vertex or to costs assigned to each vertex–center pair. Costs are assumed to be flat, i.e., they are independent of the topology of the tree. With respect to costs, MUP minimizes the difference between the maximum and minimum cost of the components of a partition and UP resorts to optimal min–max or max–min partitions. We provide polynomial time algorithms for centered and non centered versions of the MUP problem on trees when weights are assigned to each vertex. In the non centered case, our results set as polynomial the complexity of this problem which was an open question for several years. On the contrary, we prove that the centered version of MUP with flat costs is NP-complete even on trees. For the UP problem, we develop polynomial time algorithms for the max–min and min–max centered p-partition problems on trees both in the case of weight and cost-based objective functions.

1. Introduction

The essence of partitioning problems is to divide a set of units into groups or clusters so that units belonging to the same group are as similar as possible, or the clusters are as balanced as possible. In several applications such as community detection, image processing, wireless sensor networks and territorial districting [1–4], the set of units can be represented by the set of vertices of a graph G where edges represent relations between units [5].

☆ The second, third and fourth authors wish to thank the Spanish Ministry of Economía and Competitividad that has partially supported this research by FEDER grant number MTM2016-74983-C02-01.

* Corresponding author.
E-mail addresses: isabella.lari@uniroma1.it (I. Lari), puerto@us.es (J. Puerto), federica.ricca@uniroma1.it (F. Ricca), andrea.scozzari@unicusano.it (A. Scozzari).

https://doi.org/10.1016/j.disopt.2018.06.003
1572-5286/

Many optimization problems can be modeled as graph partitioning problems, with different objectives and constraints. Weights are often associated to each vertex and subsets of vertices, and an objective function based on these weights must be maximized or minimized. Several applications also require a fixed number p of clusters in the partition and/or capacity constraints on clusters. Sometimes, some vertices of the graph represent centers so that in the partition each component must contain exactly one center [6,7]. In this case, the resulting partition is called a *centered partition* of G. The classification of the vertex set of a graph into centers and non centers is common in location problems, where centers represent facilities and the other vertices represent clients (see, e.g., [8,9] and the references there in). For operational reasons, it may be also required to introduce connectivity constraints. Connected partitions are such that the subgraphs of G induced by the vertices of each cluster are connected [10].

Given a graph G, we study the problem of finding a connected partition of the set of vertices of G into p components (p-partition). Weights are assigned to the vertices of G and a weight is defined for each component of the partition. Then, the general aim is to balance the components' weights as much as possible. This kind of problems are known as *uniform* or *equipartition* problems. The objective function can be formulated in different ways. One possibility is to minimize the difference between the maximum and the minimum weight of a component (Most Uniform Partition—MUP). Alternatively, components' weights can be balanced using a min–max or max–min approach (Uniform Partition—UP). We study these problems both in the *non centered* and *centered* cases, that is, when centered partitions are considered. It is also meaningful to consider costs of assigning a vertex to a center. We assume that these costs are *flat*, i.e., they are independent of the topology of G. The goal is to find a centered partition of G into p connected components in which the costs of the components are as balanced as possible. We observe that the cost version is more general than the weight version, since the latter case corresponds to the former when each unit has the same assignment cost for all centers. In previous papers we already studied problems of this class on general graphs, providing several NP-completeness results for special classes of graphs [6,7]. Due to the difficulty of solving the above partition problems, in the literature, they have been widely studied focusing on tree graphs.

For the centered case and for arbitrary graphs, mathematical programming formulations based on an exponential number of constraints and heuristic algorithms have been presented by several authors for the problem of partitioning a given territory. Among these, we mention [11–13]. In [6,7] the authors present a wide variety of centered partitioning problems on trees with different objective functions and equipartition criteria. For each analyzed problem, the authors either provide a polynomial time algorithm, or prove that it falls in the class of difficult NP-hard problems. In particular, in [7] the authors show that the min–max and max–min problems are NP-hard even on planar graphs with vertex degree at most 3 and $p = 2$.

In the literature, equipartition problems were mainly considered in the non centered case, and many papers examined the connected p-partition problem of a graph G with a weight-based objective function. Lucertini et al. [14] presented a polynomial time algorithm for MUP on paths. Recently, Ito et al. [15] provided a polynomial time algorithm for finding an $(\ell; u)$-partition of a tree into p connected components. This is, indeed, a feasibility problem since an $(\ell; u)$-partition of a graph is such that the weight of each component is at least equal to ℓ and at most equal to u.

Perl and Schach [16] presented an elegant shifting algorithm for the max–min connected p-partition of a tree and Becker and Perl [17] found a variant of this algorithm for the min–max problem. Some other variants of the shifting algorithm are presented in subsequent papers (see, e.g., [18]). Becker et al. [19] proved that the above equipartition problems are NP-hard on grid graphs with three rows even in the case $p = 2$. For the max–min problem, polynomial time algorithms exist for ladder graphs [20], while pseudo-polynomial time algorithms on series parallel graphs and a Fully Polynomial Time Approximation Scheme (FPTAS) on interval graphs were presented in [21] and [22], respectively.

Many other criteria exist for partitioning a graph G into a fixed number of connected components as balanced as possible. Some authors consider the so called *Minimum Diameter* partitioning problem that

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.

consists of partitioning G into p connected subgraphs minimizing either the maximum difference between the largest and the smallest weight in each subgraph or the sum of these (maximum) differences (see, e.g., [23,24,13]). The variety of equipartition criteria introduced in the literature for partitioning problems is rather wide, and, obviously, it is beyond the scope of the paper to provide a complete review on this subject.

In this paper we consider a tree graph and study both the centered and non centered versions of MUP and UP problems. For MUP we provide a polynomial time algorithm for the centered and non centered cases with a weight-based objective function (Section 3.1). This establishes an important result since the computational complexity of this problem was set as an open problem for several years in the literature. On the other hand, the MUP problem becomes difficult when the centered version with flat costs is considered. In fact, we show that it is NP-hard on trees (Section 3.2). UP non centered problems on trees were already studied in [17,16]. Here we study four centered UP problems on trees, i.e., finding the max–min and the min–max connected p-partition of a tree in both cases of a cost- or weight-based objective function. For all problems we provide polynomial time algorithms (Sections 4.1 and 4.2) some of them are found by exploiting results in [17]. For the min–max cost p-centered partition problem, we propose a new polynomial time algorithm with overall time complexity of $O(n^2p)$.

2. Notation and definitions

Given a tree $T = (V, E)$, we assign non negative weights w_v to the vertices $v \in V$, with $|V| = n$, and consider an integer $2 \leq p \leq |V| - 1$. A p-partition of T is a partition of the set V into p non empty subsets, $\{\mathcal{C}_1, \ldots, \mathcal{C}_j, \ldots, \mathcal{C}_p\}$, such that each subset induces a subtree of T. The radius r of T is the minimum *eccentricity* of the vertices in V, where the eccentricity of $v \in V$ is the maximum of the distances from v to a vertex in the tree. Let T_q be the tree rooted at a vertex q. For some problems analyzed in this paper the tree does not have to be rooted, but for some other cases a root is needed in order to facilitate the algorithms. The root implies that all the edges are directed away from q and, given a vertex v, we denote by T_v the subtree of T_q rooted at v. An edge $e = (v, u)$ emanating from v in T_v means that u belongs to the set of children of v. If for a vertex v the set of its children is empty, then v is a terminal vertex or a leaf of T_q. Given a subtree T_v and an edge e emanating from v, we denote by $T_{v,e}$ the subtree of T_q formed by $T_u \cup (v, u)$.

In addition to general trees we also consider centered trees. In this case we assume that V is partitioned into two subsets S and U such that $S \subset V$ with $|S| = p$. S is the set of *centers* and $U = V \setminus S$ is the set of *units*. In centered trees we consider also a cost function $c : U \times S \to \mathbb{R}^+ \cup \{0\}$ which associates a cost c_{is} to each pair (i, s), $i \in U$, $s \in S$. We assume that the costs are *flat*, i.e., for each unit i the costs c_{is} are independent of the topology of T. A *centered partition* of T is a p-partition such that each subset contains exactly one vertex in S. The *cost of the component* $\mathcal{C}_s$ centered in s is defined as the sum of the costs c_{is} of the units $i \in \mathcal{C}_s$. The *weight of the component* $\mathcal{C}_s$ is given by the sum of the weights of the vertices $v \in \mathcal{C}_s$.

In this paper, given a tree T, we study the following problems:

(i) Find a partition of T into p connected components that minimizes the difference between the maximum and the minimum cost or weight of a component (most uniform connected p-partition problem).
(ii) Find a partition of T into p connected components that (i) maximizes the minimum cost or weight of a component (max–min connected p-partition problem) (ii) minimizes the maximum cost or weight of a component (min–max connected p-partition problem).

All the above listed problems are analyzed both in the centered and non centered cases. Problem (i) belongs to the class of MUP problems, while (ii) belongs to the class of UP problems. In particular, for the weight version of (i) we provide, for the first time in the literature, polynomial time algorithms, while for the cost version we prove that the most uniform connected p-partition problem is NP-hard even on trees. In addition, we solve UP problems by adapting already existing approaches for all but one of them. In fact, for the cost version of the min–max connected p-partition problem we introduce a new efficient polynomial time procedure.

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.

3. Most uniform partition of trees

The MUP problem was widely studied in the literature. NP-hardness results were established for arbitrary graphs and polynomial time algorithms were provided for very special classes of graphs like paths [14]. To the best of our knowledge, this problem is still open on trees. Here, we close this gap by providing a polynomial time algorithm for MUP, both in the centered and non centered cases, when a weight-based objective function is considered. Conversely, we show that in the centered case and with flat costs, the problem is NP-hard on trees. This also implies that even the feasibility problem of finding an $(\ell; u)$-partition of a tree is NP-hard.

3.1. Most uniform weighted partition problem

In this section we study the MUP problem on trees and a weight-based objective function. We solve the problem in polynomial time. The general idea of the algorithm is based on the following fact. Consider an optimal partition of the given tree T, and let $\mathcal{C}_M$ and $\mathcal{C}_m$ denote the maximum and the minimum weight components, respectively. Since the graph is a tree, there exists a unique path joining these two components and at least one edge of this path must be cut in the optimal partition. This allows to identify two subtrees T_1 and T_2 such that the maximum weight of a component in the MUP optimal partition of T is equal to the optimal value of a min–max weight partitioning problem on one of the two subtrees, while the minimum weight of a component in the MUP optimal partition of T is equal to the optimal value of a max–min weight partitioning problem on the other subtree.

Then the MUP optimal partition can be found by repeatedly solving min–max and max–min problems on all possible pairs of subtrees T_1 and T_2 that can be obtained by removing from T one edge at a time. At each iteration, the algorithm finds a min–max weight partition of T_1 in k components and a max–min weight partition of T_2 in $(p-k)$ components (and vice versa), identifying the two corresponding optimal values U and L. These two values are used to check if there exists a $(L;U)$-partition of the whole tree T into p connected components.

The following procedure solves the most uniform non centered partition problem on trees with a weight-based objective function.

---

algorithm MUP
  set $D^* := \sum_{v \in V} w_v$
  for each $e \in E$
    let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be the subtrees of T obtained by removing e
    for $k = 1, \ldots, min\{p-1, |V_1|\}$
      find a min–max k-partition of T_1 and let U be the maximum weight of a component
      find a max–min $(p-k)$-partition of T_2 and let L be the minimum weight of a component
      if $0 \leq U - L < D^*$ then
        if there exists a $(L;U)$-partition π_{LU} of T into p connected components
          set $D^* := U - L$ and $\pi^* := \pi_{LU}$
      find a max–min k-partition of T_1 and let L be the minimum weight of a component
      find a min–max $(p-k)$-partition of T_2 and let U be the maximum weight of a component
      if $0 \leq U - L < D^*$ then
        if there exists a $(L;U)$-partition π_{LU} of T into p connected components
          set $D^* := U - L$ and $\pi^* := \pi_{LU}$
  π^* is a MUP partition
  D^* is the difference between the maximum and the minimum weight of a component in π^*

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.

Theorem 1. *The MUP problem on trees can be solved in $O(p^5n^2)$ time.*

Proof. The proof is based on the following remarks.

- Let π^* be an optimal MUP partition of T into p connected components, and let $\mathcal{C}_M$ and $\mathcal{C}_m$, be two components having maximum and minimum weight, respectively. W.l.o.g. we can assume $\mathcal{C}_M \neq \mathcal{C}_m$. Since T is a tree there is a unique path $P = (v_1, \ldots, v_q)$ joining $\mathcal{C}_M$ and $\mathcal{C}_m$ such that $v_1 \in \mathcal{C}_M$, $v_q \in \mathcal{C}_m$ and $v_i \notin \mathcal{C}_M \cup \mathcal{C}_m, i \neq 1, q$.
- Since $\mathcal{C}_M$ and $\mathcal{C}_m$ are different, at least one edge (v_i, v_{i+1}) in P is cut in π^*.
- Let T_M and T_m be the two subtrees obtained by removing (v_i, v_{i+1}) from T containing $\mathcal{C}_M$ and $\mathcal{C}_m$, respectively. Let π^*_M and π^*_m be the partitions induced by π^* in T_M and T_m, respectively, and let k and $p-k$ be the corresponding number of components.
  Then π^*_M must be an optimal solution of the min–max k-partition problem on T_M, otherwise π^* would not be a MUP partition of T into p components. Similarly, π^*_m must be an optimal partition of the max–min $(p-k)$-partition problem on T_m.

It follows that π^*_M and π^*_m can be found by an exhaustive search in T which removes one edge at a time and solves the min–max k-partition problem in one subtree and the max–min $(p-k)$-partition problem on the other (and vice versa) for all possible values of k and $(p-k)$. Since these two problems are solved independently, at each iteration one must check whether the current min–max and max–min optimal values U and L are feasible or not w.r.t. the $(L;U)$-partition problem into p connected components of the whole tree T. From a complexity viewpoint, the max–min and min–max p-partition problems can be solved in $O(p^2r + pn)$ time [17] and in $O(rp(p + \log d) + n)$ time [25], respectively, where r is the radius of T and d the maximum degree of a vertex. For the $(L;U)-$ partition problem [15] provides an $O(p^4n)$ time algorithm. Since these problems must be solved twice for all edges of T and values of k in algorithm `MUP`, the overall time complexity for solving the MUP problem is $O(p^5n^2)$. □

For the centered version of the above MUP problem, it can be solved using the same algorithm, but after the introduction of modified weights for the vertices of the tree. Consider the subset $S \subset V$ of the centers of T and let $\overline{W} = \sum_{i \in U} w_i$. We define the following weights:

$$\overline{w}_v = \begin{cases} w_v & \text{if } v \in U \\ w_v + \overline{W} & \text{if } v \in S \end{cases} \tag{1}$$

Adopting these weights, any connected p partition of T containing more than one center in a component will never be optimal for MUP. Therefore, algorithm `MUP` will produce only p-centered partitions. Since the tree is centered, at each iteration of algorithm `MUP` the number of components in the two subtrees T_1 and T_2 is fixed. This implies that the overall time complexity to solve the centered version of the MUP problem is $O(n^2p^4)$.

3.2. Most uniform centered partition problem with flat costs

In the following we prove that the Most Uniform Centered Partition (MUCP) problem with flat costs is NP-complete on trees. The decision version of MUCP can be stated as follows:

Input: A tree $T = (V, E)$, two subsets $U, S \subseteq V$, with $|S| = p$, such that $U \cap S = \emptyset$ and $U \cup S = V$. Non negative integer flat costs c_{is}, $\forall\, i \in U$ and $s \in S$ and a rational $K \geq 0$.
Question: Is there a p-centered partition such that the difference between the maximum and minimum costs of the components is less than or equal to K?

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.

Theorem 2. *The Most Uniform Centered Partition problem with flat costs is NP-complete on trees.*

Proof. The reduction is from SUBSET-SUM which is a well-known NP-complete problem [26]: given n positive integers a_i, $i = 1, \dots, n$, with $\sum_{i=1}^{n} a_i = Q$, is there a subset $H \subseteq \{1, \dots, n\}$ such that $\sum_{i \in H} a_i = \frac{Q}{2}$?

Let us consider the tree T in Fig. 1, where the $2n+1$ circular vertices are the units of T, and the $n+1$ squared vertices represent the centers of T. For each unit $i = 0, \dots, n$ assign the following flat costs:

$$\begin{cases} c_{0s_k} = 0, & \text{if } k = 0 \\ c_{0s_k} = M, & \text{if } k = 1, \dots, n \\ c_{is_0} = a_i, & \forall\, i = 1, \dots, n \\ c_{is_k} = 0, & \forall\, i = 1, \dots, n \text{ and } k = i \\ c_{is_k} = M, & \forall\, i = 1, \dots, n \text{ and } k \neq i \end{cases}$$

with $M \gg Q$. To each unit $n+i$, $i = 1, \dots, n$, we assign the following costs:

$$\begin{cases} c_{(n+i)s_k} = \dfrac{Q}{2}, & \forall\, i = 1, \dots, n, \text{ with } k = i \text{ or } k = 0 \\ c_{(n+i)s_k} = M, & \forall\, i = 1, \dots, n \text{ and } k \neq i \end{cases}$$

Set $K = 0$. Assume we have a solution of SUBSET-SUM, H, then we obtain a solution for MUCP by assigning units i, $i \in H$, to center s_0, and units $n+i$, $i \in H$, to center s_i. Note that unit $i = 0$ is always assigned to s_0. We assign all the other units labeled k and $n+k$, $k \notin H$, to the corresponding centers s_k. In this way, we have a (connected) centered partition in which all the component's costs are equal to $\frac{Q}{2}$, so that the difference between the maximum and minimum cost of the components is exactly equal to $K = 0$.

Assume now that we have a solution of MUCP where the difference between the maximum and minimum costs of the components is equal to 0. Such a solution can be obtained only if the unit $i = 0$ is assigned to s_0, otherwise the component centered in s_0 has cost equal to 0, and the maximum cost of a component of the partition is equal to $M + \frac{Q}{2}$ (it is given by the component to which $i = 0$ is assigned). In addition, we have to assign a subset of units to center s_0 in order to obtain that the cost of the component centered in s_0 is equal to $\frac{Q}{2}$. Let H be the index set of units assigned to s_0. All the other units with label k, $k \notin H$, must be assigned to the corresponding centers s_k, as well as those with labels $n+k$, $k = 1, \dots, n$. Thus, we obtain a (connected) centered partition where the cost of each component is equal to $\frac{Q}{2}$, so that the difference between the maximum and minimum costs is 0. To satisfy the connectivity constraint, any other feasible centered partition has at least a component with cost greater than M and one with cost equal to 0. Thus, H is a solution of SUBSET-SUM. Hence, solving MUCP with flat costs on a tree is the same as solving SUBSET-SUM. Since it is easy to verify that MUCP is in NP, it is NP-complete. □

Corollary 1. *The problem of finding a centered $(L;U)$-partition with flat costs is NP-complete on trees.*

Proof. We can exploit the proof of Theorem 2. Actually, it suffices to consider $L = \frac{Q}{2} - \delta$ and $U = \frac{Q}{2}$, with $0 < \delta < 1$ and rational. □

4. Uniform centered partitions of trees

The main results on UP problems on trees were presented in [17,16] where the authors provide a general technique for partitioning trees with different objectives that is based on *shifting* operations and greedy decisions. Specializing this technique with a suitable choice for weights and costs, in this section we provide polynomial time algorithms for three out of our four UP problems. On the other hand, for the min–max UP problem with flat costs, we develop an efficient optimization procedure based on the solution of a sequence of feasibility problems.

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.

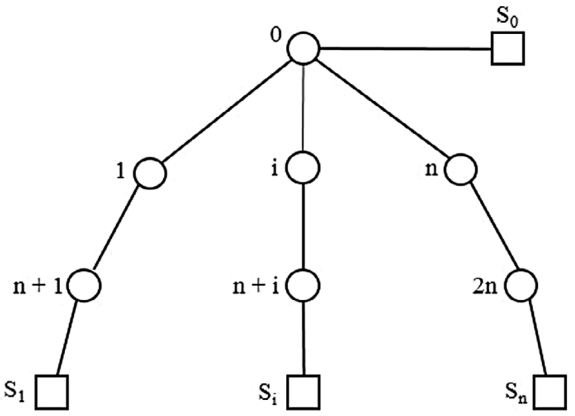

Fig. 1. The tree T used in the proof.

4.1. Max–min centered partition of trees

In this section we study the max–min (cost/weight) centered partition problem of a tree T and we show how this problem can be solved in polynomial time by using results from [17]. Given the family $\mathcal{F}$ of all the possible subsets of V, they define a weighting function $H : \mathcal{F} \to \mathbb{R}^+ \cup \{0\}$ that assigns a weight $H(\mathcal{Z})$ to each subset $\mathcal{Z}$ in $\mathcal{F}$. Among others, they solve the problem of finding a partition of T into p connected components, $\{\mathcal{Z}_1, \ldots, \mathcal{Z}_p\}$, that maximizes the minimum of $H(\mathcal{Z}_j)$, $j = 1, \ldots, p$, by applying a shifting algorithm originally proposed in [16]. We refer to this problem as *BP-max–min problem* and observe that the only difference with our problem is that [17] does not consider centered partitions. For the BP-max–min problem the shifting algorithm applies when $H(\cdot)$ is a *basic* weighting function, i.e., a function satisfying the following property: if $\mathcal{Z}_1, \mathcal{Z}_2 \in \mathcal{F}$ are such that $\mathcal{Z}_1 \subseteq \mathcal{Z}_2$ then $H(\mathcal{Z}_1) \leq H(\mathcal{Z}_2)$.

Consider our problem of finding a max–min cost centered partition $\{\mathcal{C}_1, \ldots, \mathcal{C}_p\}$ of T. Let $M = \sum_{i \in U} \max_{s \in S} c_{is}$. For a subset $\mathcal{C}$ of V we define the above weighting function $H(\cdot)$ as follows:

$$H(\mathcal{C}) = \begin{cases} M|\mathcal{C} \cap S| + \displaystyle\sum_{i \in \mathcal{C} \cap U} \max_{s \in \mathcal{C} \cap S} c_{is} & \text{if } \mathcal{C} \cap S \neq \emptyset \\ \displaystyle\sum_{i \in \mathcal{C}} \min_{s \in S} c_{is} & \text{if } \mathcal{C} \cap S = \emptyset \end{cases} \tag{2}$$

It is easy to see that the above weighting function is basic. Notice that when $\{\mathcal{C}_1, \ldots, \mathcal{C}_p\}$ is a centered partition, for a component $\mathcal{C}_s$ centered in s one has:

$$H(\mathcal{C}_s) = M + \sum_{i \in \mathcal{C}_s \cap U} c_{is} \tag{3}$$

Theorem 3. *A partition $\{\mathcal{C}_1, \ldots, \mathcal{C}_p\}$ is an optimal solution of the max–min cost centered partition problem on a tree T if and only if it is an optimal solution of the BP-max–min problem with weighting function $H(\cdot)$.*

Proof. Consider the following facts.

i. Since T is a tree (and therefore it is connected) and $S \subset V$, there always exists a centered partition.
ii. Any optimal solution of BP-max–min with weighting function $H(\cdot)$ is centered. Actually, in any partition of T into p connected components that is not centered there is always (at least) a component that does not contain any center, implying that the weight of this component is smaller than the minimum weight of any centered partition.
iii. For any centered partition, the weight $H(\mathcal{C}_j)$ of a component $\mathcal{C}_j$ centered in s_j is given by formula (3).

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.

The above facts imply that any optimal solution to our max–min cost centered partition problem with basic weighting function $H(\cdot)$ is optimal also for BP-max–min and vice versa. □

From Theorem 3 it follows that the max–min cost centered partition problem can be solved by the shifting algorithm for the BP-max–min problem in $O(p^2r+pn)$ time, where r is the radius of T. We observe that the same basic weighting function (2) and the same shifting algorithm can be applied also to solve our max–min weight centered partition problem by setting for each $i \in U$: $c_{is} = w_i$, $\forall s \in S$.

4.2. Min–max centered partition of trees

In [17] Becker and Perl also provide a shifting algorithm for the problem of finding a partition of a tree into p connected components that minimizes the maximum weight of a component (*BP-min–max problem*). The algorithm works under the condition that the weighting function $H(\cdot)$ (defined in Section 4.1) on the family $\mathcal{F}$ is *invariant*. The min–max weight centered p partition problem on T can be solved in polynomial time by exploiting the shifting algorithm in [17], as illustrated in the following.

According to [17], let v be any vertex of the rooted tree T_q and $e_1 = (v, u_1)$ and $e_2 = (v, u_2)$ two edges emanating from v in T_v. Denote by T_{v,e_1} and T_{v,e_2} the subtrees given by $T_{u_1} \cup (v, u_1)$ and $T_{u_2} \cup (v, u_2)$, respectively. Let T' denote a subtree of T_q in which v is a leaf. The weighting function $H(\cdot)$ is invariant if:

(i) $H(\cdot)$ is basic;
(ii) for any vertex v and any pair of edges $e_1 = (v, u_1)$ and $e_2 = (v, u_2)$ emanating from T_v one has:

$$H(T_{u_1} \cup (v, u_1)) \leq H(T_{u_2} \cup (v, u_2)) \;\Rightarrow\; H(T_{u_1} \cup (v, u_1) \cup T') \leq H(T_{u_2} \cup (v, u_2) \cup T').$$

where in the notation we refer to a component of the partition of T as a subtree of the rooted tree T_q.

For the min–max weight centered p partition problem on T, given a subset $\mathcal{C}$ of V, we define the weighting function $H(\cdot)$ as follows:

$$H(\mathcal{C}) = \sum_{v \in \mathcal{C}} \overline{w}_v$$

where we consider the weights $\overline{w}_v$ defined in formula (1) in order to take into account that our partition is centered. Since $H(\mathcal{C})$ is the sum of the weights of the vertices in $\mathcal{C}$, it is among the weighting functions that Becker and Perl proved to be invariant. For this reason their algorithm applies leading to the following theorem for which the proof is analogous to the one of Theorem 3.

Theorem 4. *A partition $\{\mathcal{C}_1, \ldots, \mathcal{C}_p\}$ is an optimal solution of the min–max weight centered partition problem on a tree T if and only if it is an optimal solution of the BP-min–max problem with weighting function $H(\cdot)$.*

The most efficient implementation of the shifting algorithm for the BP-min–max problem was provided by Perl and Vishkin in [25] and requires $O(rp(p + \log d) + n)$ time, where r and d are the radius and the maximum degree of a vertex, respectively.

When the min–max centered p partition on T is formulated by an objective function based on costs instead of weights, the application of the shifting algorithm in [17] is not straightforward. In fact, the weighting function $H(\cdot)$ defined in formula (2) is not invariant as the following simple example shows.

Example. Consider the simple tree T in Fig. 2a, where v, w, and z are units, with w and z two leaves, and $\bar{s}$ a center of T.

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.

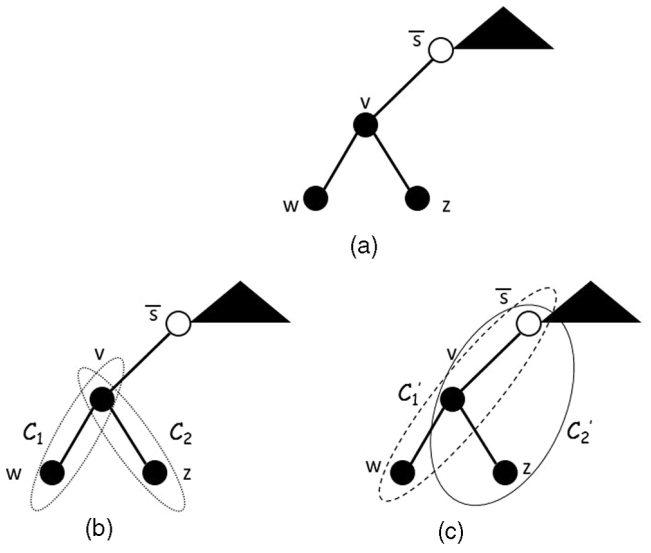

Fig. 2. The tree T used for the example.

Let $\min_{s\in S} c_{vs} = \min_{s\in S} c_{ws} = 1$ and $\min_{s\in S} c_{zs} = 2$. Let $c_{v\bar{s}} = c_{w\bar{s}} = 3$, and $c_{z\bar{s}} = 2$ be the assignment costs of the three units.

Denote by $T_{v,e_1} = (v, w) = \mathcal{C}_1$ and $T_{v,e_2} = (v, z) = \mathcal{C}_2$ the two components with $H(\mathcal{C}_1) = 2$ and $H(\mathcal{C}_2) = 3$ (Fig. 2b). According to the definition of invariant functions, consider $T' = (\bar{s}, v)$, so that $\mathcal{C}'_1 = (\bar{s}, v) \cup (v, w)$ and $\mathcal{C}'_2 = (\bar{s}, v) \cup (v, z)$ (Fig. 2c). We have $H(\mathcal{C}_1) < H(\mathcal{C}_2)$, but:

$$H(\mathcal{C}'_1) = M + 6 > H(\mathcal{C}'_2) = M + 5.$$

with $M = \sum_{i\in U} \max_{s\in S} c_{is}$. Thus, the weighting function $H(\cdot)$ in formula (2) is not invariant.

For the min–max cost centered p partition we propose a new polynomial time algorithm based on the solution of a sequence of feasibility problems. At each iteration, for a fixed $\delta > 0$, the algorithm finds a centered partition with maximum component cost bounded above by δ (δ-*centered partition*). Note that since T is a tree, a unit i cannot be assigned to a center s such that the unique path from i to s contains another center $s' \neq s$. As a consequence, we can suppose that all leaves of T are centers. For a fixed value δ, if a δ-centered partition of T exists, it can be found by visiting bottom-up T rooted at a leaf r (denoted by T_r). Let T_i be the subtree of T_r rooted at i, S_i the set of its centers, and $p(i)$ the parent of i in T_r, $i \neq r$. The idea of the algorithm is to add as much cost as possible to the components in the bottom part of the tree without exceeding the given limit δ. If a unit i can be assigned to a center in T_i, such center is selected in S_i as the one that minimizes the sum of the costs; if not, i must be assigned to the same center as $p(i)$ in $S \setminus S_i$. In this way, during the algorithm, for the current vertex i and for each center $s \in S_i$ we are able to record the minimum cost of a component containing i and s. A δ-centered partition of T exists if, at the end, all these costs are smaller than or equal to δ. During the algorithm we compute the following quantities:

- $\bar{c}(i, s)$, $i \in V$ and $s \in S$: the sum of the costs c_{hs} of the units h in T_i that must be assigned to the same center as i in any δ-centered partition of T;
- $w^*(i, s)$, $i \in V$, $s \in S_i$: the minimum cost of a component containing i and s in a centered partition of T_i whose components have cost at most δ. At the beginning we set $w^*(i, s) = M > \delta$, $i \in V$, $s \in S_i$;
- $s^*(i)$, $i \in V$: center to which vertex i is assigned to in the final δ-centered partition if it exists; notice that for any center s we have $s^*(s) = s$.

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.

We also introduce the binary indicator $r(p(i), i)$, $i \in U$, which is set to 1 when i must be necessarily assigned to the same center as its parent $p(i)$ in any δ-centered partition of T.

---

algorithm δ-`centered partition`
Part 1: verify feasibility
root T at a leaf r
for each $i \in V$ and $s \in S$ set $\bar{c}(i, s) := c_{is}$
for each $i \in U$ and $s \in S$ set $w^*(i, s) := M > \delta$
for each $s \in S$ set $w^*(s, s) := c_{ss}$
for each $i \in V \backslash \{r\}$ set $r(p(i), i) := 0$
visit T_r bottom-up starting from its leaves
  if the visited vertex is a unit i
    for each j such that $p(j) = i$ and $r(i, j) = 0$
      for each $s \in S_j$ such that $w^*(j, s) \leq \delta$ set $w^*(i, s) := w^*(j, s) + \bar{c}(i, s)$
    if $w^*(i, s) > \delta$ for all $s \in S_i$ then
      set $r(p(i), i) := 1$ and $\bar{c}(p(i), s) := \bar{c}(p(i), s) + \bar{c}(i, s)$ for all $s \in S \backslash S_i$
  else if the visited vertex is a center $s \in S$
    if $\bar{c}(s, s) > \delta$ then STOP: the problem is infeasible
Part 2: find a δ-centered partition
for each $s \in S$ set $s^*(s) := s$
visit T_r top down level by level and for each unassigned unit i
  if $r(p(i), i) = 1$ then
    set $s^*(i) := s^*(p(i))$
  else
    find a center s in T_i such that $w^*(i, s) \leq \delta$
    visit bottom up the path from s to i and for each unit j in the path set $s^*(j) := s$

Theorem 5. *Algorithm δ-`centered partition` correctly finds a δ-centered partition of T, if it exists, or it establishes that the problem is infeasible in $O(np)$ time.*

Proof. During the algorithm, for each visited vertex i and for each $s \in S_i$, $w^*(i, s)$ and $\bar{c}(s, s)$ are computed coherently with the definitions given before. This is true at the beginning when the first visited vertex is a leaf and therefore it is a center. Suppose that the values are correct at a given iteration of the algorithm. Case 1: the next vertex to be visited is a unit i. Suppose that j is a child of i and notice that in any partition where i is assigned to a center in S_j also j must be assigned to the same center. Hence in order to evaluate $w^*(i, s)$ for $s \in S_j$ one has to consider the value $w^*(j, s)$, that has been already computed by the algorithm. If $r(i, j) = 1$ then $w^*(j, s') > \delta$ for all $s' \in S_j$ and i cannot be assigned to s. Otherwise $w^*(i, s)$ is given by $w^*(j, s) + \bar{c}(i, s)$. If this value is less than or equal to δ, i can be assigned to s; otherwise i cannot be assigned to s and it must be assigned to the same center as its parent $p(i)$. In the latter case the costs of $p(i)$ to the centers in $S \setminus S_i$ must be updated by adding the cost of unit i. Therefore, the values of $w^*(i, s)$ for $s \in S_i$ and $\bar{c}(i, s)$ for $s \in S \setminus S_i$ are correct. Case 2: the next vertex to be visited is a center s and $\bar{c}(s, s) > \delta$ then in any feasible partition the cost of the component containing s will be greater than δ and then the problem is infeasible. For the complexity, note that Part 1 of the algorithm considers each pair i, s, $i \in V$ and $s \in S$ at most once and for each pair the computation requires a constant time. Hence, Part 1 has time complexity $O(np)$. Part 2 visits each vertex once and for each vertex the computation requires a time proportional to the number of centers. Hence Part 2 has time complexity $O(np)$ and the overall time complexity is $O(np)$. □

In order to solve the min–max centered partition problem, algorithm δ-`centered partition` must be repeatedly applied for different values of δ. We observe that not all possible values are relevant, but only

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.

a restricted set of them must be considered. The following algorithm finds a connected min–max centered partition of T with flat costs efficiently by implementing an appropriate scheme for the selection of the values of δ to analyze.

---

algorithm `min-max centered partition`
  let $\delta = \sum_{i \in U} min_{s \in S} c_{is}$ (δ is a lower bound of the optimal value)
  repeat
    verify the feasibility of the δ-centered partition problem by applying
    Part 1 of the algorithm δ-`centered partition`
    if the problem is infeasible then
      set δ equal to the minimum among the $w^*(i,s)$ such that $r(p(i),i) = 1$
  until the δ-centered partition is feasible
  find an optimal partition by applying Part 2 of the algorithm δ-`centered partition`

Theorem 6. *Algorithm* ***min-max centered partition*** *correctly solves the min–max cost centered partition problem on a tree in $O(n^2 p)$ time.*

Proof. Suppose that for $\delta = \delta_1$ the δ-centered partition problem is infeasible. In this case, when algorithm δ-`centered partition` stops, $\bar{c}(s', s') > \delta_1$ for some visited center s'. Let δ_2 be the minimum among the values $w^*(i,s)$ such that $r(p(i),i) = 1$, and notice that $\delta_2 > \delta_1$. By applying again algorithm δ-`centered partition` with $\delta = \delta'$, $\delta_1 < \delta' < \delta_2$, nothing would change in the algorithm w.r.t. δ_1 and, in particular, the value of $\bar{c}(s', s')$ would be equal to the one obtained for δ_1 and then the problem would be again infeasible. It follows that δ_2 is a lower bound of the optimal value of the min–max partitioning problem and can be considered as next value of δ.

Note that, if i, s is the pair such that $w^*(i,s) = \delta_2$, for all $\delta \geq \delta_2$ at the end of the algorithm we will have $r(p(i),i) = 0$ and then unit i is not considered again for finding a new value of $\delta > \delta_2$. It follows that the number of different values of δ to consider is $O(n)$ and the overall time complexity for the min–max centered cost partition problem is $O(n^2 p)$. □

An alternative approach compared to algorithm `min-max centered partition` consists in a binary search on all the possible values of δ with a time complexity of $O(np \log \bar{C})$, where $\bar{C}$ is an upper bound on the cost of a component (for example $\bar{C} = \sum_{i \in U} \max_{s \in S} c_{is}$). If we assume that the costs are bounded above by some polynomial $M(n,p)$, this approach provides an improved time complexity of $O(np \log n)$.

5. Conclusions

Equipartition criteria define an important family of partition problems. In this paper we have restricted ourselves to consider two types of equipartition problems on graphs: Most Uniform Partition and Uniform Partition problems. Most of our results are devoted to tree graphs where we provide polynomial time algorithms for: 1) the centered and non centered MUP problems with a weight-based objective function; and 2) centered UP problems with a weight or flat cost-based objective function. The former results set the polynomial complexity of MUP problem on trees which was open before in the literature. For the centered MUP problem with flat costs, we have proven that it is NP-complete even on trees.

Our results provide efficient polynomial time algorithms for equipartition problems on trees, but it is also interesting to address similar problems on general graphs, where these algorithms do not apply especially for large networks (huge number of vertices). Future research in this area should develop exact algorithms for equipartition problems on general graphs related to large data set (big data) based on integer programming

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.

formulations. We are also interested in the study of heuristic algorithms capable to provide good quality solutions whenever the exact methods are limited by the complexity of the problems.

Acknowledgment

The second, third and fourth authors acknowledge the financial support by the Spanish research grant MTM2017-74983-C02-01.

References

[1] A.A. Abbasi, M. Younis, A survey on clustering algorithms for wireless sensor networks, Comput. Commun. 30 (2007) 2826–2841.
[2] F.D. Malliarosa, M. Vazirgiannisa, Clustering and community detection in directed networks: A survey, Phys. Rep. 533 (2013) 95–142.
[3] M. Megiddo, A. Tamir, New results on the complexity of p-center problems, SIAM J. Comput. 12 (1983) 751–758.
[4] D. Sen, N. Gupta, S.K. Pal, Incorporating local image structure in normalized cut based graph partitioning for grouping of pixels, Inform. Sci. 248 (2013) 214–238.
[5] D.A. Bader, H. Meyerhenke, P. Sanders, D. Wagner (Eds.), Graph Partitioning and Graph Clustering, in: Contemporary Mathematics, vol. 588, American Mathematical Society, Providence, Rhode Island, 2013.
[6] N. Apollonio, I. Lari, J. Puerto, F. Ricca, B. Simeone, Polynomial algorithms for partitioning a tree into single-center subtrees to minimize flat service costs, Networks 51 (2008) 78–89.
[7] I. Lari, J. Puerto, F. Ricca, A. Scozzari, Partitioning a graph into connected components with fixed centers and optimizing cost-based objective functions or equipartition criteria, Networks 67 (2016) 69–81.
[8] O. Berman, D. Krass, M. Menezes, Facility reliability issues in network p-median problems: strategic centralization and co-location effects, Oper. Res. 55 (2007) 332–350.
[9] J. Puerto, F. Ricca, A. Scozzari, Reliability problems in multiple path-shaped facility location on networks, Discrete Optim. 12 (2014) 61–72.
[10] P. Hansen, B. Jaumard, C. Meyer, B. Simeone, V. Doring, Maximum split clustering under connectivity constraints, J. Classification 20 (2003) 143–180.
[11] B. Bozkaya, E. Erkut, G. Laporte, A tabu search heuristic and adaptive memory procedure for political districting, European J. Oper. Res. 144 (2003) 12–26.
[12] F. Ricca, B. Simeone, Local search algorithms for political districting, European J. Oper. Res. 189 (2008) 1409–1426.
[13] X. Tang, A. Soukhal, V. T'kindt, Preprocessing for a map sectorization problem by means of mathematical programming, Ann. Oper. Res. 222 (2014) 551–569.
[14] M. Lucertini, Y. Perl, B. Simeone, Most uniform path partitioning and its use in image processing, Discrete Appl. Math. 42 (1993) 227–256.
[15] T. Ito, T. Nishizeki, M. Schroeder, T. Uno, X. Zhou, Partitioning a weighted tree into subtrees with weights in a given range, Algorithmica 62 (2012) 823–841.
[16] Y. Perl, S. Schach, Max–min tree partitioning, J. ACM 28 (1981) 5–15.
[17] R.I. Becker, Y. Perl, The shifting algorithm technique for the partitioning of trees, Discrete Appl. Math. 62 (1995) 15–34.
[18] E. Agasi, R.I. Becker, Y. Perl, A shifting algorithm for constrained min–max partition on trees, Discrete Appl. Math. 45 (1993) 1–28.
[19] R.I. Becker, I. Lari, M. Lucertini, B. Simeone, Max–min partitioning of grid graphs into connected components, Networks 32 (1998) 115–125.
[20] R.I. Becker, I. Lari, M. Lucertini, B. Simeone, A polynomial-time algorithm for max–min partitioning of ladders, Theory Comput. Syst. 34 (2001) 353–374.
[21] T. Ito, X. Zhou, T. Nishizeki, Partitioning a graph of bounded tree-width to connected subgraphs of almost uniform size, J. Discrete Algorithms 4 (2006) 142–154.
[22] B.Y. Wu, Fully polynomial-time approximation schemes for the max–min connected partition problem on interval graphs, Discrete Math. Algorithms Appl. 4 (2012) 1250005.
[23] J. Chlebikova, Approximability of the maximally balanced connected partition problem in graphs, Inform. Process. Lett. 60 (1996) 225–230.
[24] M. Maravalle, B. Simeone, R. Naldini, Clustering on trees, Comput. Statist. Data Anal. 24 (1997) 217–234.
[25] Y. Perl, U. Vishkin, Efficient implementation of a shifting algorithm, Discrete Appl. Math. 12 (1985) 71–80.
[26] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Co., New York, NY, USA, 1979.

Please cite this article in press as: I. Lari, et al., Uniform and most uniform partitions of trees, Discrete Optimization (2018), https://doi.org/10.1016/j.disopt.2018.06.003.