Optimal Center-Based Clustering in Ultrametrics

Andrew Draganov & Rasmus Skibdahl Jørgensen

August 28, 2024

1 Introduction

This manuscript will show a general-purpose algorithm that solves k-means, k-median and k-center optimally in every ultrametric for all values of $k \in \{1, \dots, n\}$ in $\texttt{Sort}(n)$ time – the time it takes to sort $O(n)$ values in the ultrametric. We note that an ultrametric is any distance metric which admits the strong triangle inequality: for all x, y, z, we have $d(x,z) \leq \max(d(x,y), d(y,z))$.

Theorem (Informal). Let (T, Dist_T) be an ultrametric space. Then there exists a data structure representing (T, Dist_T) over which one can find the optimal k-center, k-median and k-means solutions for all values of $k \in \{1, \dots, n\}$ in $\texttt{Sort}(n)$ time. Furthermore, in each setting, the solutions are hierarchical.

We note that some variants of these results are known for specific ultrametrics. For example, [1] showed that one can optimally solve k-center on a specific ultrametric in $O(n^2)$ time. Similarly, [2] showed that k-median can be solved optimally in $O(n \log^2(n^2 + \Delta^2))$ time for hierarchically well-separated trees (HSTs) [1]. As noted in [2], their algorithm extends trivially to k-means. Nonetheless, we are not aware of a comprehensive treatment of this subject over all ultrametrics and center-based clustering objectives.

Overview of techniques. Our primary result is showing that k-means and k-median can be solved optimally in all ultrametrics. The proof of this relies on a surprising reduction – k-means and k-median in an ultrametric can be represented as a k-center problem in space that admits the strong-triangle inequality. Furthermore, it turns out that k-center can be solved almost trivially under the strong triangle inequality. Thus, by reducing to this easier case, we show that all center-based clustering problems admit simple optimal algorithms.

For k-center, we will see that the well-known strategy of farthest-first traversal [4] (which achieves a 2-approximation in Euclidean space) is actually optimal in ultrametrics. The intuition here is that the standard 2-approximation comes directly from the triangle inequality. Thus, the ultrametric's strong triangle inequality resolves the approximation error. The runtime being $\texttt{Sort}(n)$ comes from the fact that the ultrametric allows us to sort these distances quickly. As a result, we will solve k-center in an ultrametric by sorting the distances from largest to smallest and placing the corresponding centers.

Given this, we will conclude the proof by showing how to reduce the general problems of k-means and k-median clustering to this k-center algorithm. Interestingly, we will see that the *cost-decreases* of placing k-means or k-median centers *themselves* satisfy the strong triangle inequality. Additionally, greedily choosing centers which maximize these cost-decreases gives optimal k-means and k-median solutions. Putting the pieces together, these results imply that we can simply apply the k-center algorithm in the LCA-tree of cost-decreases to get optimal k-means and k-median solutions.

Notation. Throughout this manuscript, we use T to represent an arbitrary rooted tree with root r. We use η to represent arbitrary internal nodes and ℓ to represent arbitrary leaves. We also assume that every internal node η in the tree is equipped with a value, which we write by $d(\eta)$. We also use the notation $\eta_1 \preceq \eta_2$ to indicate that η_2 lies on the path from η_1 to r and use $\text{children}(\eta)$ and $\text{parent}(\eta)$ to indicate the direct children and parent of a node. Lastly, we use the notation from [3] with $\ell_1 \vee \ell_j$ denoting the lowest common ancestor (LCA) of a set of nodes/leaves and $T[\eta]$ denoting the subtree rooted at η. Thus, $T[\ell_1 \vee \ell_2]$ is the smallest subtree containing both ℓ_1 and ℓ_2.

[1] Here, $\log \Delta$ is the depth of the HST.

For notation on clustering, we define (k,z)-clustering as finding the set of centers $\mathbf{C} \in T$ with $|\mathbf{C}| = k$ which minimize

$$\text{Cost}_z(T,\mathbf{C}) = \sum_{\ell \in \text{leaves}(T)} \min_{c \in \mathbf{C}} \text{Dist}_T(\ell,c)^z.$$

This corresponds to k-median and k-means for $z=1$ and $z=2$, respectively. We also define k-center clustering as finding the $\mathbf{C}$ which minimizes

$$\text{Cost}_\infty(T,\mathbf{C}) = \max_{\ell \in \text{leaves}(T)} \min_{c \in \mathbf{C}} \text{Dist}_T(\ell,c).$$

2 Ultrametrics and LCA-Trees

Throughout the literature, the stand-out candidate for a hierarchical (dis-)similarity measure is the *ultrametric*. We will however require a looser definition, which we refer to as a *relaxed ultrametric*:

Definition 1. Let T be a finite metric space. Then a function[2] $\text{Dist}_T : T \times T \to \mathbb{R}$ is a *relaxed ultrametric* if, for all $\ell_1,\ell_2,\ell_3 \in T$, the following conditions are satisfied:

1. $\text{Dist}_T(\ell_1,\ell_2) = \text{Dist}_T(\ell_2,\ell_1)$
2. $\text{Dist}_T(\ell_1,\ell_2) \geq 0$
3. $\text{Dist}_T(\ell_1,\ell_3) \leq \max(\text{Dist}_T(\ell_1,\ell_2), \text{Dist}_T(\ell_2,\ell_3))$.

We will actually prove our optimal center-based clustering results over these relaxed ultrametrics. We note that every ultrametric is a relaxed ultrametric with the additional condition that the distance between two points is 0 if and only if they are the same point. Thus, our theoretical results will immediately apply to all ultrametrics.

In either case, the last condition in Definition 1 is called the *strong triangle inequality* and it is what allows ultrametrics to capture hierarchical relationships. Specifically, the strong triangle inequality implies that any three points in an ultrametric space must form an acute isosceles triangle:

Fact 1. Let Dist_T be a dissimilarity measure on a finite space T which satisfies the strong triangle inequality. Then for any $\ell_1,\ell_2,\ell_3 \in T$, one of the following holds:

1. $\text{Dist}_T(\ell_1,\ell_2) \leq \text{Dist}_T(\ell_1,\ell_3) = \text{Dist}_T(\ell_2,\ell_3)$
2. $\text{Dist}_T(\ell_1,\ell_3) \leq \text{Dist}_T(\ell_1,\ell_2) = \text{Dist}_T(\ell_2,\ell_3)$
3. $\text{Dist}_T(\ell_2,\ell_3) \leq \text{Dist}_T(\ell_1,\ell_2) = \text{Dist}_T(\ell_1,\ell_3)$

Proof. We prove this by contradiction. First, assume that all three are unequal, so WLOG $\text{Dist}_T(\ell_1,\ell_2) < \text{Dist}_T(\ell_1,\ell_3) < \text{Dist}_T(\ell_2,\ell_3)$. Then the strong triangle inequality does not hold, since $\text{Dist}_T(\ell_2,\ell_3) \not\leq \max(\text{Dist}_T(\ell_1,\ell_2), \text{Dist}_T(\ell_1,\ell_3))$.

Similarly, assume that the singleton edge is longer than the two others, i.e. $\text{Dist}_T(\ell_1,\ell_3) = \text{Dist}_T(\ell_2,\ell_3) < \text{Dist}_T(\ell_1,\ell_2)$. This also breaks the strong triangle inequality, since $\text{Dist}_T(\ell_1,\ell_2) \not\leq \max(\text{Dist}_T(\ell_1,\ell_3), \text{Dist}_T(\ell_2,\ell_3))$. □

As a result, for any three points in a relaxed ultrametric space, knowing two of the pairwise distances is sufficient to give the ordering of all three. For example, if two of the distances are equal, then the third must be smaller. This has the following fundamental consequence:

Lemma 1. Let (T, Dist_T) be a relaxed ultrametric space and let $\varepsilon \in \mathbb{R}_+$. Then the relation $\ell_i \sim \ell_j \iff \text{Dist}_T(\ell_i,\ell_j) < \varepsilon$ is an equivalence relation.

Proof. It is clear by the properties of a distance metric that the relation is reflexive and symmetric. To show that it is transitive, let ℓ_1,ℓ_2,ℓ_3 be any three points such that $\text{Dist}_T(\ell_1,\ell_2) < \varepsilon$ and $\text{Dist}_T(\ell_2,\ell_3) < \varepsilon$. Then, $\text{Dist}_T(\ell_1,\ell_3) = \varepsilon$ would contradict Fact 1.

□

[2] We Dist_T to represent ultrametric distances since these will later be shown to always correspond to trees. Although relaxed ultrametrics are not necessarily metrics, we use the word "distance" to imply dissimilarities for ease of readability.

Consequently, two points are in different equivalence classes with respect to threshold ε if and only if their pairwise distance is larger than ε in the ultrametric. This proves sufficient to show that all ultrametrics can be represented via trees:

Theorem 1. Let (X, Dist_T) be a finite relaxed ultrametric space. Then there exists a tree T with mappings $f : X \to \text{leaves}(T)$ and $d : T \to \mathbb{R}_{\geq 0}$ such that, for all $x_i, x_j \in X$, $\text{Dist}_T(x_i, x_j) = d\left(f(x_i) \vee f(x_j)\right)$.

We first put this in context before giving its proof. In words, Theorem 1 states that any ultrametric can be represented over the leaves of a tree, with the property that the distance between two leaves is uniquely determined by the value in their LCA. We note that variants of this theorem have been given elsewhere [5, 9, 7]; nonetheless, the results later in this manuscript require the form given above.

Importantly, this tree-representation of ultrametrics is slightly different from the one often found in the literature. It is commonly assumed that tree-representations of ultrametrics use the shortest-path distance and that all leaves are be equidistant from the root [8]. However, these requirements are significantly stricter than the tree structure we are considering here.[3] Let us now proceed to the proof:

Proof. We use Lemma 1 to construct the tree T. In essence, we will consider the equivalence classes induced by the largest distance d_{max}. By Lemma 1, all distances across these equivalence classes are d_{max}. Thus, each equivalence class is assigned to a subtree and their common parent is given value d_{max}. We then do this recursively.

Specifically, let $d_{max}(X)$ be the largest distance in (X, Dist_T) and let S be its corresponding set of equivalence classes. For each equivalence class $s_i \in S$, create a node η_i and set it to be a child of r. Assign $d(\eta) = d_{max}(X)$. We now apply this construction recursively for each of the children. The base case occurs when X has either one or two points. If there is only one point, x_i, we simply return a leaf ℓ_i. This leaf is given weight $d(\ell_i) = 0$ and we define $f(x_i) = \ell_i$. If X has two points, x_i and x_j, then we create two leaves ℓ_i and ℓ_j as children of the input node. The mapping is arbitrarily defined as $f(x_i) = \ell_i$ and $f(x_j) = \ell_j$. We assign the input node with weight $d(\eta) = \text{Dist}_T(x_i, x_j)$ and give the leaves weight 0, i.e. $d(\ell_i) = d(\ell_j) = 0$.

We verify the validity of this construction inductively. In the base case, the space (X, Dist_T) has either one or two points. We respectively represent these as a singleton node with value 0 or a rooted tree with two children, such that the value of the root is the distance between the two points. In both settings, all pairwise distances in X are preserved via the LCA values.

In the inductive step, assume that (X, Dist_T) has more than two points, that the maximal distance is d_{max}, and that all smaller distances are represented via distinct trees. By Lemma 1, the distance between any two nodes in separate trees must be d_{max}. Now consider that the construction described above defines a new node with value d_{max} and assigns the existing trees to it as children. Since this new node is a parent to the already-existing trees, their internal LCAs are not affected. However, the LCA between any nodes in separate subtrees has value d_{max}. Therefore, the entire ultrametric (X, Dist_T) is preserved.

□

We refer to this data structure as an *LCA-tree*. The following corollary gives the sufficient conditions for an LCA-tree to correspond to an ultrametric:

Corollary 1. Let T be an LCA-tree. For any leaf $\ell \in T$, let $p(\ell) = [\ell, \eta_i, \ldots, \eta_j, r(T)]$ be the path from ℓ to the root of the tree. Then the LCA-distances on T form a relaxed ultrametric if and only if the following condidtions are satisfied:

1. For all $\eta \in T$, $d(\eta) \geq 0$, and
2. For all $\eta_1, \eta_2 \in p(\ell)$, $\eta_1 \preceq \eta_2 \implies d(\eta_1) \leq d(\eta_2)$.

Proof. First, note that the LCA-distances in the tree satisfy the symmetry by the definition of LCA. Furthermore, condition (1) and (2) together ensure the non-negativity conditions required for a relaxed ultrametric. It therefore remains to show the strong triangle inequality. Let ℓ_1, ℓ_2 and ℓ_3 be three leaves in the LCA-tree. If they all have the same LCA (i.e., $\ell_1 \vee \ell_2 = \ell_2 \vee \ell_3 = \ell_1 \vee \ell_3$), then the leaves are equidistant in the LCA-tree and the strong triangle

[3] A simple example to see our framework's generality is the rooted tree with two leaves – one edge has weight 2 and the other has weight 1. Despite the shortest-path distances being ultrametric in this case, the children have differing distances to the root. Instead, Theorem 1 states that the ultrametric can be represented via a *new* tree (with potentially different structure), where the distances in the new tree are defined by the value in the two leaves' LCA. So, in the example, the root r of the new tree would have value $d(r) = 3$ and the new tree's edges have no notion of weight at all.

inequality is satisfied. Thus, assume WLOG that $\ell_1 \vee \ell_2 \preceq \ell_1 \vee \ell_3$. This implies that $\ell_1 \vee \ell_3 = \ell_2 \vee \ell_3$. This immediately implies that the strong triangle inequality is satisfied:

$$\ell_1 \vee \ell_2 \preceq \ell_1 \vee \ell_3 = \ell_2 \vee \ell_3 \quad \underset{\text{By Assumption}}{\Longrightarrow} \quad d(\ell_1 \vee \ell_2) \leq d(\ell_1 \vee \ell_3) = d(\ell_2 \vee \ell_3).$$

□

As a result of Corollary 1, if we wish to show that a tree's LCA-distances satisfy the strong triangle inequality, we essentially need to show that the tree's values are non-decreasing on paths from the leaves to the root. Going forward, we will rely exclusively on this LCA-distance representation of relaxed ultrametrics: unless stated otherwise, every discussion of ultrametrics will implicitly be through their LCA-tree representation.

3 k-Center in Ultrametrics

3.1 Structure of a Center-Based Solution

Before delving into how to solve center-based clustering objectives optimally in LCA-trees, we must first describe how cluster memberships are defined in them. Recall that a cluster is the set of points that are closest to a center. Since many of the center-to-leaf relationships are equidistant in an LCA-tree, we use the "marking" procedure from [2] to define a consistent notion of cluster attribution:

Let $\mathbf{C} = [c_1, \dots c_k]$ be k ordered centers that correspond to distinct leaves in the LCA-tree. We obtain the cluster memberships $C_i = \{\ell \in T : c_i = \arg\min_{c \in \mathbf{C}} d(\ell, c)\}$ by adding the centers in the given order and, for each center placed, marking the nodes in the tree from the corresponding leaf to its lowest unmarked ancestor. Thus, if we place center c_i in a leaf node, we go up the tree and mark every node with "C_i" until we hit a previously-marked node. Leaves are then assigned to clusters by finding their lowest marked ancestor.

Notice also that the number of optimal k-clusterings in any LCA-tree is exponential in k. To see this, consider an LCA-tree where there are several leaves below an unmarked node whose parent is marked. Regardless of which leaf the center is placed on, the distances (and therefore the costs) to the rest of the tree are equivalent. Thus, if an optimal clustering had a center on one of these leaves, we could replace it with a center on any of the other leaves without any change to the solution's optimality. Thus, when describing optimal solutions, we consider them equivalent up to such a permutation.

3.2 k-Center in LCA-trees

We now move to the properties of center-based clustering objectives in LCA-trees, starting with the k-center clustering task. Recall that the k-center objective requires finding k centers that minimize

$$\text{Cost}_\infty(T, \mathbf{C}) = \max_{\ell \in T} \min_{c \in \mathbf{C}} \text{Dist}(\ell, c).$$

Although the k-center task is NP-hard in the general setting, we will see that it can be solved optimally (and almost trivially) in an LCA-tree. To describe this more formally, however, we must define *hierarchical* clusters:,

Definition 2. A *hierarchical clustering* [6] of $\mathbf{X}$ over $k \in \{1, \dots, n\}$ is a set of nested partitions $\mathcal{P}_1, \dots, \mathcal{P}_n$ that observes:

1. For $k = 1$, $\mathcal{P}_1 = \{\mathbf{X}\}$.
2. For $1 < k \leq n$, $\mathcal{P}_k = (\mathcal{P}_{k-1} n C_i) \cup \{C_j, C_l\}$, such that $C_i = C_j \cup C_l$ with $C_j \cap C_l = \emptyset$ for $0 \leq i, j, l \leq k$ with $i \neq j \neq l$.

On an intuitive level, a hierarchical set of solutions means that the clustering in $\mathcal{P}_k$ is the same as the one at $\mathcal{P}_{k-1}$ except that a single cluster was split apart.

Farthest-First Traversal. We will solve k-center in LCA-trees using the farthest-first traversal algorithm [4]. The naive algorithm works by assigning the first center to a random leaf in the tree. For each subsequent center, we choose it from the subtree that has highest distance to the current centers. Although this algorithm provides a 2-approximation in standard k-center [4], it turns out that the change from the triangle inequality to the strong triangle inequality makes this method optimal.

Unfortunately, the naive farthest-first algorithm may take $O(n^2)$ time to obtain all clusterings for $k \in \{1,\dots,n\}$ since, when placing center c_i, we may have to search through $O(n)$ nodes to find the one with the next-highest cost. However, we can improve this to $\texttt{Sort}(n)$ time – the time it takes to sort a list of the $O(n)$ values in the LCA-tree – by noting that the nodes' values grow as we go up the tree. Thus, it is sufficient to sort the internal nodes by these costs and then place their corresponding centers in that order. The following lemma formalizes this intuition:

Lemma 2. Let T be an LCA-tree satisfying the conditions in Corollary 1. Then there is an algorithm that runs in $\texttt{Sort}(n)$ time and finds all the optimal k-center solutions on T for $k \in \{1,\dots,n\}$. Furthermore, these optimal solutions are hierarchical.

Proof. We will first show that the greedy farthest-first traversal is optimal in ultrametrics. After this, we will see a simple algorithm for accomplishing it in $\texttt{Sort}(n)$ time. Lastly, we will evidence that the optimal k-center solutions are hierarchical.

We first show that farthest-first traversal is optimal for every choice of k. Assume for contradiction that the greedy k-center solution $\mathbf{C}_g$ is not optimal. Then there must be another clustering $\mathbf{C}_o$ of k centers that is *actually* optimal, i.e. $\text{Cost}_\infty(T,\mathbf{C}_o) < \text{Cost}_\infty(T,\mathbf{C}_g)$. These two solutions must differ by at least one unmarked node. Of those nodes that are unmarked in $\mathbf{C}_o$ but marked in $\mathbf{C}_g$, let η be the one with largest value (with ties broken arbitrarily). This means that $\text{Cost}_\infty(T,\mathbf{C}_o) = d(\text{parent}(\eta))$. However, $\mathbf{C}_g$ *has* marked η *and every other node with larger value*. Therefore, we must have $\text{Cost}_\infty(T,\mathbf{C}_g) \leq d(\text{parent}(\eta))$. This gives the desired contradiction. Interestingly, this correctness proof does not depend on *which* leaf gets chosen to be the center in a subtree – just that one leaf is chosen. We now show that the farthest-first traversal can be executed in LCA-trees in $\texttt{Sort}(n)$ time.

The main idea is as follows: for every internal node, we must assign it one leaf as its *corresponding center*. We then sort the internal nodes by their values and place these corresponding centers one at a time. If a set of nodes have equal values, then all of their centers must be placed before the cost can decrease. Thus, ties can be broken arbitrarily.

Algorithm 2 does precisely this. It uses Algorithm 1 as a subroutine to find the corresponding centers for the internal nodes. This is done by depth-first search: at any given node η, Algorithm 2 assigns η's corresponding center as the corrsponding center of its first child. For η's remaining children that were not chosen, we store their value in a global dictionary. Algorithm 1 finally returns the dictionary of nodes in the tree and their values. Algorithm 2 concludes by sorting these nodes by their values from largest to smallest and placing the corresponding centers in this order.

The bottleneck of Algorithm 2 is the sorting, which occurs in $\texttt{Sort}(n) > O(n)$ time. The other steps run in $O(n)$ time. Lastly, placing centers in this way must be hierarchical. Every placed center corresponds to an unmarked node η in the tree. Since every leaf in $T[\eta]$ belonged to the same cluster, placing a new center only splits one cluster at a time. □

4 (k,z)-clustering in LCA-trees

The result for k-center essentially boils down to the optimality of the classic 2-approximation when applied in an ultrametric. We now turn to the more interesting result: that the optimal (k,z)-clustering solutions behave very similarly to the optimal k-center ones. This may be surprising given that the former's cost depends on the number of points in a cluster while the latter's does not. Nonetheless, in this section we will see that the optimal solutions to the (k,z)-clustering problem are hierarchical and can all be found in $\texttt{Sort}(n)$ time using the k-center algorithm as a subroutine. Recall that the (k,z)-clustering objective has the cost function

$$\text{Cost}_z(T,\mathbf{C}) = \sum_{\ell \in T} \min_{c \in \mathbf{C}} \text{Dist}_T(\ell,c)^z,$$

and corresponds to k-median and k-means for $z=1$ and $z=2$, respectively. We now give this chapter's primary result, which is a complete analog of the k-center one from Lemma 2:

Algorithm 1 `CorrespondingCenters`

Input: node η in an LCA-tree; dict `Costs` mapping nodes to distances;

if η is leaf **then**
    $c(\eta) = \eta$
    Return
end if
ChildCount $= 0$
for $\eta' \in$ children(η) **do**
    `CorrespondingCenters`$(\eta', $`Costs`$)$
    if ChildCount $= 0$ **then**
        $c(\eta) = c(\eta')$
    else
        `Costs`$\{\eta'\} = d(\eta)$
    end if
    ChildCount += 1
end for
Return

Algorithm 2 `Ultrametric-kCenter`

Input: LCA-tree T

T.root $=$ `CollapseTree`$(T$.root$)$
`Costs` = { }
`CorrespondingCenters`(T.root, `Costs`) // assume pass-by-reference on `Costs`
`Costs` = OrderedDict(`Costs`) // sorted from largest to smallest
for $\eta \in$ `Costs` **do**
    Place center at $c(\eta)$
end for

Theorem 2. Let T be an LCA-tree satisfying the conditions in Corollary 1 and let z be any positive integer. Then there is an algorithm that runs in $\texttt{Sort}(n)$ time and finds the optimal (k,z)-clustering solutions on T for all $k \in \{1,\ldots,n\}$. Furthermore, these solutions are hierarchical.

Optimal (k,z) Centers in Subtrees. To gain some preliminary insight into this, let us consider the structure of an optimal $(k=1,z=2)$-clustering solution in an LCA-tree. Placing the first center creates a trail of markings (from the leaf to the root of the tree) and the following lemma shows that this center is also optimal everywhere along this trail:

Lemma 3. Let $c_z = OPT_{1,z}(T)$ be an optimal $(1,z)$-clustering solution for LCA-tree T. Then for every subtree $T' \subset T$ such that $c_z \in T'$, c_z is an optimal $(1,z)$-clustering solution for T'.

Proof. We show this inductively. The base case is the trivial LCA-tree of one node where, inherently, the only choice of center is optimal and there are no subtrees. For the inductive case, consider LCA-tree T whose root has k children, such that each child has an optimal center placed within it. Our goal is to show that, if we were to have one center for all of T, the optimum would be one of the k centers in its subtrees. We therefore want to find the center that gives $\text{cost}_1(T)$ – the cost of optimally placing 1 center in T.

If we only had one center to place for all of T, that center must be in one of its subtrees. WLOG, let the optimal center for T be in subtree T_1. Thus, our cost for one center is necessarily of the form $\text{cost}_1(T) = \text{cost}_1(T_1) + \sum_{i=2}^{k} |T_i| \cdot d(\text{root}(T))^z$, where $|T_i|$ is the number of leaves in the i-th subtree of T. By the inductive hypothesis, we already had an optimal center for subtree T_1, implying that $\text{cost}_1(T_1)$ is minimized by choosing the optimal center in T_1. The other term $\sum_{i=2}^{k} |T_i| \cdot d(\text{root}(T))^z$ does not depend on where in T_1 the center is placed. Therefore, the optimal center from T_1 remains optimal for T.

□

The key thing to note about Lemma 3 is that it applies to any LCA-tree and, therefore, holds when placing the first center in an otherwise unmarked subtree. This notion proves essential enough that we give it its own definition:

Definition 3. Let T be an LCA-tree satisfying the strong triangle inequality, let η be a node in T and let z be a positive integer. Then η's *corresponding z-center* is $c_z(\eta) = OPT_{1,z}(T[\eta])$.

Overview of Proof for Theorem 2 and Notation. Before diving into the full algorithm for fast, optimal (k,z)-clustering, we first give a simple blueprint illustrating how it works. Assume we have placed the first center and have left a set of nodes unmarked. For each such unmarked node, we can determine its optimal center as well as how much the subtree's (k,z)-clustering cost would decrease by placing this center. Curiously, an application of Lemma 3 shows that these cost-decreases themselves satisfy the strong triangle inequality. Another application of Lemma 3 then allows us to show that greedily choosing the maximum cost-decrease gives an optimal (k,z)-clustering solution in an LCA-tree. As a result, we can apply the k-center algorithm onto the LCA-tree of cost decreases. This section is devoted to verifying the speed and optimality of the above blueprint.

Let us define the cost of a node as

$$\text{NodeCost}(\eta, z) = |T[\eta]| \cdot d(\text{parent}(\eta))^z,$$

where $|T[\eta]|$ is the number of leaves in the subtree rooted at η. This represents the cost contributed by η's leaves when η is unmarked but its parent is marked. In essence, this is the cost of $T[\eta]$ in a (k,z)-clustering solution if there is no center in $T[\eta]$. Similarly, we define the cost *decrease* at η as

$$\text{CostDecrease}(\eta, z) = \text{NodeCost}(\eta, z) - \text{Cost}(T[\eta], c_z(\eta)),$$

where $\text{Cost}(T[\eta], c_z(\eta)) = \sum_{\ell \in \text{leaves}(T[\eta])} \text{Dist}_T(\ell, c_z(\eta))^z$. We define the cost-decrease of the root node to be infinite.

In essence, the cost-decrease quantifies how much placing an optimal center in $T[\eta]$ would decrease the subtree's total cost. Importantly, the cost-decrease of a node η assumes that $\text{parent}(\eta)$ – the node directly above η – is marked. We will see in Lemma 6 that this is a reasonable assumption: every useful center we will place in the (k,z)-clustering setting will always have a marked parent.

Proving Theorem 2. We now proceed to the constituent lemmas which will prove Theorem 2. First, we see that all of the corresponding z-centers can be found in $O(n)$ time on an LCA-tree:

Lemma 4. Let T be an LCA-tree satisfying the strong triangle inequality. Then there exists an algorithm which, for all $\eta \in T$, finds $c_z(\eta)$ and $\text{Cost}(T[\eta], c_z(\eta))$ in $O(n)$ time. Furthermore, this algorithm stores the cost-decrease for all nodes η' for which $c_z(\eta') \neq c_z(\text{parent}(\eta'))$.

Proof. This is accomplished by Algorithm 3, which is essentially a depth-first implementation of Lemma 3's proof.

We prove by induction that Algorithm 3 finds the costs for all internal nodes. In the base case, our current node η is a leaf. Thus, η's corresponding z-center is η and the cost of η to this center is simply $d(\eta)$.

We now essentially reuse the logic from Lemma 3 to prove the inductive step. We begin the inductive step with a node η along with the costs and corresponding z-centers of each of η's children. That is, for all $\eta' \in \text{children}(\eta)$, we have access to both $c_z(\eta)$ and $\text{Cost}(T[\eta'], c_z(\eta'))$. We now seek the optimal center for η and what the cost would be in $T[\eta]$ after placing this center. By Lemma 3, we know that the optimal center for η is one of its children's corresponding z-centers. I.e., $c_z(\eta)$ must be equal to $c_z(\eta')$ for one of the children η'. We therefore test what the cost would be if we placed the center at each of the children and choose the minimum. By Lemma 3, this center must be optimal. We therefore record the cost of placing this center in η, concluding the correctness proof.

Since this is done by depth-first search, the algorithm runs in $O(n)$ time.

Algorithm 3 `GetCostDecreases`

Input: node η in an LCA-tree; dict `Costs` mapping nodes to their ; dict `CostDecreases` mapping nodes to their cost decrease;

if η is leaf **then**
    $c_z(\eta) = \eta$
    `Costs`$\{\eta\} = d(\eta)$
    Return
end if

if η is root **then**
    ParentDist = $d(\eta) + 1$
    `CostDecreases`$\{\eta\} = \infty$
else
    ParentDist = $d(\text{parent}(\eta))$
end if

SumOfCosts = $\sum_{\eta' \in \text{children}(\eta)} |T[\eta']| \cdot d(\eta)$
CostIfChosen = $\{\eta'$: 0 for $\eta' \in \text{children}(\eta)\}$
ChildCostDecreases = $\{\eta'$: 0 for $\eta' \in \text{children}(\eta)\}$
for $\eta' \in \text{children}(\eta)$ **do**
    `GetCostDecreases`(η', `Costs`, `CostDecreases`)
    CostIfChosen$\{\eta'\}$ = SumOfCosts $-|T[\eta']| \cdot d(\eta)$ + `Costs`$\{\eta'\}$
    ChildCostDecreases$\{\eta'\} = |T[\eta]|$·ParentDist - CostIfChosen$\{\eta'\}$
end for

ChosenChild = arg max(ChildCostDecreases)
$c_z(\eta) = c_z(\text{ChosenChild})$
`Costs`$\{\eta\}$ = CostIfChosen{ChosenChild}
for $\eta' \in \text{children}(\eta)$ such that η' is not ChosenChild **do**
    `CostDecreases`$\{\eta'\} = |T[\eta']| \cdot d(\eta)$
end for
Return

□

Rather than thinking about corresponding z-centers as those which minimize the cost, we will instead think of them through the equivalent notion of the centers which *maximize* the cost-decrease. The next lemma shows the peculiar property that these cost-decreases themselves form a relaxed ultrametric:

Lemma 5. Let T be an ultrametric LCA-tree which satisfies the conditions in Corollary 1 and let T' be the LCA-tree obtained by replacing all values in T by the cost-decreases. I.e., for all $\eta \in T'$, $d(\eta) = \text{CostDecrease}(\eta, z)$. Then the LCA-distances over T' also satisfy the conditions in Corollary 1.

Proof. By Corollary 1, showing that T' satisfies the strong trinagle inequality simply requires verifying that the cost-decreases are non-negative and monotonically non-decreasing along any leaf-root path in T'. We first show that they are monotonically non-decreasing.

Let η be an unmarked node with h children whose parent is marked. We now place the optimal center c_z in $T[\eta]$. WLOG, this center must land in one of η's children's subtrees. Call this child η_c, implying that $c_z(\eta) = c_z(\eta_c)$. We now show that the cost-decrease of η is greater than or equal to the cost decrease of η_c. After this, we will see that the cost decrease of η_c is in turn greater than the cost decrease of any of η's other children.

First, notice that $\text{CostDecrease}(\eta, z) \geq \text{CostDecrease}(\eta_c, z)$. We show this by separating $\text{CostDecrease}(\eta, z)$ into a sum of terms, of which η_c is a subset:

$$
\begin{aligned}
\text{CostDecrease}(\eta, z) &= \text{NodeCost}(\eta, z) - \text{Cost}(T[\eta], c_z(\eta)) \\
&= \left(|T[\eta_c]| \cdot d(\text{parent}(\eta))^z + \sum_{\substack{\eta' \in \text{children}(\eta) \\ \eta' \neq \eta_c}} |T[\eta']| \cdot d(\text{parent}(\eta))^z \right) \\
&\quad - \text{Cost}(T[\eta], c_z(\eta)) \\
&= \left(|T[\eta_c]| \cdot d(\text{parent}(\eta))^z + \sum_{\substack{\eta' \in \text{children}(\eta) \\ \eta' \neq \eta_c}} |T[\eta']| \cdot d(\text{parent}(\eta))^z \right) \\
&\quad - \left(\text{Cost}(T[\eta_c], c_z(\eta)) + \sum_{\substack{\eta' \in \text{children}(\eta) \\ \eta' \neq \eta_c}} |T[\eta']| d(\eta)^z \right) \\
&\geq \left(\text{NodeCost}(\eta_c, z) + \sum_{\substack{\eta' \in \text{children}(\eta) \\ \eta' \neq \eta_c}} |T[\eta']| \cdot d(\text{parent}(\eta))^z \right) - \\
&\quad \left(\text{Cost}(T[\eta_c], c_z(\eta)) + \sum_{\substack{\eta' \in \text{children}(\eta) \\ \eta' \neq \eta_c}} |T[\eta']| d(\eta)^z \right) \\
&= (\text{NodeCost}(\eta_c, z) - \text{Cost}(T[\eta_c], c_z(\eta))) \\
&\quad + \sum_{\substack{\eta' \in \text{children}(\eta) \\ \eta' \neq \eta_c}} |T[\eta']| \cdot (d(\text{parent}(\eta))^z - d(\eta)^z) \\
&= \text{CostDecrease}(\eta_c, z) + \sum_{\substack{\eta' \in \text{children}(\eta) \\ \eta' \neq \eta_c}} |T[\eta']| \cdot (d(\text{parent}(\eta))^z - d(\eta)^z) \\
&\geq \text{CostDecrease}(\eta_c, z),
\end{aligned}
$$

where both inequalities are due to $d(\text{parent}(\eta)) \geq d(\eta)$.

Lastly, we consider the cost decreases of the other children of η. Let $\eta_o \neq \eta_c$ be any other child of η. Then by Lemma 3, we have $\text{CostDecrease}(\eta_c, z) \geq \text{CostDecrease}(\eta_o, z)$. This concludes showing that the costs are monotonically non-decreasing along paths to the root.

It remains to show that the cost-decreases are necessarily non-negative. For this, we rely on the fact that the original distances in T are non-negative. Since we already know that they are monotonically non-decreasing, we therefore only have to show that the cost-decrease of placing a center at a leaf is non-negative. To this end, let ℓ be any leaf. By

Corollary 1. Then $\text{CostDecrease}(\ell, z) = \text{NodeCost}(\ell, z) - \text{Cost}(T[\ell], c_z(\ell))$. However, $\text{Cost}(T[\ell], c_z(\ell)) = d(\ell)$ while $\text{NodeCost}(\ell, z) \geq d(\text{parent}(\ell))$. Thus, $\text{CostDecrease}(\ell, z) \geq d(\text{parent}(\ell)) - d(\ell) \geq 0$.

□

We note that the cost-decrease at a leaf ℓ is not necessarily 0. Interpreting the cost-decreases as a relaxed ultrametric, this means that $\text{Dist}_T^{cost-decrease}(\ell, \ell) \neq 0$. This is why we required the definition of relaxed ultrametrics rather than standard ultrametrics.

The next lemma shows that, not only do these cost-decreases satisfy the conditions in Corollary 1, they are also monotonically *increasing* in all relevant settings. In words, Lemma 6 is saying that if $T[\eta_1]$ has non-zero cost, then placing a center there decreases this cost by a non-zero amount.

Lemma 6. Let T be an ultrametric LCA-tree satisfying the conditions in Corollary 1 and let z be a positive integer. For any leaf $\ell \in T$, let $p(\ell) = [\ell, \eta_i, \ldots, \eta_j, r(T)]$ be the path from ℓ to the root of the cost-decrease LCA-tree. Then for all $\eta_1, \eta_2 \in p(\ell)$ such that $d(\eta_1) > 0$, $\eta_1 \preceq \eta_2 \implies \text{CostDecrease}(\eta_1) < \text{CostDecrease}(\eta_2)$.

Proof. By Lemma 5, we know that the cost decreases are monotonically non-decreasing along paths to the root. Thus, it remains to show that the cost-decrease at a node η_1 is non-zero if $d(\eta_1) > 0$. To do this, we first decompose the cost-decrease at η:

$$\begin{aligned}
\text{CostDecrease}(\eta_1) &= \text{NodeCost}(\eta_1) - \text{Cost}(T[\eta_1], c_z(\eta_1)) \\
&= |T[\eta_1]| \cdot d(\text{parent}(\eta_1)) - \text{Cost}(T[\eta_1], c_z(\eta_1)) \\
&= d(\text{parent}(\eta_1)) \cdot \left(\sum_{\eta' \in \text{children}(\eta_1)} |T[\eta']| \right) - \text{Cost}(T[\eta_1], c_z(\eta_1)).
\end{aligned}$$

Now, let η_c be the child of η_1 containing $c_z(\eta_1)$. Much as in the proof of Lemma 5, we rewrite the above in terms of the costs in η_c and the costs in the other children:

$$\begin{aligned}
\text{CostDecrease}(\eta_1) &= d(\text{parent}(\eta_1)) \cdot \left(|T[\eta_c]| + \sum_{\substack{\eta' \in \text{children}(\eta_1) \\ \eta' \neq \eta_c}} |T[\eta']| \right) \\
&\quad - \text{Cost}(T[\eta_c], c_z(\eta_1)) - d(\eta_1) \cdot \left(\sum_{\substack{\eta' \in \text{children}(\eta_1) \\ \eta' \neq \eta_c}} |T[\eta']| \right) \\
&= \left(\sum_{\substack{\eta' \in \text{children}(\eta_1) \\ \eta' \neq \eta_c}} |T[\eta']| \right) \cdot (d(\text{parent}(\eta_1)) - d(\eta_1)) \\
&\quad + d(\text{parent}(\eta_1)) \cdot |T[\eta_c]| - \text{Cost}(T[\eta_c], c_z(\eta_1)) \\
&\geq d(\text{parent}(\eta_1)) \cdot |T[\eta_c]| - \text{Cost}(T[\eta_c], c_z(\eta_1)) \\
&\geq d(\text{parent}(\eta_1)) \cdot |T[\eta_c]| \\
&\geq d(\eta_1) \cdot |T[\eta_c]| \\
&> 0
\end{aligned}$$

where the first inequality is by the fact that $d(\eta_1) \leq d(\text{parent}(\eta_1))$, the second is due to the fact that the costs are strictly non-negative, the third is by the fact that $d(\text{parent}(\eta_1)) > d(\eta_1)$, and the fourth inequality is by the assumption that $d(\eta_1) > 0$.

□

Lemma 6 allows us to address the fact that the cost-decrease at a node η was defined under the assumption that η's parent is marked. By Lemma 6, the cost decreases are either strictly increasing along paths to the root or are 0. Thus, if

two nodes have equivalent non-zero cost-decreases, their subtrees must be disjoint.[4] As a result, when we go through the nodes sorted by their cost-decreases, it will always be the case that the next node we place will either have its parent marked or will have a cost-decrease of 0 (in which case it is irrelevant in terms of the optimal solutions).

We finally move to our last lemma, which shows that greedily maximizing cost-decreases results in an optimal (k,z)-clustering over the LCA-tree.

Lemma 7. The optimal (k,z)-clustering solution over an LCA-tree can be obtained by greedily choosing the k centers that, at each step, maximize the cost-decrease.

Proof. We show this by contradiction. Consider that there is a solution that was obtained greedily and another, different one that is actually optimal. We now map every center in the "optimal" solution to its closest center in the "greedy" one and observe how the optimal solution's cost changes. There are two cases that can occur: either all of the greedy centers receive one optimal center each, or there is at least one greedy center that receives more than one optimal center and another that receives none.

We start with the case where each greedy center c_g has one optimal center c_o mapped to it. By definition, c_o must be in the set of points that are assigned to c_g. Thus, it is either (a) in $T[c_g]$ or (b) in another subtree whose parent was marked by c_g. In case (a), Lemma 3 states that the greedy algorithm must have chosen c_o. In case (b), by the greedy algorithm, $T[c_g]$ has greater cost minimization than $T[c_o]$. Thus, we can decrease the cost of the optimal solution by replacing c_o by c_g, giving a contradiction. Therefore, we conclude that if one optimal center was mapped to each greedy center, then the solutions must be equivalent.

It remains to consider the case where more than one optimal center was mapped to a greedy center c_g. WLOG, let there be two optimal centers c_o^1 and c_o^2 that are mapped to c_g. By a similar argument as above, c_g must be the same as one of these optimal centers, i.e. $c_g = c_o^1$. By extension, $c_o^2 \neq c_g$. Now consider the greedy center elsewhere in the tree that had no optimal center mapped to it. Call this center c_g'. This means that the greedy algorithm had both $T[c_o^2]$ and $T[c_g']$ available to it but chose $T[c_g']$. Thus, $\text{CostDecrease}(T[c_o^2]) < \text{CostDecrease}(T[c_g'])$. We can therefore decrease the cost of the optimal solution by replacing center c_o^2 by center c_g'. This gives the desired contradiction.

□

This brings us to the primary result of this chapter, restated from before:

Theorem 2. Let T be an LCA-tree satisfying the conditions in Corollary 1 and let z be any positive integer. Then there is an algorithm that runs in $\texttt{Sort}(n)$ time and finds the optimal (k,z)-clustering solutions on T for all $k \in \{1,\ldots,n\}$. Furthermore, these solutions are hierarchical.

Proof. We use Algorithm 3 from Lemma 4 to find the cost-decreases for (k,z)-clustering in the LCA-tree. By Lemma 5, these satisfy the strong triangle inequality. Furthermore, by Lemma 7, greedily choosing centers that maximize the cost-decreases gives an optimal (k,z)-clustering.

Thus, running farthest-first traversal on the cost-decrease LCA-tree will give the partitions for (k,z)-clustering solutions. However, we must be a bit careful here: while running a naive farthest-first traversal on the cost-decrease LCA-tree will give the *partitions* of the optimal (k,z)-clustering solutions, it will not necessarily give the correct *centers*. This is due to the fact that the farthest-first traversal is optimal regardless of which center we pick for every subtree. Luckily, we have already addressed this. When considering the LCA-tree of cost-decreases, we have a mapping between every cost-decrease and the center which induces it. Thus, we will perform the farthest-first traversal by placing the nodes' corresponding z-centers. This ensures that both the partition *and* the centers align with the optimal (k,z)-clustering solutions.

Putting this all together, our final algorithm – Algorithm 4 – is quite simple. We first run Algorithm 3 to return a list of nodes and their cost-decreases. Importantly, Algorithm 3 returns only the cost-decreases for those nodes η' with $c_z(\eta') \neq c_z(\text{parent}(\eta'))$. We then sort this list in $\texttt{Sort}(n)$ time. By the discussion after Lemma 6's proof, we can safely go through this list and place the nodes' corresponding z-centers: the nodes with non-zero cost-decrease will always have their parent marked. The nodes with zero cost-decrease come at the end of the sorted list and do not affect the optimality of the soluiton. By Lemma 3, each corresponding z-center is immediately optimal in every node that it marks. Thus, Algorithm 4 optimally solves the (k,z)-clustering objective in LCA-trees which satisfy the conditions in Corollary 1. The bottleneck in this algorithm remains the time to sort the $O(n)$ internal values in the LCA-tree.

□

[4] Formally, for two nodes $\eta_1, \eta_2 \in T$ with $d(\eta_1) > 0$ and $d(\eta_2) > 0$, we have that $d(\eta_1) = d(\eta_2) \implies T[\eta_1] \cap T[\eta_2] = \emptyset$.

Algorithm 4 `Ultrametric-kz`

Input: LCA-tree T

```
Costs, CostDecreases = { }, { }
GetCostDecreases(T.root, Costs, CostDecreases) // pass-by-reference
CostDecreases = OrderedDict(CostDecreases)
for η ∈ CostDecreases do
    Place center at c_z(η)
end for
```

References

[1] Anna Beer et al. "Connecting the Dots–Density-Connectivity Distance unifies DBSCAN, k-Center and Spectral Clustering". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 80–92.

[2] Vincent Cohen-Addad et al. "Parallel and efficient hierarchical k-median clustering". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 20333–20345.

[3] Sanjoy Dasgupta. "A cost function for similarity-based hierarchical clustering". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 2016, pp. 118–127.

[4] Sariel Har-peled. *Geometric Approximation Algorithms*. USA: American Mathematical Society, 2011. ISBN: 0821849115.

[5] Bruno Leclerc. "Description combinatoire des ultramétriques". In: *Mathématiques et Sciences humaines* 73 (1981), pp. 5–37.

[6] Guolong Lin et al. "A general approach for incremental approximation and hierarchical clustering". In: *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. 2006, pp. 1147–1156.

[7] Abdolreza Mirzaei and Mohammad Rahmati. "A novel hierarchical-clustering-combination scheme based on fuzzy-similarity relations". In: *IEEE Transactions on Fuzzy Systems* 18.1 (2009), pp. 27–39.

[8] Roderick DM Page and Edward C Holmes. *Molecular evolution: a phylogenetic approach*. John Wiley & Sons, 2009.

[9] IA Pestunov, SA Rylov, and VB Berikov. "Hierarchical clustering algorithms for segmentation of multispectral images". In: *Optoelectronics, Instrumentation and Data Processing* 51 (2015), pp. 329–338.