Approximation Algorithms for Hierarchical Location Problems*

C. Greg Plaxton

November 2002

Abstract

We formulate and (approximately) solve hierarchical versions of two prototypical problems in discrete location theory, namely, the metric uncapacitated k-median and facility location problems. Our work yields new insights into hierarchical clustering, a widely used technique in data analysis. First, we show that every metric space admits a hierarchical clustering that is within a constant factor of optimal at every level of granularity with respect to the average (squared) distance objective. Second, we provide a natural solution to the leaf ordering problem encountered in the traditional dendrogram-based approach to the visualization of hierarchical clusterings.

*Department of Computer Science, University of Texas at Austin, Austin, TX 78712. This research was supported by NSF Grant CCR–9821053. Email: `plaxton@cs.utexas.edu`.

1 Introduction

Inspired by the recent work of Dasgupta [4] on a hierarchical version of the k-center problem, we formulate hierarchical versions of the metric uncapacitated k-median and facility location problems, two prototypical problems in discrete location theory. Before defining and addressing the hierarchical versions of these problems, we review the definitions of the k-center, k-median, and facility location problems. We also review certain "incremental" versions of the k-center and k-median problems, and introduce a corresponding incremental version of the facility location problem. The incremental versions of these problems represent a natural intermediate step towards defining their hierarchical versions, as will be seen in Section 3.

1.1 Preliminaries

For any real $\alpha \geq 1$, we say that a distance function d defined over a set of points satisfies the *α-approximate triangle inequality* if, for any triple of points x, y, and z, $d(x,z) \leq \alpha(d(x,y) + d(y,z))$. We define an *$\alpha$-approximate metric space* as a set of points with an associated distance function d that satisfies positivity ($d(x,y) > 0$ unless $x = y$, in which case $d(x,y) = 0$), symmetry ($d(x,y) = d(y,x)$), and the α-approximate triangle inequality. Our motivation for assuming such a relaxed triangle inequality is that squaring each of the distances in a given metric space yields a 2-approximate metric space. More generally, raising the distances of a metric space to any constant power yields an α-approximate metric space for some constant $\alpha \geq 1$. Consequently, the various constant-factor approximation algorithms that we develop in this paper for α-approximate metric spaces immediately imply constant-factor approximation algorithms for related problems on metric spaces in which the objective function is altered by raising each distance to some constant power. In keeping with the foregoing motivation, we will assume throughout the paper that the parameter α governing the relaxed triangle inequality is a constant.

In this paper we will define approximation versions of various optimization problems. As a convenient shorthand, throughout this paper we define an approximation algorithm for a given problem to be *nice* if and only if it is constant-factor approximate and runs in polynomial time. Remark: The constant factor in the approximation bound is allowed to depend on the constant α governing the relaxed triangle inequality.

Throughout the remainder of the paper, we fix an arbitrary α-approximate metric space with associated nonempty point set U and distance function d. We let n denote $|U|$, we define an *index* as an integer in the range 1 to n inclusive, and we define a *scaling factor* as a nonnegative real. Each point x has an associated nonnegative *weight* $w(x)$ and *value* $v(x)$. For any set of points X, we let $w(X) = \sum_{x \in X} w(x)$ and $v(X) = \sum_{x \in X} v(x)$.

For any point x, nonempty sets of points X and Y, and scaling factor λ, we define

$$
\begin{aligned}
d(x,Y) &= \min_{y \in Y} d(x,y), && (1)\\
\mathit{radius}(X,Y) &= \max_{x \in X} d(x,Y), && (2)\\
\mathit{error}(X,Y) &= \sum_{x \in X} d(x,Y) \cdot w(x), && (3)\\
\mathit{cost}_\lambda(X,Y) &= \lambda \cdot \mathit{error}(X,Y) + v(Y). && (4)
\end{aligned}
$$

Remark: We occasionally abuse our notation slightly by identifying a singleton set with its lone element. For example, we generally write $\mathit{error}(X,x)$ instead of $\mathit{error}(X,\{x\})$.

For any nonempty set of points X and integer k, $1 \leq k \leq |X|$, we let $\mathit{radius}_k(X)$ (resp., $\mathit{error}_k(X)$) denote the minimum, over all subsets Y of X such that $|Y| = k$, of $\mathit{radius}(X,Y)$ (resp., $\mathit{error}(X,Y)$). Similarly, for any nonempty set of points X and scaling factor λ, we let $\mathit{cost}_\lambda(X)$ denote the minimum, over all nonempty subsets Y of X, of $\mathit{cost}_\lambda(X,Y)$.

In Sections 1.2 through 1.5 below, we define a number of location problems and review the prior work on these problems. In our discussions of prior work, we restrict our attention to the important special case $\alpha = 1$, since most of the existing work assumes a strict triangle inequality. Section 1.6 gives an outline of the remainder of the paper.

1.2 The k-center and k-median problems

A nonempty set of points X is said to achieve a *radius (resp., error) ratio* of a if $radius(U, X)$ (resp., $error(U, X)$) is at most a times $radius_{|X|}(U)$ (resp., $error_{|X|}(U)$). Given an index k, the *k-center (resp., k-median) problem* asks us to determine a set of k points with minimum radius (resp., error) ratio. A k-center (resp., k-median) algorithm is *a-approximate* if it computes a set of k points with radius (resp, error) ratio a.

We now give a brief overview of the approximability results known for the k-center and k-median problems. The farthest point technique of Gonzalez [5] yields a simple greedy 2-approximate k-center algorithm running in $O(nk)$ time. Hochbaum and Shmoys [7] match this factor-of-2 approximation bound (albeit with a somewhat worse running time) using a more general approximation technique that is applicable to a certain class of "bottleneck" problems that includes k-center. Hochbaum and Shmoys [7] also show that no polynomial time k-center algorithm can achieve an approximation factor better than 2 unless $\mathbf{P} = \mathbf{NP}$. Thus, the approximability of k-center is well understood. The situation with respect to the k-median problem is somewhat more complicated. The first nice k-median algorithm is due to Charikar *et al.* [3]. That result has subsequently been improved in terms of both quality of approximation and running time. Currently, the best approximation factor associated with any nice k-median algorithm is $3 + \varepsilon$, where ε is an arbitrarily small positive constant; this result is due to Arya *et al.* [1]. Jain *et al.* [8] show that there is no nice $(1+2/e)$-approximate k-median algorithm unless **NP** $\subseteq$ **DTIME**$[n^{O(\log \log n)}]$. The reader is referred to [8] for a more complete survey of prior work on the k-median problem.

1.3 The incremental center and median problems

We define a *rank function* as a numbering of the points from 0 to $n - 1$. A rank function r is said to achieve a *radius (resp., error) ratio* of a if for any index k, $radius(U, \{x \in U \mid r(x) < k)$ (resp., $error(U, \{x \in U \mid r(x) < k))$, is at most a times $radius_k(U)$ (resp., $error_k(U)$). The *incremental center (resp., median) problem* asks us to determine a rank function r with minimum radius (resp., error) ratio. An incremental center (resp., median) algorithm is a-approximate if it computes a rank function with radius (resp., error) ratio a.

The farthest point technique of Gonzalez [5] provides a 2-approximate $O(n^2)$-time incremental center algorithm. Given the hardness result for the k-center problem, no polynomial time incremental center algorithm can achieve a better radius ratio unless $\mathbf{P} = \mathbf{NP}$. The incremental median problem is addressed in [10], where it is motivated within an online framework and referred to as the online median problem. The incremental k-median algorithm of Mettu and Plaxton [10] runs in $O(n^2)$ time if the ratio of the maximum interpoint distance to the minimum interpoint distance is $2^{O(n)}$, and achieves a cost ratio of approximately 30. More recently, Mettu and Plaxton [11] have presented the fastest (randomized) nice k-median algorithm known. That algorithm runs in $O(nk)$ time for k between $\log n$ and $\frac{n}{\log^2 n}$; see [11] for the general time bound.

1.4 The facility location problem

We say that a nonempty set of points X has a *cost ratio of a* with respect to a given scaling factor λ if $cost_\lambda(U, X) \leq a \cdot cost_\lambda(U)$. The *facility location problem* asks us to determine a nonempty set of points with minimum cost ratio with respect to a given scaling factor. A facility location algorithm is a-approximate if it computes a set of points with cost ratio a with respect to any given scaling factor.

The first nice facility location algorithm is due to Shmoys *et al.* [12]. That algorithm has subsequently been improved in terms of both quality of approximation as well as running time. Currently, the best approximation bound established for any nice facility location algorithm is approximately 1.52; this result is due to Mahdian *et al.* [9]. Guha and Kuller [6] show that there is no nice 1.463-approximate facility location algorithm unless **NP** $\subseteq$ **DTIME**$[n^{O(\log \log n)}]$. The fastest nice facility location algorithm known is the $O(n^2)$-time greedy algorithm presented in [10], which achieves an approximation ratio of 3. The reader is referred to [9] for a more complete survey of prior work on the facility location problem.

1.5 The incremental facility location problem

A *threshold sequence* is a nondecreasing sequence of values $0 = t_1 \leq t_2 \leq \cdots \leq t_n$ drawn from $\mathbf{R} \cup \{\infty\}$.

We say that a rank function r and threshold sequence $0 = t_1 \leq t_2 \leq \cdots \leq t_n$ achieve a *cost ratio* of a if for any scaling factor λ,

$$cost_\lambda(U, \{x \in U \mid r(x) < k\}) \quad \leq \quad a \cdot error_k(U) \tag{5}$$

where k is the largest index such that $t_k \leq \lambda$.

The *incremental facility location problem* asks us to determine a rank function and threshold sequence with minimum cost ratio. An incremental facility location algorithm is said to be a-approximate if it computes a rank function and threshold sequence with cost ratio a.

There is no prior work on the incremental facility location problem as we are introducing it for the first time in the present paper.

1.6 Outline of the remainder of the paper

The remainder of the paper is organized as follows. Section 2 presents a nice incremental facility location algorithm. Section 3 develops hierarchical versions of the k-center, k-median, and facility location problems. As discussed in Section 3, the work of Dasgupta [4] provides a nice hierarchical center algorithm. In Section 4, we present a simple algorithm for converting a good solution to the incremental median (resp., facility location) problem into a good solution for the hierarchical median (resp., facility location) problem. This property is captured by our main techincal lemma, Lemma 4.10. In Section 5, we use Lemma 4.10 and the incremental median result of Mettu and Plaxton [10] to establish our main theorem with respect to the hierarchical median problem. Similarly, in Section 6, we use Lemma 4.10 and the incremental facility location result of Section 2 to establish our main theorem with respect to the hierarchical facility location problem.

2 A Nice Incremental Facility Location Algorithm

In this section we prove Theorem 1 below. Theorem 1 provides a key building block for the nice hierarchical facility location algorithm of Section 6. (The hierarchical facility location problem is defined in Section 3.4.)

Theorem 1 *There is a nice incremental facility location algorithm.*

Let $\mathcal{A}$ be any existing c-approximate nice facility location algorithm, where c is some positive constant. A number of such algorithms have been presented in the literature, though the presentation is typically restricted to the special case $\alpha = 1$ (i.e., the strict form of the triangle inequality is assumed). In order to make use of such an algorithm in the present context, we need to ensure that it can be modified to yield a constant factor guarantee for any constant α. Fortunately, this is invariably a straightforward exercise. For example, it is easy to verify that the simple $O(n^2)$-time facility location algorithm presented in [10] has this property.

Let $\mathcal{I}$ denote a given instance of the incremental facility location problem. Thus for any scaling factor λ, $(\mathcal{I}, \lambda)$ is an instance of the facility location problem.

If $|U| = 1$, or every point has value zero, or every point has weight zero, then the theorem is straightforward to prove. Therefore, in what follows, we assume that none of these conditions hold. Let v^- and v^+ denote the minimum and maximum nonzero point values, respectively. Let w^- denote the minimum nonzero point weight. Let W denote the sum of the point weights. Let d^- and d^+ denote the minimum and maximum interpoint distances.

We will prove Theorem 1 by using $\mathcal{A}$ as a subroutine in an $8c$-approximate nice incremental facility location algorithm $\mathcal{B}$. (Remark: The factor of 8 can easily be improved to $4 + \varepsilon$, for an arbitrarily small positive constant ε, and perhaps further. Our current goal is to simply establish some constant approximation bound.) We begin by studying optimal or near-optimal solutions to the facility location instance $(\mathcal{I}, \lambda)$ for various ranges of λ.

First let us consider the case where λ is sufficiently large. In particular, let us assume that $\lambda \geq \frac{v^+}{d^- w^-}$. In this case, we claim that $X = \{x \mid w(x) > 0\}$ is an optimal solution to the facility location instance $(\mathcal{I}, \lambda)$. To see this, let Y be an arbitrary solution and note that $\mathit{error}(U, X) = 0$ and $\mathit{error}(U, Y) \geq \mathit{error}(X, Y) \geq d^- w^- \cdot |X \setminus Y|$, so $\lambda(\mathit{error}(U, Y) - \mathit{error}(U, X)) \geq v^+ \cdot |X \setminus Y|$. Furthermore, $v(X) - v(Y) \leq v^+ \cdot |X \setminus Y|$. Thus $\mathit{cost}_\lambda(U, X) \leq \mathit{cost}_\lambda(U, Y)$ for $\lambda \geq \frac{v^+}{d^- w^-}$.

Now let us consider the case where λ is sufficiently small. In particular, let us assume that $\lambda \leq \frac{v^-}{d^+ W}$. We consider two subcases. For the first subcase, assume there exists a point x such that $v(x) = 0$. In this subcase we claim that $X = \{x \mid v(x) = 0\}$ is an optimal solution to $(\mathcal{I}, \lambda)$. To see this, let Y be an arbitrary solution, and observe that: if $Y \subseteq X$ then $\mathit{error}(U, X) \leq \mathit{error}(U, Y)$ and $v(X) = v(Y) = 0$, so $\mathit{cost}_\lambda(U, X) \leq \mathit{cost}_\lambda(U, Y)$; if $|Y \setminus X| > 0$, then $\mathit{error}(U, X) - \mathit{error}(U, Y) \leq d^+ W$ and $v(Y) - v(X) \geq v^-$, so $\mathit{cost}_\lambda(U, X) \leq \mathit{cost}_\lambda(U, Y)$ by the case assumption. For the second subcase, assume that $v(x) > 0$ for every point x. In this subcase we claim that the solution $X = \{x\}$, where x is a point such that $v(x) = v^-$, is within a factor of two of optimal. To see this, note that $\mathit{cost}_\lambda(U, Y) \geq v(Y) \geq v^-$ for any solution Y, while $\mathit{error}(U, x) \leq d^+ W$, so that $\mathit{cost}_\lambda(U, x) \leq \lambda d^+ W + v^- \leq 2v^-$ by the case assumption.

We now define a sequence of scaling factors $\langle \lambda_i \mid 0 \leq i < \ell \rangle$, where $\lambda_i = \frac{v^+}{4^i d^- w^-}$ and ℓ is the least integer such that $\lambda_{\ell-1} \leq \frac{v^-}{2d^+ W}$. Thus $\ell = \Theta(\log \frac{d^+ v^+ W}{d^- v^- w^-})$, which is bounded by a polynomial in the size of the input. We compute a solution X_i for each facility location instance $(\mathcal{I}, \lambda_i)$, $0 \leq i < \ell$, as follows. For $i = 0$ we use the approach discussed above for the case where $\lambda \geq \frac{v^+}{d^- w^-}$. Thus X_0 has optimal cost with respect to any scaling factor λ greater than or equal to λ_0. For $i = \ell - 1$ we use the approach discussed above for the case where $\lambda \leq \frac{v^-}{d^+ W}$. Thus $X_{\ell-1}$ has a cost ratio of 2 with respect to any scaling factor less than or equal to $2\lambda_{\ell-1}$. For each i such that $0 < i < \ell - 1$, we run $\mathcal{A}$ on the instance $(\mathcal{I}, \lambda_i)$ to obtain a solution X_i with cost ratio c.

Let $\lambda'_0 = \infty$, $\lambda'_i = 2\lambda_i$ for $0 < i < \ell$, and $\lambda'_\ell = 0$. Then the claims established in the preceding paragraph, along with Lemma 2.1 below, immediately imply that for every i, $0 \leq i < \ell$, the solution X_i has cost ratio $2c$ with respect to any scaling factor λ such that $\lambda'_{i+1} \leq \lambda < \lambda'_i$.

Lemma 2.1 *If X is a solution to the facility location instance $(\mathcal{I}, \lambda)$ with cost ratio a, then for any scaling factor λ' such that $\lambda/2 \leq \lambda' \leq 2\lambda$, X is a solution to the facility location instance $(\mathcal{I}, \lambda')$ with cost ratio $2a$.*

Proof: If $\lambda \leq \lambda' \leq 2\lambda$ then the result follows since $\mathit{cost}_{\lambda'}(U) \geq \mathit{cost}_\lambda(U)$ and $\mathit{cost}_{\lambda'}(U, X) \leq 2 \cdot \mathit{cost}_\lambda(U, X)$. Similarly, if $\lambda/2 \leq \lambda' \leq \lambda$, the result follows since $\mathit{cost}_{\lambda'}(U) \geq \mathit{cost}_\lambda(U)/2$ and

$cost_{\lambda'}(U, X) \leq cost_{\lambda}(U, X)$. ■

We now inductively define an increasing sequence of integers $0 = a_0 < a_1 < \cdots < a_m$ as follows. For each successive positive integer i, we define a_i as the least integer such that

$$2 \cdot cost_{\lambda_{a_i}}(U, X_{a_i}) \leq cost_{\lambda_{a_{i-1}}}(U, X_{a_{i-1}})$$

if such an integer exists; otherwise, we set a_i to ℓ and terminate the sequence. By the analysis of the preceding paragraph, coupled with the observation that the cost of a solution does not increase if the scaling factor is decreased, we obtain that for every i, $0 \leq i < m$, the solution X_{a_i} has cost ratio $4c$ with respect to any scaling factor λ such that $\lambda'_{a_{i+1}} \leq \lambda < \lambda'_{a_i}$.

For each i, $0 \leq i < m$, let $Y_i = \cup_{i \leq j < m} X_{a_j}$ and note that

$$v(Y_i) \leq \sum_{i \leq j < m} v(X_{a_j})$$

and

$$error(U, Y_i) \leq error(U, X_{a_i}),$$

so

$$\begin{aligned} cost_{\lambda}(U, Y_i) &\leq \sum_{i \leq j < m} cost_{\lambda}(U, X_{a_j}) \\ &\leq 2 \cdot cost_{\lambda}(U, X_{a_i}) \end{aligned}$$

for any scaling factor λ. Combining this with the claim of the previous paragraph, we obtain that for every i, $0 \leq i < m$, the solution Y_i has cost ratio $8c$ with respect to any scaling factor λ such that $\lambda'_{a_{i+1}} \leq \lambda < \lambda'_{a_i}$.

Thus we have obtained a sequence of solutions $Y_{m-1} \subseteq \cdots \subseteq Y_0$ for which Y_{m-1} has cost ratio $8c$ with respect to any scaling factor λ such that $0 = \lambda'_{a_m} \leq \lambda < \lambda'_{a_{m-1}}$, Y_{m-2} has cost ratio $8c$ with respect to any scaling factor λ such that $\lambda'_{a_m-1} \leq \lambda < \lambda'_{a_{m-2}}$, and so on up to Y_0, which has cost ratio $8c$ with respect to any scaling factor λ such that $\lambda'_{a_1} \leq \lambda < \lambda'_{a_0} = \infty$. Given such a sequence of solutions it is straightforward to compute a rank function and threshold sequence with cost ratio $8c$. This completes the proof of Theorem 1.

3 Hierarchical Clustering and Some Related Notions

Hierarchical clustering is a widely used technique in data analysis. In Section 3.1 below, we review the definition of a hierarchical clustering and describe the standard dendrogram-based approach to depicting a given a hierarchical clustering. Section 3.2 introduces a closely related structure that we refer to as a hierarchical assignment. Section 3.3 defines a special case of a hierarchical assignment that we refer to as a hierarchical ordering. Section 3.4 uses the notion of a hierarchical ordering to define several hierarchical location problems.

3.1 Hierarchical Clustering

A *clustering* is a partition of U into a number of nonempty sets, or *clusters*. A k-clustering is a clustering with k clusters. The *radius (resp., error) of a k-clustering* with associated clusters X_i, $0 \leq i < k$, is defined as $\max_{0 \leq i < k} radius_1(X_i)$ (resp., $\sum_{0 \leq i < k} error_1(X_i)$).

A *hierarchical clustering* is a set of n clusterings containing one k-clustering for each index k, and such that for any index k less than n, the $(k+1)$-clustering can be transformed into the k-clustering by merging some pair of clusters.

Question 3.1 *Does every metric space admit a hierarchical clustering for which each associated k-clustering has radius (resp., error) within a constant factor of optimal?*

Dasgupta [4] answered the radius version of Question 3.1 in the affirmative. He left open the question of whether a similar result holds with respect to error. In Section 3.3 we define the notion of a hierarchcial ordering and formulate a stronger version of Question 3.1 with respect to hierarchical orderings.

We remark that there are $\prod_{2\leq k\leq n}\binom{k}{2} = n!(n-1)!2^{1-n}$ distinct hierarchical clusterings of U, since there is a unique n-clustering and there are $\binom{k}{2}$ different merge operations that can be applied to any k-clustering to obtain a $(k-1)$-clustering. Furthermore, the sequence of $n-1$ merges performed in successively transforming the n-clustering into the 1-clustering induce an n-leaf binary tree in which each leaf corresponds to a point and each of the $n-1$ internal nodes corresponds to a merge. Thus it is natural to consider depicting a hierarchical clustering using a standard binary tree diagram. The shortcoming of such a representation is that information regarding the relative order of the merges is, in general, lost. For example, in a binary tree in which several nodes appear at the same level, we cannot tell in which order the corresponding merges are performed.

A *dendrogram* is a drawing of a binary tree that preserves the total order on the internal nodes (induced by the merge operations) by ensuring that no two internal nodes appear at the same height on the page. In addition, the n leaves are normally arranged along a horizontal line at the bottom of the tree.

Remark: Sometimes the height of an internal node not only encodes the relative order of the merges, but is in fact proportional to some distance measure between the two clusters being merged. This sort of approach is well-suited to the depiction of hierarchical clusterings obtained via agglomerative heuristics (e.g., single, complete, or average linkage) that repeatedly merge the two closest clusters (according to some distance measure such as closest pair, farthest pair, or average distance) and for which it can be proven that the distances associated with successive merges are nondecreasing.

The primary appeal of the dendrogram representation of a hierarchical clustering is that it enables one to visualize the data at any desired level of granularity. To visualize the k-clustering associated with some desired value of k, one simply scans the dendrogram for the height at which a horizontal line leaves $k-1$ internal nodes above and $n-k$ internal nodes below. Note that the k tree edges cut by such a horizontal line lead downwards to the roots of k subtrees. The k sets of leaves associated with these k subtrees form the desired k-clustering.

An issue that arises in generating a dendrogram representation of a given hierarchical clustering is that there is more than one dendrogram corresponding to a given hierarchical clustering. More precisely, it is well known that there are 2^{n-1} different dendrograms corresponding to a given hierarchical clustering. This factor arises because exchanging the left and right subtrees of any internal node in a dendrogram yields an alternative encoding of the same hierarchical clustering. The problem of determining which of the 2^{n-1} possible dendrograms to use to represent a given hierarchical clustering is sometimes called the *leaf ordering problem*. Various approaches have been proposed for addressing the leaf ordering problem. For example, Bar-Joseph *et al.* [2] have recently presented an $O(n^3)$-time dynamic programming algorithm that can be used to compute a leaf ordering minimizing the sum of the distances between adjacent points in the ordering. In Section 3.3 we suggest a natural alternative approach to the leaf ordering problem. We also describe how our approach can be used in combination with any given leaf ordering algorithm.

3.2 Hierarchical Assignment

An *assignment* is a function from U to U. A k*-assignment* is an assignment with a range of size k. The *radius (resp., error) of an assignment* σ is defined as $\max_{x\in U} d(x,\sigma(x))$ (resp., $\sum_{x\in U} d(x,\sigma(x))\cdot w(x)$).

A *hierarchical assignment* is a set of n assignments containing one k-assignment for each index k, and such that for any index k less than n, there exists a pair of points x and y for which the $(k+1)$-

assignment can be transformed into the k-assignment by reassigning to x all points assigned to y. Note that this transformation may be viewed as an "oriented merge" of the two sets of points mapped to x and y in the $(k+1)$-assignment. (We consider the merge to be oriented because the union of these sets of points is assigned to x, and not y, in the k-assignment.)

A notable difference between a hierarchical assignment and a hierarchical clustering is that whereas there is only one possible n-clustering of U, there are $n!$ possible n-assignments, one corresponding to each permutation. Furthermore, for $k > 1$, there are $k(k-1)$ different oriented merge operations that can be applied to any k-assignment to obtain a $(k-1)$-assignment. It follows that there are exactly $(n!)^2(n-1)!$ distinct hierarchical assignments of U.

We define a *parent function p with respect to a given rank function* r as a mapping from U to U such that $p(x) = x$ if $r(x) = 0$ and $r(p(x)) < r(x)$ otherwise.

The foregoing discussion suggests the following *permutation-rank-parent* representation in which a hierarchical assignment with associated k-assignment σ_k, $1 \le k \le n$ is represented by specifying the following information: (1) The permutation σ_n; (2) The rank function r such that the range of σ_k is equal to $\{x \mid r(x) < k\}$; (3) The parent function p with respect to r such that for any index k less than n, the oriented merge operation transforming σ_{k+1} into σ_k reassigns to $p(x)$ all points assigned to x, where $r(x) = k$.

Note that there are $n!$ choices for the permutation σ_n and $n!$ choices for the rank function r. Furthermore, for every choice of σ_n and r, there are $(n-1)!$ choices for the parent function p. Thus there are $(n!)^2(n-1)!$ possible permutation-rank-parent representations, one for each hierarchcial assignment.

3.3 Hierarchical Orderings

We define a *hierarchical ordering* as a hierarchical assignment for which the associated k-assignment is idempotent for all k. Note that the identity assignment is the only idempotent n-assignment on a set of n points. Furthermore, for any index $k < n$, if the $(k+1)$-assignment associated with a hierarchical assignment is idempotent, then so is the k-assignment. Thus we can equivalently define a hierarchical ordering as a hierarchical assignment for which the associated n-assignment is the identity assignment. Thus the permutation-rank-parent representation for hierarchical assignments described in Section 3.2 corresponds to a rank-parent representation for hierarchical orderings, and there are exactly $n!(n-1)!$ hierarchical orderings.

Question 3.2 *Does every metric space admit a hierarchical ordering for which each associated k-assignment has radius (resp., error) within a constant factor of optimal?*

The following view of a hierarchical ordering may be useful in order to better understand the relationship between Question 3.2 above and Question 3.1 posed (and answered, for the radius case) by Dasgupta [4]. A hierarchical ordering may be interpreted as a hierarchical clustering in which the points of each cluster are assigned to a unique "representative" point in the cluster, subject to the additional constraint that when two clusters X and Y are merged, the representative of the resulting cluster is required to be chosen as either the representative of X or the representative of Y. If we were to drop the latter constraint, there would be no difference between the hierarchical ordering questions posed above and the corresponding hierarchical clustering questions posed by Dasgupta. But by constraining the choice of representative, we only make it more difficult to remain within a constant factor of optimal for all indices k.

For the radius version of the problem, the α-approximate triangle inequality implies that for any cluster X and point x in X, $\mathit{radius}(X, x) \le 2\alpha \cdot \mathit{radius}_1(X)$. Given that we are assuming α to be a constant, this implies that a given metric space admits a hierarchical ordering for which each associated k-assignment has radius within a constant factor of optimal if and only if it admits a hierarchical clustering for which each associated k-clustering has radius within a constant factor of optimal. So, Dasgupta's work [4] immediately provides a positive answer to the radius version of Question 3.2.

For the error version of the problem, which is the primary focus of the present paper, note that the (weighted) sum of distances to the representative of a given cluster can vary dramatically (by a factor essentially as large as the diameter of the metric space) depending on the choice of cluster representative. Thus the error version of Question 3.2 is stronger than the error version of Question 3.1 in that a positive answer to the former question immediately implies a positive answer to the latter question, but not vice versa.

In Section 5 we resolve the error version of Question 3.2 in the affirmative, thereby also providing a positive answer to the error version of Question 3.1. (In fact, for any constant α, we provide a positive answer to Question 3.2 for any α-approximate metric space.)

Let us now briefly return to the leaf ordering problem mentioned at the end of Section 3.1. Earlier we saw that the leaf ordering problem arises because there are 2^{n-1} different dendrograms corresponding to a given hierarchical clustering. But the number of dendrograms is exactly equal to the number of hierarchical orderings, so if we encode a hierarchical ordering as a dendrogram by adopting the convention that the leftmost leaf in each subtree is the representative of the cluster corresponding to that subtree, then the leaf ordering problem goes away.

On the other hand, there may be applications in which the flexibility associated with the leaf ordering problem is viewed as advantageous, since it allows us the opportunity to optimize some auxiliary objective function in the choice of the particular dendrogram to be used to represent a given hierarchical clustering. In such a context, if we wish to represent a hierarchical ordering instead of a hierarchical clustering, it may be preferable to apply a given leaf ordering technique, and then to use the following modified dendrogram diagram to indicate the representative of each cluster. In a typical dendrogram, when two clusters are merged, a horizontal line is drawn that connects the roots of the two clusters, and a vertical line is drawn from the center of this horizontal line upward, to represent the root of the merged cluster. Instead, the vertical line representing the new root can be drawn so that it simply extends the vertical line associated with the representative. With this modified dendrogram diagram, we can apply an arbitrary leaf ordering heuristic and still represent any given hierarchical ordering.

3.4 Hierarchical Location Problems

A hierarchical ordering is said to achieve a *radius (resp., error) ratio* of a if each associated k-assignment has radius (resp., error) at most a times $radius_k(U)$ (resp., $error_k(U)$). The *hierarchical center (resp., median) problem* asks us to determine a hierarchical ordering with minimum radius (resp., error) ratio. A hierachical center (resp., median) algorithm is a-approximate if it is guaranteed to return a solution with radius (resp., error) ratio a.

As indicated earlier, Dasgupta's work provides a nice hierarchical center algorithm. (Dasgupta only considers the case $\alpha = 1$, but his work is easily extended to handle an arbitrary constant α.) In Section 5, we provide a nice hierarchical median algorithm.

A hierarchical ordering, together with a threshold sequence $0 = t_1 \leq t_2 \leq \cdots \leq t_n$, achieves a *cost ratio* of a if for any scaling factor λ, if k is the largest index such that $\lambda \geq t_k$, then the k-assignment associated with the hierarchical ordering has cost at most a times $cost_\lambda(U)$. The *hierarchical facility location problem* asks us to determine a hierarchical ordering and threshold sequence with minimum cost ratio. A hierarchical facility location algorithm is a-approximate if it computes a hierarchical ordering and threshold sequence with cost ratio a. In Section 6 we present a nice hierarchical facility location algorithm.

In Section 3.3 we discussed two ways to represent a hierarchical ordering as a dendrogram. It is worth remarking that the solution to an instance of the hierarchical facility location problem, that is, a hierarchical ordering and associated threshold sequence, also has a natural dendrogram representation, since we can use the heights of the internal nodes of the dendrogram to encode the threshold sequence.

4 An Error-Preserving Parent Function

Throughout this section, we assume a fixed (and arbitrary) rank function that numbers the points in U from 0 to $n-1$. For the sake of brevity, we use the term "parent function" to refer to any parent function with respect to this rank function. In order to streamline our notation, throughout this section we identify each point with its rank. Thus, throughout this section, an expression such as "point i" refers to the point with rank i, where $0 \leq i < n$. As an additional notational convenience, for any natural number i, we let $[i]$ denote the set $\{j \mid 0 \leq j < i\}$. For example, in this section we use the expression $[n]$ to refer to the set of points U.

As discussed in Section 3.3, once we specify a parent function p to go along with the rank function fixed above, we have specified a hierarchical ordering. For any parent function p and index k, let σ_k^p denote the k-assignment associated with the hierarchical ordering determined by p, and let τ_k^p denote the assignment such that for any point i,

$$\tau_k^p(i) = \begin{cases} i & \text{if } i < k, \\ p(i) & \text{otherwise.} \end{cases} \tag{6}$$

Lemma 4.1 *For any parent function p, σ_n^p is the identity assignment and*

$$\sigma_k^p = \tau_k^p \sigma_{k+1}^p.$$

for any index k less than n.

Proof: The claim that σ_n^p is the identity assignment is immediate. The remaining claim would also be immediate if the condition $i < k$ appearing in Equation 6 were changed to $i \neq k$. By the definition of σ_n^p, the range of σ_n^p is $[k]$ for any parent function p and index k. Thus, for any parent function p and index k less than n, the assignment $\tau_k^p \sigma_{k+1}^p$ is not altered if the condition $i < k$ appearing in Equation 6 is changed to $i \neq k$, completing the proof. ∎

For any parent function p and point i, we inductively define the set T_i^p in terms of the sets T_j^p associated with points $j > i$ as follows:

$$T_i^p = \{i\} \cup \{T_j^p \mid p(j) = i\}.$$

Lemma 4.2 *For any parent function p and index k, $\{T_i^p \mid p(i) < k \leq i\}$ is a partition of $\{i \mid k \leq i < n\}$.*

Proof: We prove the claim by reverse induction on k. The base case, $k = n$, is trivial. For the induction step, let k be any index less than n, and note that

$$\{i \mid p(i) < k \leq i\} = (\{i \mid p(i) < k+1 \leq i\} \cup \{k\}) \setminus \{i \mid p(i) = k\},$$

so the claim follows by the induction hypothesis and the definition of T_k^p. ∎

The following lemma gives a useful recharacterization of of the error associated with σ_k^p for any parent function p and index k.

Lemma 4.3 *For any parent function p and index k, the error of assignment σ_k^p is equal to*

$$\sum_{i:p(i)<k\leq i} error(T_i^p, p(i)).$$

Proof: See Appendix A. ■

The remainder of this section is organized as follows. Section 4.1 presents a simple algorithm for computing a "good" parent function with respect to our arbitrary fixed choice of rank function. Section 4.2 shows that for any index k, the parent function computed by this algorithm minimizes the error of the assignment σ_k^p to within a constant factor.

4.1 Algorithm

Our algorithm for determining a "good" parent function p proceeds by computing $p(i)$ for successively lower values of i, ranging from $n-1$ down to 1. (Recall that $p(0) = 0$ for any parent function.) Hence T_i^p is fully determined by the time we are ready to compute $p(i)$, so that T_i^p can be used in the computation of $p(i)$. In particular, we set $p(i)$ to

$$\min\{j \in [i] \mid d(i,j) = d(i,[i]) \ \vee\ d(i,j) \cdot w(T_i^p) \leq c_1 \cdot \mathit{error}(T_i^p, i)\} \tag{7}$$

where c_1 is a sufficiently large constant to be specified later. (We ultimately choose $c_1 = 2\alpha + 1$.) It is straightforward to give an $O(n^2)$-time implementation of the above parent function computation.

4.2 Analysis

Throughout this section, we let p denote the particular parent function computed by the algorithm of Section 4.1.

The following lemma is a straightforward consequence of the α-approximate triangle inequality.

Lemma 4.4 *For any point z and nonempty sets of points X and Y, we have*

$$\frac{d(z,Y) \cdot w(X)}{\alpha} - \mathit{error}(X,z) \leq \mathit{error}(X,Y) \leq \alpha\,[d(z,Y) \cdot w(X) + \mathit{error}(X,z)]$$

Proof: See Appendix B. ■

Lemma 4.5 *For any nonzero point i such that $d(i,p(i)) = d(i,[i])$ and $d(i,p(i)) \cdot w(T_i^p) > c_1 \cdot \mathit{error}(T_i^p, i)$, we have*

$$\mathit{error}(T_i^p, p(i)) \quad < \quad \frac{\alpha^2(c_1+1)}{c_1 - \alpha} \cdot \mathit{error}(T_i^p, [i]).$$

Proof: Note that

$$\begin{aligned} \mathit{error}(T_i^p, [i]) \quad &\geq \quad \frac{d(i,[i]) \cdot w(T_i^p)}{\alpha} - \mathit{error}(T_i^p, i) \\ &= \quad \frac{d(i,p(i)) \cdot w(T_i^p)}{\alpha} - \mathit{error}(T_i^p, i), \end{aligned}$$

where the first inequality follows from Lemma 4.4. Lemma 4.4 also implies

$$\mathit{error}(T_i^p, p(i)) \quad \leq \quad \alpha[d(i,p(i)) \cdot w(T_i^p) + \mathit{error}(T_i^p, i)].$$

The claim of the lemma follows since $d(i,p(i)) \cdot w(T_i^p) > c_1 \cdot \mathit{error}(T_i^p, i)$. ■

Lemma 4.6 *For any nonzero point i such that $d(i, p(i)) \cdot w(T_i^p) \leq c_1 \cdot error(T_i^p, i)$, we have*

$$error(T_i^p, p(i)) \quad \leq \quad \alpha(c_1 + 1) \cdot error(T_i^p, i).$$

Proof: Immediate from Lemma 4.4. ∎

Lemma 4.7 *For any nonzero point i such that $d(i, p(i)) \cdot w(T_i^p) \leq c_1 \cdot error(T_i^p, i)$, and $p(i) \neq 0$, we have*

$$error(T_i^p, p(i)) \quad < \quad \frac{\alpha^2(c_1 + 1)}{c_1 - \alpha} \cdot error(T_i^p, [p(i)]).$$

Proof: By the minimality of our choice of $p(i)$ as specified in Equation 7, we have

$$d(i, j) \cdot w(T_i^p) \quad > \quad c_1 \cdot error(T_i^p, i)\}$$

for all j in $[p(i)]$, and hence

$$d(i, [p(i)]) \cdot w(T_i^p) \quad > \quad c_1 \cdot error(T_i^p, i)\}.$$

Thus

$$\begin{aligned} error(T_i^p, [p(i)]) \quad &\geq \quad \frac{d(i, [p(i)]) \cdot w(T_i^p)}{\alpha} - error(T_i^p, i) \\ &> \quad \left(\frac{c_1}{\alpha} - 1\right) \cdot error(T_i^p, i), \end{aligned}$$

where the first inequality follows from Lemma 4.4. The lemma then follows from Lemma 4.6. ∎

Lemma 4.8 *For any point i such that $p(i) \neq 0$, we have*

$$error(T_i^p, p(i)) \quad \leq \quad \frac{\alpha^2(c_1 + 1)}{c_1 - \alpha} \cdot error(T_i^p, [p(i)]).$$

Proof: If $d(i, p(i)) = d(i, [i])$ and $d(i, p(i)) \cdot w(T_i^p) > c_1 \cdot error(T_i^p, i)$, then the desired inequality follows from Lemma 4.5 and the observation that $[p(i)] \subseteq [i]$.

Otherwise, $d(i, p(i)) \cdot w(T_i^p) \leq c_1 \cdot error(T_i^p, i)$, and the result follows from Lemma 4.7. ∎

Let

$$c_2 \quad = \quad \frac{\alpha^3(c_1 + 1)^2}{c_1 - \alpha}.$$

Lemma 4.9 *For any nonzero point i, we have*

$$error(T_i^p, p(i)) \quad \leq \quad c_2 \cdot error(T_i^p, [i]).$$

Proof: If $d(i, p(i)) = d(i, [i])$ and $d(i, p(i)) \cdot w(T_i^p) > c_1 \cdot error(T_i^p, i)$, then the desired inequality follows from Lemma 4.5.

Otherwise, $d(i, p(i)) \cdot w(T_i^p) \leq c_1 \cdot error(T_i^p, i)$, and Lemma 4.6 implies

$$error(T_i^p, p(i)) \quad \leq \quad \alpha(c_1 + 1) \cdot error(T_i^p, i).$$

The result then follows since

$$\begin{aligned}
error(T_i^p, i) &= error(i,i) + \sum_{j:p(j)=i} error(T_j^p, i) \\
&\leq \frac{\alpha^2(c_1+1)}{c_1-\alpha} \cdot \sum_{j:p(j)=i} error(T_j^p, [i]) \\
&\leq \frac{\alpha^2(c_1+1)}{c_1-\alpha} \cdot \left(error(i,[i]) + \sum_{j:p(j)=i} error(T_j^p, [i]) \right) \\
&= \frac{c_2}{\alpha(c_1+1)} \cdot error(T_i^p, [i]).
\end{aligned}$$

(The first step follows from the definition of T_i^p and the observation that $error(i,i) = 0$. The second step follows from Lemma 4.8 since $i \neq 0$. The final step follows from the definition of T_i^p.) ■

Lemma 4.10 *For any index k, the error of σ_k^p is at most $c_2 \cdot error([n],[k])$.*

Proof: By Lemma 4.3, the error of σ_k^p is

$$\begin{aligned}
\sum_{i:p(i)<k\leq i} error(T_i^p, p(i)) &\leq \sum_{i:p(i)<k\leq i} c_2 \cdot error(T_i^p, [i]) \\
&\leq c_2 \cdot \sum_{i:p(i)<k\leq i} error(T_i^p, [k]) \\
&= c_2 \cdot error([n],[k]).
\end{aligned}$$

(The first step follows from Lemma 4.9. The second step follows since $k \leq i$. The third step follows from Lemma 4.2.) ■

In order to minimize the approximation ratio of c_2 associated with the preceding lemma, we set $c_1 = 2\alpha + 1$ and obtain $c_2 = 4\alpha^3(\alpha+1)$.

5 A Nice Hierarchical Median Algorithm

Theorem 2 *There is a nice algorithm for the hierarchical median problem.*

Proof: Immediate from Lemma 4.10 and the incremental median algorithm of Mettu and Plaxton [10].[1] ■

6 A Nice Hierarchical Facility Location Algorithm

Theorem 3 *There is a nice algorithm for the hierarchical facility location problem.*

Proof: Immediate from Theorem 1 and Lemma 4.10. ■

[1] See also the full version of [10], accepted to *SIAM Journal on Computing* and available at the author's website, for details regarding the extension of the online median result to α-approximate metric spaces for any constant α.

Acknowledgments

The author would like to thank Sanjoy Dasgupta, Xiaozhou Li, Ramgopal Mettu, Yu Sun, and Arun Venkataramani for their valuable comments on earlier drafts of this manuscript. Also, the author would like to thank Joydeep Ghosh for suggesting the modified dendrogram diagram discussed at the end of Section 3.3.

References

[1] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 21–29, July 2001.

[2] Z. Bar-Joseph, E. D. Demaine, D. K. Gifford, A. M. Hamel, T. S. Jaakkola, and N. Srebro. K-ary clustering with optimal leaf ordering for gene expression data. In *Proceedings of the 2nd Workshop on Algorithms in Bioinformatics*, September 2002.

[3] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65:129–149, 2002.

[4] S. Dasgupta. Performance guarantees for hierarchical clustering. In J. Kivinen and R. H. Sloan, editors, *Proceedings of the 15th Annual Conference on Computational Learning Theory*, volume 2375 of *Lecture Notes in Computer Science*. Springer, July 2002.

[5] T. F. González. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[6] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228–248, 1999.

[7] D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184, 1985.

[8] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 731–740, May 2002.

[9] M. Mahdian, Y. Ye, and J. Zhang. Improved approximation algorithms for metric facility location problems. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, September 2002.

[10] R. R. Mettu and C. G. Plaxton. The online median problem. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 339–348, November 2000.

[11] R. R. Mettu and C. G. Plaxton. Optimal time bounds for approximate clustering. In *Proceedings of the 18th Conference on Uncertainty in Artifical Intelligence*, pages 344–351, August 2002.

[12] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, May 1997.

A Proof of Lemma 4.3

For any parent function p and index k, we now define an associated assignment $\tilde{\sigma}_k^p$ as follows. If $i < k$, we let $\tilde{\sigma}_k^p(i) = i$. Otherwise, appealing to Lemma 4.2, we define $\tilde{\sigma}_k^p(i)$ as the unique point j such that i belongs to T_j^p and $p(j) < k \leq j$.

For any parent function p and index k, let $\tilde{\tau}_k^p$ denote the assignment

$$\tilde{\tau}_k^p(i) = \begin{cases} i & \text{if } p(i) < k, \\ p(i) & \text{otherwise.} \end{cases} \tag{8}$$

Lemma A.1 *For any parent function p, $\tilde{\sigma}_n^p$ is the identity assignment and*

$$\tilde{\sigma}_k^p = \tilde{\tau}_k^p \tilde{\sigma}_{k+1}^p$$

for any index k less than n.

Proof: The claim that $\tilde{\sigma}_n^p$ is the identity assignment is immediate. For the rest of the lemma, fix a parent function p and an index k less than n. We now complete the proof by arguing that

$$\tilde{\sigma}_k^p(i) = \tilde{\tau}_k^p(\tilde{\sigma}_{k+1}^p(i)) \tag{9}$$

for all points i. We consider three cases.

First, suppose that $i < k$. In this case, it is immediate that $\tilde{\sigma}_k^p$, $\tilde{\tau}_k^p$, and $\tilde{\sigma}_{k+1}^p$ all map i to i, so Equation 9 holds.

Next, suppose that $i = k$. We claim that $\tilde{\sigma}_k^p$, $\tilde{\tau}_k^p$, and $\tilde{\sigma}_{k+1}^p$ all map k to k, so Equation 9 holds as in the preceding case. The claim is immediate for $\tilde{\sigma}_{k+1}^p$. Since $p(k) < k$, the claim also holds for $\tilde{\tau}_k^p$. To see that $\tilde{\sigma}_k^p(k) = k$, note that k belongs to T_k^p and $p(k) < k$.

Finally, suppose that $i > k$. Let j denote $\tilde{\sigma}_{k+1}^p(i)$. Thus i belongs to T_j^p and $p(j) < k + 1 \leq j$, or equivalently, $p(j) \leq k < j$. Also, the RHS of Equation 9 is equal to $\tilde{\tau}_k^p(j)$. We now complete our analysis by considering two subcases.

For the first subcase, suppose that $p(j) = k$. Then $T_j^p \subseteq T_k^p$. Furthermore, $p(k) < k$, so the LHS of Equation 9 is equal to k. Furthermore, the subcase assumption implies that the RHS is equal to k.

For the second subcase, suppose that $p(j) < k$. Then i belongs to T_j^p and $p(j) < k < j$, so the LHS of Equation 9 is equal to j. Furthermore, the subcase assumption implies that the RHS is equal to j. ■

Lemma A.2 *For any parent function p and index k such that $k < n$, we have*

$$\tau_k^p \tau_{k+1}^p = \tau_k^p \tilde{\tau}_k^p.$$

Proof: For any point i, $\tau_{k+1}^p(i) = \tilde{\tau}_k^p(i)$ unless $p(i) < k < i$, in which case $\tau_{k+1}^p(i) = p(i)$ and $\tilde{\tau}_k^p(i) = i$. The lemma follows since the condition $p(i) < k \leq i$ implies that $\tau_k^p(i) = \tau_k^p(p(i)) = p(i)$. ■

Lemma A.3 *For any parent function p and index k, we have*

$$\sigma_k^p = \tau_k^p \tilde{\sigma}_k^p.$$

Proof: We prove the claim by reverse induction on k. The base case, $k = n$, holds since σ_n^p, τ_n^p, and $\tilde{\sigma}_n^p$ are all equal to the identity assignment. For the induction step, assume that $\sigma_{k+1}^p = \tau_{k+1}^p \tilde{\sigma}_{k+1}^p$ for some index k less than n, and note that

$$\begin{aligned}
\sigma_k^p &= \tau_k^p \sigma_{k+1}^p \\
&= \tau_k^p \tau_{k+1}^p \tilde{\sigma}_{k+1}^p \\
&= \tau_k^p \tilde{\tau}_k^p \tilde{\sigma}_{k+1}^p \\
&= \tau_k^p \tilde{\sigma}_k^p.
\end{aligned}$$

(The first step follows from Lemma 4.1. The second step follows from the induction hypothesis. The third step follows from Lemma A.2. The last step follows from Lemma A.1.) ∎

We are now ready to complete the proof of Lemma 4.3. For any parent function p and index k, the error of assignment σ_k^p is

$$\begin{aligned}
\sum_{i \in [n]} d(i, \sigma_k^p(i)) \cdot w(i) &= \sum_{i \in [n]} d(i, \tau_k^p(\tilde{\sigma}_k^p(i))) \cdot w(i) \\
&= \sum_{i \in [k]} d(i, i) \cdot w(i) + \sum_{k \le i < n} d(i, \tau_k^p(\tilde{\sigma}_k^p(i))) \cdot w(i) \\
&= \sum_{i : p(i) < k \le i} \sum_{j \in T_i^p} d(j, \tau_k^p(\tilde{\sigma}_k^p(j))) \cdot w(j) \\
&= \sum_{i : p(i) < k \le i} \sum_{j \in T_i^p} d(j, \tau_k^p(i)) \cdot w(j) \\
&= \sum_{i : p(i) < k \le i} \sum_{j \in T_i^p} d(j, p(i)) \cdot w(j) \\
&= \sum_{i : p(i) < k \le i} \mathit{error}(T_i^p, p(i)).
\end{aligned}$$

(The first step follows from Lemma A.3. For the second step, note that $\tilde{\sigma}_k^p(i) = \tau_k^p(i) = i$ for all i in $[k]$. For the third step, note that the first summation vanishes since $d(i, i) = 0$, and the second summation can be rewritten as a double summation using Lemma 4.2. For the fourth step, note that $j \in T_i^p$ where $p(i) < k \le i$ implies $\tilde{\sigma}_k^p(j) = i$. For the fifth step, note that $k \le i$ implies $\tau_k^p(i) = p(i)$. The last step follows from Equation 3.)

B Proof of Lemma 4.4

In the arguments that follow, let σ denote an assignment mapping each point in U to a nearest point in Y. To establish the lower bound on $\mathit{error}(X, Y)$, let x be an arbitrary point in X, and note that

$$\begin{aligned}
d(x, Y) &= d(x, \sigma(x)) \\
&\ge \frac{d(z, \sigma(x))}{\alpha} - d(x, z) \\
&\ge \frac{d(z, Y)}{\alpha} - d(x, z),
\end{aligned}$$

where the first inequality follows from the α-approximate triangle inequality. The lower bound now follows by multiplying through by $w(x)$ and summing over all x in X:

$$\mathit{error}(X, Y) = \sum_{x \in X} d(x, Y) \cdot w(x)$$

$$\begin{aligned} &\geq \sum_{x \in X} \left(\frac{d(z,Y)}{\alpha} - d(x,z) \right) \cdot w(x) \\ &= \frac{d(z,Y)}{\alpha} \cdot w(X) - \mathit{error}(X,z). \end{aligned}$$

The argument to establish the upper bound on $\mathit{error}(X,Y)$ is similar. Let x be an arbitrary point in X, and note that

$$\begin{aligned} d(x,Y) &\leq d(x,\sigma(z)) \\ &\leq \alpha \left[d(z,\sigma(z)) + d(x,z) \right] \\ &= \alpha \left[d(z,Y) + d(x,z) \right], \end{aligned}$$

where the second inequality follows from the α-approximate triangle inequality. The upper bound now follows by multiplying through by $w(x)$ and summing over all x in X:

$$\begin{aligned} \mathit{error}(X,Y) &= \sum_{x \in X} d(x,Y) \cdot w(x) \\ &\leq \sum_{x \in X} \alpha \left[d(z,Y) + d(x,z)) \right] \cdot w(x) \\ &= \alpha \left[d(z,Y) \cdot w(X) + \mathit{error}(X,z) \right]. \end{aligned}$$