On hierarchical diameter-clustering, and the supplier problems

Aparna Das Claire Kenyon

Abstract

We study the problem of hierarchical clustering to minimize the diameter of clusters of a given point set, and the hierarchical version of the 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 one (online) due to Charikar, Chekuri, Feder and Motwani, are essentially the same algorithm when points are considered in the same order. We also prove that the analysis of those algorithms is tight.

We exhibit a new lower bound for hierarchical clustering.

We then turn to the k-supplier problem. We study the online hierarchical version of the problem, and design the first constant-factor approximation algorithm for that problem.

1 Introduction

Clustering is the partition of data points into disjoint clusters according to similarity [1, 10]. However, there are many application domains, where clusters have subclusters, these have subsubclusters, and so on. A *hierarchical clustering* of n data points is a recursive partitioning (or clustering) of the points into $1, 2, 3, 4, \dots, n$ clusters such that the $(i+1)$st clustering is obtained by dividing one of the clusters of the ith clustering into two parts, thus making the clustering gradually more fine-grained ([5], section 10.9). This viewpoint has long been popular among statisticians, biologists (particularly taxonomists) and social scientists [11].

One common measure of the quality of a clustering is the maximum cluster diameter ("complete-linkage" clustering). We focus on the hierarchical diameter-clustering problem: finding a hierarchical clustering where the value of a clustering is the maximum cluster diameter.

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 clustering of S into k clusters (i.e. k-clustering) and $A_k(S)$ is the value of the k-clustering constructed by algorithm A. (If A is randomized, then $A_k(S)$ should be replaced by $E(A_k(S))$).

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], whereas the algorithm proposed in [2] may be partially inspired by a k-clustering algorithm by Hochbaum and Shmoys [9], and 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-approximation deterministic algorithm and a $2e$-approximation 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 *doubling algorithm* and to the algorithm from [4] as the *farthest algorithm*. Here are the main results from [4, 2]. Note that the algorithms define one center for each cluster and also achieve the same approximation bounds for a closely related objective function, namely, minimizing the maximum cluster radius.

Theorem 1. *[4] For the hierarchical diameter-clustering problem, there exist an 8-approximation deterministic algorithm and a $2e$-approximation randomized algorithm.*

[2] For the online hierarchical diameter-clustering problem, there exist an 8-approximation deterministic algorithm and a $2e$-approximation randomized algorithm.

In this paper, 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 doubling algorithm considers the points in arbitrary order. Assuming some conditions which remove the non-deterministism, we prove that the clustering produced by the farthest algorithm is always a refinement of the clustering produced by the doubling algorithm, both in the the deterministic and in the randomized cases. (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 are at distance equal to the diameter. Also assume that the 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 doubling (deterministic or randomized) algorithm.

With this interpretation, we see that the farthest algorithm is always at least as good as the doubling algorithm, and so the competitive ratio of the farthest algorithm can be seen as a corollary of the competitive ratio of the doubling algorithm. Could it be that the farthest algorithm is actually better? Our second contribution is to answer this question in the negative by proving that the analysis of [4] is tight.

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

This means that the 8-approximation upper bound for the farthest algorithm is tight, and, by the refinement theorem, that the 8-approximation upper bound for the doubling algorithm is also tight. Proving tightness of the analysis of the randomized variants is an open problem.

Can the approximation ratio be improved? We turn to the problem of the best approximation ratio achievable by any algorithm for hierarchical clustering, with no computational restrictions. In other words, the algorithm is allowed to have non-polynomial running time. In our third contribution, we prove that no deterministic algorithm can achieve better than a 2-approximation, and no randomized algorithm can achieve better than a $(3/2)$-approximation. (Note that the lower bounds proved in [2] apply to the online model only and thus are incomparable to our lower bounds.) It is an open problem whether these bounds can be improved.

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

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

In the (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 *hierarchical* supplier problem, we require that $S_{i-1} \subseteq S_i$ and that the $(i-1)$st assignment is obtainable by reassigning, for each $s \in S_i \setminus S_{i-1}$, all the customers previously assigned to s to the same supplier in S_{i-1}.

In the still more difficult *online* hierarchical supplier problem, the set of possible suppliers is known in advance but new customers arrive as time goes on, so C is a *sequence* of customers. This 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 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 distances d. Let Δ denote the diameter of the set.

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

Defining levels. Let level$(1) = 0$, and for $i \in [2, n]$, let level$(i) = \lfloor \log_2(\Delta / d_i) \rfloor + 1$.

Defining the π'-tree. Define a tree, called the π'-tree, as follows. Point 1 is the root of the tree. For each $i \in [2, n]$, the parent $\pi'(i)$ of i is the point closest to i among the points whose level is strictly less than level(i). Connect each point $i \in [2, n]$ to its parent $\pi'(i)$ with an edge.

Output. For each k, the k-clustering is obtained from the π'-tree by deleting edges from i to the parent of i, for $i \in [2, k]$, and returning the resulting connected components.

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.

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

In addition [4] proves that:

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

[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: $\text{level}(1) = 0$ and $\text{level}(i) = \lfloor \ln(\Delta/d_i) + r \rfloor + 1$. The monotonicity properties are unchanged; the two inequalities are replaced by the following.

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

2.2 Review of the doubling algorithm from [2]

Here the input consists of a *sequence* of n points $\{x_1, \dots x_n\}$ with associated distance metric d. 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. Remove all tree nodes at depth less than d, and return the resulting connected components.

The following property is 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

Let S be the set of points. We assume that the starting point in the farthest algorithm is a point x_i for which there exists another point x_j such that $d(x_i, x_j) = \Delta$, and that points arrive to the doubling algorithm in the order given by the labels defined in the first step of the farthest algorithm. In order to relate the algorithms, we assume that they break ties in the same way. Specifically in the occurrence of a tie when choosing parents, the point with the larger label is favored.

First, given the π' tree constructed by the farthest algorithm, we define a certain tree T, which we will prove satisfies property 1. Then we will prove that it also satisfies the insertion rule, hence is exactly the T^+ tree constructed by the 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 π' construct T^+

(1) Let $\ell =$ the maximum level in π'

(2) **For** each level $i = 0, \ldots \ell$

(3) Let L_i denote the points on level i

(4) Let $S = L_0 \cup L_1 \cup \ldots \cup L_i$

(5) Insert all points in S into depth i of T^+

(6) **For** each point p at depth i of T^+

(7) If $level(p) = i$ create edge between p and the copy of $\pi'(p)$ at depth $i-1$

(8) Else create edge between p and the copy of p at depth $i-1$

Figure (1) shows the π' tree and the corresponding T^+ constructed by algorithm (2.3).

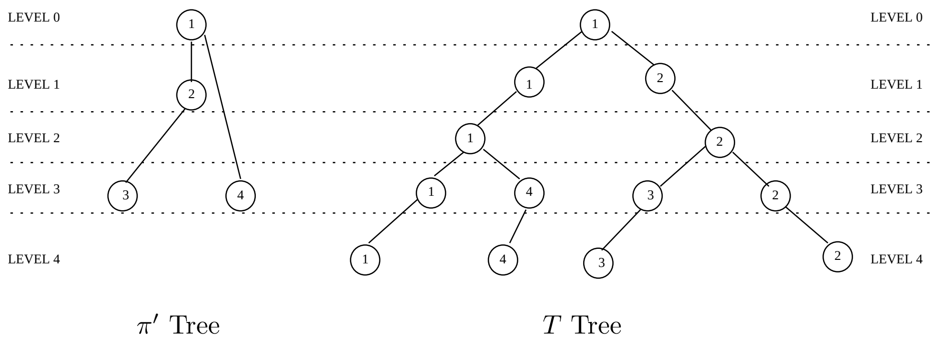


Figure 1: π'-Tree to C^+ Tree

Let T be the tree constructed by Algorithm (2.3).

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 (6) of algorithm (2.3), the parent of p in T is the copy of p at depth $d-1$. Thus Property 1 follows trivially since $d(p,p) = 0$.

If on the other hand $d = \text{level}(p)$, then by step (8) of algorithm (2.3) $\text{parent}(p) = \pi'(p)$. From equation (2), $d(p, \pi'(p)) \leq \Delta/2^{\text{level}(p)-1}$. Substituting $\text{level}(p) = d$ and $\pi'(p) = \text{parent}(p)$, we get: $d(p, \text{parent}(p)) \leq \Delta/2^{d-1}$. $\square$

Lemma 2. *T satisfies the insertion rule.*

Proof. By step (6) of algorithm (2.3) 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 $\text{level}(\pi'(p)) < \text{level}(p)$, $\pi'(p)$ must be at depth $d-1$. By 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 in T according to the insertion rule.

To show that insertion rule is satisfied we need to show that no other point at a depth higher than $d-1$ is close enough to p to be its parent. Let q' be any point at depth $j > d-1$. Note that $q' \in \{1, \ldots, p-1\}$ since these are the only points in T when p arrives. 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 $\text{level}(p) = d$ and equation (1) 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. $\square$

Thus T is the tree constructed by the doubling algorithm when the assumptions of Theorem 2 hold.

Given k, the farthest algorithm always returns a clustering $F(k)$ with exactly k clusters, while the doubling algorithm returns a clustering $D(k)$ with $\leq k$ clusters. We first show the two clusterings are the same when they both have exactly k clusters. The refinement property then follows easily.

Lemma 3. *Let k be such that T has a level L with exactly k vertices, then the clusterings $F(k)$ and $B(k)$ 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 doubling algorithm, where the cluster are defined by the k points at level L in T. Since level L 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 point i is the center of D_i, T contains a i-to-x path $P = (i = p_1, p_2, \ldots p_l = x)$. 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 $x \in F_i$. $\square$

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

Proof.

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

Similarly, on input k_2, the 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), $F(k_2) = B(k_2)$. By definition of a hierarchical clustering, $F(k_1) \preceq F(k_2)$ as $k_1 < k_2$. Thus we have $F(k_1) \preceq F(k_2) = B(k_2)$. □

2.4 Proof of the Refinement Theorem, randomized version

Suppose the random parameter r in the randomized versions of the farthest and the 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 (2.3).

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 which, for some values of k, is worse than the optimal k-clustering by a factor of at least $8 - 4\epsilon$.

Let $n = -2\log(\epsilon)$. The input set S will contain $2n + 9$ points. We denote the nine standard points as $p_1, p_1', p_2, p_3, p_3', p_4, p_4, p_5, p_5'$ and the $2n$ additional points as $q_1, q_1', q_2, q_2', \ldots q_n, q_n'$. 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'.

We take $k = 5$ and compare the value of the 5-clustering produced by the farthest algorithm's 5-clustering with the value of the optimal 5-clustering. It is easy to verify that the optimal 5-clustering of S consists of the following clusters:

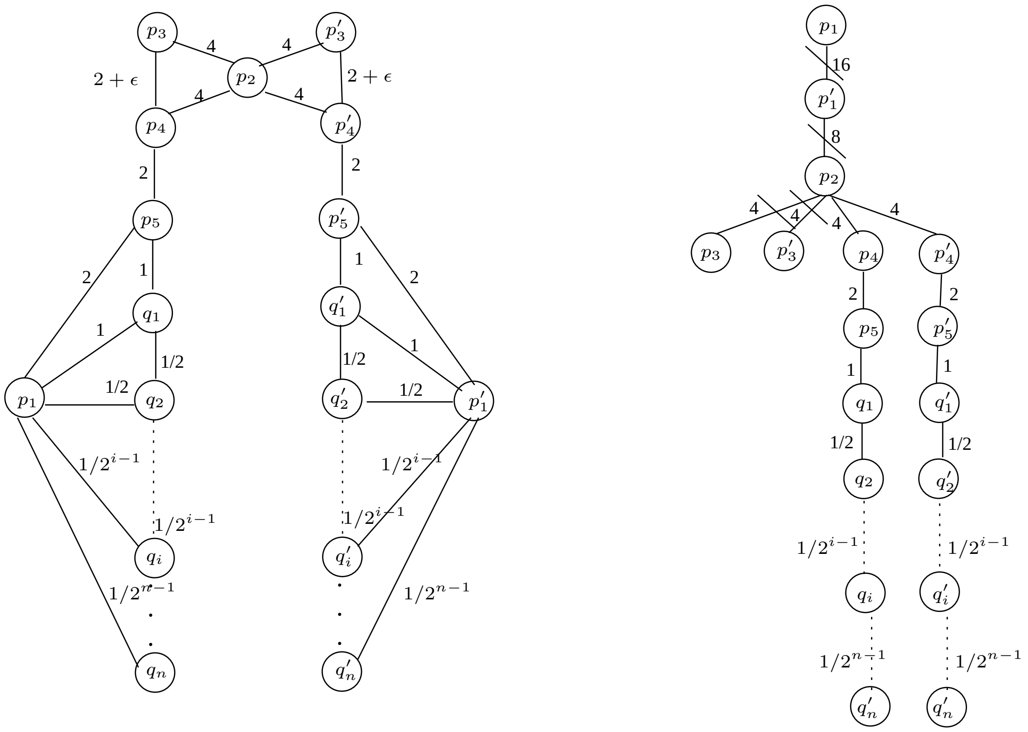
$$\Big\{ \big(p_1, p_5, q_1, q_2 \ldots q_n\big), \big(p_1', p_5', q_1', q_2', \ldots, q_n'\big), \big(p_2\big), \big(p_3, p_4\big), \big(p_3', p_4'\big) \Big\}$$

where clusters $\big(p_3, p_4\big)$ and $\big(p_3', p_4'\big)$ 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 $16 - (2/2^{n-1})$. During the execution, ties between point a p and its counterpart p' are broken arbitrarily. To do the labeling, the algorithm starts with point p_1 and obtains the following ordering:

$$p_1, p_1', p_2, p_3, p_3', p_4, p_4', p_5, p_5', q_1, q_1', \ldots q_n, q_n'.$$

The algorithm uses the diameter $\Delta = 16$ to define the levels of the points. For $i \in [1, n]$ we get $level(q_i) = level(q_i') = i - 4$. To define the π'-tree, the algorithm creates edges between each point $p \neq 1$ and the closest point to p at a strictly lower level. The resulting π'-tree is shown in figure



(a) Points in S

(b) π'-Tree

Figure 2: Graph for Tight Example

(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:

$$\left\{ (p_1), (p'_1), (p_3)\,(p'_3), (p_4, p_5, q_1 \dots q_n,\ 2,\ p'_4, p'_5, q_1 \dots q'_n,) \right\}$$

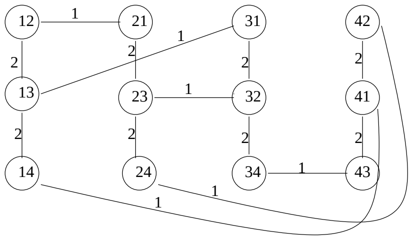
Note that the last cluster has diameter $16 - 2/2^{n-1}$, which is the distance from q_n to q'_n. This proves the Tightness theorem.

4 Proof of the Hierarchical Lower Bound Theorem

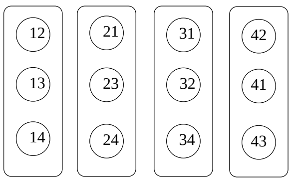
Let S denote the input points p_{ij} for $i, j \in [1, 4]$ and $i \neq j$ shown in figure (4). The distances between the points are: $d(p_{ij}, p_{ji}) = 1$ and $d(p_{ij_1}, p_{ij_2}) = 2$. (This resembles but is not the same as the example in [2], where the authors focus on the online setting). It is 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

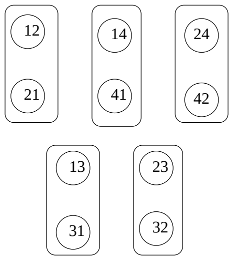
Let A be any deterministic hierarchical clustering algorithm.



Graph of Points



Optimal 4−Clustering



Optimal 6−Clustering

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) 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 supplier closest 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) doubling algorithm's tree T^+.

When a new customer c arrives, we find the supplier s who is closest to c. If s is already active, assign c to s. Otherwise if s is not active, first we add s to T^+ (using the deterministic or randomized doubling algorithm) and then assign c to s.

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 a customer c with closest supplier s, assign c to s_i for $i \in [1, k']$, if either $s = s_i$ or s is in the subtree below s_i in depth d of T^+.

5.2 The deterministic analysis

Suppose that depth d was the largest depth containing at most k active suppliers. Let s (at depth d) be the active supplier that customer c was assigned to, and $s_0, s_1, \ldots s_p$ be sequence of the suppliers on the c-to-s path. Then s_0 is the supplier closest to c and $s_p = s$. 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 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 c-to-s path are strictly increasing, 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. In the following lemma we prove that $8\mathrm{OPT}_k \geq \Delta/(2^d)$. 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, $s_1, s_2, \ldots, s_{k+1}$. From Lemma 4, we have $\mathrm{OPT}_k \geq \delta/4$. 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 $s_1, s_2, \ldots, s_{k+1}$ be the first $k+1$ active suppliers, $\delta = \min_{1 \leq i < j \leq k+1} d(s_i, s_j)$, and let OPT_k be the maximum distance from a customer to a supplier in the optimal k-supplier solution. Then $\delta \leq 4OPT_k$.*

Proof. Let d be the largest depth in T^+ with at most k active suppliers, depth $d+1$ contains at least $k+1$ active suppliers, $s_1, s_2, \ldots, s_{k+1}$, each of which 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

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

□

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 $e^r\Delta/e^{d+1} \leq \delta < e^r\Delta/e^d$. 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. *Proceedings of the 29th Symposium on the Theory of Computing*, 1997.

[3] Marek Chrobak, Claire Kenyon, John Noga and Neal E. Young. Online Medians via Online Bribery. In *LATIN*, 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. *Oper. Res. Lett.*, 3:285–288, 1985.

[7] T. F. Gonzalez. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science*, Vol. 38, No. 2-3, June 1985, pp. 293–306.

[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 *Symposium on Discrete Algorithms (SODA).* ACM-SIAM, 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`.