Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in DISCRETE MATHEMATICS

Electronic Notes in Discrete Mathematics 55 (2016) 37–40

www.elsevier.com/locate/endm

Algorithms for uniform centered partitions of trees

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

[a] *Sapienza University of Rome, Italy*

[b] *IMUS, Instituto de Matemáticas de la Universidad de Sevilla, Spain*

[c] *University Niccolò Cusano, Rome, Italy*

Abstract

In this paper we provide polynomial time algorithms for the problem of finding uniform centered partitions of a tree, that is, partitions that are as balanced as possible either w.r.t. the costs or to the weights of their components.

Keywords: Tree partitioning, centered partitions, flat costs, min-max criteria, uniform partitions.

1 Introduction, notation and definitions

Let $T = (V, E)$ be a tree with $|V| = n$. 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* (*facilities*) and $U = V \backslash S$ is the set of *units* (*clients*). We consider 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 these costs are *flat*, i.e., they are independent of the topology of T. We also consider a nonnegative weight w_i associated to each i in U. A *centered partition* of T is a partition of the set V into p non empty subsets, $\{\mathcal{C}_1, \ldots, \mathcal{C}_p\}$, such that each subset induces a subtree of T and contains exactly one center. 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 units $i \in \mathcal{C}_s$. We consider the flat costs and study the following two problems: i) *max-min cost* centered partition problem, that is, find a centered partition of T that maximizes the minimum cost of a component; ii) *min-max cost* centered partition problem, that is, find a centered partition of T that minimizes the maximum cost of a component.

Replacing the minimum and maximum cost by the minimum and maximum weight of a component we obtain the following variants of the above problems: iii) *max-min weight*

http://dx.doi.org/10.1016/j.endm.2016.10.010

1571-0653/© 2016 Elsevier B.V. All rights reserved.

centered partition problem; iv) *min-max weight* centered partition problem. For problems i)-iv) we provide polynomial time algorithms: for i), iii) and iv) we adapt already existing approaches, while for ii) we suggest a new procedure. This kind of problems are known as *uniform* partition problems, and they have been widely studied in the literature on trees [2,5]. In this paper we focus on the particular case of finding uniform centered partitions. In previous papers we already studied problems of this class on general graphs, providing several NP-completeness results for other types of graphs [1,3]. In particular, we proved that all the above problems are NP-complete even on planar bipartite graphs with vertex degree at most 3 and $p = 2$, and this motivates our interest for studying them now on trees.

2 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 [2], where Becker and Perl provide a general technique for partitioning trees with different objectives that is based on *shifting* operations and greedy decisions. 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 the 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 the $H(\mathcal{Z}_j)$, $j = 1, \ldots, p$, by applying a shifting algorithm originally proposed in [5]. We refer to this problem as *BP-max-min problem* and observe that the only difference with our problem is that [2] 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 generic subset $\mathcal{C}$ of V we introduce the following weighting function:

$$H(\mathcal{C}) = \begin{cases} M|\mathcal{C} \cap S| + \sum\limits_{i \in \mathcal{C} \cap U} \max\limits_{s \in \mathcal{C} \cap S} c_{is} & \text{if } \mathcal{C} \cap S \neq \emptyset \\ \sum\limits_{i \in \mathcal{C}} \min\limits_{s \in S} c_{is} & \text{if } \mathcal{C} \cap S = \emptyset \end{cases} \tag{1}$$

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:

$$H(\mathcal{C}_s) = M + \sum_{i \in \mathcal{C}_s \cap U} c_{is} \tag{2}$$

Theorem 2.1 *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 $H(\cdot)$.*

From Theorem 2.1 it follows that the max-min cost centered partition problem can be solved by the shifting algorithm for the BP-max-min problem in $O(p^2 r + pn)$ time, where r is the radius of T. We observe that the same basic weighting function (1) and the same

shifting algorithm can be applied also to solve our max-min weight centered partition problem by setting for each $i \in U$: $c_{is} = w_i$, $\forall s \in S$.

3 Min-max centered partition of trees

In [2] 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*) that applies when the weighting function $H(\cdot)$ is *invariant* (see [2] for the definition). Our min-max weight centered partition problem on T can be solved in polynomial time by exploiting the shifting algorithm in [2]. Let $W = \sum_{i \in U} w_i$ and assign the following weights:

$$\overline{w}_v = \begin{cases} w_v \text{ if } v \in U \\ W \text{ if } v \in S \end{cases} \tag{3}$$

It can be shown that the resulting weighting function that assigns to a component $\mathcal{C}$ a weight $\overline{H}(\mathcal{C}) = \sum_{v \in \mathcal{C}} \overline{w}_v$ is invariant.

Theorem 3.1 *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 $\overline{H}(\cdot)$.*

The most efficient implementation of the shifting algorithm for the BP-min-max problem was provided by Perl and Vishkin in [6] and requires $O(rp(p + \log d) + n)$ time, where r and d are the radius of T and the maximum degree of a vertex, respectively.

Finally, for the min-max cost centered partition problem we propose a new polynomial time algorithm based on the solution of a sequence of feasibility problems in which, at each iteration, a centered partition with maximum component cost bounded above by a quantity δ (δ-*centered partition*) must be identified. 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 assigning costs; if not, i must be assigned to the same center as its parent $p(i)$ in $S \backslash S_i$. 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, but at most the one containing s, have cost at most δ. At the beginning we set $w^*(i,s) = M > \delta$, $i \in V$, $s \in S_i$.

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. For any given δ, after a suitable initialization of the above quantities, the algorithm performs the following visit of T_r:

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
return $r(p(i), i)$, $\forall\, i \in U$

If a δ-centered partition exists, it can be found by a top-down visit of T_r using $r(p(i), i)$.

Theorem 3.2 *A δ-centered partition of T can be found in $O(np)$ time.*

By a binary search on all the possible values of δ one can find the min-max cost centered partition in $O(np \log \bar{C})$ time, where $\bar{C}$ is an upper bound on the cost of a component (for example $\bar{C} = \sum_{i \in U} \max_{s \in S} c_{is}$). Let $f(\delta)$ be the maximum cost of a component in a δ-centered partition of T. It is easy to see that $f(\delta)$ is an increasing stepwise linear function of δ whose number of steps is bounded above by 2^n. Using the approach in [4] one can search over the different δ values in an overall time complexity of $O(n^2 p)$.

References

[1] N. Apollonio, I. Lari, J. Puerto, F. Ricca, and B. Simeone, Polynomial algorithms for partitioning a tree into single-center subtrees to minimize flat service costs, *Networks* **51** (2008) 78–89.

[2] R.I. Becker and Y. Perl, The shifting algorithm technique for the partitioning of trees, *Disc. Appl. Math.* **62** (1995) 15–34.

[3] I. Lari, J. Puerto, F. Ricca and A. Scozzari, Partitioning a graph into connected components with fixed centers and optimizing cost-based objective functions or equipartition criteria, *Networks* **1** (2016) 69–81.

[4] M. Megiddo and A. Tamir, New results on the complexity of p-center problems, *SIAM J. Comput.* **12** (1983) 751–758.

[5] Y. Perl and S. Schach, Max-min tree partitioning, *J. of the ACM* **28** (1981) 5–15.

[6] Y. Perl and U. Vishkin, Efficient implementation of a shifting algorithm, *Disc. Appl. Math.* **12** (1985) 71–80.