

# Introduction

[This chapter is under construction!]

Clustering is a fundamental unsupervised learning tool. It can be used to search data for hidden structure, to summarize data and to classify new unseen data according to prior findings. Since applications are manifold, clustering also offers a rich variety of algorithmic problems. In this lecture, we will study clustering problems through the lens of algorithm theory. Thus, we are equipped with a mathematical formulation that allows us to judge the quality of a clustering, and our goal is to find a clustering that has the best possible quality.

The clustering problems studied in this lecture share some properties: Most of them are *partitional* clustering problems, i.e., we get some input data and want to partition it into disjoint subsets of associated elements. Furthermore, most problems that we study are *center-based*, which means that we do not only want the partitioning, but we want a set of *centers* and an assignment of points to centers which then implicitly defines the partitioning. All clustering problems are *geometric* clustering problems, which means that we have geometric information about the input data (which we then call input *points*), i.e., we know the distances between any two points.

For our study of geometric clustering problems, we need to recall the definition of *metric spaces*. The most known metric space is the Euclidean space  $\mathbb{R}^d$ , where two points  $x, y \in \mathbb{R}^d$  with  $x = (x_1, \dots, x_d)^T$  and  $y = (y_1, \dots, y_d)^T$  have distance  $\|x - y\| = \sqrt{\sum_{i=1}^d (x_i - y_i)^2}$ . It is a special case of the following more general definition.

**Definition 1.** Let  $X$  be any non-empty set and  $d : X \times X \rightarrow \mathbb{R}$  be a function on  $X$ . Then  $d$  is a metric on  $X$  if it holds for all  $x, y, z \in X$  that

1.  $d(x, y) \geq 0$  and  $d(x, y) = 0$  if and only if  $x = y$ ,
2.  $d(x, y) = d(y, x)$ , and
3.  $d(x, y) \leq d(x, z) + d(z, y)$ .

We call a pair  $(X, d)$  consisting of a finite set  $X$  and a metric  $d$  a metric space.

We refer to Condition 3 in Definition 1 as the *triangle inequality*. Usually we call elements of  $X$  *points*. Furthermore, we extend any metric  $d$  to point sets by setting

$$d(x, A) = \inf_{y \in A} d(x, y)$$

for any  $A \subseteq X$  and

$$d(A, B) = \inf_{x \in A, y \in B} d(x, y).$$

for any  $A, B \subseteq X$ .

The Euclidean space  $\mathbb{R}^d$  with the Euclidean distance is one example for a metric space. We will encounter it later in the lecture when we study the  $k$ -means problem. This modeling is common in machine learning and statistics, where  $k$ -means comes up as an important problem. However, clustering is also a classical topic in graph theory. Here, it is more common to study the case of *finite metric spaces*. The input then consists of a *finite* point set  $P$  and a distance function  $d : P \times P \rightarrow \mathbb{R}$  which is metric, i.e.  $(P, d)$  form a metric space. In this scenario,  $d$  simply consists of  $|P|^2$  numbers that give the distance for every pair of points in  $P$ . The can thus be viewed as an undirected complete weighted graph: the vertices are the points in  $P$ , and the weight of edge  $\{x, y\}$  is  $d(x, y) = d(y, x)$  for all  $x, y \in P$ .

In this context, the the metric space  $(P, d)$  can also be implicitly given by a *graph metric*: For any (not necessarily complete) graph  $G = (V, E)$  weighted by a function  $w : V \times V \rightarrow \mathbb{R}^{\geq 0}$ , we define the *shortest path distance*  $d_s(x, y)$  between  $x, y \in V$  to be the length of the shortest path between  $x$  and  $y$  in  $G$ . One can show that  $d_s$  is a metric on  $V$ , and thus  $G$  and  $w$  implicitly define the metric space  $(V, d_s)$ .

All problems studied in this lecture are minimization problems. As it turns out, for most of them, it is not known how to compute optimal solutions in polynomial time, and it may well be impossible to do so, since the problems are NP-hard. For this reason, we will resort to approximation algorithms that find a reasonably good solution instead of an optimal one.

Recall that an  $\alpha$ -approximation algorithm for a minimization problem is an algorithm that outputs solutions with a value that is at most  $\alpha$  times the value of an optimal solution. Let  $\mathfrak{I}$  be the set of all possible input instances to an optimization problem, let  $ALG$  be an algorithm that computes a feasible solution  $S^{ALG}(I)$  for any  $I \in \mathfrak{I}$ . For any  $I \in \mathfrak{I}$ , let  $S^*(I)$  be an optimal solution for  $I$ . For a minimization problem, suppose that  $c(S)$  is the cost of a solution  $S$ . Then  $ALG$  is an  $\alpha$ -approximation algorithm if

$$\sup_{I \in \mathfrak{I}} \frac{c(S^{ALG}(I))}{c(S^*(I))} \leq \alpha.$$

Whenever we talk about  $\alpha$ -approximation algorithms, we implicitly mean algorithms that have a polynomial worst-case running time. All problems considered in this lecture can be solved optimally given exponential running time.

Our studies will also include *randomized* approximation algorithms. For randomized algorithms,  $c(S^{ALG}(I))$  or  $v(S^{ALG}(I))$ , respectively, is a random variable. We might either want to achieve that it is close to the optimum value with high probability, or

that its expected value is close to the optimum value. We choose the second alternative. Thus, we say that a randomized algorithm  $ALG$  is an  $\alpha$ -approximation algorithm in expectation for a minimization problem if

$$\sup_{I \in \mathcal{I}} \frac{\mathbf{E}[c(S^{ALG}(I))]}{c(S^*(I))} \leq \alpha.$$

As for deterministic algorithms, we also require randomized approximation algorithms to have a polynomial worst-case running time.

## 1.1 Mathematical facts and notations

This section collects mathematical facts that we need throughout the lecture.

**Fact 2.** For all  $r \in \mathbb{R}$  with  $r \neq 1$  and all  $n \geq 0$ ,

$$\sum_{i=0}^n r^i = \frac{1 - r^{n+1}}{1 - r}.$$

If  $|r| < 1$ , then

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1 - r}.$$

For a number  $i \in \mathbb{N}^{\geq 1}$ , we sometimes use the abbreviation  $[i] = \{1, \dots, i\}$  to denote the set of all numbers from 1 to  $i$ .

## The happy world of $k$ -center

We start with the study of the  $k$ -center problem. It is a classical problem from graph theory, and it is one of the rare problems where we precisely know the best possible approximation ratio under the assumption that  $P \neq NP$ . This approximation ratio can be achieved by two elegant approximation algorithms.

### 2.1 The metric $k$ -center problem

The input to the  $k$ -center problem is a finite metric space  $(P, d)$  and a number  $k \in \mathbb{N}^{\geq 1}$ . The  $k$ -center problem is a center-based clustering problem, meaning that we are searching for a set of centers  $C$  that give a good clustering when every point is assigned to the center closest to it. Center-based clustering problems differ in the evaluation of the quality of a center set. The  $k$ -center problem is a so-called min-max problem: We are minimizing the maximum distance of any point to its closest center.

**Definition 3.** Let  $(P, d)$  be a metric space and  $k \in \mathbb{N}^{\geq 1}$  be a number. The  $k$ -center

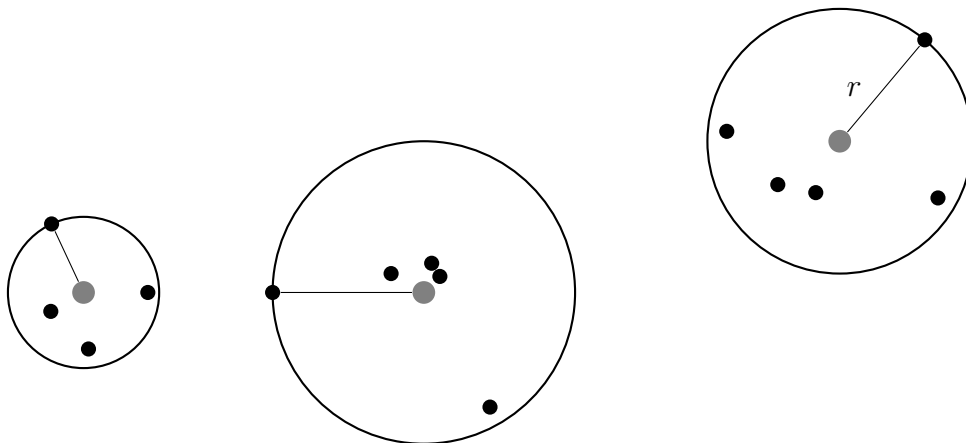


Figure 2.1: An example for a  $k$ -center instance and solution with  $k = 3$ .

problem is to compute a set  $C \subseteq P$  with  $|C| = k$  that minimizes

$$r(C) := \max_{x \in P} \min_{c \in C} d(x, c).$$

We call the value  $\max_{x \in P} \min_{c \in C} d(x, c)$  either  $k$ -center objective value or maximum radius of the solution  $C$ . Figure 2.1 shows a point set, three chosen centers and the radii of the resulting three clusters. The maximum radius of the depicted solution is labeled by  $r$ .

## 2.2 A simple and elegant approximation

Gonzalez' algorithm [Gon85] gives a greedy 2-approximation for the  $k$ -center problem. The algorithm starts by picking an arbitrary point  $c_1 \in P$  as the first center. Then it does  $k - 1$  iterations of the following: Choose the point  $c_i$  which is farthest away from the previously chosen centers  $\{c_1, \dots, c_{i-1}\}$  to be the center. This is repeated until  $k$  centers are chosen.

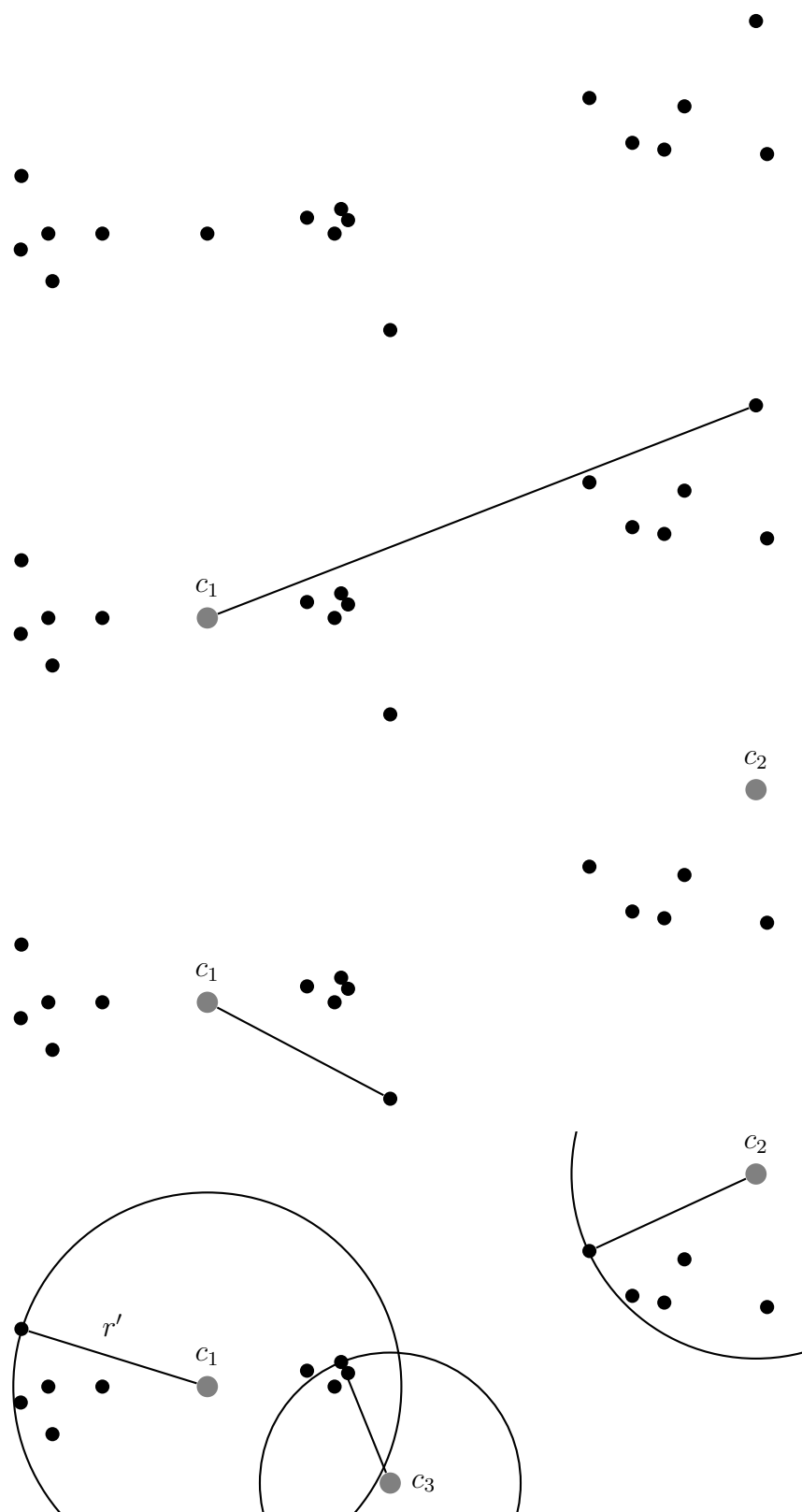
**farthest-first-traversal**(a finite metric space  $(P, d)$ ,  $k \in \mathbb{N}^{\geq 1}$ )

1. Choose a center  $c_1$  arbitrarily from  $P$  and set  $C^1 = \{c_1\}$
2. **For**  $i = 2$  **to**  $k$  **do**
3.     Choose  $c_i$  as (one of) the points maximizing  $d(x, C^{i-1})$ ,  
i.e.,  $c_i = \arg \max\{d(x, C^{i-1}) \mid x \in P\}$
4.     Set  $C^i = C^{i-1} \cup \{c_i\}$
5. **Return**  $C = \{c_1, \dots, c_k\}$

Figure 2.2 shows an example run of Algorithm **farthest-first-traversal** with  $k = 3$ . The first center is chosen arbitrarily. Center 2 and 3 are then chosen as points with maximum distance to the previously chosen centers. This means that the algorithm has the tendency to choose points that are at the border of a cluster. The resulting solution is worse than the solution depicted in Figure 2.1, but its maximum radius  $r'$  only slightly larger than  $r$  from the previous solution. The following theorem shows that in general, the maximum radius is at most twice the maximum radius of an optimal solution.

**Theorem 4.** *Algorithm **farthest-first-traversal**  $((P, d), k)$  computes a 2-approximation for the  $k$ -center problem.*

*Proof.* Let  $C$  be the solution computed by **farthest-first-traversal** $((P, d), k)$ . This solution has radius  $r(C) = \max_{x \in P} d(x, C)$ . Then there is a point  $y \in P$  that actually has distance  $r(C)$  to its closest center in  $C$ , i.e.,  $d(y, C) = \min_{c \in C} d(y, c) = r(C)$ . Now we observe that the distance between  $y$  and its closest center can only get shorter during the execution of the algorithm, i.e.,  $d(y, C^i) \geq d(y, C^k)$  for all  $i \in \{1, \dots, k\}$ . Thus,  $d(y, C^i) \geq r(C)$  is true for all  $i \in \{1, \dots, k\}$ . Since Gonzalez'

Figure 2.2: An example run of farthest first traversal with  $k = 3$ .

algorithm always chooses the point that has maximum distance to the previously chosen centers,  $d(y, C^i) \geq r(C)$  implies  $d(c_{i+1}, C^i) \geq r(C)$  for all  $i \in \{1, \dots, k-1\}$ . This means that the centers and  $y$  form a set of  $k+1$  points with pairwise distance  $r(C)$ . Any solution has to have a cluster with two of these points, and this cluster then has at least radius  $r(C)/2$ . Since the optimal solution has a maximum radius of at least  $r(C)/2$ , and the computed solution has a radius of  $r(C)$ , we conclude that the computed solution is a 2-approximation.  $\square$

**Lemma 5.** *Algorithm `farthest-first-traversal`( $P, d$ ) can be implemented to run in time  $\mathcal{O}(|P| \cdot k)$ .*

*Proof.* Line 1 takes constant time. The loop in lines 2-4 is executed  $k-1$  times. For line 3, the algorithm needs to find the point whose closest center is farthest away. To keep track of this, we use an array  $c_{\min}$  which stores the closest center (so far) for each point, i.e. we preserve the equality  $c_{\min}(x) = \arg \min\{c \in C^{i-1} \mid d(x, c)\}$  for all  $x \in P$ . The array is initialized in time  $\mathcal{O}(|P|)$  before starting the loop. In the loop, when we choose a new center  $c_i$ , we compare  $d(x, c_i)$  with  $d(x, c_{\min}(x))$  and update  $c_{\min}(x)$  accordingly. This takes constant time for every point, i.e., updating  $c_{\min}$  takes time  $\mathcal{O}(|P|)$ . We can find the point with the maximum  $c_{\min}$  value simultaneously or find it in another  $\mathcal{O}(|P|)$  operations after the updating is finished. Thus, lines 2-4 take time  $\mathcal{O}(|P|k)$ .  $\square$

## Was part of lecture 1, but is not used until later

The following definitions will later be part of the respective chapters.

**Definition 6.** *Let  $(P, d)$  be a metric space and  $k \in \mathbb{N}^{\geq 1}$  be a number. The  $k$ -median problem is to compute a set  $C \subseteq P$  with  $|C| = k$  that minimizes*

$$\sum_{x \in P} \min_{c \in C} d(x, c).$$

**Definition 7.** *Let  $(P, d)$  be a metric space and  $f : P \rightarrow \mathbb{N}^{\geq 1}$  be a function of opening costs. The facility location problem is to compute a set  $C \subseteq P$  that minimizes*

$$\sum_{x \in P} \min_{c \in C} d(x, c) + \sum_{c \in C} f(c).$$

**Definition 8.** *Let  $P \subset \mathbb{R}^d$  be a finite set of points from the Euclidean space and  $k \in \mathbb{N}^{\geq 1}$  be a number. The  $k$ -means problem is to compute a set  $C \subseteq P$  with  $|C| = k$  that minimizes*

$$\sum_{x \in P} \min_{c \in C} \|x - c\|^2.$$

– end of lecture 1 –

## 2.3 A matching lower bound

The  $k$ -center problem is NP-hard. This has been observed simultaneously by Hochbaum [Hoc84] and Hsu and Nemhauser [HN79], who also both discovered the inapproximability result that we will see below. Both their proofs rely on a reduction from the *dominating set problem*.

**Definition 9.** Let  $G = (V, E)$  be an undirected unweighted graph. A set  $D \subset V$  is a dominating set for  $G$  if every vertex in  $V$  has a neighbor in  $D$ , i.e., for all  $u \in V$ , either  $u \in D$  or there exists a  $v \in D$  such that  $\{u, v\} \in E$ .

The dominating set problem is to decide for a given graph  $G$  and a number  $k \in \mathbb{N}^{\geq 1}$  whether a dominating set  $D$  for  $G$  exists that has cardinality at most  $k$ , i.e.,  $|D| \leq k$ .

The dominating set problem is NP-hard; it is one of the problems shown to be NP-hard in the book by Garey and Johnson [GJ79]. It is closely related to the  $k$ -center problem.

**Lemma 10.** The  $k$ -center problem is NP-hard. Furthermore, it is NP-hard to compute an  $\alpha$ -approximation for the  $k$ -center problem for any  $\alpha < 2$ .

*Proof.* Let  $G = (V, E)$ ,  $k$  be an input to the dominating set problem. We define an input for the  $k$ -center problem by setting  $P = V$  and setting the distance  $d(x, y)$  between  $x \in P$  and  $y \in P$  to 1 if  $\{x, y\} \in E$ , and to 2 otherwise. This is a metric because no function  $d : P \times P \rightarrow \{1, 2\}$  can violate the triangle inequality. Assume that we are given an algorithm that decides the decision variant of the  $k$ -center problem, i.e., for a given weighted graph, a number of centers and a value  $t$ , it decides whether a solution to the  $k$ -center problem of radius at most  $t$  exists. We apply this algorithm to the input  $P$ ,  $d$ ,  $k$  and  $t = 1$ . We output YES if the algorithm finds a clustering of radius at most 1 and NO otherwise.

Now we argue that this reduction is correct. First, assume that a dominating set  $D$  of size at most  $k$  exists in  $D$ . Then every vertex in  $V$  is either in  $D$  or it has a neighbor in  $D$ . Consider  $D$  as a center set in  $G'$ . Then every point in  $P$  either has distance 0 to its closest center (if it is in  $D$  itself), or it has distance 1 (if it has a neighbor in  $D$ ). Thus, the maximum radius of the solution is 1. Second, assume that the algorithm found a center set  $C$  with maximum radius 1 in  $G'$ . Then  $C$  is a dominating set in  $G$ : Every vertex has distance  $\leq 1$  to at least one vertex in  $C$ , so it is either in  $C$  or a neighbor of a point in  $C$ .

This shows that the  $k$ -center problem is NP-hard. But that is not all: We observe that the reduction nearly immediately also gives an inapproximability result. That is because the only possible distances between two distinct points in  $G'$  are 1 or 2. Thus, any approximation algorithm with an approximation ratio  $\alpha < 2$  has no other option than providing an optimal solution. More precisely we can formulate the above reduction in the following way: Assume that we are given an algorithm that computes an  $\alpha$ -approximation for the  $k$ -center problem for  $\alpha < 2$ . Apply this algorithm to  $P$ ,  $d$ ,  $k$ . If the maximum radius of the solution is 1, then output YES, otherwise, output NO. By the same arguments as above, this reduction is correct and shows that it is NP-hard to compute an  $\alpha$ -approximation for the  $k$ -center problem for  $\alpha < 2$ .  $\square$



## 2.4 Incremental and hierarchical clustering

In this section, we consider *incremental* and *hierarchical* clustering. The basic idea behind these notions is that instead of one clustering for one fixed number of centers, we want multiple clusterings, indeed, even *one clustering for each possible value of  $k$* . The two notions are related, but differ in the details of the desired clusterings.

Incremental clusterings are motivated by the scenario where we want to place centers, but we do not know how many we will build, or we cannot build them all at the same time. An incremental clustering thus gives an ordering of the input points, and the first  $k$  points in this order form the center set for the  $k$ -center clustering with  $k$  centers.

In the following, we will compare incremental and hierarchical clusterings with optimal clusterings for a fixed number of centers. For this purpose we refer to an optimal solution for the  $k$ -center problem (with  $k$  centers) by  $C_k^{\text{opt}}$ .

**Definition 11.** *Let  $(P, d)$  be a metric space. An incremental clustering is an ordering  $x_1, \dots, x_{|P|}$  of all points in  $P$ .*

*For any  $k \in [|P|]$ , the solution to the  $k$ -center problem on  $P$  induced by such an incremental clustering is the center set  $\{x_1, \dots, x_k\}$ .*

*We say that an incremental clustering is  $\alpha$ -competitive if for all  $k \in [|P|]$ ,*

$$r(\{x_1, \dots, x_k\}) \leq \alpha \cdot r(C_k^{\text{opt}}),$$

*where  $C_k^{\text{opt}}$  is an arbitrary optimum solution with  $k$  centers.*

Our notion of competitiveness in Definition 11 compares the individual clusterings with optimal clusterings with the same number of clusters. Notice that we do not actually know the values  $r(C_k^{\text{opt}})$ , and can in general not compute them (assuming  $P \neq NP$ ), so we can not compute the competitiveness of a given incremental solution.

In general, a 1-competitive incremental clustering does not even exist since the optimal clusterings may contradict each other in their choice of centers.

However, we observe that Gonzalez' algorithm can be used to compute an incremental clustering if we simply continue using it until all points have become a center. This means that the approximation algorithm which is optimal under the condition  $P \neq NP$  can directly be used to compute incremental clusterings of the same quality.

**full-farthest-first-traversal**(a finite metric space  $(P, d)$ )

1. Choose a center  $c_1$  arbitrarily from  $P$ , set  $C^1 = \{c_1\}$
2. **For**  $i = 2$  **to**  $|P|$  **do**
3.     Set  $R_i := \max_{x \in P} d(x, C^{i-1})$
4.     Choose a  $c_i$  as a point with  $d(c_i, C^{i-1}) = R_i$ , set  $C^i = C^{i-1} \cup \{c_i\}$
5. **Return**  $c_1, \dots, c_{|P|}$  and  $R_2, \dots, R_{|P|}$

The only real change compared to `farthest-first-traversal` is that the algorithm `full-farthest-first-traversal` does not have  $k$  as a parameter, runs the main loop until  $|P|$  centers are chosen and returns the numbering instead of one fixed center set. The other changes are for our convenience: We name the distance of center  $c_i$  to the previously chosen centers  $R_i$ . This will prove helpful later.

**Corollary 12.** *The algorithm `farthest-first-traversal`  $((P, d))$  computes a 2-competitive incremental  $k$ -center clustering. In particular,  $R_k \leq 2 \cdot r(C_{k-1}^{\text{opt}})$  holds for all  $k \in \{2, \dots, |P|\}$  and optimal solutions  $C_k^{\text{opt}}$ .*

*Proof.* This follows directly from Theorem 4. □

Later in this lecture, we see that it is a lucky coincidence that a best-possible approximation algorithm for the  $k$ -center problem already computes an incremental solution. For other objectives, this is not the case, and obtaining incremental clusterings is much more involved.

Hierarchical clusterings are related to incremental clusterings, yet they are motivated differently. Here, the *clusterings* have to be related. More precisely, we want that for all  $k$ , the clustering with  $k$  clusters can be obtained from the clustering with  $k + 1$  clusters by merging two of the clusters. We call this *hierarchical compatibility*. Until now, we have had a fairly center-based view on clusterings. That makes sense in all scenarios where we want to summarize the data by the centers, and thus, the centers are really what we are interested in. However, in some scenarios it is the *partitioning itself* that we are curious about. We want to know how the data points themselves cluster into groups, and the center is just a means to judge the (dis)similarity of the points in the same cluster. This is the case for hierarchical clustering. We want to see the data clustered into groups at different levels of granularity.

**Definition 13.** *Let  $(P, d)$  be a metric space. A hierarchical  $k$ -center clustering consists of  $|P|$  partitionings  $\mathcal{H}_1, \dots, \mathcal{H}_{|P|}$  of  $P$  that satisfy*

- $|\mathcal{H}_k| = k$  for all  $k \in \{1, \dots, |P|\}$ , in particular this means that  $\mathcal{H}_1 = \{P\}$  and  $\mathcal{H}_{|P|} = \{\{x\} \mid x \in P\}$ ,
- and  $\mathcal{H}_k = \mathcal{H}_{k-1} \setminus \{A\} \cup \{A_1, A_2\}$  for all  $k \in \{2, \dots, |P|\}$ , with  $A \in \mathcal{H}_{k-1}$  and two clusters  $A_1, A_2 \in \mathcal{H}_k$  with  $A = A_1 \dot{\cup} A_2$ .

Let  $\mathcal{H}_k$  be one of the clusters in the hierarchy, and assume that we name its clusters  $C_1, \dots, C_k$ . Then we define  $r(\mathcal{H}_k)$  to be the maximum radius of the clusters if we choose centers optimally, i.e.,  $r(\mathcal{H}_k) := \max_{i \in [k]} \min_{c \in C_i} \max_{x \in C_i} d(x, c)$ .

We say that an hierarchical clustering  $\mathcal{H}_1, \dots, \mathcal{H}_{|P|}$  is  $\alpha$ -competitive if

$$r(\mathcal{H}_k) \leq \alpha \cdot r(C_k^{\text{opt}})$$

for all  $k \in [|P|]$  and optimal solutions  $C_k^{\text{opt}}$ .

Definition 13 is fairly complex. Each  $\mathcal{H}_k$  is a *partitioning* of  $P$ , i.e., it contains  $k$  sets that are disjoint and whose union is  $P$ . To compute the cost of such a clustering, we

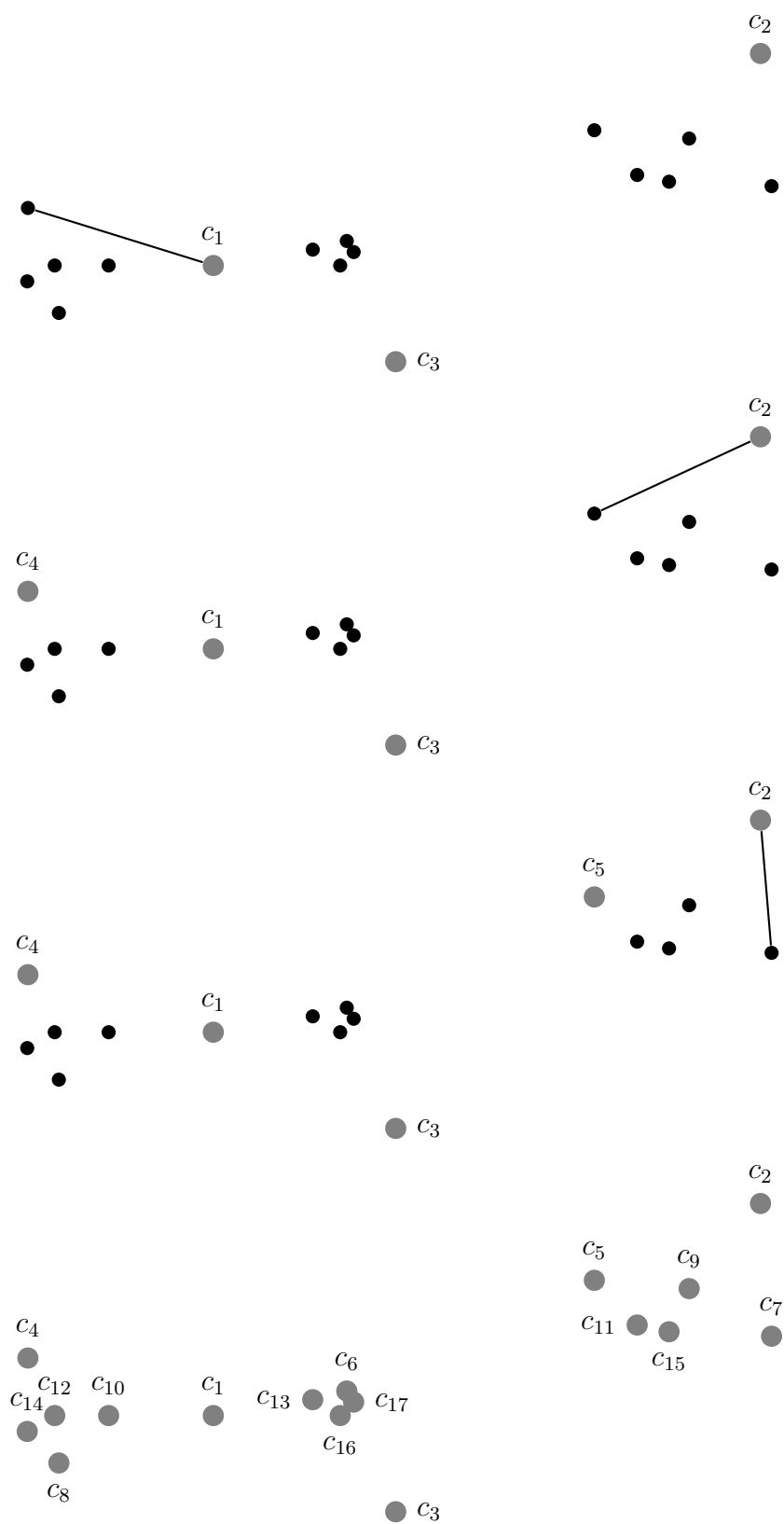


Figure 2.3: Continuation of Figure 2.2 as an example for a full farthest first traversal.

consider optimal centers: For each cluster in a clustering, we pick the point in the cluster which minimizes the 1-center cost of that cluster. These centers may change between the levels of the hierarchy. So our notion of hierarchical clustering is different from incremental clustering in two ways: Firstly, we require the clusters themselves to be hierarchically compatible, which is a stronger requirement than having nested center sets. But secondly, we do not require the centers to stay the same, so here we require less. For each level, we may completely change our choice of centers.

Gonzalez' algorithm will again be useful for our goal to find a good hierarchical clustering. However, this time the farthest first traversal is only the first building block of the algorithm. The incremental clustering computed by Gonzalez' algorithm has the properties that we investigated above, in particular,  $\{x_1, \dots, x_j\}$  is a 2-approximative solution for the  $k$ -center problem with  $k = j$ . However, the clusters induced by  $\{x_1, \dots, x_j\}$  for different values of  $j$  are in general not hierarchically compatible with each other. For example, the cluster that was associated with  $x_j$  in the  $j$ -clustering may split up when assigned optimally to the first  $j - 1$  centers.

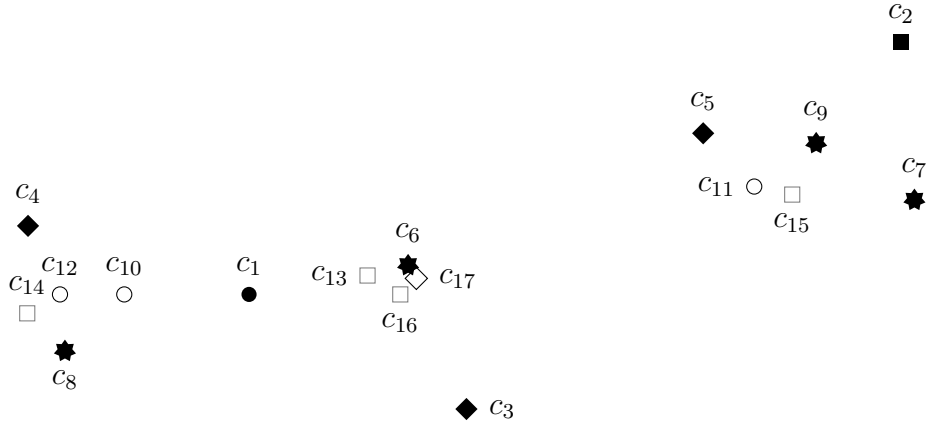
Still, we will make use of the incremental clustering. Say that the full farthest first traversal outputs the order  $x_1, \dots, x_{|P|}$  and the values  $R_2, \dots, R_{|P|}$ . We now find a good hierarchical clustering in three more steps.

**I. Levels of granularity** The full farthest first traversal provides us with  $|P|$  clusterings of decreasing cost: The first one is the 1-center clustering where every point is assigned to  $x_1$ , the second clustering is induced by assigning every point to either  $x_1$  or  $x_2$ , and so on. The costs of these solutions are  $R_2, \dots, R_{|P|}$  for  $k = 1, \dots, |P| - 1$ , and 0 for the last clustering, where every point has its own cluster. We now group the clusterings into levels with the goal that all clusterings assigned to the same level have roughly the same cost. The largest  $R_i$  value is  $R_2$ , and we also call this value  $R$ , i.e.,  $R = R_2$ . The formal definition of the levels is:

$$L_0 = \{x_1\}, \quad L_j = \left\{ x_i \mid R_i \in \left( \frac{R}{2^j}, \frac{R}{2^{j-1}} \right] \right\} \text{ for all } j \geq 1.$$

The first point  $x_1$  is special because it does not have an  $R_i$  value (these start at  $i = 2$ ). Thus,  $x_1$  gets its own special level  $L_0$ . The remaining points are partitioned into the levels according to their  $R_i$  in an exponentially decreasing manner, based on  $R = R_2$ .  $L_1$  contains all points  $x_i$  with  $R/2 < R_i \leq R$ . This in particular includes  $x_2$  since  $R_2 = R$ , and it may or may not include further points.  $L_3$  contains all  $x_i$  with  $R/4 < R_i \leq R/2$ , and in general,  $L_j$  contains all  $x_i$  with  $R/2^j < R_i \leq R/2^{j-1}$ . Figure 2.4 depicts the resulting levels for the points in our ongoing example.

Notice that we do not give an upper bound on  $j$ , i.e., on the maximum level index. Indeed, we do not know any such bound: The cost of the  $(|P| - 1)$ -center clustering may be arbitrarily small compared to the cost of the 1-center clustering. However, at most  $|P|$  of the levels are non-empty because there are only  $|P|$  points that are assigned to levels. Furthermore notice that for all points in the same level, the  $R_i$  value is of a comparable magnitude, more precisely, they are all within a factor of 2 of the left interval boundary.



$$\begin{aligned} \bullet L_0 &= \{c_1\} & \blacksquare L_1 &= \{c_2\} & \blacklozenge L_2 &= \{c_3, c_4, c_5\} & \star L_3 &= \{c_6, c_7, c_8, c_9\} \\ \circ L_4 &= \{c_{10}, c_{11}, c_{12}\} & \square L_5 &= \{c_{13}, c_{14}, c_{15}, c_{16}\} & \diamond L_6 &= \{c_{17}\} \end{aligned}$$

Figure 2.4: Continuation of Figure 2.3, showing the levels of all points.

For use in the next paragraph, we define a function  $L : \{x_1, \dots, x_{|P|}\} \rightarrow \mathbb{N}^{\geq 1}$  which gives the index of the level that a point is in, i.e.,  $L(x_i) = j$  when  $x_i \in L_j$ .

**II. The parent function** We need the levels of granularity in order to define a mapping that we call the *parent function*. It assigns a parent to every point in  $P$ . For every  $i \in \{2, \dots, |P|\}$ , we define

$$\text{parent}(x_i) := \arg \min \{d(x_i, y) \mid y \in \bigcup_{j=0}^{L(x)-1} L_j\}.$$

For  $x_1$ , we set  $\text{parent}(x_1) := x_1$ , so  $x_1$  is mapped to itself. For all  $x_i \neq x_1$ , the parent of  $x_i$  is its closest point among all points in a level with lower index than  $L(x_i)$ . Notice that this means that the parent is a point  $x_\ell$  with  $\ell < i$ , and that among all points  $x_\ell$  with  $\ell < i$  we exclude exactly those that are in the same level as  $x_i$  itself. In our example, the parents are as follows:

$$\begin{array}{c} i \\ \text{parent}(x_i) \end{array} \begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\ \hline - & 1 & 1 & 1 & 2 & 3 & 2 & 4 & 5 & 8 & 5 & 8 & 6 & 12 & 11 & 6 & 6 \end{array}$$

Notice that the parent does not have to be in the level directly before the level of  $x$ , it can be in any level with a lower index. For example, the parent of  $c_{16} \in L_5$  in our example is  $c_6 \in L_3$ . This is important, since it allows us to show the following bound for the distance between a point and its parent.

**Lemma 14.** For all  $x \in P$ ,

$$d(x, \bigcup_{j'=0}^j L_{j'}) \leq \frac{R}{2^j} \quad \text{and} \quad d(x, \text{parent}(x)) \leq \frac{R}{2^{L(x)-1}}.$$

*Proof.* This is a property of the farthest first traversal. Let  $x_\ell$  be the point with highest index in  $L_j$ . Then all points are at distance  $R_{\ell+1}$  from  $\{x_1, \dots, x_\ell\} = \bigcup_{j'=0}^j L_{j'}$ , because  $R_{\ell+1}$  is the distance that  $x_{\ell+1}$  had to this solution. Furthermore,  $x_{\ell+1}$  is not in  $L_j$ , so its distance  $R_{\ell+1}$  is at most  $R/(2^j)$ . We conclude that

$$d(x, \bigcup_{j'=0}^j L_{j'}) \leq R_{\ell+1} \leq \frac{R}{2^j}$$

for all  $x \in P$ . Furthermore,  $\text{parent}(x)$  is the closest point to  $x$  in  $\bigcup_{j=0}^{L(x)-1} L_j$ , so it follows directly that

$$d(x, \text{parent}(x)) = d(x, \bigcup_{j'=0}^{L(x)-1} L_{j'}) \leq \frac{R}{2^{L(x)-1}}$$

is true for all  $x \in P$ . □

**III. The full algorithm** The final algorithm computes the above described levels and the parent function, and then uses the parent function to compute a hierarchical clustering. This is the full pseudo code:

**hierarchical-kcenter**(a finite metric space  $(P, d)$ )

1. Compute  $x_1, \dots, x_{|P|}$  and  $R_2, \dots, R_{|P|}$  by calling **full-farthest-first-traversal** $((P, d))$
2. Set  $R := R_2$  and set  $L_0 := \{x_1\}$ ,  $L(x_1) = 0$
3. **For all**  $j \geq 1$ , set  $L_j = \{x_i \mid R_i \in (\frac{R}{2^j}, \frac{R}{2^{j-1}}]\}$ , and set  $L(x_i) = j$  iff  $x_i \in L_j$
4. **For all**  $x \in P$ , set  $\text{parent}(x) := \arg \min\{d(x, y) \mid y \in \bigcup_{j=0}^{L(x)-1} L_j\}$
5. Set  $C_i = \{x_i\}$  and  $c(C_i) = x_i$  for all  $i \in [|P|]$
6. Set  $\mathcal{H}_{|P|} = \{C_i \mid i \in [|P|]\}$
7. **For**  $k = |P| - 1$  to 1 **do**
8.     Notice that  $x_{k+1}$  is the center of the cluster  $C_{k+1}$  to be reassigned
9.     Let  $x_p = \text{parent}(x_{k+1})$  be the parent of  $x_{k+1}$ , notice  $x_p \in C_p$
10.    Set  $C_p = C_p \cup C_k$  and  $\mathcal{H}_k = \{C_i \mid i \in [k]\}$
11. **Return**  $\mathcal{H}_1, \dots, \mathcal{H}_{|P|}$

In Lines 1-4, the algorithm computes the incremental clustering by using Gonzalez' algorithm, then derives the levels of all points and computes the parent function. The remaining steps compute the actual hierarchical clustering. The process starts with  $\mathcal{H}_{|P|}$ , the partitioning into singleton clusters, i.e., into  $C_i = \{c_i\}$ . Now the algorithm creates the clusterings with less clusters iteratively. It always removes the last clustering (according to the numbering), and merges it with a clustering that is lexicographically before it. This is done by considering the parent  $c_p = \text{parent}(c_{j+1})$

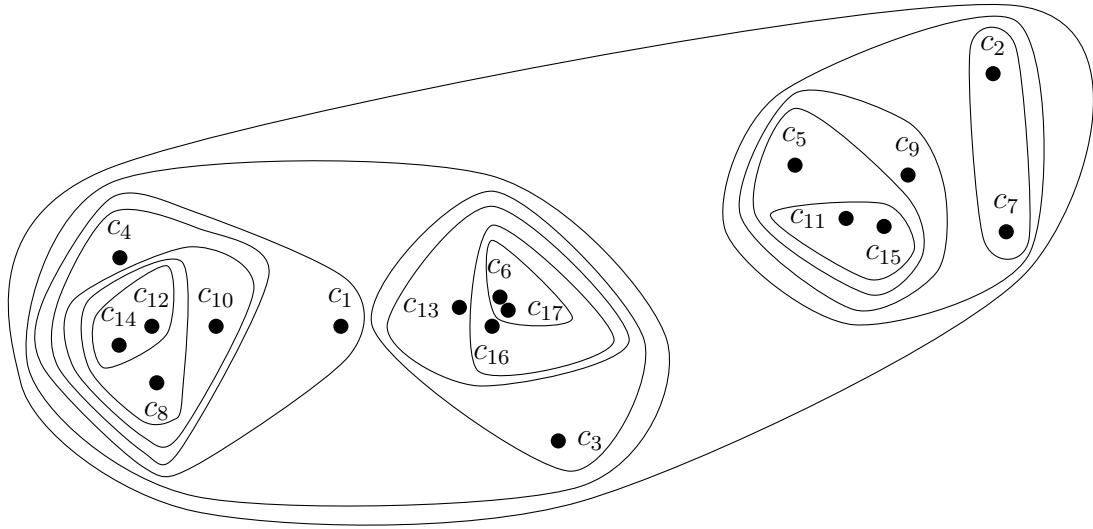


Figure 2.5: Continuation of Figure 2.4, depicting the resulting hierarchical clustering.

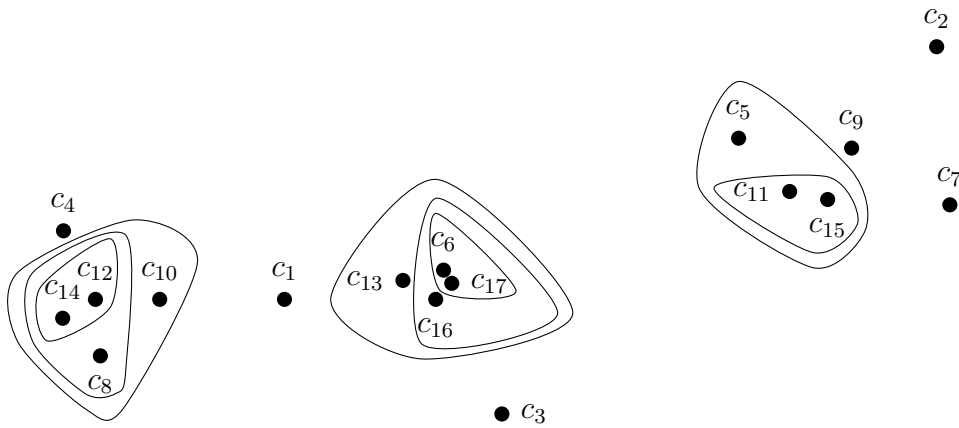


Figure 2.6: Continuation of Figure 2.5, depicting the clustering for  $k = 9$ .

and adding the points in  $C_j$  to  $C_p$  (which is the cluster that contains  $c_p$  and was originally initialized by  $C_p = \{c_p\}$ ). The cluster  $C_{j+1}$  is then dissolved, and the clustering with  $i$  clusters consists of the updated clusters  $C_1, \dots, C_j$ . Figure 2.5 shows the hierarchical clustering that this procedure creates for our example point set (notice that the picture does not uniquely define all intermediate clusterings, although the algorithm does, since the order is not depicted for unrelated sets). Figure 2.6 shows one specific clustering from the hierarchy, namely the clustering for  $k = 9$ .

– end of Lecture 2 –

By Lemma 14, we know that the first time that a point is reassigned (then, to its parent), the distance to its new center is relatively small: It is at most  $R/(2^{L(x)-1})$ . Before we proceed, let us briefly discuss what this means. Assume that  $x = x_i$ . Since  $x$  is on level  $L(x)$ , we know that  $R_i \in (\frac{R}{2^{L(x)}}, \frac{R}{2^{L(x)-1}}]$ . Now notice that  $R_i$  is the cost of the Gonzalez' solution for  $k = i$ . At the point when we reassign  $x = x_i$  for the

first time, we compute the  $(i - 1)$ -clustering. The cost of this clustering is  $R_i$ . We conclude that the  $(i - 1)$ -clustering of Gonzalez costs  $R_i > \frac{R}{2^{L(x)}} = 2 \cdot (R/(2^{L(x)-1}))$ . So, compared to the Gonzalez' solution,  $x$  now has at most twice the distance to its closest center in the  $(i - 1)$ -clustering. [Since Gonzalez' itself is a 2-approximation, this means that the distance between  $x$  and its closest center is at most 4 times the optimum maximum radius.]

Now the crucial question is: What does it mean for the radius that points are potentially reassigned multiple times? The key feature of the algorithm is that two solutions in different levels differ exponentially with respect to their cost. Thus, if a point is reassigned again and again, then the additional distance incurred by the first reassignments plays a smaller and smaller role compared to the later clusterings. Consider  $c_{15}$  in our ongoing example. Its parent is  $c_{11}$ , the parent of  $c_{11}$  is  $c_5$ , the parent of  $c_5$  is  $c_2$ , and the parent of  $c_1$  is  $c_1$ . This means that  $c_{15}$  is reassigned four times. Say we are interested in the clustering for  $k = 4$ , which is created when  $c_5$  stops being a center, which in particular means that  $c_{15}$  is reassigned for the third time, namely to  $c_2$ . By the triangle inequality, we know that

$$d(c_{15}, c_2) \leq d(c_{15}, c_{11}) + d(c_{11}, c_5) + d(c_5, c_2).$$

Furthermore, we know that  $c_{15} \in L_5$ ,  $c_{11} \in L_4$ ,  $c_5 \in L_2$  and  $c_2 \in L_1$ , implying that

$$\begin{aligned} & d(c_{15}, c_{11}) + d(c_{11}, c_5) + d(c_5, c_2) \\ & \leq \frac{R}{2^{L(c_{15})-1}} + \frac{R}{2^{L(c_{11})-1}} + \frac{R}{2^{L(c_5)-1}} \\ & \leq \frac{R}{2^4} + \frac{R}{2^3} + \frac{R}{2^1} \leq R. \end{aligned}$$

Since  $c_5 \in L_2$ ,  $R_5 \in (\frac{R}{2^2}, \frac{R}{2^1}]$ , meaning that the 4-clustering costs at least  $R/4$ . So even though we did three reassignments, the bound only doubled compared to the distance between  $c_5$  and its direct parent. The following theorem shows that in general, the solution for any  $k$  has a radius which is at most 4 times the radius of Gonzalez' solution with the same number of clusters. Since Gonzalez' is guaranteed to be a 2-approximation, we conclude that the hierarchical  $k$ -center algorithm is 8-approximative.

**Theorem 15.** *Algorithm `hierarchical-kcenter`(( $P, d$ )) computes a hierarchical clustering on  $(P, d)$  which is 8-approximative as defined in Definition 13.*

*Proof.* The output of Algorithm `hierarchical-kcenter` is a hierarchical clustering by definition. We now fix a number  $k \in \mathbb{N}^{\geq 1}$  and show that the  $k$ -clustering  $\mathcal{H}_k$  contained in the hierarchy has a maximum radius which is at most 4 times as large as the maximum radius of the  $k$ -clustering computed by Gonzalez' algorithm.

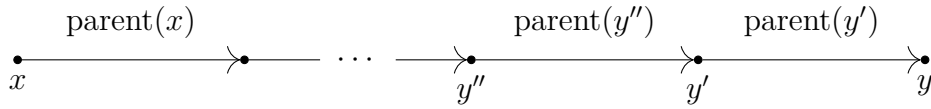
To show this, we briefly use an extension of the parent function which we define inductively. The function  $\text{parent}^1$  is just the parent function, defined by  $\text{parent}^1(x) = \text{parent}(x)$  for all  $x \in P \setminus \{x_1\}$ , and  $\text{parent}(x_1) = x_1$ . The function  $\text{parent}^0$  is the identity, i.e.,  $\text{parent}^0(x) = x$  for all  $x \in P$ . The function  $\text{parent}^i$  for  $i > 1$  results from applying the parent function  $i$  times. More precisely,  $\text{parent}^i$  results from  $\text{parent}^{i-1}$  by setting  $\text{parent}^i(x) = \text{parent}(\text{parent}^{i-1}(x))$  for all  $x \in P$  with  $x \neq x_1$  and  $\text{parent}^i = x_1$ .



**Claim 16.** Let  $x \in P$ ,  $i \geq 1$ ,  $y' = \text{parent}^{i-1}(x)$  and  $y = \text{parent}(y') = \text{parent}^i(x)$ . Then

$$d(x, y) \leq \frac{R}{2^{L(x)-1}} + \frac{R}{2^{L(x)-2}} + \frac{R}{2^{L(x)-3}} + \cdots + \frac{R}{2^{L(y')-1}} = \sum_{j=L(y')}^{L(x)} \frac{R}{2^{j-1}}.$$

*Proof.* We prove the claim inductively. For  $i = 1$ , the claim follows from Lemma 14. Assume that that  $i \geq 2$  and that the claim is true for  $i-1$ . If  $\text{parent}^{i-1}(x) = \text{parent}^i(x)$ , then we are done by induction hypothesis. Otherwise, set  $y'' = \text{parent}^{i-2}(x)$ , thus we also know that  $\text{parent}(y'') = y'$ . Notice that  $y$  is in a lower level than  $y'$ , i.e.  $L(y) < L(y')$ , by the definition of the parent function, and similarly,  $L(y') < L(y'')$ .



By induction hypothesis,

$$d(x, y') \leq \frac{R}{2^{L(x)-1}} + \frac{R}{2^{L(x)-2}} + \frac{R}{2^{L(x)-3}} + \cdots + \frac{R}{2^{L(y'')-1}}.$$

By the triangle inequality,  $d(x, y) \leq d(x, y') + d(y', y)$ . By Lemma 14,  $d(y', y) \leq \frac{R}{2^{L(y')-1}}$ . Combining these statements shows the claim.  $\square$

We want to bound the maximum radius of the  $k$ -clustering. This clustering results from reassigning the points  $x_{|P|}, \dots, x_{k-1}$ , potentially multiple times. Let  $x$  be one of these points, and let  $y'$  be the last point among its parents which is in  $\{x_{|P|}, \dots, x_{k+1}\}$ , and  $y = \text{parent}(y')$  be the first one that is in  $\{x_k, \dots, x_1\}$ . By Claim 16, we know that  $d(x, y) \leq \sum_{j=L(y')}^{L(x)} \frac{R}{2^{j-1}}$ . We can simplify this by using Fact 2:

$$d(x, y) \leq \sum_{j=L(y')}^{L(x)} \frac{R}{2^{j-1}} \leq \frac{R}{2^{L(y')-1}} \sum_{j=0}^{\infty} \left(\frac{1}{2}\right)^j = \frac{R}{2^{L(y')-1}} \cdot \frac{1}{1/2} = \frac{R}{2^{L(y')-2}}.$$

The radius of the  $k$ -clustering  $\{x_1, \dots, x_k\}$  is  $R_{k+1}$ . Now the crucial observation is that since  $y'$  is still in  $\{x_{|P|}, \dots, x_{k+1}\}$ , its level  $L(y')$  is at least  $L(x_{k+1})$ , i.e.,  $L(y') \geq L(x_{k+1})$ , and

$$d(x, y) \leq \frac{R}{2^{L(y')-2}} \leq \frac{R}{2^{L(x_{k+1})-2}}.$$

Since  $R_{k+1} \in \left(\frac{R}{2^{L(x_k)}}, \frac{R}{2^{L(x_{k+1})-1}}\right]$ , we get that

$$d(x, y) \leq \frac{R}{2^{L(x_{k+1})-2}} \leq 4 \cdot \frac{R}{2^{L(x_k)}} \leq R_{k+1}.$$

This is true for all points  $x \in P$ , thus the maximum radius of the solution is at most 4 times the maximum radius of the Gonzalez' solution, which is at most twice times the maximum radius of an optimal solution. The theorem follows.  $\square$