Faster Algorithms for k-Median Problems in Trees

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. {rbenkocz, binay}@cs.sfu.ca. *

[2] Department of Computer Science, University of California, Riverside, CA 92521. marek@cs.ucr.edu. **

[3] Department of Computer Science, University of Nevada, Las Vegas, NV 89154-4019. ***

[4] Instytut Informatyki, Uniwersytet Warszawski, Banacha 2, 02–097, Warszawa, Poland, and Department of Computer Science, University of Liverpool, Liverpool L69 7ZF, UK. rytter@csc.liv.ac.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. The number k is a constant. In this paper we give the first sub-quadratic time algorithms (in fact linear time within a poly-logarithmic factor) for directed trees, for balanced undirected trees, and for general 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 $cost(F) = \sum_{x \in G} \min_{u \in F} w_x d_{xu}$. 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 [16]), and

* Research supported by NSERC and MITACS.

** Research supported by grant EPSRC GR/N09077 (UK), NSF grants CCR-9988360, CCR-0208856, and ME-103/INT-9600919 from MŠMT (Czech Republic) and NSF (USA).

*** Research supported by grant EPSRC GR/N09077 (UK).

† Research supported by grant EPSRC GR/N09077.

most of the past work focused on efficient heuristics (see, for example, [21]) and approximation algorithms [3, 12, 11, 4].

Trees. For special graphs such as trees, interval, or circular arc graphs, the k-median problem can be solved in polynomial time. In 1979 Kariv and Hakimi [20] presented a dynamic programming algorithm for trees that runs in time $O(k^2n^2)$. Hsu [19] gave an algorithm with running time $O(kn^3)$. Later on, Tamir [26] 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 for arbitrary k. Recently, Shah and Farach-Colton [24] 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-median is possible, the main difficulty being to handle the cardinality constraint on the set of facilities.

Some other special cases of the problem have been investigated. The case $k = 2$ was considered by Gavish and Sridhar [17] who presented an algorithm with running time $O(n \log n)$, and by Breton [9] who gives a much simpler algorithm with the same complexity. The dynamic version of the case $k = 2$ was studied by Auletta *et al.* in [5][5]. Hassin and Tamir [18] 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.* [6].

Directed Treees. The k-median problem for directed trees was introduced by Li *et al.* [23] as a mathematical model for optimizing the placement of web proxies to minimize average latency. In a directed tree, all edges are oriented towards the root. The distance d_{xu} between any node x and any of its ancestors u extends in a natural way here, and $d_{xu} = \infty$ when u is not an ancestor of x. Intuitively, service can only be provided following the direction of edges. The objective is, as in the undirected case, to minimize the total distance from every client to the closest facility.

In [23], 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.* [27], who 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 [26] can be adapted to reduce the time complexity to $O(kn^2)$. In Chrobak *et al.* [13], efficient algorithms for $k = 2$ and $k = 3$ were given. For $k = 2$, the algorithm runs in time $O(n \log n)$, and for $k = 3$, it runs in time $O(n \log^2 n)$.

Our Results. In this paper, we partially solve the problem left open by Shah and Farach-Colton [24] regarding a sub-quadratic algorithm for k-median in a tree T. We describe a tree decomposition and a template of recursive functions that allows a sub-quadratic computation of a k-median in any tree, provided that the following query is answered in sub-linear time: the 1, 2, … $k - 1$ optimal

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

median in certain subtrees of T. We refer to this query as *the dynamic j-median* problem, $1 \leq j \leq k-1$. We are able to solve the dynamic j-median in sub-linear time for i) any value of k in a directed tree, ii) any k in an undirected balanced binary tree, and iii) $k = 3$ in an arbitrary tree.

Our Approach. We use the paradigm of undiscretized dynamic programming which was first employed by Langerman *et al.* [25] for the problem of placing filters in a multi-cast tree, and then was adapted to facility location problems by Shah *et al.* [24] and Chrobak *et al.* [13].

Before we describe the recursive functions used, observe that any set of k facilities partitions the set of nodes into k connected components. Every component has the property that its set of vertices are closest to the same facility, and the facility is also the optimum 1-median of the component. The facility is sometimes said to *cover* the vertices in the component. Therefore, finding the optimum k-median in a tree is equivalent to finding a set of $k-1$ edges whose removal disconnects the tree into k connected components that are optimal for the k-median. The $k-1$ edges are referred to as *split* or *cut* edges, and the set of optimum k medians or the optimum $k-1$ split edges are sometimes called *minimizers*. Our goal is to find a set of vertices, or split edges, or a combination of vertices and split edges that are optimal for the k-median problem. The technique of identifying split edges is extensively used in solving median problems in trees for particular values of k [9, 10, 17].

Without loss of generality, we assume our tree T to be binary. If it is not, we can make it binary with linear time preprocessing by adding a linear number of vertices and edges of weight zero [26, 24]. We define two recursive functions, $COST$ and f_z, each associated with a subtree T' of T.

- $COST(T', \mathcal{F}_j)(\alpha)$ – represents the contribution to the final k-median cost of the vertices in T' given the restrictions that,
  - a facility from $T \setminus T'$ serves $root(T')$, the root of T', at distance α from $root(T')$, and
  - a set of split edges $F_j \in \mathcal{F}_j$ are selected optimally from T' given the constraint above.

  $\mathcal{F}_j$ is the set of j split edges that are minimizers for T' as α varies from 0 to ∞.
- $f_z(T', F_j)$ – represents the contribution to the final k-median cost of vertices in T' given that
  - $z \in T'$ is a facility covering $root(T')$, and
  - F_j is a set of l split edges that is a minimizer for T' under the restriction mentioned above.

  Observe that for a given T' and a fixed $z \in T'$, F_j is clearly defined, although it might not be unique. Since we are not concerned with finding all optimal k-median solutions, any minimizer F_j is sufficient for our purposes.

Formally, the definition of $COST$ function can be expressed as

$$
\begin{aligned}
COST(T',\mathcal{F}_j)(\alpha) = \min_{\substack{F_j \in E(T') \\ |F_j|=j}} & COST(T',F_j)(\alpha) = \\
= \min_{\substack{F_j \in E(T') \\ |F_j|=j}} & \Big\{ \sum_{X \in \mathcal{K}(F_j)} MedCost(X,1) + \\
+ & \sum_{v \in T' \setminus \mathcal{K}(F_j)} w(v)\big(\alpha + d(v, root(T'))\big) \Big\} = \\
= \min_{\substack{F_j \in E(T') \\ |F_j|=j}} & \big\{ a(F_j) + \alpha b(F_j) \big\},
\end{aligned}
$$

where $\mathcal{K}(F_j)$ is the set of j components, excluding the component containing $root(T')$, obtained from T' by removing edges F_j; $MedCost(X, p)$ is the cost of the p median of subtree X; and $a(F_j)$, $b(F_j)$ are constants depending only on the choice of F_j. Therefore, the following lemma can be stated.

Lemma 1. *$COST(T',\mathcal{F}_j)(\alpha)$ is a concave, piecewise linear function.*

As a note for those familiar with Tamir's algorithm [26] and Shah *et al.* [24], our $COST$ function is very similar to the undiscretized function F_v, and f_z to G_v, but they are not identical. Also, observe that while $COST$ is continuous, f_z is discrete. In Shah *et al.* [24] both functions are undiscretized, but the G_v obtained is a discontinuous step function, and thus it is discrete by nature.

We can now explain how to use f_z and $COST$ to find an optimal k-median solution for tree T. Let u and v be the children of $root(T)$ (assume for now that they both exist). Denote $T_u^c = T \setminus T_u$, $T_v^c = T \setminus T_v$, and $r_T = root(T)$. In the optimal solution, r_T is covered by either a facility in T_u^c or by one in T_v^c. This does not exclude the possibility that r_T itself is a facility. Then clearly,

$$
MedCost'(T,k) = \min_{0 \le j \le k-1} \Big\{ \min_{z \in T_v^c} \Big\{ f'_z(T_v^c, F_{k-1-j}) + COST(T_u^c, \mathcal{F}_j)\big(d(z, r_T)\big) \Big\} \Big\}. \tag{1}
$$

We write $MedCost'$ because the above is just a partial computation of the optimum k-median. The roles of T_u^c and T_v^c need to be exchanged and the smallest of the two values obtained must be chosen. Similarly, we write f'_z due to a technicality. T_u^c and T_v^c are not disjoint subtrees, therefore the contribution of r_T is added twice, once in the $COST$ function and once in the f_z function. To fix this, we just subtract the contribution of r_T from f_z, hence the modified expression f'_z in (1).

If r_T has only one child, then practically the $COST$ function in (1) is not needed. However, as it is discussed later, $COST$ functions are needed for the computation of f_z. There is one final observation. If T is rooted at its 1-median m_T, then it can be shown that at least one facility must exist in both T_u^c and T_v^c. Then, (1) can be slightly simplified because the case where no facility is placed in one of T_u^c or T_v^c need not be tested.

To finalize the description of the k-median algorithm, we need to show how to recursively compute the $COST$ and f_z functions, and to analyze the time required by all these operations. Key to our approach is the use of a tree decomposition of T that gives us a good bound on the complexity of $COST$ function, and allows us to compute $COST$ and f_z relatively easy. Note that the complexity of f_z is obviously linear. It is here, in fact, where our method and that of Shah *et al.* [24] diverge. In the later case, the authors leave tree T untouched but provide efficient data structures to maintain the recursive functions F_v and G_v. This makes difficult to handle functions where the cardinality of the median set is an additional parameter.

2 The Tree Decomposition

Tree decompositions have been previously used in the literature to solve efficiently various problems in trees. Probably the most well known of them is the centroid decomposition (see for example [15]). For our problem, we choose the spine tree decomposition (STD) of [7] because of its ability to reflect the structure of the underlying tree – an important feature when computing $COST$ and f_z functions. Similar decompositions exist, such as the top-tree decomposition of Alstrup *et al.* [1, 2], the decomposition structure of Boland [8], the centroid path decomposition of Cole *et al.* [14]. We describe the STD here briefly by presenting first a view closer to the centroid decomposition.

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 (otherwise we can simply switch the two subtrees). By T_u we denote the subtree of T rooted at u. The left and right children of u are denoted by *left*(u) and *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 R_{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 $T_u = T_{u\rho(u)}$. 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 *decomposition tree* for T, which we denote by $\mathsf{Dec}(T)$. Each node of $\mathsf{Dec}(T)$ is of the form $\langle u, v\rangle$, where v is a rightist descendant of u. This node $\langle u, v\rangle$ represents the tree T_{uv}. The root of $\mathsf{Dec}(T)$ 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 $\mathsf{Dec}(T)$.

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 = $ *left*(u).

Nodes with two children: For $u \neq v$, pick a node $\mu \in R_{uv}$ 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 = $ *right*(μ).

For $\langle u, v\rangle \in \mathsf{Dec}(T)$, define $m_{uv} = |T_{uv}|$. 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 $m_{u\mu} \leq$

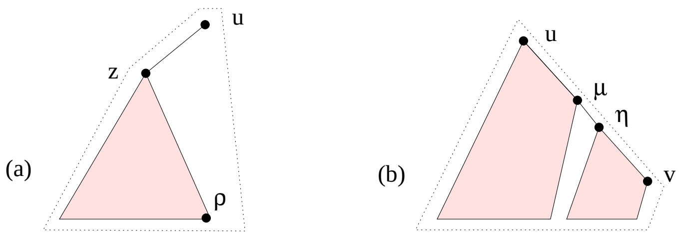

Fig. 1. Two cases in the construction of the decomposition tree. (a) A node $\langle u, u\rangle$ with one child $\langle z, \rho\rangle$. (b) A node $\langle u, v\rangle$ with two children $\langle u, \mu\rangle$ and $\langle \eta, v\rangle$.

$\frac{2}{3}m_{uv} + 1$ (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 considered unbalanced (more specifically, right-balanced but not left-balanced). The following results can be shown (proofs are omitted due to space constraints):

Lemma 2. *Let $\langle u, v\rangle \in \mathsf{Dec}(T)$ and $u \neq v$. If T_{uv} is a full subtree, that is $v = \rho(u)$, then $\langle u, v\rangle$ is balanced.*

Lemma 3. *Suppose that $\langle u, v\rangle \in \mathsf{Dec}(T)$ 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 $\langle \eta, \eta\rangle$.*

By Lemmas 2 and 3, on any root-to-leaf path no more than three unbalanced nodes can occur consecutively. This implies the following lemma.

Lemma 4. *The height of $\mathsf{Dec}(T)$ is $O(\log n)$, and its total size, $\sum_{\langle u,v\rangle \in \mathsf{Dec}(T)} |T_{uv}|$, is $O(n \log n)$.*

With the above decomposition in mind, the STD can be easily introduced. STD is a decomposition centered around a path (spine) in the original tree T. An additional data structure – a weight balanced binary search tree – is built on top of the spine, i.e. the leaves of this balanced search tree coincide with the vertices on the spine. The search tree built is such that the total ordering implied over its leaves corresponds exactly with the total ordering defined by the spine. In other words, listing the leaves from left to right gives the same sequence of nodes as listing the vertices on the spine from root to leaf. The connected components obtained from T by removing the spine, are decomposed recursively.

The properties of the decomposition described at the beginning of this section can be instilled into STD if the following are satisfied (see Fig. 2).

1. A spine is always a rightist path.

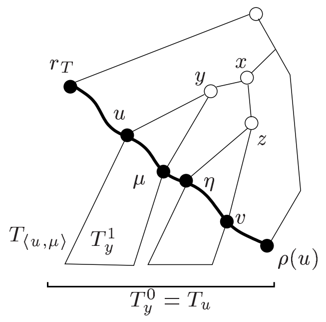

Fig. 2. Correspondence between $\mathsf{Dec}(T)$ and STD. The spine is highlighted

2. Let x be an internal node of a search tree with y and z as left, resp. right children. Let u and v be leaves of the search tree such that u (v) is the leftmost (rightmost) spine node with x as ancestor. Let μ (η) be the rightmost (leftmost) spine node with y (z) as ancestor. x is associated implicitly with tree $T_{\langle u,v\rangle}$, y with $T_{\langle u,\mu\rangle}$, and z with $T_{\langle \eta,v\rangle}$. $T_{\langle u,v\rangle}$, $T_{\langle u,\mu\rangle}$, and $T_{\langle \eta,v\rangle}$ are chosen exactly as in $\mathsf{Dec}(T)$.
3. A subtree with one child in $\mathsf{Dec}(T)$ is decomposed recursively in STD.

We denote $T_{\langle u,v\rangle}$ by T^1_x (subtree below spine), u by x_C (close to the root), v by x_F (far from root), T_u by $T^0_y = T^0_x$ (subtree along spine).

3 Computing *COST* Function

We are now interested in determining the recurrence relation for $COST$ function. Several variations of $COST$ functions are computed and stored at the internal nodes of the STD. They are all needed for the computation of the final $COST$ and f_z functions. These are restricted and unrestricted $COST$ functions, rooted at x_C or x_F, computed for T^1_x or T^0_x. More precisely, let $T_x(x_C)$ ($T_x(x_F)$) in $COST$ function denote that the external median is situated at distance α from x_C (x_F) and that it covers x_C (x_F). Denote by $\mathcal{F}^R_j$ a set whose members are the set of j cut edges from T^1_x such that i) each j cut edges from $F^R_j \in \mathcal{F}^R_j$ is restricted to among the edges in T^1_x situated below the spine path, and ii) $COST(T^1_x(x_C), \mathcal{F}^R_j)(\alpha)$ is minimized. Observe that

$$COST(T^0_x(x_C), \mathcal{F}^R_j)(\alpha) = COST(T^1_x(x_C), \mathcal{F}^R_j)(\alpha) + A + \alpha B,$$

where A is the sum of the weighted distance of all nodes, hanging between x_F and the leaf spine, to x_C, and B represents their weight. It is easy to see that the partition of the abscissa by the linear pieces of $COST(T^1_x(x_C), \mathcal{F}^R_j)(\alpha)$ doesn't change when a linear function in α is added. Thus, the value of $COST(T^0_x(x_C), \mathcal{F}^R_j)(\alpha)$ for any α can be obtained in constant time from $COST(T^1_x, \mathcal{F}^R_j)(\alpha)$ once A and B are known, and vice versa.

When the restriction imposed on j deleted edges in $F_j^R \in \mathcal{F}_j^R$ to lie below spine edges is removed, the choice of the set of j cut edges $\mathcal{F}_j$ is defined slightly different. If all the edges in some $F_j \in \mathcal{F}_j$ lie below the spine, then they must lie in T_x. If one of the edges in F_j is a spine edge, instead of restricting all the remaining ones to be in T_x, we do the following. We force the p cut edges reachable from the root to lie in T_x, while the remaining q cut edges form the optimal $q+1$-median of the component $T(e)$ split by the spine edge e reachable from the root. Formally,

$$COST(T_x^0(x_C), \mathcal{F}_j)(\alpha) = \min\Big\{COST(T_x^0(x_C), \mathcal{F}_j^R)(\alpha),$$
$$\min_{\substack{e\in\text{spine}\\ p\leq j-1}} \{COST(T_x^0(x_C) \setminus T(e)(x_C), F_p^R)(\alpha) + MedCost(T(e), j-p)\}\Big\}. \quad (2)$$

Since we are concerned with computing $COST(T_u^C(m_T), \mathcal{F}_l)(\alpha)$, we describe a recurrence relation for $COST(T_x^0(x_C), \mathcal{F}_j)(\alpha)$ in terms of $COST(T_y^0(y_C), \mathcal{F}_j)(\alpha)$ and $COST(T_z^0(z_C), \mathcal{F}_j)(\alpha)$. Here y and z are the children of the search tree node of x with y towards the root.

$$COST(T_x^0(x_C), \mathcal{F}_j)(\alpha) = \begin{cases} 0, & \text{if } x \text{ is a leaf;} \\ w(T_x)\alpha + \sum_{v\in T_x} w(v)d(v, x_C), & \text{if } j = 0; \\ \min\{C_1 + C_2\}, & \text{otherwise,} \end{cases} \quad (3)$$

where

$$C_1 = \min_{p+q=j} \Big\{COST(T_y^1(y_C), \mathcal{F}_p^R)(\alpha) + COST(T_z^0(z_C), \mathcal{F}_q)(\alpha + d(z_C, x_C))\Big\}$$
$$C_2 = COST(T_y^0(y_C), \mathcal{F}_j)(\alpha).$$

C_1 captures the case where no spine edge is cut from T_y and possibly some spine edges are cut from T_z, and C_2 the situation in which possibly some spine edges are cut from T_y.

We can prove the following result (proof omitted in the short version of the paper).

Lemma 5. *Let $S_j(T_x)$ denote the size of concave function $COST(T_x^0(x_C), \mathcal{F}_j)(\alpha)$. Then*

$$S_j(T_x) \leq \begin{cases} 1, & \text{if } j = 0 \\ c|T_x| - 1, & \text{if } j = 1 \\ c|T_x| \log|T_x| \Big(1 + \log|T_x|\Big)^{j-2}, & \text{if } j \geq 2 \end{cases} \quad (4)$$

where c is some constant.

We now have a bound on the complexity of $COST$ function and a recurrence relation that can be used in the actual computation. The computation proceeds bottom-up, by merging the description of the cost function stored at the children

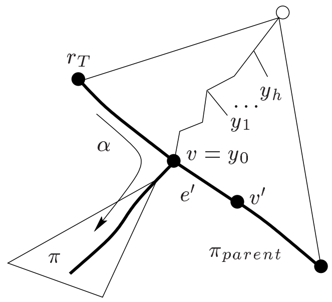

Fig. 3. Updating $COST$ from spine π to the parent spine

of the current node. Here, merging means computing the minimum of several concave piecewise linear functions. This can be easily done in time linear in the total size of the concave functions, if the linear pieces of both functions are given in sorted order (details omitted).

Computing the restricted $COST$ function is easier. Since by definition, no cut edges lie on the spine, both T_z^1 and T_y^1 are reachable from the external median at distance α from x_C. Then,

$$COST(T_x^1(x_C), \mathcal{F}_j^R)(\alpha) = \\ \min_{p+q=j} \{COST(T_z^1(z_C), \mathcal{F}_p^R)(\alpha) + COST(T_y^1(y_C), \mathcal{F}_q^R)(\alpha + d(y_C, z_C)).$$

Both versions of the restricted cost function must be computed, rooted at x_C and at x_F.

When $COST(\cdot, \mathcal{F}_l)(\alpha)$ is computed for an entire spine, it becomes $COST(\cdot, \mathcal{F}_l^R)(\alpha)$ assigned to a leaf on the parent spine. The unrestricted cost function at the parent spine for the same leaf can be computed as follows. First, the parent spine edge adjacent to the leaf is split. This corresponds to edge $e' = (v, v')$ in Fig. 3. Then, with π_{parent} as current spine, the unrestricted cost function for v is obtained using

$$COST(T_v^0(v), \mathcal{F}_j)(\alpha) = \min\Big\{ \min_{p+q=j} \{COST(T_v^0(v), \mathcal{F}_p^R)(\alpha) + \\ + MedCost(T_{v'}^0(v'), q+1),\ COST(T_v^1(v), \mathcal{F}_p^R) + A + \alpha B \Big\}, \quad (5)$$

where $T_{v'}^0(v')$ is the component obtained by splitting edge e', and A is the sum of the weighted distance of all nodes in $T_{v'}^0(v')$ to v and B represents their weight. Here, the difficulty lies in evaluating $MedCost(T_{v'}^0(v'), q+1)$ in sub-linear time. We call this problem the dynamic j-median. Solving it for arbitrary trees and arbitrary values of k is key to providing a sub-quadratic algorithm for the k-median problem in trees.

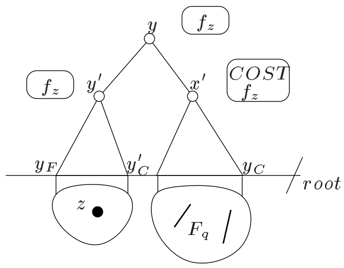

Fig. 4. Computing the list of values $f_z(T_x(x_C), F_l)$ using the $COST$ and f_z functions recursively

4 Computing Function f_z

The basic idea behind computing f_z is to use merging as in computing the $COST$ functions. Again, every internal node y of the STD is associated with a list of values $f_z(T_y(y_C), F_l)$. With this notation, we specify that $z \in T_y^1$, z covers y_C (no cut edge is chosen from path $\pi(z, y_C)$), and l cut edges are chosen from T_y^0 optimally. In other words, $f_z(T_y(y_C), F_l)$ denotes the $(l+1)$-median of T_y^0 when $z \in T_y^1$ is one of the medians covering y_C.

Consider Fig. 4 where we are interested in computing the f_z list at node y. Observe that in the list already computed for y' we have not considered the situation in which some cut edges for $z \in T_{y'}^1$ are chosen from $T_{x'}^1$. Therefore we have

$$f_z\big(T_y(y_C), F_l\big) = \min\Big\{ \min_{p+q=l} \Big\{ f_z\big(T_{y'}(y'_C), F_p\big) + COST\big(T_{x'}^1(x'_F), \mathcal{F}_q^R\big)\big(d(z, x'_F)\big) \Big\}, \\ f_z\big(T_{y'}(y'_C), F_l\big) \Big\},$$

where $COST(T_{x'}^1(x'_F), \mathcal{F}_q^R)\alpha$ denotes the restricted $COST$ function computed with respect with x'_F as root. The list of f_z values can be merged whenever the $COST$ functions are merged.

When processing a spine is finished, we need to update the list of f_z values so that they correctly reflect the freedom of choosing cut edges optimally from the components towards the leaf of the parent spine. This can be done in a way similar to the operations described so far. Details are omitted in this version of the paper.

5 Complexity of the Algorithm

Lemma 5 bounds the complexity of the $COST$ function stored at any internal node of the STD, and is the key result used in proving the running time and storage space complexity of our approach. Let $E_{KM}(n)$ denote the complexity of solving the dynamic j-median from (5). Then we have the following result (proof omitted).

Theorem 1. *The k-median problem on arbitrary trees is solved by our algorithm in $O((kn\log^2 n)(k+(1+\log n)^{k-3})+knE_{KM}(n))$ time and $O(kn\log n(1+\log n)^{k-3})$ space.*

6 The Dynamic j-median

From Equation (5) we notice that we need the j-median cost in $T^0_{e'} = T^0_{v'}$. Taking Fig. 3 as reference, assume we have to compute $MedCost(T^0_v, j)$ for $0 \le j \le k-1$. Computing j-median cost in T^0_v amounts to finding the optimal $j-1$ cuts in T^0_v. Let $y_1, y_2, \ldots, y_t$ be the children of the nodes of the path $\sigma(v, root)$ where each T_{y_i} does not contain v, and let $y_0 = v$.

Assume z is the median covering y_0 in $T^0_{y_0}$, and z is restricted to belong to $T^1_{y_h}$. It is easy to see that

$$MedCost(T^0_{y_0}, j) = \min_{0\le h\le t} \min_{\forall z\in T_{y_h}} f_z(T^0_{y_0}(y_0), F_j).$$

Here, $f_z(T^0_{y_0}(y_0), F_j)$ is the optimal cost of a $j+1$ median in $T^0_{y_0}$ when z is one of the medians forced to lie in $T^1_{y_h}$ and z must cover y_0. On the other hand, we have access to $f_z(T^0_{y_h}(y_h), F_j)$. Then $f_z(T^0_{y_0}(y_0), F_j)$ can be obtained from $f_z(T^0_{y_h}(y_h), F_j)$ by updating the later with the contribution of vertices in $\bigcup_{i=0}^{h-1} T^1_{y_i}$. This contribution can be evaluated using the restricted $COST$ function stored at these nodes. Formally,

$$f_z(T^0_{y_0}(y_0), F_j) = \min_{0\le p\le j-1} \Big\{ (f_z(T^0_{y_h}(y_{h_C}), F_p) + \min_{\sum_{i=0}^{h-1} q_i = j-p} \big(\sum_{i=0}^{h-1} COST(T^R_{y_i}(y_{i_F}), F_{q_i})(d(z, y_{i_F}))\big)\Big\}. \quad (6)$$

We need to compute this value in amortized poly-logarithmic time.

Until now, we have not been able to solve this problem in general. We are able to solve it for some special cases. These are

(a) $MedCost(T^0_{e'}, j)$, $j = 0, 1$.
(b) $MedCost(T^0_{e'}, j)$, $\forall j$ when T is a directed tree where each node points to its parent, thereby implying the direction along which one can move.
(c) $MedCost(T^0_{e'}, j)$, $\forall j$ when T is a balanced tree.

6.1 Case $k = 3$

The algorithm for this case is quite involved. We state, without further discussion the following theorem. Note that the space complexity is that of Theorem 1.

Theorem 2. *the 3-median problem in trees can be solved in $O(n\log^3 n)$ time.*

6.2 Directed Trees

When the tree is directed, all edges point from the root of T towards the leaves, and any facility can serve only its successor nodes. Therefore, in the k-median, the root node is always a facility. There is no restricted $COST$ function defined relative to the leaf node x_F of any internal node of the STD, and no list f_z needs to be computed.

The main algorithm for computing the k median then uses only the $COST$ function at the root node r_T to select

$$\min_{p+q=k-1} \{COST(T_u^C, \mathcal{F}_p)(0) + COST(T_v^C, \mathcal{F}_q)(0)\}.$$

The $COST$ functions are merged exactly the same way as in the undirected case, and the j-median cost in $T_{y_0}^0$ can be found using only $COST$ functions, for any node y_0 on a spine. Since T is directed, one of the facilities realizing the optimal solution must be placed at y_0. Thus, we can compute the optimal j-median cost in $T_{y_0}^0$ by evaluating $COST(T_{y_0}^0(y_0), \mathcal{F}_j)(\alpha)$ at $\alpha = 0$. This can easily be done in $O(k \log n)$ time since $COST(T_{y_0}^0(y_0), \mathcal{F}_j)(\alpha)$ can be obtained from the $COST$ functions stored at nodes $y_1, y_2, \ldots, y_t$. The result can be summarized by the following theorem.

Theorem 3. *The k-median problem in a directed graph can be solved by the above algorithm in $O(kn \log^{k-1} + k^2 n \log n)$ time and $O(kn \log^{k-2} n)$ space.*

6.3 Balanced Trees

As for the 3-median problem, we state only the main result.

Theorem 4. *The k-median problem in a balanced undirected tree can be solved in $O(k^2 n \log^{k-1} n + n \log n)$ time and $O(kn \log^{k-2} n)$ space.*

7 Conclusions

In this paper, we give the first sub-quadratic algorithms for several k-median problem on trees, namely the k-median in directed trees, the k-median in balanced undirected trees, and the 3-median problem in arbitrary trees. Our technique uses undiscretized dynamic programming formulations similar to those of Shah *et al.* [24]. Our success in tackling the k-median problem is mainly due to a tree decomposition structure used on the input tree. This is different from the approach of Shah *et al.* [24] who concentrate on data structures for maintaining the dynamic programming functions. Although large amounts of details had to be left out, we hope that the main ideas of our algorithm have been conveyed.

References

1. S. Alstrup, J. Holm, K. de Lichtenbarg, and M. Thorup. Minimizing diameters of dynamic trees. In *ICALP*, (1997), 270–280.
2. S. Alstrup, J. Holm, and M. Thorup, Maintaing center and median in dynamic trees. *Scandanavian Workshop on Algorithmic Theory*, (2000), 46–56.
3. 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.
4. 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.
5. 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.
6. V. Auletta, D. Parente, and G. Persiano. Placing resources on a growing line. *Journal of Algorithms*, 26:87–100, 1998.
7. R.R. Benkoczi and B.K. Bhattacharya. Spine tree decomposition. Technical Report CMPT1999-09, School of Computing Science, Simon Fraser University, Canada, 1999.
8. R.P. Boland. Polygon Visibility Decompositions with Applications. PhD. Thesis, University of Ottawa, Ottawa, Canada, 2002.
9. D. Breton. Facility location optimization problems in trees. M.Sc. Thesis, School of Computing Science, Simon Fraser University, Canada, 2002.
10. R.E. Burkard, E. Çela, and H. Dollani. 2-Medians in Trees with Pos/Neg Weights. *Discr. Appl. Math.*, 105:51–71, 2001.
11. 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.
12. 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)*, 1999.
13. 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.
14. R. Cole, M. Farach-Colton, R. Hariharan, T. Przytycka, and M. Thorup. An $O(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. *J. Comput*, 30(5):1385:1404, 2000.
15. G.N. Frederickson and D.B. Johnson. Finding k-th paths and p-centers by generating and searching good data structures. *J. of Algorithms*, 4:61–80, 1983.
16. M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-completeness.* W.H. Freeman and Co., 1979.
17. R. Gavish and S. Sridhar. Computing the 2-median on tree networks in $O(n \log n)$ time. *Networks*, 26:305–317, 1995.
18. R. Hassin and A. Tamir. Improved complexity bounds for location problems on the real line. *Operation Research Letters*, 10:395–402, 1991.
19. W.L. Hsu. The distance-domination numbers of trees. *Operation Research Letters*, 1:96–100, 1982.

20. 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.
21. 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.
22. 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 00–00, 1998.
23. 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.
24. 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)*, 2002.
25. 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.
26. A. Tamir. An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19:59–64, 1996.
27. 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.
28. G. Woeginger. Monge strikes again: optimal placement of web proxies in the internet. *Operations Research Letters*, 27:93–96, 2000.