Faster Algorithms for k-Medians in Trees (Extended Abstract)

Robert Benkoczi[1], Binay Bhattacharya[1], Marek Chrobak[2], Lawrence L. Larmore[3], and Wojciech Rytter[4]

[1] School of Computing Science, Simon Fraser University, Burnaby, BC, V5A 1S6, Canada – Research supported by NSERC and MITACS.
[2] Department of Computer Science, University of California, Riverside, CA 92521 – Research supported by grants EPSRC GR/N09077 (UK), CCR-9988360, CCR-0208856 (NSF) and ME-103/INT-9600919 from MŠMT (Czech Rep. and NSF).
[3] Department of Computer Science, University of Nevada, Las Vegas, NV 89154-4019 – Research supported by grant EPSRC GR/N09077 (UK).
[4] Instytut Informatyki, Uniwersytet Warszawski, Banacha 2, 02–097, Warszawa, Poland, and Department of Computer Science, and New Jersey Institute of Technology, Newark, New Jersey 07102-1982, USA – Research supported by grant EPSRC GR/N09077 (UK).

Abstract. In the k-median problem we are given a connected graph with non-negative weights associated with the nodes and lengths associated with the edges. The task is to compute locations of k facilities in order to minimize the sum of the weighted distances between each node and its closest facility. In this paper we consider the case when the graph is a tree. We show that this problem can be solved in time $O(n\text{polylog}(n))$ for the following cases: (i) directed trees (and any fixed k), (ii) balanced undirected trees, and (iii) undirected trees with $k = 3$.

1 Introduction

In the *k-median problem* we are given a graph G in which each node u has a non-negative weight w_u and each edge (u, v) has a non-negative length d_{uv}. We extend this notation to arbitrary $u, v \in G$, so that d_{uv} is the minimum distance between u, v in G. Our goal is to find a set F of k vertices that minimizes $cost_F(G) = \sum_{x \in G} \min_{u \in F} w_x d_{xu}$. The optimal cost of G with k facilities is $cost_k(G) = \min \{cost_F(G) \ : \ |F| \leq k\}$. We think of the nodes of G as customers, with each customer x having a demand w_x for some service. F is a set of k *facilities* that provide this service. It costs d_{xu} to provide a unit of service to the customer at x from the facility located at u. We wish to place k facilities in G so that the overall service cost is minimized. This optimal set is called the *k-median* of G. The k-median problem has been a subject of study for several decades. The general case is known to be NP-hard (see Problem ND51 in [9]), and most of the past work focused on efficient heuristics (see, for example, [14]) and approximation algorithms [1, 7, 6, 2].

Trees. The case when G is a tree has been also extensively studied. In 1979 Kariv and Hakimi [13] presented a dynamic programming algorithm for this problem that runs in time $O(k^2n^2)$. Hsu [12] gave an algorithm with running time $O(kn^3)$. More recently, Tamir [19] showed that the Kariv-Hakimi algorithm is faster than previously thought, namely that its running time is only $O(kn^2)$. This is the best upper bound known to this date.

Some other special cases of the problem have been investigated. The case $k = 2$ was considered by Gavish and Sridhar [10] who presented an algorithm with running time $O(n \log n)$. The dynamic version of the case $k = 2$ was studied by Auletta *et al.* in [3][5]. Hassin and Tamir [11] gave an $O(kn)$-time algorithm for k facilities when the tree is a path. Another $O(kn)$-time algorithm for the path was later given by Auletta *et al.* [4]. Recently, Shah and Farach-Colton [17] reported the first sub-quadratic algorithms for solving several facility location problems in trees. However, they leave open the question whether a sub-quadratic algorithm for k-medians is possible.

Directed Trees. In *rooted directed trees* the edges are directed towards the root. Let T be a directed tree with n nodes and root r. If y is the parent of x, denote by d_{xy} the length of edge (x, y). The length function extends in a natural way to any pair x, y where y is an ancestor of x. (If y is not an ancestor of x, we assume $d_{xy} = \infty$.) Our goal is to find a set of nodes F of size at most k that minimizes the quantity: $cost_F(T) = \sum_{x \in T} \min_{u \in F \cup \{r\}} w_x d_{xu}$. Note that in this case we actually place $k + 1$ facilities in the tree, since the "facility" at the root is fixed.

The k-median problem for directed trees was introduced by Li *et al.* [16] to model the placement of web proxies to minimize average latency. In [16], the authors presented a $O(k^2n^3)$-time algorithm for this problem. This was later improved to $O(k^2n^2)$ by Vigneron *et al.* [20]. In [20] the authors also noted that the Kariv-Hakimi algorithm can be modified to find k-medians in directed trees in time $O(k^2n^2)$. Tamir (private communication) observed that his method from [19] can be adapted to reduce the time complexity to $O(kn^2)$. In Chrobak *et al.* [8], efficient algorithms were given for $k = 2$ and $k = 3$, with complexities $O(n \log n)$ and $O(n \log^2 n)$, respectively. In the special case when the tree is a line, Li *et al.* [15] gave an algorithm with running time $O(kn^2)$. This was subsequently improved by Woeginger [21] to $O(kn)$. (This case can also be solved in time $O(kn)$ using the method from [11].)

Our results. We provide several results that address the question whether k-medians can be found in sub-quadratic time, left open by Shah and Farach-Colton [17]. We show sub-quadratic algorithms for the following versions: (i) directed trees (for any fixed k), (ii) undirected balanced binary trees, and (iii) arbitrary trees for $k = 3$. The running time of our algorithms is bounded by $O(n \,\mathrm{polylog}(n))$. For directed trees, if the weights are bounded by a constant, our algorithm runs in time $O(n \log^2 n)$.

[5] In [3], the authors also claimed a linear-time algorithm; however, their running time analysis is flawed.

Our approach. Our algorithms are based on a combination of divide-and-conquer and dynamic programming techniques.

The divide-and-conquer method is based on a recursive decomposition of the given tree called the *spine decomposition* [5]. The idea is to decompose a given tree T into subtrees T_{uv}, where v is a descendant of u and T_{uv} contains all descendants of u that are not descendants of v. This decomposition has depth $O(\log n)$ and its total size is $O(n \log n)$.

Some previous algorithms for k-medians in trees [10, 15, 13, 19] were based on *ancestor-based* dynamic programming. They computed, recursively, an optimal cost of placing j facilities in T_u, the subtree of T rooted at u, under the assumption that there is a facility at an ancestor z of u. We use an alternative approach, referred to as *undiscretized dynamic programming*, which was applied by Langerman *et al.* [18] for placing filters in a multi-cast tree, and by Shah *et al.* [17] and Chrobak *et al.* [8] for facility location and k-medians. Instead of using an ancestor z as a parameter of the cost functions, we use a real-valued argument $\alpha \geq 0$ that represents the distance d_{uz}. We compute these cost functions bottom-up, and in order of increasing j, for all subtrees T_{uv} in the spine decomposition.

A cost function for j facilities in T_{uv} is a minimum of a number of linear functions, with each linear function corresponding to a choice of j facilities. The crucial problem in the analysis of our algorithm is to determine a good estimate on the complexity (the number of line segments) of these cost functions. Our key result is that this complexity is $O(m \operatorname{polylog}(m))$, for $m = |T_{uv}|$. It is known [8] that for $j = 2$ this number is $O(m)$, and we conjecture that it is linear for any constant j. We point out that in the facility location problem studied in [17], the complexity of the resulting dynamic programming functions can be easily shown to be linear (see Lemma 2.1 in [17]).

Both in the directed and undirected case, we can think of the k-median problem as the problem of partitioning T into subtrees (k subtrees in the undirected case, and $k+1$ in the directed case), for which the sum of the 1-median costs is minimized. The directed case turns out to be easier, mainly because a subtree T_u cannot contain a facility that serves a node that is not in T_u.

2 Spine Decomposition

Throughout the paper, T is a tree (directed or not) with root r and n nodes. Without loss of generality, we assume that T is a binary tree in which each node is either a leaf or has two children. We can also assume that T is *rightist*, that is, the size of the right subtree of any node is at least the size of its left subtree.

For any subtree T' of T we write $|T'|$ for the size of T' (the number of nodes). By T_u we denote the subtree of T rooted at u. The left and right children of u are denoted by $\mathit{left}(u)$ and $\mathit{right}(u)$. We say that $v \in T_u$ is a *rightist* descendant of u if the path from u to v contains only right links. We denote this path by S_{uv} and by T_{uv} we denote the tree $T_u - T_{right(v)}$. The leaf of T that is the rightmost descendant of u is denoted $\rho(u)$. Thus $T_u = T_{u\rho(u)}$. The path from u to $\rho(u)$ is called a *spine*. We will refer to subtrees T_u as *full* subtrees and subtrees T_{uv}, where $v \neq \rho(u)$, as *partial* subtrees.

We now define a *spine decomposition tree* for T [5], which we denote by $\mathsf{SD}(T)$. (Our presentation of spine decomposition is different from the one in [5], but is essentially equivalent.) Each node of $\mathsf{SD}(T)$ is of the form $\langle u, v\rangle$, where v is a rightist descendant of u, and it represents the tree T_{uv}. The root of $\mathsf{SD}(T)$ is $\langle r, \rho(r)\rangle$. Other nodes $\langle u, v\rangle$ are defined as follows:

Leaves: If $u = v$ and u is a leaf of T then $\langle u, u\rangle$ is a leaf of $\mathsf{SD}(T)$.

Nodes with one child: If $u = v$ and u is not a leaf of T then $\langle u, u\rangle$ has one child $\langle z, \rho(z)\rangle$, where $z = \mathit{left}(u)$.

Nodes with two children: For $u \neq v$, pick a node $\mu \in S_{uv}$ as follows. If $|T_{uu}| > \frac{1}{2}|T_{uv}|$, let $\mu = u$. Otherwise, let μ be the farthest node from u for which $|T_{u\mu}| \leq \frac{2}{3}|T_{uv}|$. (Note that $\mu \neq v$.) The left child of $\langle u, v\rangle$ is $\langle u, \mu\rangle$ and the right child is $\langle \eta, v\rangle$, where $\eta = \mathit{right}(\mu)$.

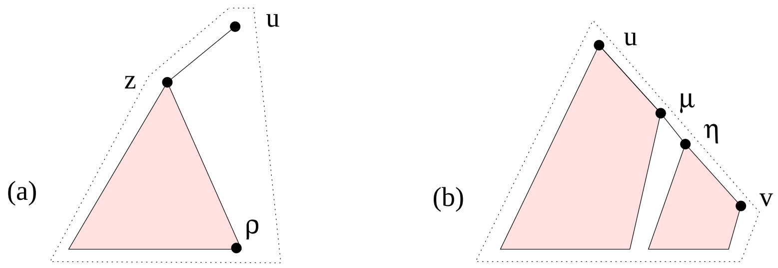

Fig. 1. Two cases in the construction of the spine decomposition tree.

Let $m_{uv} = |T_{uv}|$. If $\langle u, v\rangle$ has two children, say $\langle u, \mu\rangle$ and $\langle \eta, v\rangle$, then we call $\langle u, v\rangle$ *left-balanced* (resp. *right-balanced*) if $m_{u\mu} \leq \frac{2}{3}m_{uv} + 1$ (resp. $m_{\eta v} \leq \frac{2}{3}m_{uv} + 1$). We define $\langle u, v\rangle$ to be *balanced*, if it is both left- and right-balanced. Nodes that have one child are right-balanced but not left-balanced.

Lemma 1. *Let $\langle u, v\rangle \in \mathsf{SD}(T)$ and $u \neq v$. If T_{uv} is a full subtree, that is $v = \rho(u)$, then $\langle u, v\rangle$ is balanced.*

Proof. To simplify notation, let $m = m_{uv}$. Since T is rightist, we have $m_{uu} - 1 \leq m - m_{uu}$, so $m_{uu} \leq \frac{1}{2}(m+1)$. If $m_{uu} = \frac{1}{2}(m+1)$ then the claim is trivial, so we can assume $m_{uu} \leq \frac{1}{2}m$. Then, by the choice of μ, $m_{u\mu} \leq \frac{2}{3}m$ and $m_{u\eta} > \frac{2}{3}m$. If $\eta = v$ then $m_{\eta v} = 1$, and we are done. For $\eta \neq v$, denoting $t = \mathit{right}(\eta)$, the inequality $m_{u\eta} > \frac{2}{3}m$ implies $m_{tv} < \frac{1}{3}m$. Thus, since T is rightist, $m_{\eta v} \leq 2m_{tv}+1 < \frac{2}{3}m+1$.

Lemma 2. *Suppose that $\langle u, v\rangle \in \mathsf{SD}(T)$ is an unbalanced node with two children $\langle u, \mu\rangle$ and $\langle \eta, v\rangle$. Then one of the following cases holds:*

(a) *$\langle u, v\rangle$ is right-balanced and $\mu = u$, that is, the left child of $\langle u, v\rangle$ is $\langle u, u\rangle$.*

(b) $\langle u, v\rangle$ *is left-balanced and* $\eta = v$*, that is, the right child of* $\langle u, v\rangle$ *is* $\langle v, v\rangle$.
(c) $\langle u, v\rangle$ *is left-balanced,* $\eta \neq v$*, the right child of* $\langle \eta, v\rangle$ *is right-balanced and its left child is* $\langle \eta, \eta\rangle$.

Proof. If $m_{uu} > \frac{1}{2}m$ then $\mu = u$, and $m_{\eta v} < \frac{1}{2}m$, so $\langle u, v\rangle$ satisfies condition (a), and we are done.

We can assume now that $m_{uu} \leq \frac{1}{2}m$. By the construction, $m_{u\mu} \leq \frac{2}{3}m$, so $\langle u, v\rangle$ is left-balanced. Thus, by the assumption of the lemma, $\langle u, v\rangle$ is not right-balanced, that is $m_{\eta v} > \frac{2}{3}m + 1$.

We have two subcases. If $\eta = v$, then the right child of $\langle u, v\rangle$ is $\langle v, v\rangle$, and case (b) holds. For $\eta \neq v$, let $t = \mathit{right}(\eta)$. Since $m_{\eta v} > \frac{2}{3}m + 1$ and $m_{u\eta} > \frac{2}{3}m$ (by the choice of μ), we get $m_{\eta\eta} = m_{u\eta} + m_{\eta v} - m = m_{u\eta} + \frac{1}{2}m_{\eta v} - m + \frac{1}{2}m_{\eta v} > \frac{2}{3}m + \frac{1}{2}(\frac{2}{3}m + 1) - m + \frac{1}{2}m_{\eta v} > \frac{1}{2}m_{\eta v}$. So the children of $\langle \eta, v\rangle$ will be $\langle \eta, \eta\rangle$ and $\langle t, v\rangle$. Further, $m_{tv} < \frac{1}{2}m_{\eta v} \leq \frac{2}{3}m_{\eta v} + 1$, so $\langle \eta, v\rangle$ is right-balanced, so (c) holds.

By Lemmas 1 and 2, on any root-to-leaf path no more than three unbalanced nodes can occur consecutively. This implies the following lemma.

Lemma 3. *The height of* $\mathsf{SD}(T)$ *is* $O(\log n)$*, and its total size,* $\sum_{\langle u,v\rangle \in \mathsf{SD}(T)} |T_{uv}|$*, is* $O(n \log n)$.

3 Directed Trees

Cost functions. We define two cost functions as follows:
$P^j_{uv}(\alpha) =$ the optimal cost of T_u with at most j facilities, such that (i) at least one of the facilities is in S_{uv}, and (ii) u is served by a facility that is above u at distance α.
$Q^j_{uv}(\alpha) =$ the optimal cost of T_{uv} with at most j facilities, such that (i) no facility is in S_{uv}, and (ii) u is served by a facility that is above u at distance α.

For the cost functions for full subtrees we will simplify notation and write $P^j_u(\alpha) = P^j_{u\rho(u)}(\alpha)$ and $Q^j_u(\alpha) = Q^j_{u\rho(u)}(\alpha)$. Note the asymmetry between $P^j_{uv}(\cdot)$ and $Q^j_{uv}(\cdot)$: in the definition of $P^j_{uv}(\cdot)$ the facilities can be located in the whole subtree T_u, while for $Q^j_{uv}(\cdot)$ only in T_{uv}. However, in both cases the "top" facilities – those that do not have any other facilities above them – are located in T_{uv}. This is, in fact, the crucial idea behind the dynamic programming formulation.

Our ultimate goal is to compute the cost functions at the root, that is $P^k_r(\cdot)$, $Q^k_r(\cdot)$. The optimal cost of T with k facilities is $\mathit{cost}_k(T) = \min\{\, P^k_r(0)\,,\, Q^k_r(0)\,\}$.

Recurrence equations. We now set up recurrence equations for our cost functions. There are two base cases: when $j = 0$ and when $\langle u, v\rangle$ is a leaf of $\mathsf{SD}(T)$.

Consider first the case $j = 0$. For any node $\langle u, v\rangle \in \mathsf{SD}(T)$, $P^0_{uv}(\alpha)$ is undefined, and $Q^0_{uv}(\alpha) = \alpha W_{uv} + Q_{uv}$, where $W_{uv} = \sum_{x\in T_{uv}} w_x$, and $Q_{uv} = Q^0_{uv}(0) = \sum_{x\in T_{uv}} w_x d_{xu}$.

If $\langle u, u\rangle$ is a leaf of $\mathsf{SD}(T)$ (that is, u is a leaf of T), then for $j = 1, \ldots, k$, set $P^j_{uu}(\alpha) = 0$, and $Q^j_{uu}(\alpha)$ is undefined.

For $j > 0$ and any non-leaf node $\langle u, v\rangle$ of $\mathsf{SD}(T)$, we want to express each cost function for j facilities at $\langle u, v\rangle$ in terms of cost functions for $j' \leq j$ at the descendands of $\langle u, v\rangle$ in $\mathsf{SD}(T)$, or in terms of any cost functions for $j' < j$.

Suppose that $\langle u, v\rangle \in \mathsf{SD}(T)$, for $u \neq v$, has two children $\langle u, \mu\rangle$ and $\langle \eta, v\rangle$, where $\eta = \mathit{right}(\mu)$. Then

$$P^j_{uv}(\alpha) = \min \left\{ \begin{array}{l} P^j_{u\mu}(\alpha) \\ \min_{i=0,\ldots,j-1} \left\{ Q^i_{u\mu}(\alpha) + P^{j-i}_{\eta v}(\alpha + d_{\eta u}) \right\} \end{array} \right\}, \tag{1}$$

$$Q^j_{uv}(\alpha) = \min_{i=0,\ldots,j} \left\{ Q^i_{u\mu}(\alpha) + Q^{j-i}_{\eta v}(\alpha + d_{\eta u}) \right\}. \tag{2}$$

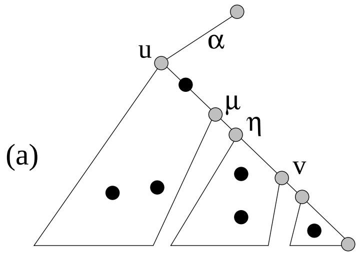

Fig. 2. Illustration of recurrence equation (1). We show two placements of facilities that correspond to the two terms in the minimum.

The second case is when we have a node $\langle u, u\rangle \in \mathsf{SD}(T)$ with one child $\langle z, \rho(z)\rangle$, where $z = \mathit{left}(u)$. Then

$$P^j_{uu}(\alpha) = \min \left\{ P^{j-1}_u(0), Q^{j-1}_u(0) \right\} \tag{3}$$

$$Q^j_{uu}(\alpha) = \alpha w_u + \min \left\{ P^j_z(\alpha + d_{zu}), Q^j_z(\alpha + d_{zu}) \right\}. \tag{4}$$

Note that to determine $P^j_{uu}(\alpha)$ we use information from an *ancestor* $\langle u, \rho(u)\rangle$ of $\langle u, u\rangle$ in $\mathsf{SD}(T)$, but for a smaller number of facilities.

The definition of the cost functions for the leaves and for $j = 0$ should be obvious. We give a brief justification for (1), the arguments for other recurrences are similar. Supopse that F is a set of j facilities that realizes $P^j_{uv}(\alpha)$, that is $F \subseteq T_u$, $|F| = j$, $F \cap R_{uv} \neq \emptyset$, and $\mathit{cost}_F(T^\alpha_u) = P^j_{uv}(\alpha)$. There are two cases (see Figure 2). If $F \cap R_{u\mu} \neq \emptyset$, then F is accounted for by the top term in the minimum, $\mathit{cost}_F(T^\alpha_u) = P^j_{u\mu}(\alpha)$. Otherwise, $F \cap R_{\eta v} \neq \emptyset$. Let $F' = F \cap T_{u\mu}$, $i = |F'|$ and $F'' = F \cap T_\eta$. We have $F' \cap R_{u\mu} = \emptyset$, $F'' \cap R_{\eta v} \neq \emptyset$, and $\mathit{cost}_F(T^\alpha_u) = \mathit{cost}_{F'}(T^\alpha_{u\mu}) + \mathit{cost}_{F''}(T^{\alpha + d_{\eta u}}_\eta)$, so this F is accounted for by the term $Q^i_{u\mu}(\alpha) + P^{j-i}_{\eta v}(\alpha + d_{\eta u})$.

Theorem 1. *The above recurrence equations for cost functions $P^j_{uv}(\cdot)$ and $Q^j_{uv}(\cdot)$ are correct. In particular, $\mathit{cost}_k(T) = \min \left\{ P^k_r(0), Q^k_r(0) \right\}$.*

Minimizers. Consider a set $F \subseteq T^{\alpha}_{uv}$ of j facilities. Then $cost_F(T^{\alpha}_{uv}) = A\alpha + B$, where $B = cost_F(T^0_{uv})$, and $A = \sum_{x \in X} w_x$, for a set of nodes $X \subseteq T_{uv} - F$ that do not have any ancestors in F. We conclude that all functions $P^j_{uv}(\cdot)$ and $Q^j_{uv}(\cdot)$ are lower envelopes of a collection of lines, and thus they are piece-wise linear, continuous and convex.

By the *complexity* of a piece-wise linear function we mean the number of its line segments. If $f(\cdot)$ and $g(\cdot)$ are two piece-wise linear functions, then the complexity of each function $f(\cdot) + g(\cdot)$ and $\min\{f(\cdot), g(\cdot)\}$ is at most the sum of the complexities of $f(\cdot)$ and $g(\cdot)$.

We say that a set of at most j nodes $F \subseteq T_u$ is a *P^j_{uv}-minimizer* at α if $F \cap S_{uv} \neq \emptyset$ and $cost_F(T^{\alpha}_u) = P^j_{uv}(\alpha)$. F is a *Q^j_{uv}-minimizer* at α if $F \subseteq T_{uv} - S_{uv}$ and $cost_F(T^{\alpha}_{uv}) = Q^j_{uv}(\alpha)$. F will be called a *(j, u, v)-minimizer* if it is either a P^j_{uv}-minimizer or a Q^j_{uv}-minimizer, for some α.

The complexity of each cost function $P^j_{uv}(\cdot)$ and $Q^j_{uv}(\cdot)$ is equal to the number of the respective minimizers (assuming that the ties are broken in an appropriate way). So we are interested in the following question: How many different minimizers we may have? Our *Linearity Conjecture* is that this number is that for any fixed $j \geq 1$ and each $\langle u, v\rangle$, the number of (j, u, v)-minimizers is $O(|T_{uv}|)$. For $j = 1$ this is is trivially true. The conjecture is supported by the result from [8], where it was proved that it indeed holds for $j = 2$. (Technically, the definition of minimizers in [8] was slightly different.) Although we were unable to prove this conjecture, we show how to use the recurrence equations for the cost functions to get a weaker bound.

Theorem 2. *Assume that k is fixed. Then, for any j and each $\langle u, v\rangle \in \mathsf{SD}(T)$, the number of (j, u, v)-minimizers is $O(m \log^{j-1} m)$, where $m = |T_{uv}|$.*

The proof of this theorem will appear in the final version.

The cost functions are piece-wise linear and non-decreasing. Each slope value is a sum of at most n weights, so there are only $O(Bn)$ possible slopes. Thus:

Lemma 4. *Assume that the node weights are positive integers from a range $1, 2, \ldots, B$, where B is a constant. Then the number of minimizers is $O(n)$.*

The Algorithm for directed trees. Our algorithm uses the recurrence equations developed earlier to compute the cost functions for all $j = 0, \ldots, k$ and all nodes $\langle u, v\rangle$ of the decomposition tree $\mathsf{SD}(T)$.

Algorithm k-MedDirTree
  Compute the spine decomposition tree $\mathsf{SD}(T)$
  Initialize the cost functions for $j = 0$ and at the leaves of $\mathsf{SD}(T)$
  For $j = 1, 2, \ldots, k$ **do**
    For all $\langle u, v\rangle \in \mathsf{SD}(T)$ in the bottom-up order **do**
      compute functions $P^j_{uv}(\cdot)$ and $Q^j_{uv}(\cdot)$
      using the recurrences (1), (2) and (3)
  Return $cost_k(T) = \min\left\{P^k_r(0), Q^k_r(0)\right\}$

Initialization. The only non-trivial question is how to compute efficiently all functions $Q^0_{uv}(\alpha) = \alpha W_{uv} + Q_{uv}$. Recall that $W_{uv} = \sum_{x \in T_{uv}} w_x$, and $Q_{uv} = Q^0_{uv}(0) = \sum_{x \in T_{uv}} w_x d_{xu}$. If $\langle u, v\rangle$ has two children $\langle u, \mu\rangle$ and $\langle \eta, v\rangle$, then we have $Q_{uv} = Q_{u\mu} + Q^0_{\eta v}(d_{\eta u})$ and $W_{uv} = W_{u\mu} + W_{\eta v}$. If $v = u$ and $\langle u, u\rangle$ has one child $\langle z, \rho\rangle$ then $Q_{uu} = Q^0_{z\rho}(d_{zu})$, and $W_{uu} = W_{z\rho} + w_u$. Using these recurrences we can compute all functions $Q^0_{uv}(\cdot)$ in linear time by traversing $\mathsf{SD}(T)$ bottom-up.

Running time. We represent each cost function by a list of its line segments, sorted in order of increasing α. Then the sum and the minimum of two cost functions can be computed in time linear in the total size of these functions. This means that for a given $\langle u, v\rangle \in \mathsf{SD}(T)$ the computation of functions $P^j_{uv}(\cdot)$, $Q^j_{uv}(\cdot)$ according to the recurrences (1), (2), (3), and (4) can be done in time proportional to the size of the cost functions for this particular node $\langle u, v\rangle$ (recall that k is a constant). Thus, by Theorem 2, denoting this cost by $\mathit{time}(j, u, v)$, we have $\mathit{time}(j, u, v) = O(|T_{uv}|\mathrm{polylog}(n))$. For any given j, the running time of the inner loop is bounded by the sum of $\mathit{time}(j, u, v)$ over all $\langle u, v\rangle \in \mathsf{SD}(T)$. This time is $O(n\,\mathrm{polylog}(n))$ due to the fact that the total size of all these subtrees is $O(n \log n)$, according to Lemma 3. Thus the overall running time is also $O(n\,\mathrm{polylog}(n))$.

If the weights are positive integers bounded by a constant, as in Lemma 4, or if the linearity conjecture holds, then $\mathit{time}(j, u, v) = O(|T_{uv}|)$.

Theorem 3. *Algorithm* k-MedDirTree *solves correctly the k-median problem. Further, assuming that k is a constant:*

(a) *Algorithm* k-MedDirTree *runs in time $O(n\,\mathrm{polylog}(n))$.*

(b) *If the linearity conjecture is true, then Algorithm* k-MedDirTree *runs in time $O(n \log^2 n)$.*

(c) *If the weights are positive integers bounded by a constant then Algorithm* k-MedDirTree *runs in time $O(n \log^2 n)$.*

4 Undirected Trees

As in the directed case, we need to partition the tree into k subtrees, in order to minimize the total 1-median cost of these subtrees. The difference is that now the facility in each subtree does not need to be its root.

Another way to view the problem is to find a set of k nodes R that are the roots of the subtrees in the partition. We refer to those nodes as *cut-nodes*, and to the edges between them (except for r) and their parents as *cut-edges*. (For consistency, we refer to r as a cut-node as well.)

We now define several cost functions:

$P^j_{uv}(\alpha) =$ the optimal cost of T_u with at most j facilities, such that (i) at least one of the cut-nodes is in S_{uv}, and (ii) some nodes in T_u may be served by a facility located outside T_u at distance α from u.

$\overleftarrow{Q}^j_{uv}(\alpha) =$ the optimal cost of T_{uv} with at most j facilities, such that (i) there are no cut-nodes in S_{uv}, and (ii) some nodes in T_{uv} may be served by a facility located outside T_{uv} at distance α from u.

$\overrightarrow{Q}^j_{uv}(\alpha) =$ the optimal cost of T_{uv} with at most j facilities, such that (i) there are no cut-nodes in S_{uv}, and (ii) some nodes in T_{uv} may be served by a facility located outside T_{uv} at distance α from v.

$C^j_{uv}(s) =$ the optimal cost of T_u with at most j facilities, such that u is served by a facility at $s \in T_{uv}$.

Let also $C^j_u = \min_{s\in T_u} C^j_u(s)$, which is simply $cost_j(T_u)$, the cost of T_u with j facilities. Our goal is to compute C^k_r. The complete set of recurrences will be given in the full paper; here we show how to compute $P^j_{uv}(\alpha)$ and $C^j_{uv}(s)$.

If the children of $\langle u, v\rangle$ are $\langle u, \mu\rangle$ and $\langle \eta, v\rangle$, then:

$$P^j_{uv}(\alpha) = \min \begin{Bmatrix} P^j_{u\mu}(\alpha) \\ \min_{i=0,\ldots,j-1} \left\{ \overleftarrow{Q}^i_{u\mu}(\alpha) + P^{j-i}_{\eta v}(\alpha + d_{\eta u}) \right\} \end{Bmatrix}$$

$$C^j_{uv}(s) = \begin{cases} C^j_{u\mu}(s) & \text{if } s \in T_{u\mu} \\ \min_{i=0,\ldots,j} \left\{ \overrightarrow{Q}^i_{u\mu}(d_{\mu s}) + C^{j-i}_{\eta v}(s) \right\} & \text{if } s \in T_{\eta v} \end{cases}$$

Otherwise, when $v = u$, and $\langle u, u\rangle$ has one child $\langle z, \rho\rangle$, then $P^j_{uu}(\alpha) = C^j_u$ and

$$C^j_{uu}(s) = \begin{cases} \min \left\{ P^{j-1}_z(d_{zu}), \overleftarrow{Q}^{j-1}_z(d_{zu}) \right\} & \text{if } s = u \\ C^j_z(s) + w_u d_{us} & \text{if } s \in T_z \end{cases}$$

Currently, we have not been able to design an algorithm that would evaluate those recurrences in subquadratic time for the general case. The crucial difference between the undirected and directed case is the formula for C^j_u that involves minimization over the whole tree T_u. However, we show how to do it in two special cases: when $k = 3$ or when the tree is balanced. The proofs will appear in the full version of this paper.

Theorem 4. *The 3-median problem in trees can be solved in time $O(n \log^3 n)$.*

Theorem 5. *The k-median problem in a balanced undirected tree (that is, a tree of depth $O(\log n)$) can be solved in time $O(k^2 n \log^{k-1} n + n \log n)$ and space $O(kn \log^{k-2} n)$.*

References

1. S. Arora, P. Raghavan, and S. Rao. Approximation schemes for euclidean k-medians and related problems. In *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC'98)*, pages 106–113, 1998.

2. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Mungala, and V. Pandit. Local seach heuristic for k-median and facility location problems. In *Proc. 16th Annual ACOM Symposium on Computing*, pages 21–29, 2001.
3. V. Auletta, D. Parente, and G. Persiano. Dynamic and static algorithms for optimal placement of resources in a tree. *Theoretical Computer Science*, 165:441–461, 1996.
4. V. Auletta, D. Parente, and G. Persiano. Placing resources on a growing line. *Journal of Algorithms*, 26:87–100, 1998.
5. R.R. Benkoczi and B.K.Bhattacharya. Spine tree decomposition. Technical Report CMPT1999-09, School of Computing Science, Simon Fraser University, Canada, 1999.
6. M. Charikar and S. Guha. Improved combinatorial algorithms for facility location and k-median problems. In *Proc. 40th Symposium on Foundations of Computer Science (FOCS'99)*, pages 378–388, 1999.
7. M. Charikar, S. Guha, E. Tardos, and D. Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC'99)*, pages 1–10, 1999.
8. M. Chrobak, L. Larmore, and W. Rytter. The k-median problem for directed trees. In *Proc. 26th International Symposium on Mathematical Foundations of Computer Science (MFCS'01)*, number 136 in Lecture Notes in Computer Science, pages 260–271, 2001.
9. M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-completeness.* W.H. Freeman and Co., 1979.
10. R. Gavish and S. Sridhar. Computing the 2-median on tree networks in $O(n \log n)$ time. *Networks*, 26:305–317, 1995.
11. R. Hassin and A. Tamir. Improved complexity bounds for location problems on the real line. *Operation Research Letters*, 10:395–402, 1991.
12. W.L. Hsu. The distance-domination numbers of trees. *Operation Research Letters*, 1:96–100, 1982.
13. O. Kariv and S.L. Hakimi. An algorithmic approach to network location problems II: The p-medians. *SIAM Journal on Applied Mathematics*, 37:539–560, 1979.
14. M.R. Korupolu, C.G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37:146–188, 2000.
15. B. Li, X. Deng, M. Golin, and K. Sohraby. On the optimal placement of web proxies on the internet: linear topology. In *Proc. 8th IFIP Conference on High Peformance Netwworking (HPN'98)*, pages 485–495, 1998.
16. B. Li, M.J. Golin, G.F. Italiano, X. Deng, and K. Sohraby. On the optimal placement of web proxies in the internet. In *IEEE InfoComm'99*, pages 1282–1290, 1999.
17. R. Shah and M. Farach-Colton. Undiscretized dynamic programming: faster algorithms for facility location and related problems on trees. In *Proc. 13th Annual Symposium on Discrete Algorithms (SODA)*, pages 108–115, 2002.
18. R. Shah, S. Langerman, and S. Lodha. Algorithms for efficient filtering in content-based multicast. In *Proc. 9th Annual European Symposium on Algorithms (ESA)*, pages 428–439, 2001.
19. A. Tamir. An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19:59–64, 1996.
20. A. Vigneron, L. Gao, M. Golin, G. Italiano, and B. Li. An algorithm for finding a k-median in a directed tree. *Information Processing Letters*, 74:81–88, 2000.
21. G. Woeginger. Monge strikes again: optimal placement of web proxies in the internet. *Operations Research Letters*, 27:93–96, 2000.