On hierarchical diameter-clustering, and the supplier problem

Aparna Das Claire Kenyon

Abstract

Given a data set in metric space, we study the problem of hierarchical clustering to minimize the maximum cluster diameter, and the hierarchical k-supplier problem with customers arriving online.

We prove that two previously known algorithms for hierarchical clustering, one (offline) due to Dasgupta and Long and the other (online) due to Charikar, Chekuri, Feder and Motwani, are essentially the same algorithm when points are considered in the same order. We show that the analysis of both algorithms are tight and exhibit a new lower bound for hierarchical clustering. Finally we present the first constant factor approximation algorithm for the online hierarchical k-supplier problem.

1 Introduction

Clustering is the partitioning of data points into disjoint clusters (or groups) according to similarity [1, 10]. For example if the data points are books, a two clustering might consist of the clusters fiction, and non-fiction. In this way, clustering provides a way to view large amounts of data concisely. In many application domains it is useful to build a partitioning that starts with broad categories which are gradually refined more and more, thus allowing the the data to be viewed simultaneously at different levels of conciseness. This calls for a *hierarchical* or nested clustering of the data where clusters have subclusters, these have subsubclusters, and so on. For example a hierarchical clustering might first separate the books into clusters fiction and non-fiction, then separate the fiction cluster into classics and non-classics and the non-fiction cluster into math, science and history, and so on. More formally, a *hierarchical clustering* of n data points is a recursive partitioning of the points into $1, 2, 3, 4, \ldots, n$ clusters such that the $(k + 1)$th clustering is obtained by dividing one of the clusters of the kth clustering into two parts, thus making the clustering gradually more fine-grained ([5], section 10.9). This framework has long been popular among statisticians, biologists (particularly taxonomists) and social scientists [11].

A criteria which is commonly used to measure of the quality of a clustering is the maximum cluster diameter, where the diameter of a cluster is the distance between the two farthest points in the cluster. The goal is to find clusterings which minimize the maximum cluster diameter. This captures the notion that similar points should be placed in the same cluster while dissimilar points are separated. In this paper, we focus on the hierarchical diameter-clustering problem: finding a hierarchical clustering where the value of the clustering is the maximum cluster diameter.

Every associated k-clustering of the hierarchical clustering should be close to the optimal k-clustering, where the optimal k-clustering is the one that minimizes the maximum cluster diameter. The competitive ratio compares each k-clustering of the hierarchical clustering to the optimal k-clustering. The competitive ratio of a hierarchical clustering algorithm A is the supremum, over

n and over input sets S of size n, of the quantity $\max_{k\in[1,n]} A_k(S)/\mathrm{OPT}_k(S)$, where $\mathrm{OPT}_k(S)$ is the value of the optimal k-clustering of S and $A_k(S)$ is the value[1] of the k-clustering constructed by algorithm A. Thus a hierarchical clustering algorithm with a small competitive ratio, produces k-clusterings which are close to the optimal for all $1 \leq k \leq n$.

The hierarchical diameter-clustering problem was studied in work by Dasgupta and Long [4] and by Charikar, Chekuri, Feder and Motwani [2]. A simple and commonly used algorithm for this problem is the greedy "agglomerative" algorithm [5], which starts with n singletons clusters and repeatedly merges the two clusters whose union has smallest diameter. However, it is proved in [4] that this algorithm has competitive ratio $\Omega(\log k)$. The authors then propose a better, constant-factor algorithm, inspired by the "divisive" k-clustering algorithm of Gonzales [7]. The algorithm proposed in [2] is coalescent and may be partially inspired by a k-clustering algorithm by Hochbaum and Shmoys [9]. Superficially the two papers look quite different. Quoting [4]: "the earlier work of [2] uses similar techniques for a loosely related problem, and achieves the same bounds". Indeed, both papers present a 8 competitive deterministic algorithm and a $2e$ competitive randomized variant. Additionally, the algorithm from [2] focuses on online clustering, where points arrive one by one in an arbitrary sequence. We refer to the algorithm from [2] as the *tree-doubling algorithm* and to the algorithm from [4] as the *farthest algorithm*. Here are the main results from [4, 2].

Theorem 1. *(***Previous Results***)*

Result from [4]: The farthest algorithm is 8-competitive, in its deterministic form and $2e$-competitive in its randomized form for the hierarchical diameter-clustering problem.

Result from [2]: The tree doubling algorithm is 8-competitive, in its deterministic form and $2e$-competitive in its randomized form for the hierarchical diameter-clustering problem.

Our first contribution is to formally relate the two algorithms. Their specification contains some non-deterministic choices: the farthest algorithm starts from an arbitrary point, and the tree-doubling algorithm considers the points in arbitrary order. Assuming some conditions which remove the non-determinism, we prove that both in the deterministic and in the randomized cases the clustering produced by the farthest algorithm is always a refinement of the clustering produced by the tree-doubling algorithm, where refinement is defined as follows:

Definition 1. *A partition $F_1, F_2, \dots F_l$ is a refinement of a partition $D_1, D_2, \dots D_k$ iff $\forall i \leq l$, $\exists j \leq k$ such that $F_i \subseteq D_j$.*

Interestingly, both algorithms could actually be viewed as a coarser version of the greedy agglomerative algorithm used in practice.

Theorem 2. (Refinement) *Assume that the first two points labeled by the farthest algorithm have distance equal to the diameter of the input. Also assume that the tree-doubling algorithm considers points in the order in which they were labeled by the farthest algorithm. Moreover, in the randomized setting, assume that the two algorithms choose the same random value r.*

Then, for every k, the k-clustering produced by the farthest (deterministic or randomized) algorithm is a refinement of the k-clustering produced by the tree-doubling (deterministic or randomized) algorithm.

[1]If A is randomized, then $A_k(S)$ should be replaced by $E(A_k(S))$.

With this interpretation, we see that the competitive ratio of the farthest algorithm can be seen as a corollary of the competitive ratio of the tree-doubling algorithm. Could it be that the farthest algorithm is actually better? We answer this question in the negative by proving that the analysis of the farthest algorithm in [4] is tight.

Theorem 3. (Tightness) *The competitive ratio of the deterministic farthest algorithm is at least* 8.

This means that the 8 competitive ratio upper bound for the farthest algorithm is tight, and, by the refinement theorem, the 8 competitive ratio upper bound for the tree-doubling algorithm is also tight. Proving tightness of the randomized variants are open.

Can the competitive ratio be improved? We turn to the question of what is the best competitive ratio achievable for any hierarchical clustering algorithm with no computational restrictions. In other words, what is the best we can expect from a hierarchical clustering algorithm if it is allowed to have non-polynomial running time. We prove that no deterministic algorithm can achieve a competitive ratio better than 2, and no randomized algorithm can achieve competitive ratio better than (3/2). (Note that the lower bounds proved in [2] apply to the online model only and thus are incomparable to our lower bounds.)

Theorem 4. (Hierarchical lower bound) *No deterministic (respectively randomized) hierarchical clustering algorithm can have competitive ratio better than* 2 *(respectively better than* 3/2*), even with unbounded computational power.*

How general are these techniques? In our final contribution, we extend the tree-doubling algorithm to design the first constant factor approximation algorithm for the *online hierarchical supplier problem.*

In the standard (offline, non hierarchical) *k-supplier problem*, we are given a set S of suppliers and a set C of customers, with customer-supplier distances. We wish to select a set S_k of k suppliers and an assignment of each customer c to a supplier $f(c)$ in S_k so as to minimize the maximum distance from any customer to its supplier, $\max_{c \in C} d(c, f(c))$. For example, the suppliers are the fixed database templates against which we are comparing the data (customers) and which we use for classification. A 3-approximation algorithm for the k-supplier problem is mentioned in [8].

In the more difficult *online hierarchical* supplier problem, the set S of suppliers is known in advance but new customers arrive as time goes on, so C is a *sequence* of customers. When a new customer arrives, it is either assigned to one of the existing open suppliers, or it opens a new supplier. If opening a new supplier results in more than k open suppliers then two existing open suppliers merge their customer lists, and one of them closes. This requirement ensures that the hierarchical condition is satisfied, i.e that $S_{i-1} \subseteq S_i$ and that for each supplier $s \in S_i \setminus S_{i-1}$, all the customers assigned to s are assigned to the same supplier in S_{i-1}. For example, suppose customers arrive over time to use resources and we would like to dynamically increase/decrease the total number of resources allocated without having to do extensive recomputation. Using the hierarchical supplier solution, this only requires splitting/merging the customers currently assigned to one of the resources. The online model is an increasingly important framework for clustering problems, when very large amounts of data are gathered over time and needs to be analyzed and categorized on the fly (see [13] for example).

Using the tree-doubling algorithm as a subroutine, we obtain a constant-factor approximation algorithm for the online hierarchical supplier problem. (Note that in the offline case, we could equivalently have used the farthest algorithm as a subroutine. In fact, we conjecture that in the offline case, a similar result may also be obtainable using methods from [3, 12].)

Theorem 5. (Online hierarchical supplier) *For the online hierarchical k-supplier problem, there exist a deterministic* 17*-approximation algorithm and a randomized* $(1{+}4e) = 11.87$*-approximation algorithm.*

2 Proof of the Refinement Theorem

2.1 Review of the farthest algorithm from [4]

The input is a set of n points $\{x_1, \dots x_n\}$ with associated distance metric d. The algorithm has three main steps:

Labeling the points. Take an arbitrary point and label it 1. Give label i for $i \in \{2, \dots, n\}$, to the point which is farthest away from the previously labeled points. Let d_i denote the distance from i to the previous $i-1$ labeled points, i.e $d_i = \min_{1 \le j \le i-1} d(i,j)$.

Assigning levels to labelled points. For labelled point 1, set level(1) = 0. For labelled point $i \in \{2, \dots, n\}$, set $\text{level}(i) = \lfloor \log_2(d_2/d_i) \rfloor + 1$, where $d_2 = d(1,2)$.

Organizing labelled points into a tree. Organize the points into a tree, which we refer to as the Π'-tree, in the following way: Place point 1 as the root of the tree. For each point $i \in \{2, \dots, n\}$, define its parent, $\pi'(i)$, to be the point closest to i among the points with level strictly less than level(i). Insert points $i \in \{2, \dots, n\}$ into the Π' tree in order of increasing levels connecting each point i with an edge to its parent $\pi'(i)$.

The hierarchical clustering is represented implicitly in the Π'-tree. To obtain the k-clustering (of the hierarchical clustering) remove edges $(i, \pi'(i))$, for $i \in \{2, \dots, n\}$ from the Π'-tree. Deleting $k-1$ edges splits the Π'-tree into k connected components and we return these connected components as the k clusters.

It is easy to verify that this defines a hierarchical clustering, and [4] proves that it satisfies the following properties. The distances $(d_i)_i$ are a monotone non-increasing sequence, and the levels $(\text{level}(i))_i$ are a monotone non-decreasing sequence. The definition of levels imply the following bounds on d_i.

$$d_2/2^{\text{level}(i)} < d_i \le d_2/2^{\text{level}(i)-1} \tag{1}$$

In addition [4] proves that:

$$d(i, \pi'(i)) \le d_2/2^{\text{level}(i)-1} \tag{2}$$

[4] also present a randomized variant of the farthest algorithm, where the only difference is in the definition of levels. A value r is chosen uniformly at random from the interval $[0,1]$, and the levels are now defined by: level(1) = 0 and $\text{level}(i) = \lfloor \ln(d_2/d_i) + r \rfloor + 1$. The monotonicity properties are unchanged; and the two inequalities are replaced by the following.

$$e^r d_2/e^{\text{level}(i)} < d_i \le e^r d_2/e^{\text{level}(i)-1} \quad \text{and} \quad d(i, \pi'(i)) \le e^r d_2/e^{\text{level}(i)-1}. \tag{3}$$

2.2 Review of the tree-doubling algorithm from [2]

Here the input consists of a *sequence* of n points $\{x_1, \dots x_n\}$ with associated distance metric d. Let Δ denote the diameter of the points. The algorithm considers the points one by one in an online fashion and maintains a certain infinite rooted tree which we will refer to as the T^+ tree. Each node in T^+ is associated to a point, and the set of nodes associated to the same point forms an infinite path in the tree. The first point is placed at depth 0 as the root of T^+, and a copy of this point is placed at each depth $d > 0$ along with a parent edge to the copy at depth $d - 1$. When a new point p arrives we use the following insertion rule.

(Insertion rule) *Find the largest depth d that contains a point q such that $dist(p, q) \leq \Delta/2^d$. Point p is inserted into depth $d_p = d + 1$ with a parent edge to q.*

A copy of p is placed at each depth $d > d_p$ with a parent edge to the copy of p at depth $d - 1$.

To obtain a k-clustering, find the maximum depth d in T^+ which has at most k nodes. Delete all tree nodes at depth less than d. This leaves $\leq k$ subtrees rooted at the points at depth d. Delete all multiple copies of points from the subtrees and return these as the clusters.

[2] proves that the following properties are maintained as nodes are added to T^+:

Property 1. (Close-parent property) *Points at depth d are at distance at most $\Delta/2^{d-1}$ from their parents.*

Property 2. (Far-cousins property) *Points at depth d are at distance greater than $\Delta/2^d$ from one another.*

[2] also presents a randomized variant, where the only difference is in the insertion rule. A value r is chosen uniformly at random from the interval $[0, 1]$, and the insertion rule is now: *Find the largest depth d that contains a point q such that $dist(p, q) \leq e^r\Delta/e^d$. Point p is inserted into depth $d_p = d + 1$ with a parent edge to q.*

The properties are replaced by the following.

Property 3. *Points at depth d are at distance at most $e^r\Delta/e^{d-1}$ from their parents , and at distance greater than $e^r\Delta/e^d$ from one another.*

2.3 Proof of the Refinement Theorem, deterministic version

To relate the farthest and tree doubling algorithms we first make some assumptions about their nondeterministic choices. The farthest algorithm starts its labelling at an arbitrary point. We will assume the first point labelled by the farthest algorithm is at distance Δ from the second point labelled by the algorithm and thus $d_2 = \Delta$. The tree doubling algorithm receives its input points in an arbitrary order. We assume that points arrive to the tree-doubling algorithm in the order they are labeled by the farthest algorithm. Lastly we assume that ties are broken in the same way by the two algorithms. Specifically if points q, q' are tied to be the parent of some point, the point with the larger label (in the case of the farthest algorithm) or the point which arrived first (in the case of the tree doubling algorithm) is favored.

Our proof is based on an alternative construction of the tree doubling T^+ tree. Our construction, given in algorithm 1, builds a tree T based on the farthest algorithm's Π' tree. We prove that our construction is consistent with the tree doubling algorithm's insertion rule and hence T could have legitimately been constructed by the tree-doubling algorithm. Finally we argue that the k-clustering defined by the Π' tree is a refinement of the k-clustering defined by T.

Given the Π' tree of the farthest algorithm, the following algorithm constructs a T^+ tree.

```
Algorithm 1: Given $\Pi'$ construct $T^+$
(1)  Let $T$ be an empty tree
(2)  Let $\ell$ = the maximum level of the points in $\Pi'$
(3)  For each level $i = 0, \ldots \ell$
(4)      Let $L_i$ denote the points with level $i$
(5)      Let $S = L_0 \cup L_1 \cup \ldots \cup L_i$
(6)      Insert each $p \in S$ at depth $i$ of $T$ with an edge to:
(7)          The copy of $\pi'(p)$ at depth $i-1$, if $level(p) = i$ or
(8)          The copy of $p$ at depth $i-1$, if $level(p) \neq i$
(9)  Return $T$
```

Figure 1 shows the Π' tree and the corresponding tree T constructed by algorithm 1.

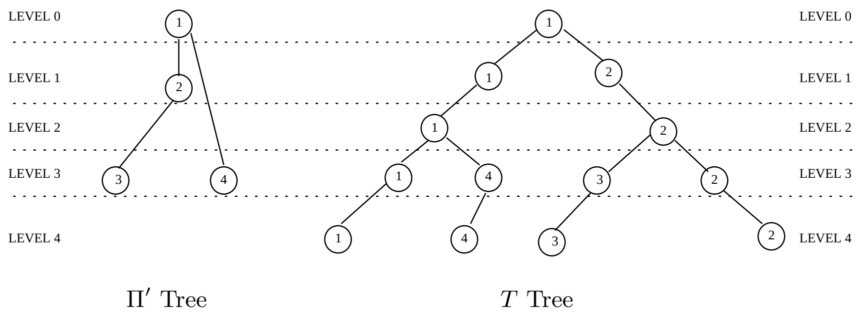


Figure 1: Π'-Tree to C^+ Tree

Let T be the tree constructed by Algorithm 1.

Lemma 1. *T satisfies the close-parent property.*

Proof. Let p be any point at depth d in T.

If $d \neq \text{level}(p)$, then by step (8) of algorithm 1, the parent of p in T is the copy of p at depth $d-1$. Thus the close parent property follows trivially in this case since $d(p,p) = 0$.

If on the other hand $d = \text{level}(p)$, then by step (7) of algorithm 1 $\text{parent}(p) = \pi'(p)$. Applying equation 2 with $d_2 = \Delta$, we have that, $d(p, \pi'(p)) \leq \Delta/2^{\text{level}(p)-1}$. Substituting $\text{level}(p) = d$ and $\pi'(p) = \text{parent}(p)$, we get: $d(p, \text{parent}(p)) \leq \Delta/2^{d-1}$. □

Lemma 2. *T satisfies the insertion rule.*

Proof. By steps (3-6) of algorithm 1 if $\text{level}(p) = d$, then p appears in T for the first time at depth d and its parent q is $\pi'(p)$.

Since $\text{level}(\pi'(p)) < \text{level}(p)$, a copy of $\pi'(p)$ must be at depth $d-1$ in T. Since T satisfies the close-parent property, $d(p, \pi'(p)) \leq \Delta/2^{d-1}$. Thus $\pi'(p)$ is qualified (distance-wise) to be the parent of p according to the insertion rule.

To show that insertion rule is satisfied we need to show that when p first arrives, there was no other point at a depth higher than $d-1$ which was close enough to p to be its parent. Let q' be any point at depth $j > d-1$, which arrived before p. Note that $q' \in \{1, \ldots, p-1\}$ since by assumption points arriving in the order they are labelled by the farthest algorithm. We need to show that $d(p,q') > \Delta/2^j$. Note that by definition of d_p, $d(p,q') \geq \min_{j\in[1,p-1]} d(p,j) = d_p$. Using the fact that level$(p) = d$ and equation 1 with $d_2 = \Delta$ we get

$$d_p > \Delta/2^{\text{level}(p)} = \Delta/2^d$$

Combining the two statements above we have that

$$d(p,q') \geq d_p > \Delta/2^d \geq \Delta/2^j,$$

where the last inequality follows since $j > d-1 \Rightarrow j \geq d$. Since $d(p,q') > \Delta/2^j$ point q' cannot be parent of p. □

We have shown that T satisfies the insertion rule and thus it can be constructed by the tree-doubling algorithm when the assumptions of Theorem 2 hold. Thus for the rest of the proof assume that the tree doubling algorithm constructs T.

Given k, the farthest algorithm removes exactly $k-1$ edges from the Π' tree and returns a clustering $F(k)$ with exactly k clusters. The tree-doubling algorithm looks for the deepest level of the T tree with at most k nodes and thus returns a clustering $D(k)$ with $\leq k$ clusters. We first show the two clusterings $D(k)$ and $F(k)$ are equivalent when they both have exactly k clusters. The refinement property then follows easily.

Lemma 3. *Let k be such that the tree doubling tree T has a depth d with exactly k vertices, then the clusterings $F(k)$ and $D(k)$ are the same.*

Proof. Let $F_1, \ldots F_k$ be the clusters returned by the farthest algorithm, where F_i contains point i. Let $D_1, \ldots D_k$ be the clusters returned by the tree-doubling algorithm, where the cluster are defined by the k points at depth d in T. Since depth d contains exactly k vertices, the monotonicity of $(\text{level}(i))_i$ implies that these points must be exactly the points $1, \ldots, k$. We will show that for any $1 \leq i \leq k$ if a point, x, is in D_i then $x \in F_i$. Since the k-clustering is a partition of the points, this immediately implies that $D_i = F_i$ for all $1 \leq i \leq k$.

Let x be a point in D_i. Since D_i contains the points in the subtree under i, there is a i-to-x path $P = (i = p_1, p_2, \ldots p_l = x)$ in T. Let $S = (i = s_1, s_2, \ldots s_m = x)$ be the sequence of points obtained by deleting all repetitions of points from P. By the construction of T we have that $s_j = \pi'(s_{j+1})$ for all $1 \leq j \leq m$ which implies that S is a valid i-to-x path in the Π'-tree. Since level L of T contains all points in $\{1, \ldots, k\}$, only point i can appear in sequence S. Thus none of the points $\{1, \ldots, k\}$ except i are in the i-to-x path in Π' tree. This implies that $x \in F_i$. □

Corollary 1. *Let k_1 be an input such that $D(k_1)$ has strictly less than k_1 clusters. Let k_2 be the minimum input such that $k_2 > k_1$ and $D(k_2)$ has exactly k_2 clusters. Then $D(k_1) \preceq F(k_1) \preceq D(k_2)$, where $A \preceq B$ stands for "B is a refinement of A".*

Proof. Let $k < k_1$ be the number of clusters in $D(k_1)$. Thus $D(k_1) = D(k)$ and T has a level with exactly k vertices. By Lemma 3, $F(k) = D(k)$. By definition of a hierarchical clustering, $F(k) \preceq F(k_1)$ as $k < k_1$. Thus we have $D(k_1) = D(k) = F(k) \preceq F(k_1)$.

Similarly, on input k_2, the tree-doubling algorithm produces a clustering with exactly k_2 clusters which implies that T has a level with exactly k_2 vertices. By Lemma 3, $F(k_2) = D(k_2)$. By definition of a hierarchical clustering, $F(k_1) \preceq F(k_2)$ as $k_1 < k_2$. Thus we have $F(k_1) \preceq F(k_2) = D(k_2)$. □

2.4 Proof of the Refinement Theorem, randomized version

Suppose the random parameter r in the randomized versions of the farthest and the tree-doubling algorithms are chosen to be the same value. Then Lemma 3 and Corollary 1 also apply to the randomized algorithms. The only change to the analysis is to use inequalities 3 instead of inequalities 2 and 1 in the proof of correctness for algorithm 1.

2.5 Nondeterministic Choices

To prove the refinement theorem, we made some assumptions about the nondeterministic choices of the two algorithms. But how much do these choices affect the performance of the algorithms?

The first point chosen by the farthest algorithm determines the value of d_2 and this in turn determines the level threshold of the Π' tree, i.e level one contains the points which are at distance $[d_2, d_2/2)$ from previously labelled points and level two contains points which are at distance $[d_2/2, d_2/2^2)$ from previously labelled points and so on. The initial point can affect the performance of the farthest algorithm by a factor up to 8. This can be demonstrated on the example we present in section 3, figure 2. On this example when the farthest algorithm chooses initial point p_1 it outputs a 5-clustering which has cost arbitrarily close to 8OPT. However the optimal 5-clustering can be obtained if p_4 is chosen as the initial point.

Points arrive to the tree doubling algorithm in an arbitrary order. How much can the ordering of points affect the performance of the tree doubling algorithm? By the refinement theorem, if points arrive in the order labelled by the farthest algorithm, there is always a way to break ties so that the tree doubling clustering is no better than the farthest clustering. However the arrival order of point can help the tree doubling algorithm perform better than the farthest algorithm. We demonstrate this on the tight example presented in 3, figure 2. If the points arrive as labelled by the farthest algorithm, the tree doubling and the farthest 5-clustering have cost 8OPT, while if the order starts with p_2, p_5, p_5', the tree doubling 5-clustering is

$$\Big\{(p_2), (p_3), (p_3')\,(p_1, p_4, p_5, q_1 \dots q_n), (p_1', p_4', p_5', q_1' \dots q_n')\Big\}$$

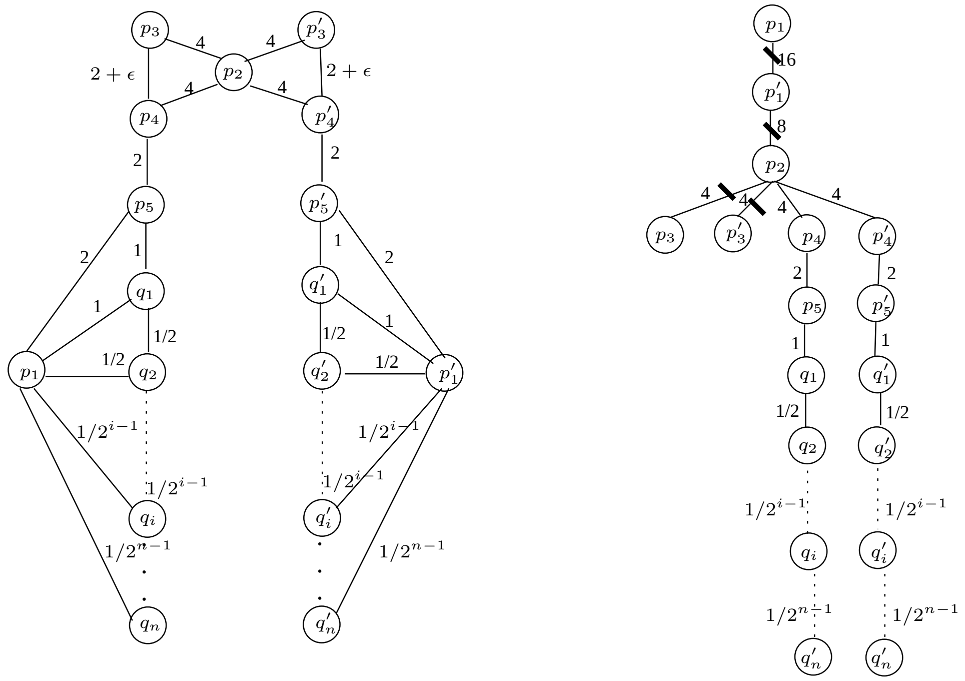
which has cost 2OPT.

Combining these observations, we see that the farthest algorithm can produce clusterings which are 8 times better than the tree doubling algorithm clusterings if the farthest algorithm starts with the best possible initial point and the tree doubling is given its points in the worst possible ordering. On the other hand the tree doubling clusterings can be 4 times better than the farthest clusterings when its points are ordered favorably and the farthest algorithm starts at the worst possible intial point.

3 Proof of the Tightness Theorem

We will prove that, for any $\epsilon > 0$, there exists an input on which the algorithm produces a hierarchical clustering where the $k = 5$ clustering is worse than the optimal 5-clustering by a factor of at least $8 - 4\epsilon$.

Choose any $\epsilon > 0$ and let $n = 2\log(1/\epsilon)$. The input set S will have $2n+9$ points; nine standard points, $p_1, p_1', p_2, p_3, p_3', p_4, p_4', p_5, p_5'$, and $2n$ additional points $q_1, q_1', q_2, q_2', \dots q_n, q_n'$. The distance among these points are shown in figure 2(a). Note that the distance from q_i to q_{i+1} and the distance from p_1 to q_i for $i \in [1, n-1]$ is $1/2^i$ and the same holds for the distance from q'_i to q'_{i+1} and the distance from p'_1 to q'_i.



(a) Points in S (b) Π'-Tree

Figure 2: Graph for Tight Example

It is easy to verify that the optimal 5-clustering of S is:

$$\Big\{ (p_1, p_5, q_1, q_2 \dots q_n), (p'_1, p'_5, q'_1, q'_2, \dots, q'_n), (p_2), (p_3, p_4), (p'_3, p'_4) \Big\}$$

where clusters (p_3, p_4) and (p'_3, p'_4) have the largest diameter of $2 + \epsilon$.

We carry out the steps of farthest algorithm and show that its 5-clustering can have a cluster of diameter $16 - (2/2^{n-1})$. For the labeling, the algorithm starts with point p_1 and obtains the following ordering:

$$p_1, p'_1, p_2, p_3, p'_3, p_4, p'_4, p_5, p'_5, q_1, q'_1, \dots q_n, q'_n$$

Thus $d(p_1, p'_1) = 16 = \Delta$ is used to define the levels for the points. We get: $level(p_1) = 0$, $level(p'_1) = 1$, $level(p_2) = 2$, p_3, p'_3, p_4, p'_4 all have level 3, p_5 and p'_5 have level 4 and for $i \in [1, n]$, $level(q_i) = level(q'_i) = i + 4$. For each point $p \neq p_1$, define $\pi'(p)$ to be the closest point to p at a strictly lower level. To define the Π'-tree, the algorithm connects each point $p \neq p_1$ with an edge to its parent $\pi'(p)$. The resulting Π'-tree is shown in figure 2(b). To obtain a 5-clustering, the algorithm removes edges $(p_i, \pi'(p_i))$ for $p_i \in \{p'_1, p_2, p_3, p'_3\}$ which yields the clustering:

$$\Big\{ (p_1), (p'_1), (p_3) (p'_3), (p_4, p_5, q_1 \dots q_n,\ p_2,\ p'_4, p'_5, q_1 \dots q'_n,) \Big\}$$

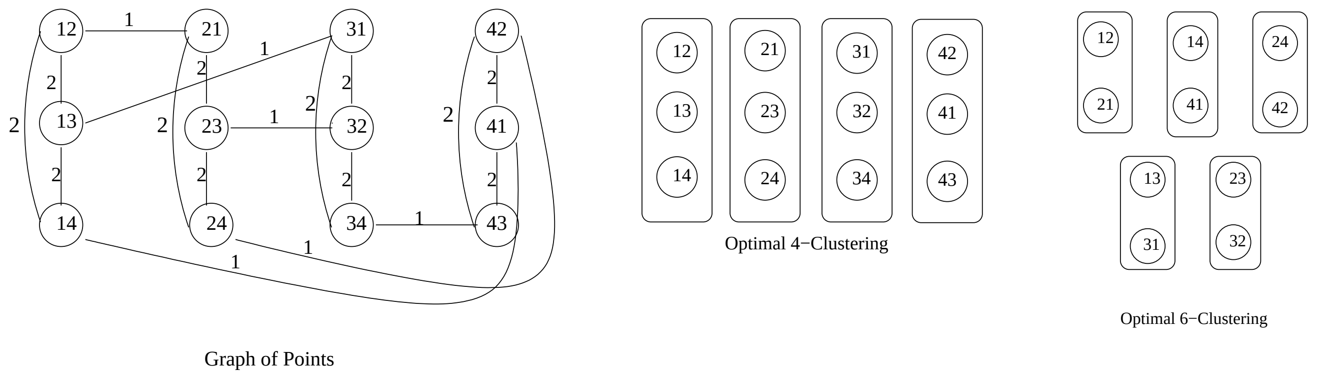


Figure 3: Lower Bound Example

Note that the last cluster has diameter $16 - 2/2^{n-1}$, which is the distance from q_n to q'_n. This proves the Tightness theorem.

4 Proof of the Hierarchical Lower Bound Theorem

What is the best competitive ratio a hierarchical clustering algorithm can hope to achieve? We demonstrate an input set on which every deterministic hierarchical clustering algorithm obtains a competitive ratio at least 2 and every randomized algorithm obtains a competitive ratio at least 1.5 even if they have unbounded computational power.

Let S denote the input points p_{ij} for $i, j \in [1, 4]$ and $i \neq j$ shown in figure 3. The distances between the points are: $d(p_{ij}, p_{ji}) = 1$ and $d(p_{ij}, p_{ik}) = 2$. (This resembles but is not the same as the example in [2], where the authors focus on the online setting). It is easy to verify that the optimal 6-clustering consists of the six pairs $p_{ij}p_{ji}$ each of diameter 1. Let $B_i = \{p_{ij} | j \in [1, 4], i \neq j\}$. Observe that B_i for $i \in [1, 4]$ is the optimal 4-clustering with each cluster having diameter 2.

4.1 The deterministic lower bound

Let A be any deterministic hierarchical clustering algorithm.

Case 1: Suppose A produces the optimal 6-clustering. Then A's clusters must be the 6 pairs $p_{ij}p_{ji}$. Since A is a hierarchical clustering algorithm, it must merge some of these pairs to obtain the 4-clustering. Merging any two of these pairs results in a cluster of diameter 4, giving us a competitive ratio of at least $A(4)/\mathrm{OPT}(4) = 4/2 = 2$.

Case 2: Suppose A does not produce the optimal 6-clustering. Then some cluster in A's 6-clustering consist of points other than some pair $p_{ij}p_{ji}$. This cluster must have diameter ≥ 2. Thus the competitive ratio for A is at least $A(6)/OPT(6) \geq 2/1 = 2$.

4.2 The randomized lower bound

Let B be any randomized hierarchical clustering algorithm. Let p be the probability that B outputs the optimal 6-clustering. Thus the maximum diameter is 1 with probability p, and at least 2 with probability $1 - p$. (See analysis for the deterministic scenario). We compute the expected competitive ratio of B for $k = 4$ and $k = 6$, and by definition, the expected competitive ratio of B over all values of k is at least the maximum of these two values.

For $k = 4$ the competitive ratio is

$$E(B_4(S))/\mathrm{OPT}_4(S) \geq (4p + 2(1-p))/2 = 1 + p,$$

where the first inequality follows from the fact that when B chooses the optimal 6-clustering, its 4-clustering will have a cluster of diameter ≥ 4.

For $k = 6$ the competitive ratio is

$$E(B_6(S))/\mathrm{OPT}_6(S) \geq (p + 2(1-p))/1 = 2 - p$$

The expected competitive ratio is $\max(1+p, 2-p) \geq 1.5$.

5 Proof of the Online Hierarchical Supplier Theorem

Our online algorithm for k-supplier will use the (online) tree-doubling algorithm as a subroutine. Note that we could equivalently have used the farthest algorithm if we were designing an off-line algorithm.

5.1 The algorithm

We denote a supplier as *active* if it is the closest supplier to one of the current customers. Throughout the algorithm, we will maintain a hierarchical clustering of the active suppliers by inserting them into the (deterministic or randomized) tree-doubling algorithm tree T^+.

When a new customer c arrives, we find the supplier s who is closest to c. If s is not yet in T^+, we mark s as an active supplier and add s to T^+ (using the deterministic or randomized tree-doubling algorithm).

To obtain a hierarchical k-supplier solution, find the largest depth d in T^+ which contains $k' \leq k$ active suppliers $s_1, s_2, \ldots s_{k'}$ and output these suppliers. For each customer c with closest supplier s_0, assign c to s_i for $i \in [1, k']$, if $s_0 = s_i$ or if s_0 is in the subtree below s_i in depth d of T^+.

5.2 The deterministic analysis

Suppose d is the largest depth containing at most k active suppliers. Let s (at depth d) be the supplier that customer c was assigned to and s_0 be the active supplier that c is closest to. Then there is a s_0-to-s path in T^+. Let $s_0, s_1, \ldots s_p$ be the sequence of the suppliers on the s_0-to-s path, where $s_p = s$. By the triangular inequality, the distance from c to s can be bounded as:

$$d(c, s) \leq d(c, s_0) + \sum_{i=0}^{p-1} d(s_i, s_{i+1}) \tag{4}$$

Let Δ be the maximum distance between any two suppliers. By the close-parent property of the tree-doubling algorithm, the distance from s_i to s_{i+1} for $i \in [0, p-1]$ is at most $\Delta/2^{depth(s_i)-1}$. Since the depths of suppliers on the s_0-to-s path are strictly decreasing, and s_{p-1} is on level $d+1$, we have that,

$$\sum_{i=0}^{p-1} d(s_i, s_{i+1}) \leq \frac{\Delta}{2^{depth(s_{p-1})-1}}(1 + 1/2 + 1/4 + \ldots) \leq 2\frac{\Delta}{2^d} \tag{5}$$

Now we derive two lower bounds for OPT_k. First, since s_0 is the closest supplier to c, we have that $\mathrm{OPT}_k \geq d(c, s_0)$. Next, since d is the largest depth in T^+ with at most k active suppliers, depth $d+1$ contains at least $k+1$ active suppliers, $s_1, s_2, \ldots, s_{k+1}$. Using Lemma 4, we have $\mathrm{OPT}_k \geq \delta/4$ where $\delta = \min_{1\leq i<j\leq k+1} d(s_i, s_j)$. By the Far-Cousins property of T^+, δ is at least $\Delta/2^{d+1}$. Applying these bounds we obtain

$$d(c,s) \leq d(c,s_0) + \frac{2\Delta}{2^d} \leq \mathrm{OPT}_k + 4\delta \leq 17\ \mathrm{OPT}_k.$$

Lemma 4. *Let d be the largest depth in T^+ with at most k active suppliers and let $s_1, s_2, \ldots, s_{k+1}$ be active suppliers at depth $d+1$. Let $\delta = \min_{1\leq i<j\leq k+1} d(s_i, s_j)$, and OPT_k be the maximum distance from a customer to a supplier in the optimal k-supplier solution. Then $\delta \leq 4OPT_k$.*

Proof. Since suppliers $s_1, s_2, \ldots, s_{k+1}$ are active, each of them is the closest supplier to some customer c_i. The solution OPT_k uses at most k suppliers, so it will have to assign two of those customers, c_i and c_j, to the some supplier s^*. Thus,

$$\mathrm{OPT}_k \geq \max(d(c_i, s^*), d(c_j, s^*)) \geq (d(c_i, s^*) + d(c_j, s^*))/2$$

Applying the triangle inequality on $d(s_i, s_j)$ we have that:

$$d(s_i, s_j) \quad \leq \quad d(s_i, c_i) + d(c_i, s^*) + d(s^*, c_j) + d(c_j, s_j)$$

Using the fact that s_i is the closest supplier to c_i and s_j is closest for c_j, we obtain

$$\delta \leq d(s_i, s_j) \leq 2(d(c_i, s^*) + d(c_j, s^*)) \leq 4\mathrm{OPT}_k.$$

$\square$

5.3 The randomized analysis

Equation 4 still holds. Instead of Equation 5 we now have:

$$\sum_{i=0}^{p-1} d(s_i, s_{i+1}) \leq \frac{e^r\Delta}{e^{depth(s_{p-1})-1}}(1 + 1/e + 1/e^2 + \ldots) \leq \frac{e}{e-1}\frac{e^r\Delta}{e^d}.$$

Now, by Property 3 the minimum distance δ between $s_1, \ldots, s_{k+1}$ satisfies $e^r\Delta/e^{d+1} < \delta \leq e^r\Delta/e^d$. Write $\delta = e^\epsilon e^r \Delta/e^{d+1}$, where ϵ is distributed uniformly in $[0,1)$. In expectation we have

$$E(e^r\Delta/e^{d+1}) = \delta \int_0^1 e^{-\epsilon} d\epsilon = \delta \frac{e-1}{e}.$$

Lemma 4 still holds, so we finally get:

$$E(d(c,s)) \leq d(c,s_0) + \frac{e}{e-1} E(\frac{e^r\Delta}{e^d}) \leq \mathrm{OPT}_k + \frac{e}{e-1} e\delta \frac{e-1}{e} \leq (1+4e)\ \mathrm{OPT}_k$$

References

[1] P. Arabie, L. J. Hubert and G. De Soete, editors. *Clustering and Classification.* World Scientific, River Edge, NJ, 1998.

[2] M. Charikar, C. Chekuri, T. Feder and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 626–635, 1997.

[3] Marek Chrobak, Claire Kenyon, John Noga and Neal E. Young. Online Medians via Online Bribery. *Lecture Notes in Computer Science* 3887:311-322 (2006); Latin American Theoretical Informatics, 2006.

[4] S. Dasgupta, P. Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences,* 70(4):555-569, 2005.

[5] R. O. Duda, P. E. Hart, and D. G. Sork. *Pattern Classification.* Wiley and Sons, 2001.

[6] M. E. Dyer and A. M. Frieze. A simple heuristic for the p-center problem. *Operations Research Letters.*, 3:285–288, 1985.

[7] T. F. Gonzalez. Clustering to Minimize the Maximum Intercluster Distance. In *Proceedings of the 17th Annual ACM Symposium on the Theory of Computing*, 38:293-306, 1985.

[8] D.S Hochbaum. Various Notions of Approximations: Good, Better, Best and More. In D.S Hochbaum, editor, *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company. 1996.

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

[10] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[11] L. Kaufman and Peter J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, NY, 1990.

[12] Guolong Lin, Chandrashekhar Nagarajan, Rajmohan Rajamaran, and David P. Williamson. A general approach for incremental approximation and hierarchical clustering. In *Proceedings of the seventeenth annual ACM-SIAM Symposium on Discrete algorithm (SODA)*, pages 1147-1156, 2006.

[13] NSF Workshop Report on *Emerging Issues in Aerosol Particle Science and Technology* (NAST), UCLA, 2003, chapter 1, section 18, "Improved and rapid data analysis tools (Chemical Characterization)". Available at `http://www.nano.gov/html/res/NSFAerosolParteport.pdf`.