Connected k-Center and k-Diameter Clustering

Lukas Drexler
Heinrich-Heine Universität Düsseldorf, Germany

Jan Eube
Universität Bonn, Germany

Kelin Luo
Universität Bonn, Germany

Heiko Röglin
Universität Bonn, Germany

Melanie Schmidt
Heinrich-Heine Universität Düsseldorf, Germany

Julian Wargalla
Heinrich-Heine Universität Düsseldorf, Germany

Abstract

Motivated by an application from geodesy, we study the *connected k-center problem* and the *connected k-diameter problem*. These problems arise from the classical k-center and k-diameter problems by adding a side constraint. For the side constraint, we are given an undirected *connectivity graph* G on the input points, and a clustering is now only feasible if every cluster induces a connected subgraph in G. Usually in clustering problems one assumes that the clusters are pairwise disjoint. We study this case but additionally also the case that clusters are allowed to be non-disjoint. This can help to satisfy the connectivity constraints.

Our main result is an $O(1)$-approximation algorithm for the disjoint connected k-center and k-diameter problem for Euclidean spaces of low dimension (constant d) and for metrics with constant doubling dimension. For general metrics, we get an $O(\log^2 k)$-approximation. Our algorithms work by computing a non-disjoint connected clustering first and transforming it into a disjoint connected clustering.

We complement these upper bounds by several upper and lower bounds for variations and special cases of the model.

2012 ACM Subject Classification Theory of computation → Facility location and clustering

Keywords and phrases Approximation algorithms, Clustering, Connectivity constraints

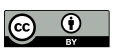
Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.50

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2211.02176`

Funding This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 390685813; 459420781.

Acknowledgements The authors thank anonymous reviewers of a previous draft for helpful comments and pointing out relevant related work. We thank Jürgen Kusche and Christian Sohler for raising the problem and for fruitful discussion on the modeling. We also thank Xiangyu Guo for the discussion on the algorithm design and analysis.



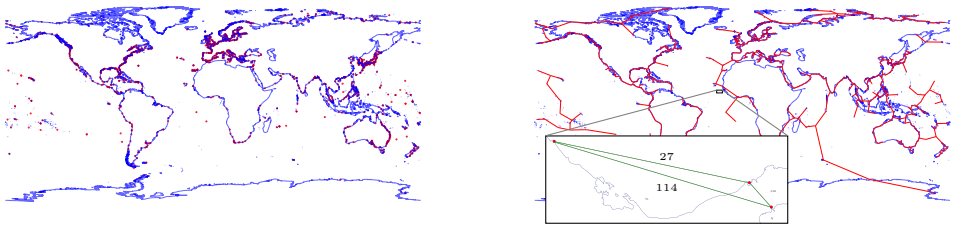


Figure 1 Gauge stations around the globe, with station location data from PSMSL (`http://www.psmsl.org/data/obtaining/`), plotted onto the map from the Natural Earth data set (`https://www.naturalearthdata.com/downloads/10m-physical-vectors/10m-coastline/`). Highlighted are three stations in Central America, and the numbers are Fréchet distances computed on the curves defined by sea levels between 1953 and 1968.

1 Introduction

Clustering problems occur in a wide range of application domains. Because of the general importance and interesting combinatorial properties, well-known k-clustering problems like k-center, k-median, and k-means have also been vastly studied in theory. These problems are NP-hard and APX-hard, but many constant-factor approximation algorithms for them are known. All k-clustering problems ask to partition a set of points (usually in a general metric space or in Euclidean space) into k clusters, often by picking k centers and assigning every point to its closest center. The clusters are then evaluated based on the distances between the points and their corresponding centers. For example in the case of k-center, the objective is to minimize the maximum distance between any point and its closest center.

In applications, clustering problems are often subject to side constraints. Consequently, clustering with side constraints has also become a thriving topic for designing approximation algorithms. Probably the most known example is clustering with capacities where the number of points in a cluster is limited. Notice how this constraint prevents us from assigning points to their closest center because there might not be enough space. So, for example, uniform capacitated (center-based) clustering consists of finding k centers and an assignment of points to those centers such that every center gets at most U points (and then evaluating the desired objective). Finding a constant factor approximation for uniform capacitated k-median clustering is a long standing open problem. Other constraints that have been studied are for example lower bounds (here, a cluster has to have a certain minimum number of points, so it may be beneficial to open less than k clusters) and clustering with outliers (here we are allowed $k + z$ clusters, but z of them have to be singletons, i.e. outliers). There are also results on constraints that restrict the choice of centers, for example by demanding that the centers satisfy a given matroid constraint. Among the newer clustering problems with constraints are those that evolve around aspects of fairness. These constraints are typically more complex and can either be point-based or center-based. Each constrained clustering problem, old or new, comes with a unique combinatorial structure, giving rise to a plethora of insights on designing approximation algorithms.

In this paper, we study a constraint that stems from the area of sea level geodesy but which is also of interest for other domains (discussed briefly below). For the application that motivated our work, consider the left picture in Figure 1. We see the location of tide gauge stations around the globe from the PSMSL data set [13, 9]. At every station, sea level heights have been collected over the years, constituting monthly time series. These records

can be used to reconstruct regional or global mean sea levels. However, the tide gauges have usually been constructed for practical purposes and not for sea level science. As a result, they are unevenly distributed over the globe. One way out of this is to replace clusters of tide gauges by representative records to thin out the data set. Our general goal is therefore to cluster the tide gauges into a given number k of clusters. However, the objective is not based on the gauge stations' geographic distance but on the time series. We wish to combine gauge stations with similar time series into one, i.e., when we cluster, we want to find clusters where the center's time series is similar to the records collected at the tide gauges represented by that center. We can model the distance between time series by a metric distance measure for time series or curves (like the Fréchet distance). As the objective we pick k-center, so we want to minimize the maximum distance between the center and the points that are replaced by it. Now we get to the complication: The gauge stations are *also* points on the map. We do not want to have points in the same cluster that are geographically very far away.

It is not immediately clear how to best model this scenario. We could resort to bicriteria approximation and look for solutions where both the time series of points in a cluster are similar and the radius of clusters is small, by either looking at the Pareto front or weighting the two objectives. Alternatively, we could fix a threshold and limit the geographic distance between centers and points, i.e., demand that a point x can only be assigned to center c if its geographic distance is at most some T. Both modelings have the drawback that they really only capture the distance on the map, while in reality, we would like to have somewhat coherent clusters that correspond to non-overlapping areas on the map. Indeed, we might be fine with having points of large geographic distance in the same cluster if all points 'between' them are also in the same cluster (i.e., that larger area of the sea behaves very similar with respect to the gauge station measurements).

The modeling that we study incorporates this via a preprocessing step. We assume that the points have been preprocessed such that we get a connectivity graph like shown on the right in Figure 1. The graph on the map was computed by finding a minimum spanning tree of the points, but it could be computed in other ways, too. The important part is that it captures a neighborhood structure. To model coherence, we now demand that clusters are *connected* in this graph. Figure 2 gives an example.

▶ **Problem 1.** *In a connected k-clustering problem, we are given points V, a metric d on V, a number k, and an unweighted and undirected connectivity graph $G = (V, E)$. A feasible solution is a partitioning of V into k clusters $C_1, \dots, C_k$ which satisfies that for every $i \in \{1, \dots, k\}$ the subgraph of G induced by C_i is* connected.

For the connected k-center problem, a solution also contains centers $c_1, \dots, c_k$ corresponding to the clusters $C_1, \dots, C_k$ and the objective is to minimize the maximum radius $\max_{i\in[k], x\in C_i} d(x, c_i)$. For the connected k-diameter problem the objective is to minimize the maximum diameter $\max_{i\in[k]} \max_{x,y\in C_i} d(x, y)$. It is easy to see that the connected k-clustering problem generalizes the classic k-center and k-diameter problem whose connectivity graph G is a complete graph.

Interestingly, the connected k-center problem was independently defined in an earlier paper by Ge et al. [4] (previously unknown to us. We thank the anonymous reviewer who pointed us to this reference). In that paper, connected clustering is motivated in the context of applications where both attribute and relationship data is present. It is applied to scenarios of community detection and gene clustering, showing the wide applicability of the modeling. We discuss their work further in the related work section.

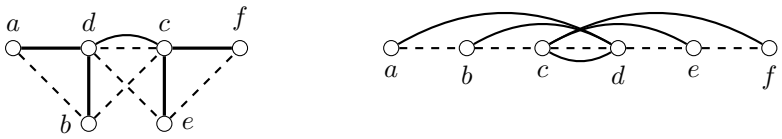


Figure 2 An example. The solid edges form the metric: Vertices connected by a solid edge have distance 1 and all other distances are 2. The dashed edges form the connectivity graph. Both pictures show the same graph. The optimal k-center solution with centers $\{c, d\}$ and clusters $\{a, b, d\}$ and $\{c, e, f\}$ is not connected. Any optimal (disjoint) connected k-center solution has radius 2.

Disjoint vs non-disjoint clusters, restricted graph classes

Notice that we demand that the C_i are *disjoint*. For some clustering problems with constraints the objective value can be decreased when we are allowed to assign points to more than one cluster: For example, lower bounds are easier to satisfy when points can be reused. The same is true for connected clustering: It is easier to satisfy connectivity when we can put important points into multiple clusters. For our application, we want to have disjoint clusters, but we still study the variation for completeness and also since it allows for better approximation algorithms that can be at least tested for their usefulness in the application (e.g., leaving it to the user to resolve overlaps). Notice that in Figure 2, allowing non-disjoint clusters enables the solution $\{c, d\}$ with clusters $\{a, b, c, d\}$, $\{c, d, e, f\}$ which has cost 1.

▶ **Definition 2.** *We distinguish between connected k-clustering* with disjoint clusters *and* with non-disjoint *clusters, referring to whether the clusters C_i have to be pairwise disjoint or not.*

Finally, we observe that in our application the connectivity graph is not necessarily arbitrary. Depending on the way that we build the graph, it could be a tree (the minimum spanning tree) or even a line (if we follow the coast line). Thus, we are interested in the problem on restricted graph classes as well.

Results and techniques

Our main result is an approximation algorithm that works for both the disjoint connected k-center problem and the disjoint connected k-diameter problem for general connectivity graphs G. For general metrics, the algorithm computes an $O(\log^2 k)$-approximation. If the metric has bounded doubling dimension D, the approximation ratio improves to $O(2^{3 \cdot D})$, and for Euclidean spaces, to $O(d \cdot 2^d)$. To obtain these results we first compute a non-disjoint clustering. Then we develop a method using a concept of a layered partitioning (see Definition 10) to make the clusters disjoint. We show how to obtain such a partitioning for different metrics. Both steps are novel and form the main contribution of this paper. In addition, in the full version of this paper we study how to compute well-separated partitions if the number of clusters is small, particularly when $k = 2$.

We also study restricted connectivity graphs (lines, stars and trees) and also the easier case of non-disjoint connected clustering. In this context we discuss greedy algorithms and obtain hardness results via reductions. The rest hardness proof in the full version of this paper is technically more involved. An overview of our results is given in Table 1, the more details are given in Section 2.

Table 1 An overview of the bounds shown in this paper and the literature for connected k-clustering. The notation $[\ell, u]$ stands for a lower bound ℓ and an upper bound u on the best possible approximation factor (achievable in polynomial time and assuming $P \neq NP$). Results marked by "*" are proven in the full version of this paper.

Restriction \ Objective	k-Center		k-Diameter	
	disjoint	non-disjoint	disjoint	non-disjoint
G is a line	1 Ge et al. [4]	1 Cor. 4	1 Cor. 4	
G is a star / tree		$[2, 2]$ Cor. 7, Lem. 9	$[2, 2]$ Lem. 5, Thm. 6	$[2, 2]$ Cor. 7, Lem. 9
Doubling dimension D	$O(2^{3D})$ Thm. 23		$O(2^{3D})$ Thm. 23	
L_p metric in dimension d	$O(d \cdot 2^d)$ Thm. 20		$O(d \cdot 2^d)$ Thm. 20	
No Restrictions	$[3^*, O(\log^2 k)]$ Thm. 18		$[2, O(\log^2 k)]$ Lem. 5, Thm. 18	

Related work

The k-center problem and the k-diameter problem are both NP-hard to approximate better than by a factor of 2 (see [10, 7] for k-center, k-diameter follows along the same lines). There are two popular 2-approximation algorithms for k-center which both also work for k-diameter with the same approximation guarantee [5, 8]. There are various results on side constraints for k-center and related k-clustering problems, including [1, 2, 3, 11] and many others. A more extensive list of results is contained in the full version of this paper, and we only review closely related work in the following. The connected k-center problem with disjoint clusters has been introduced and studied by Ge et al. [4][1]. Besides other results, Ge et al. present a greedy algorithm for the problem and claim that it computes a 6-approximation. In the full version of this paper we present an example showing that this greedy algorithm actually only obtains an $\Omega(k)$-approximation. The greedy algorithm is based on the approach of transforming a non-disjoint clustering into a disjoint one. In this transformation, it does not change the centers, i.e., it uses the given centers of the non-disjoint clustering also as centers for the disjoint clustering. In addition, we prove in the full version of this paper a lower bound showing that no algorithm based on transforming a non-disjoint clustering into a disjoint one with the same centers can compute an $O(1)$-approximation. Hence, without fundamental changes of the algorithm, no $O(1)$-approximation can be obtained. We even show that in general the optimal non-disjoint clustering can be better than the optimal disjoint clustering by a factor of $\Omega(\log \log k)$. Hence, if one uses only the radius of an optimal non-disjoint clustering as a lower bound for the radius of an optimal disjoint clustering, one cannot show a better approximation factor than $\Omega(\log \log k)$. To the best of our knowledge, no other approximation algorithms with provable guarantees for the connected k-center or k-diameter problem are known.

Ge et al. introduce the connected k-center problem to model clustering problems where both attribute and relationship data is present. They perform experiments in the context of gene clustering and community detection and demonstrate that for both these applications modelling them as connected clustering problems leads to superior results compared to standard clustering formulations without connectivity constraint. For community detection for example, they construct datasets from DBLP[2] where researchers are supposed to be

[1] We thank an anonymous reviewer for pointing us to this reference
[2] https://dblp.org/

clustered according to their main research area. Based on keyword frequencies they defined a distance measure for the researchers. At the same time, the coauthor network can be used as a connectivity graph. The advantage of connected clustering compared to traditional models is that it naturally takes into account both the distance measure and the coauthor network. For their experiments, Ge et al. develop a heuristic called NetScan for the connected k-center problem with disjoint clusters, which is reminiscent of the k-means method, and efficient on large datasets. In their experiments, the outcomes of this heuristic were significantly better than the outcomes of state-of-the-art clustering algorithms that take into account either only the distance measure or only the coauthor network. The work of Ge et al. has attracted some attention and it is cited in many other articles on community detection and related subjects.

Furthermore, Ge et al. show that already for $k = 2$, the connected k-center problem with disjoint clusters is NP-hard. They also argue that it is even NP-hard to obtain a $(2 - \epsilon)$-approximation for any $\epsilon > 0$. Additionally they give an algorithm based on dynamic programming with running time $O(n^2 \log n)$ that solves the connected k-center problem with disjoint clusters optimally when the connectivity graph is a tree.

Gupta et al. [6] study the connected k-median and k-means problem and prove upper and lower bounds on their approximability. Related to our motivation, Liao and Peng [12] consider the connected k-means problem to model clustering of spatial data with a geographic constraint. They develop a local-search based heuristic and conduct an experimental evaluation.

Outline

In Section 2 we discuss the general setting and results for restricted graph classes. Section 3 covers the case of non-disjoint connected clustering. Then in Section 4, we show the results on the connected clustering problems for general connectivity graphs and disjoint clusterings.

2 Setup and review of results on restricted graph classes

For all approximation algorithms in this paper, we use the following well-known framework for k-center approximation due to Hochbaum and Shmoys [8]. It is built upon the following fact: For the k-diameter or k-center problem (connected or not), the value of the cost function is always equal to one of the at most n^2 different distances between two points in V where $n = |V|$. This is true because it is either the distance between two points in the same cluster (k-diameter) or it is the distance between a point and its center (k-center). Thus, a standard scheme to follow is to sort these distances in time $O(n^2 \log n)$ and then search for the optimum value by binary search. The problem then reduces to finding a subroutine for the following task.

▶ **Problem 3.** *If there is a solution which costs r for a given r, find a solution that costs at most $\alpha \cdot r$. Otherwise, report that r is too small.*

An algorithm that solves this task can easily be turned into an α-approximation by searching for the smallest r for which the algorithm returns a solution. The running time of the resulting algorithm is $O(n^2 \log n)$ for the preprocessing plus $O(\log n)$ times the running time of the subroutine.

Lines, stars and trees

Connected k-clustering demands that the clusters are connected in a given connectivity graph G. How tricky is this condition? Maybe it can actually *help* to solve the problem? This is true if G is very simple, i.e., a line. We include the following proof as a warm-up.

▶ **Corollary 4.** *When the connectivity graph G is a line graph, then the connected k-center problem and the connected k-diameter problem can be solved optimally in time $O(n^2 \log n)$ both with disjoint and non-disjoint clusters. This is true even if the distances are not a metric.*

Proof. We only show how to solve the connected k-center problem with non-disjoint clusters. The full proof can be found in the full version of this paper. The line graph G is defined by vertices $V = \{v_1, v_2, ..., v_n\}$ and edges $E = \{\{v_i, v_{i+1}\} \mid i \in \{1, \dots, n-1\}\}$. Assume that r is given.

Notice that any connected cluster is a subpath of G. We start by precomputing for every v_i how far a cluster with center at v_i can stretch to the left and right: Let a_i be the smallest ℓ such that $d(v_j, v_{j'}) \leq r$ for all $j, j' \in \{\ell, \dots, i\}$ and let b_i be the largest ℓ such that $d(v_j, v_{j'}) \leq r$ for all $j, j' \in \{i, \dots, \ell\}$. We can compute all a_i and all b_i in time $O(n^2)$. Now we cut the line into clusters. We start by finding an index i with $a_i = 1$ for which b_i is as large as possible because we have to cover the first vertex and want to cover as many other vertices as possible. We place a center at v_i and know that all vertices until v_{b_i} are covered by the cluster. Now we know that the next cluster has to contain v_{b_i+1}, so we search for an i' which satisfies $b_i + 1 \in \{a_{i'}, \dots, b_{i'}\}$, if there are multiple, we take the one with maximum $b_{i'}$. This finds the center which covers v_{b_i+1} and the largest number of additional vertices. We place a center at $v_{i'}$. It may be that $i' < i$ as in Figure 2) and thus the clusters have to overlap (recall that we are in the non-disjoint case). The process is iterated until v_n is covered. If the number of clusters is more than k, we report that r was too small, otherwise, we report the clustering. This way we solve Problem 3 for $\alpha = 1$ in time $O(n^2)$. ◀

For trees, k-center and k-diameter differ. Surprisingly, the connected k-diameter problem is already NP-hard if G is a star. We prove the following lemma by a reduction from the uniform minimum multicut problem on stars in the full version of this paper.

▶ **Lemma 5.** *Let $\epsilon > 0$. Assuming $P \neq NP$, there is no $(2 - \epsilon)$-approximation algorithm for the connected k-diameter problem with disjoint clusters even if G is a star.*

Notice how the connected k-diameter problem with G being a star is thus very different from the k-diameter problem where the *metric* is given by a graph metric that is a star. The latter problem can be solved optimally by sorting the edges by weight and then deleting the $k - 1$ most expensive edges to form k connected components which form an optimal clustering. Say we have distances $d(e_1) \geq d(e_2) \geq \dots \geq d(e_n)$, then this optimal clustering has cost $d(e_k) + d(e_{k+1})$. However, any clustering that keeps an edge from $\{e_1, \dots, e_{k-1}\}$ costs at least $d(e_{k+1}) + d(e_{k-1}) \geq d(e_k) + d(e_{k+1})$ since it deletes at most $k - 1$ edges.

Ge et al. [4] show that the connected k-center problem is still solvable optimally for trees by dynamic programming.

▶ **Theorem 6** (Ge et al. [4])**.** *When the connectivity graph G is a tree, then the connected k-center problem with disjoint clusters can be solved optimally in time $O(n^2 \log n)$. This is true even if the distances are not a metric.*

It follows immediately that the connected k-diameter problem with disjoint clusters on trees can be 2-approximated by the same algorithm because the diameter of the produced solution is always at most twice the radius. This is interesting because our reduction in Lemma 5 shows that this is tight, i.e., using the dynamic programming algorithm for k-diameter achieves the best possible approximation ratio (assuming $P \neq NP$).

3 General G, non-disjoint clusters

The connected k-center and k-diameter problems with non-disjoint clusters behave similarly to the unconstrained versions. On the positive side, there is a 2-approximation; on the negative side, it is NP-hard to approximate these problems better than 2. In contrast to the case of disjoint clusters, APX-hardness starts with stars for *both* k-center and k-diameter. We show this via reductions from clique cover and set cover in the full version of this paper.

▶ **Corollary 7.** *Let $\epsilon > 0$. Assuming $P \neq NP$, there is no $(2-\epsilon)$-approximation algorithm for the connected k-diameter problem with non-disjoint clusters, even if G is a star. The same is true for the connected k-center problem with non-disjoint clusters.*

For the positive result, the classical result by Hochbaum and Shmoys [8] can be used. We discuss it in detail because we need it as a basis for our algorithms. For the unconstrained k-center problem, Problem 3 for $\alpha = 2$ can be solved as follows: Given input V, k, and a radius r, one picks an arbitrary point $x \in V$ and puts all nodes within distance $2r$ of x into one cluster. When r is at least the radius of the optimal k-clustering, this cluster will contain all nodes that are in the same optimal cluster as x. The cluster is then removed from V and the process is repeated until all nodes are covered. If the number of clusters is at most k, the solution is returned, otherwise, it is reported that r was too small.

This algorithm can easily be adapted to the connected k-center problem with non-disjoint clusters by the following observation: Let x and y be two nodes from the same optimal cluster with center c and radius r. Then x and y are connected in the connectivity graph by a path that contains only nodes within distance $2r$ from x and y. So the algorithm is: When a node x is selected, put all nodes into a cluster that have distance at most $2r$ from x and are reachable from x in the connectivity graph via a path on which all nodes have a distance of at most $2r$ from x. This set can be determined by the BFS-type algorithm `ComputeCluster` (see Algorithm 1 with $R = 2r$). Say the resulting cluster is T. Do not remove T from G

Algorithm 1 ComputeCluster(G, M, R, c).

Input: points V, graph $G = (V, E)$, metric $M = (V, d)$, radius R, node $c \in V$
$T \leftarrow \{c\}$;
$N \leftarrow \{u \in V \setminus T \mid \exists v \in T, (v, u) \in E : d(u, c) \leq R\}$;
while $N \neq \emptyset$ **do**
  $T \leftarrow T \cup N$;
  $N \leftarrow \{u \in V \setminus T \mid \exists v \in T, (v, u) \in E : d(u, c) \leq R\}$;
Output: cluster T

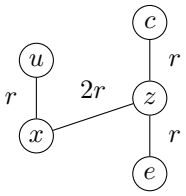
but only mark all nodes in T as covered. As long as there are uncovered nodes, pick an arbitrary such node and form a cluster of radius $2r$ around it (in general this cluster will also contain nodes that are already covered). This will result in at most k connected clusters with radius $2r$ if r is at least the radius of an optimal connected k-clustering. We call this algorithm `GreedyClustering` and we give its pseudocode as Algorithm 2. In general, the sets T_c computed by this algorithm are not disjoint but the centers are pairwise distinct.

▶ **Lemma 8.** *Let r^* denote the radius of an optimal connected k-center clustering with non-disjoint clusters. For $r \geq 2r^*$, Algorithm 2 computes a center set C with $|C| \leq k$.*

Proof. Consider a node $c \in V$ that is chosen as a center by the algorithm and the optimal cluster O node c is contained in. This cluster is centered around some node c' and has a radius of at most r^*. Hence, by the triangle inequality all nodes in O have a distance of at

Algorithm 2 GreedyClustering(G, M, r).

Input: graph $G = (V, E)$, metric $M = (V, d)$, radius r
$C \leftarrow \emptyset$; `// center nodes`
$V' \leftarrow V$; `// uncovered nodes`
while $V' \neq \emptyset$ **do**
  select a node $c \in V'$ and add it to C;
  $T_c \leftarrow$ ComputeCluster(G, M, r, c);
  $V' \leftarrow V' \setminus T_c$;
Output: centers C, sets T_c for all $c \in C$



The optimal connected 2-clustering has centers x and z with clusters $\{x, u\}$ and $\{z, c, e\}$ and a radius of r. The greedy algorithm started with x forms $\{x, u, z\}$ as the first cluster. After that, only c and e remain. Without z, they are not connected anymore and have to go into different clusters.

Figure 3 An example where greedy disconnects an optimum cluster.

most $2r^*$ from c. Also since O is connected, all nodes in O are reachable from c. In particular, all nodes in O are reachable from c on paths that contain only nodes within distance $2r^*$ of c. This implies that for $r \geq 2r^*$, the set T_c is a superset of the optimal cluster O. Since the centers in Algorithm 2 are chosen among the uncovered nodes, all chosen centers must be from distinct optimal clusters. This implies that there can be at most k centers in C. ◀

The same algorithm works for the connected k-diameter problem when `ComputeCluster` is evoked with $R = r$ (not $2r$) if r is at least the optimal diameter. By adding all points in distance r to the cluster of the chosen center x, it is ensured that the optimum cluster is added if r is at least the optimum value (since the distance between two points is then at most r). Furthermore, the resulting cluster has diameter at most $2r$ by the triangle inequality.

▶ **Lemma 9.** *There exists a 2-approximation algorithm for the connected k-center problem with non-disjoint clusters and also for the connected k-diameter problem with non-disjoint clusters.*

4 General G, disjoint clusters

The disjoint case for general connectivity graphs is more challenging. To keep the presentation simple, we focus in the following on the connected k-center problem: Given an unweighted graph $G = (V, E)$ and a metric space $M = (V, d)$ with $d : V \times V \to \mathbb{R}$, find k node-disjoint connected subgraphs of G (clusters) that cover all vertices and minimize the maximum radius of these subgraphs. An adaptation to the connected k-diameter problem can be found in the full version of this paper.

We start with the algorithm `GreedyClustering` from the previous section on the non-disjoint case. Notice that in general, the output of this algorithm is not node-disjoint. We could opt to delete the nodes in T computed by Algorithm 1 to enforce disjointness, however, the problem is this: The first cluster that the algorithm forms around a vertex x is guaranteed to be a superset of the optimal cluster that x is contained in. It might be a strict superset and contain a node that belongs to a different optimal cluster. This node will get removed from G together with all other nodes in the cluster around x. However, its removal might

make the optimal cluster it is contained in unconnected. This is problematic because then k connected clusters might not suffice anymore to cover all points from G even if we guessed the optimal radius r correctly. See Figure 3 for an example where this happens.

4.1 Making the clusters disjoint

In this section we describe how to transform the set of non-disjoint clusters computed by `GreedyClustering` into a set of pairwise disjoint clusters that cover all points at the cost of increasing the radius or diameter. This transformation has to be performed very carefully in order to not increase the radius or diameter by too much.

Let C with $|C| \leq k$ denote the set of centers around which the non-disjoint clusters have been formed by the algorithm and let r denote their radius. The following two observations are helpful: (1) When two centers are more than $2r$ apart then their corresponding clusters are disjoint. (2) If a set of centers have pairwise distance at most L then merging the corresponding clusters results in a single cluster with radius at most $r + L$ and diameter at most $2r + L$.

If it is possible to partition the centers into groups such that all centers within the same group have a distance of at most L and all centers from different groups have a distance of more than $2r$, we could make the clusters disjoint as follows: as long as there are two non-disjoint clusters whose centers are in the same group of the partition, merge them into a single cluster. In the end, the algorithm will return no more than $|C| \leq k$ clusters. By isolating some singletons as new clusters, we obtain a solution with exactly k clusters as required without worsening the solution. After this, all clusters whose centers are in the same group are disjoint (if not they would have been merged) and clusters whose centers are in different groups are disjoint because their centers are far enough from each other. Hence, such a partition results in a solution with disjoint clusters with radius $r + L$ and diameter $2r + L$. A key idea in our algorithm for the general case is to find such a partition of the centers in C with small L. However, observe that this is not possible in general. A simple counterexample would be that all centers are equally spaced on a line with distance r between two consecutive centers. Then all centers have to be in the same group and L would be $(k-1)r$, resulting in an approximation factor of $\Omega(k)$.

To circumvent this problem, we do not partition all centers from C at once but we start with a partition of a subset of C that satisfies the properties above (i.e., centers in the same group have distance at most L, while centers in different groups have a distance of more than $2r$). We call this the first layer of the partition. Then we remove all centers contained in the first layer from C and proceed with the remaining centers analogously: Let C' denote the set of centers not contained in the first layer. We find a partition of a subset of C' that satisfies the properties above and call this the second layer of the partition. We repeat this process until all points from C are in some layer. We call such a partition a *well-separated partition*. Figure 4 shows possible partitions for the example above.

▶ **Definition 10.** *Let $M = (C, d)$ be a metric and $r > 0$. An* r-well-separated partition *with $\ell \in \mathbb{N}$ layers and with parameters $(h_1, \ldots, h_\ell)$ is a partition of C into groups $\{C_{1,1}, \ldots, C_{1,\ell_1}\}, \{C_{2,1}, \ldots, C_{2,\ell_2}\}, \ldots, \{C_{\ell,1}, \ldots, C_{\ell,\ell_\ell}\}$ with the following properties.*

- **(i)** *The groups cover all points from C, i.e., $\bigcup_{i\in[\ell], j\in[\ell_i]} C_{i,j} = C$.*
- **(ii)** *The groups are pairwise disjoint, i.e., $\forall i, i', j, j'$ with $i \neq i'$ or $j \neq j'$, $C_{i,j} \cap C_{i',j'} = \emptyset$.*
- **(iii)** *For $i \in [\ell]$, we call the sets $C_{i,1}, \ldots, C_{i,\ell_i}$ the* sets on layer i. *Two different sets from the same layer are more than $2r$ away, i.e., $\forall i \in [\ell], v \in C_{i,j}, v' \in C_{i,j'}$ with $j \neq j'$, $d(v, v') > 2r$.*
- **(iv)** *For $i \in [\ell]$, the maximum diameter of a group on layer i is at most h_i, i.e., $\max_j \max_{v,v'\in C_{i,j}} d(v, v') \leq h_i$.*

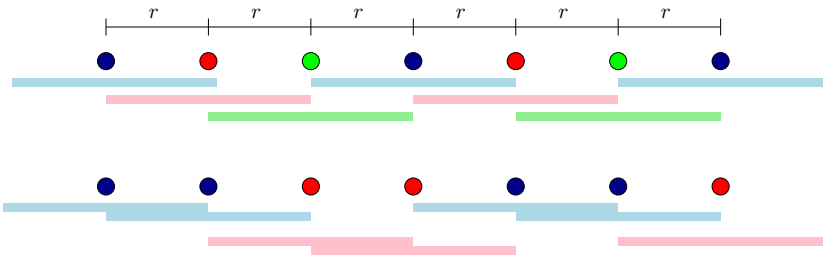


Figure 4 We consider an instance with 7 centers on a line where consecutive centers have a distance of r. The top figure shows a well-separated partition of this instance with $L = 0$ and $\ell = 3$ layers. The colors depict the different layers and the colored rectangles depict the clusters of radius R around these centers. On the blue layer there are, e.g., three groups where each group consists of a single blue center. The bottom figure shows a well-separated partition of the same instance with $L = r$ and $\ell = 2$. The blue layer contains two groups of two centers each, while the red layer contains two groups, one with two centers and one with only one center.

It is not clear at first glance why a well-separated partition is helpful for obtaining a solution with disjoint clusters. For every layer of the partition, we can use the reasoning above. That is, we merge all non-disjoint clusters whose centers are in the same group to obtain disjoint clusters with radius $r + L$ and diameter $2r + L$. However, a cluster is then only disjoint from all clusters on the same layer but in general not from clusters on other layers (see Figure 4). A main ingredient of our algorithm is a non-trivial way to merge clusters on different layers. For this, we add the layers one after another. Consider the case of two layers. The clusters from the first layer are disjoint from each other. We add the clusters of the second layer one after another. For each cluster from the second layer, we first check with which clusters from the first layer it overlaps. If there is more than one, we split the cluster from the second layer into multiple parts and merge the parts with different clusters from the first layer with which they overlap. This is done in such a way that the final result is a set of disjoint connected clusters. We prove with an inductive argument that the radius and diameter of these clusters is $O(\ell \cdot L)$, where ℓ denotes the number of layers of the well-separated partition.

The following lemma describes an algorithm that adjusts the clusters layer by layer to make them pairwise disjoint.

▶ **Lemma 11.** *Consider an instance $(G = (V, E), M = (V, d), k)$ of the connected k-center problem and assume that Algorithm 2 computes a center set $C \subseteq V$ with $|C| \leq k$ for some radius r. Furthermore, let an r-well-separated partition of C with ℓ layers and parameters $(h_1, \ldots, h_\ell)$ be given. Then we can efficiently find a feasible solution for the connected k-center problem with disjoint clusters with radius at most $(2\ell - 1)r + \sum_{i=1}^{\ell} h_i$.*

Proof. According to Definition 10 and Algorithm 2, we have the following properties:

(i) $\bigcup_{i\in[\ell],j\in[\ell_i]} C_{i,j} = C$
(ii) $\forall i, i', j, j'$ with $i \neq i'$ or $j \neq j'$: $C_{i,j} \cap C_{i',j'} = \emptyset$
(iii) $\forall i \in [\ell], c \in C_{i,j}, c' \in C_{i,j'}$ with $j \neq j'$: $d(c, c') > 2r$ and $T_c \cap T_{c'} = \emptyset$
(iv) $\forall i \in [\ell]$, $j \in [\ell_i]$, $c, c' \in C_{i,j}$: $d(c, c') \leq h_i$
(v) $\bigcup_{i\in[\ell],j\in[\ell_i]} \bigcup_{c\in C_{i,j}} T_c = V$

In the first step, we adjust the clusters by merging all non-disjoint clusters whose centers belong to the same group. To be precise, for each group $C_{i,j}$ we do the following: As long as there are two different centers $c \in C_{i,j}$ and $c' \in C_{i,j}$ with $T_c \cap T_{c'} \neq \emptyset$, we remove c' from

$C_{i,j}$ and replace T_c by $T_c \cup T_{c'}$. That is, we merge the two clusters T_c and $T_{c'}$ and define c as its center. Since centers in the same group on layer i have a distance of at most h_i, after this step the clusters in each group $C_{i,j}$ are pairwise disjoint and have a radius of at most $r + h_i$ and a diameter of at most $2r + h_i$. They are still connected because we only merge connected clusters that have at least one node in common.

Since clusters in different groups of the same layer are pairwise disjoint anyway, all clusters on the same layer are pairwise disjoint after this step. Hence, in the next step we only need to describe how clusters from different layers can be made disjoint. For this, it will be helpful to view the clusters as trees. To make this more precise, consider a cluster T_c with center c. We know that the subgraph of G induced by T_c is connected. For any cluster T_c we choose an arbitrary spanning tree in this induced subgraph and consider c to be the root of this tree. Let $\mathcal{T}_i$ denote the set of all such trees in the i-th layer for $i \in [\ell]$. In the following we will use the terms clusters and trees synonymously. By abuse of notation we will use T_c to denote both the cluster with center c and the spanning tree with root c, depending on the context.

For every $i \in [\ell]$, all trees in $\mathcal{T}_i$ are node-disjoint. We will now describe how to ensure that trees on different layers are also node-disjoint. For this, we will go through the layers $i = 1, 2, \ldots, \ell$ in this order and replace $\mathcal{T}_i$ by an adjusted set of trees $\mathcal{T}'_i$. We will construct these trees so that at each step $i \in [\ell]$ all trees from $\cup_{j\in[i]}\mathcal{T}'_j$ are pairwise disjoint. Furthermore, at step i the radius of any tree from $\cup_{j\in[i]}\mathcal{T}'_j$ will be bounded from above by $(2i-1)r + \sum_{j\in[i]} h_j$. Finally, our construction ensures that in the end, the trees in $\cup_{i\in[\ell]}\mathcal{T}'_i$ cover all nodes in V. Hence, these trees form a feasible solution to the connected k-center problem with disjoint clusters with the desired radius.

We set $\mathcal{T}'_1 = \mathcal{T}_1$. Then for $i = 1$, the desired properties are satisfied because the trees on layer 1 are pairwise disjoint and have a radius of at most $r + h_1$. Now assume that the properties are true for some i and let us discuss how to ensure them also for $i + 1$. We start with $\mathcal{T}'_{i+1} = \emptyset$ and add trees to it one after another. Consider an arbitrary tree $T \in \mathcal{T}_{i+1} = (V', E')$ with center c and let $V^* \subseteq V'$ denote the nodes that also occur in some tree $T' \in \mathcal{T}'_j$ for some $j \in [i]$. Observe that any node from V^* can be contained in at most one such tree T' because by the induction hypothesis all trees in $\cup_{j\in[i]}\mathcal{T}'_j$ are pairwise disjoint. If V^* is empty then the tree T is disjoint from all trees in $\cup_{j\in[i+1]}\mathcal{T}'_j$ and does not need to be adjusted. In this case we simply add it to $\mathcal{T}'_{i+1}$.

If V^* contains only a single node v then we merge the tree T with the unique tree T' from $\mathcal{T}'_j$ for some $j \leq i$ that also contains node v, i.e., we replace T' by $T \cup T'$ in $\mathcal{T}'_j$. Tree T' has a radius of at most $(2i-1)r + \sum_{j\in[i]} h_j$. Since the diameter of T is at most $2r + h_{i+1}$, the radius of the union of T and T' with respect to the center of T' is at most (see Figure 5)

$$(2r + h_{i+1}) + (2i-1)r + \sum_{j\in[i]} h_j = (2(i+1)-1)r + \sum_{j\in[i+1]} h_j. \tag{1}$$

Now consider the case that V^* contains more than one node. In this case we cannot simply merge T with some tree from $\cup_{j\in[i]}\mathcal{T}'_j$ because the resulting tree would not be disjoint from the other trees. We also cannot merge all trees that contain nodes from V' into a single cluster because the radius of the resulting cluster could be too large. Instead we split the tree T into multiple components and we merge these components separately with different trees from $\cup_{j\in[i]}\mathcal{T}'_j$. For each node $v \in V^*$ that is not the root c of T we consider the path from c to v and let e denote the last edge on this path (i.e., the edge leading to v). We remove edge e from the tree T and thereby split the tree T into two components. Since we do this for every node from $V^* \setminus \{c\}$, the tree T will be split into $|V^* \setminus \{c\}| + 1$ pairwise disjoint connected components. Each of these components that does not contain the root c contains

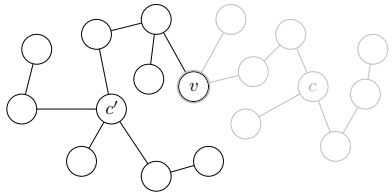


Figure 5 This figure shows the tree T' with center c' in black and the tree T with center c in gray. These trees have node v in common. When T and T' are merged into a single tree, the radius of this new tree with respect to c' is larger than the radius of T' by at most the diameter of T.

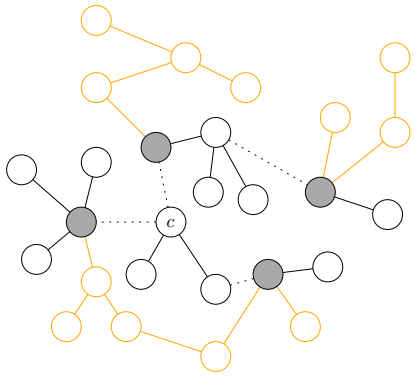


Figure 6 This figure shows the tree T in black. The nodes in V^* are marked gray and the edges that are removed from T are shown dotted. The orange trees depict the trees on lower layers that contain the nodes from V^* and with which the corresponding components are merged.

exactly one node from V^*. Hence, for each of these components there is a unique tree from $\cup_{j\in[i]}\mathcal{T}'_j$ from which it is non-disjoint. We merge every component with the tree from which it is non-disjoint (see Figure 6). In the component that contains the root, only the root might belong to V^*. If this is the case, we merge it with the unique tree from $\cup_{j\in[i]}\mathcal{T}'_j$ from which it is non-disjoint. Otherwise, we add this component to $\mathcal{T}'_{i+1}$. Since T has a diameter of at most $2r + h_{i+1}$, the same is true for each of the components. By the induction hypothesis, each tree from $\cup_{j\in[i]}\mathcal{T}'_j$ has a radius of at most $(2i-1)r + \sum_{j\in[i]} h_j$. Hence, as in (1), the radius of the merged clusters is bounded from above by $(2(i+1)-1)r + \sum_{j\in[i+1]} h_j$. ◀

▶ **Corollary 12.** *If there exists a polynomial-time algorithm that computes for any metric (C, d) and any r an r-well-separated partition with ℓ layers and parameters $(h_1, \ldots, h_\ell)$ then there exists an approximation algorithm for the connected k-center problem with disjoint clusters that achieves an approximation factor of $4\ell - 2 + 2\sum_{i=1}^{\ell} h_i/r$.*

Proof. To obtain the desired approximation factor, we first determine the smallest r for which Algorithm 2 returns a center set C with $|C| \leq k$. Due to Lemma 8, this radius r will be at most $2r^*$, where r^* denotes the radius of an optimal connected k-clustering with non-disjoint clusters. Let r^*_D denote the radius of an optimal connected k-clustering with disjoint clusters. Then $r^*_D \geq r^* \geq r/2$. According to Lemma 11, the polynomial-time algorithm for computing an r-well-separated partition can then be used to compute a connected k-clustering with disjoint clusters and radius at most $(2\ell-1)r + \sum_{i\in[\ell]} h_i$. The approximation factor of this k-clustering is

$$\frac{(2\ell-1)r + \sum_{i\in[\ell]} h_i}{r^*_D} \leq \frac{(2\ell-1)r + \sum_{i\in[\ell]} h_i}{r/2} = 4\ell - 2 + 2\sum_{i\in[\ell]} \frac{h_i}{r}.$$ ◀

The same algorithm that we developed in this sections for the connected k-center problem can also be used for the connected k-diameter problem without any modifications. Only the analysis of the approximation factor needs to be adapted slightly.

Lemma 8 is changed as follows.

▶ **Lemma 13.** *Let r^* denote the diameter of an optimal connected k-diameter clustering with non-disjoint clusters. For $r \geq r^*$, Algorithm 2 computes a center set C with $|C| \leq k$.*

Observe that the diameter of the clusters T_c that are computed by Algorithm 2 for some r can be at most $2r$.

A straightforward adaption of Lemma 11 yields the following result.

▶ **Lemma 14.** *Consider an instance $(G = (V, E), M = (V, d), k)$ of the connected k-diameter problem and assume that Algorithm 2 computes a center set $C \subseteq V$ with $|C| \leq k$ for some radius r. Furthermore, let an r-well-separated partition of C with ℓ layers and parameters $(h_1, \ldots, h_\ell)$ be given. Then we can efficiently find a feasible solution for the connected k-diameter problem with disjoint clusters with diameter at most $(4\ell - 2)r + h_1 + 2\sum_{i=2}^{\ell} h_i$.*

Overall we obtain the following corollary.

▶ **Corollary 15.** *If there exists a polynomial-time algorithm that computes for any metric (C, d) and any r an r-well-separated partition with ℓ layers and parameters $(h_1, \ldots, h_\ell)$ then there exists an approximation algorithm for the connected k-diameter problem with disjoint clusters that achieves an approximation factor of $4\ell - 2 + h_1/r + 2\sum_{i=2}^{\ell} h_i/r$.*

4.2 Finding well-separated partitions

With the discussion above, finding a good approximation algorithm is reduced to finding an efficient algorithm for computing an r-well-separated partition with small L and few layers. For general metrics, we present an efficient algorithm that computes a well-separated partition with $L = O(r \cdot \log k)$ and $\ell = O(\log k)$. This yields a clustering of radius and diameter $O(r \cdot \log^2 k)$. Details can be found in the proof of Theorem 18 in the next section. We give better results for computing well-separated partitions for L_p-metrics and metric spaces with bounded doubling dimension in Theorem 20 and Theorem 23. Overall, we get the following results.

▶ **Theorem 16.** *There exists an $O(\log^2 k)$-approximation algorithm for the connected k-center problem with disjoint clusters and for the connected k-diameter problem with disjoint clusters. The approximation ratio improves*

- *to $O(2^{3 \cdot \dim(M)})$ if the metric space has bounded doubling dimension $\dim(M)$, and*
- *to $O(d \cdot 2^d)$ if the distance is an L_p-metric in $\mathbb{R}^d$.*

It is an intriguing question if better well-separated partitions exist for general metrics and for the special metrics that we have considered. By our framework, better partitions would immediately give rise to better approximation factors.

We prove in the full version a lower bound of $\Omega(\log \log k)$ for our algorithmic framework. To be precise, we construct an instance in a general metric space together with a set of k centers C that could be produced by the algorithm `GreedyClustering` such that even the optimal disjoint solution with centers C is worse than the optimal disjoint solution for arbitrary centers by a factor of $\Omega(\log \log k)$. Hence, to prove a constant-factor approximation in general metric spaces, one cannot rely on the centers chosen by `GreedyClustering`.

4.2.1 Well-separated partitions in general metrics

According to Corollaries 12 and 15, we only need to find an efficient algorithm for computing an r-well-separated partition to obtain an approximation algorithm for the connected k-center and k-diameter problem.

Algorithm 3 computes an r-well separated partition layer by layer. For each layer it creates the groups in a greedy fashion: At the beginning of a layer i, the set U' of all nodes that are not assigned to previous layers is considered. The goal is to assign as many of these nodes to the current layer i as possible. For this, we start with an arbitrary node $u \in U'$ that is not assigned to any previous layer and we create a group around u. First the group consists only of u itself. Then we iteratively augment the group by adding all nodes to the group that have a distance of at most $2r$ from some node that already belongs to the group. We repeat this augmentation step multiple times one after another. We stop when the number of new nodes that join the group is smaller than twice the number of nodes that have already been added to the group for the first time. Then the group around u is finished and added to layer i. All nodes in the group are removed from U'. Furthermore, we also remove all nodes that have a distance of at most $2r$ from this group from U'. These nodes have to be assigned to other layers that are created later to ensure property (iii) in Definition 10. As long as U' is not empty, we repeat the process to create another group on layer i. The pseudocode is shown as Algorithm 3.

Algorithm 3 PartitionGeneralMetric$((C, d), r)$.

Input: metric (C, d), radius r
$U \leftarrow C$; // nodes that still have to be assigned
$i \leftarrow 0$;
while $U \neq \emptyset$ **do**
  $i \leftarrow i + 1$; // start a new layer
  $j \leftarrow 0$;
  $U' \leftarrow U$; // nodes that could still be assigned on i-th layer
  while $U' \neq \emptyset$ **do**
    $j = j + 1$; // create a new group in i-th layer
    select a node $u \in U'$, $C_{i,j} \leftarrow \{u\}$ and $N_0(u) \leftarrow \{u\}$;
    $U' \leftarrow U' \setminus \{u\}$;
    $U \leftarrow U \setminus \{u\}$;
    $s = 1$;
    while $s \neq 0$ *and* $U' \neq \emptyset$ **do**
      $N_s(u) \leftarrow \{x \in U' \mid \exists v \in N_{s-1}(u) : d(v, x) \leq 2r\}$;
      // nearby nodes of nodes $C_{i,j}$ in U'
      if $|N_s(u)| \geq 2 \cdot |C_{i,j}|$ **then**
        $C_{i,j} \leftarrow C_{i,j} \cup N_s(u)$; // add nearby nodes to $C_{i,j}$
        $U' \leftarrow U' \setminus N_s(u)$;
        $U \leftarrow U \setminus N_s(u)$;
        $s = s + 1$;
      else
        $U' \leftarrow U' \setminus N_s(u)$; // nearby nodes cannot be on i-th layer
        $s = 0$; // end group of node u
Output: $\{C_{1,1}, C_{1,2}, \ldots\}, \{C_{2,1}, C_{2,2}, \ldots\}, \ldots$

▶ **Lemma 17.** *Let (C, d) be an arbitrary metric with $k := |C|$ and $r > 0$. Let $\ell = 1 + \lfloor \log_{\frac{3}{2}}(k) \rfloor$ and $h = 4r\lfloor \log_3 k \rfloor$. The output of Algorithm 3 is an r-well-separated partition with at most ℓ layers and parameters $(h, \dots, h)$.*

Proof. Let $\{C_{1,1}, \dots, C_{1,\ell_1}\}, \{C_{2,1}, \dots, C_{2,\ell_2}\}, \dots, \{C_{\ell,1}, \dots, C_{\ell,\ell_\ell}\}$ denote the output of Algorithm 3. The algorithm ensures that every point from C is contained in exactly one group $C_{i,j}$ because when nodes are deleted from U in Line 18 they have been added to $C_{i,j}$ in Line 16. Furthermore U' is always a subset of U and so no node can be assigned to multiple clusters. Furthermore, Lines 14 and 21 ensure that nodes in different groups of the same layer are more than $2r$ apart. This shows that the properties (i), (ii), and (iii) in Definition 10 are satisfied.

Next we show property (iv) that the maximum diameter of every group is h. As long as the number of nearby nodes in $N_s(u)$ is at least twice the number of the previously grouped nodes in $\cup_{t=1}^{s-1} N_t(u)$, we add these nearby nodes to the current group. As long as this is true we have

$$|N_s(u)| \geq 2 \cdot \sum_{t=0}^{s-1} |N_t(u)|.$$

Together with $|N_0(u)| = 1$, this implies $|\cup_{t=1}^{s} N_t(u)| \geq 3^s$ for every s by a simple inductive argument. Since this set cannot contain more than $k = |C|$ nodes, we have $C_{i,j} = \bigcup_{s=1}^{h} N_s(u)$ for some $h \leq \lfloor \log_3 k \rfloor$. For any $s \geq 1$, any node in $N_s(u)$ has a distance of at most $2r$ from some node in $N_{s-1}(u)$. Since u is the only node in $N_0(u)$, this implies that any node has a distance of at most $2rh$ from u. Hence, the diameter of every group is at most $4rh \leq 4r\lfloor \log_3 k \rfloor$. This shows property (iv) in Definition 10.

Now it only remains to bound the number of layers of the partition. When a new layer is started, U' is set to U, the set of yet unassigned nodes in Line 6. When a group is formed then its current neighbors $N_s(u)$ get removed from U' in Line 21. These are exactly the nodes that do not get assigned to the current layer and have to be assigned to other layers afterwards. Since line 21 is only reached if $|N_s(u)|$ is smaller than twice $|C_{i,j}|$, at least one third of the initially unassigned nodes get assigned to groups on the current layer and at most two thirds are postponed to other layers afterwards. This implies that after ℓ layers, there are no more than $(\frac{2}{3})^\ell \cdot k$ nodes left to be assigned. Hence, the number of layers cannot be more than $1 + \lfloor \log_{\frac{3}{2}}(k) \rfloor$. ◀

Based on Corollaries 12 and 15, it is now easy to prove the following theorem.

▶ **Theorem 18.** *There exists an $O(\log^2 k)$-approximation algorithm for the connected k-center problem and for the connected k-diameter problem with disjoint clusters.*

Proof. According to Lemma 17, one can efficiently compute for any metric an r-well-separated partition with at most ℓ layers and parameters $(h, \dots, h)$ for $\ell = 1 + \lfloor \log_{\frac{3}{2}}(k) \rfloor = O(\log k)$ and $h = 4r\lfloor \log_3 k \rfloor = O(r \cdot \log k)$.

By Corollary 12 this implies that we can efficiently find a solution for the connected k-center problem with disjoint clusters with approximation factor

$$4\ell - 2 + 2\sum_{i=1}^{\ell} h/r = O(\ell + \ell h/r) = O(\log k + \log^2 k) = O(\log^2 k).$$

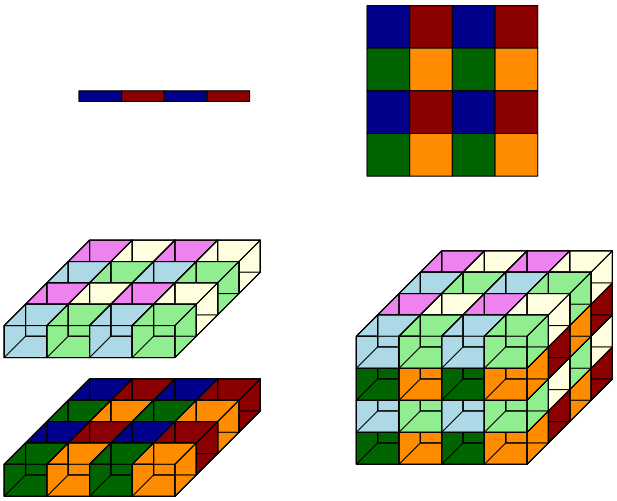


Figure 7 In the upper row, colorings for $d = 1$ and $d = 2$ are shown. In the lower row on the right, a coloring for $d = 3$ is shown. It is composed of alternatingly using the 2-dimensional colorings shown on the left.

By Corollary 15, it also implies that we can efficiently find a solution for the connected k-diameter problem with disjoint clusters with approximation factor

$$4\ell - 2 + h_1/r + 2\sum_{i=2}^{\ell} h_i/r = O(\ell + \ell h/r) = O(\log k + \log^2 k) = O(\log^2 k). \quad \blacktriangleleft$$

4.2.2 Well-separated partitions in Euclidean metrics

In this section, we study how to compute an r-well-separated partition if the metric is an L_p-metric in the d-dimensional space $\mathbb{R}^d$ for some $p \in \{1, 2, \ldots, \infty\}$.

▶ **Lemma 19.** *For any L_p-metric in $\mathbb{R}^d$, an r-well-separated partition with 2^d layers and parameters $(h, \ldots, h)$ with $h = 3d^{1/p}r$ can be computed in polynomial time.*

Proof. First we partition the space $\mathbb{R}^d$ into d-dimensional hypercubes with side length $3r$. These hypercubes are chosen such that they are pairwise disjoint and that they cover the entire space. Then we color these hypercubes such that no two neighboring hypercubes get the same color where also diagonal neighbors are taken into account (see Figure 7). Based on this coloring we create the following r-well-separated partition: each color corresponds to one layer of the partition and within a layer all nodes that belong to the same hypercube form a group. Since the distance of two hypercubes of the same color is at least $3r$, property (iii) of Definition 10 is satisfied. Properties (i) and (ii) are satisfied because the hypercubes partition the space $\mathbb{R}^d$. Finally, the diameter of any of the hypercubes is bounded from above by $(\sum_{i=1}^{d}(3r)^p)^{1/p} = 3d^{1/p}r$, which also proves property (iv).

It remains to bound the number of layers, i.e., the number of different colors necessary to color the hypercubes. One can prove by induction that 2^d colors are sufficient. For $d = 1$, one simply colors the hypercubes alternatingly with two different colors. For $d \geq 2$, we first pick two different colorings of $\mathbb{R}^{d-1}$ with 2^{d-1} colors each such that the two colorings do not have a color in common. Then we color the hypercubes in $\mathbb{R}^d$ by alternatingly using one of the two $(d-1)$-dimensional colorings. This way, we obtain a coloring of the hypercubes in $\mathbb{R}^d$ with 2^d colors (see Figure 7). ◀

Based on Corollaries 12 and 15, it is now easy to prove the following theorem.

▶ **Theorem 20.** *For any L_p-metric in $\mathbb{R}^d$, there exists an $O(d \cdot 2^d)$-approximation algorithm for the connected k-center problem and for the connected k-diameter problem with disjoint clusters.*

Proof. According to Lemma 19, we can efficiently compute an r-well-separated partition with 2^d layers and parameters $(h, \dots, h)$ for $h = 3d^{1/p}r$.

By Corollary 12 this implies that we can efficiently find a solution for the connected k-center problem with disjoint clusters with approximation factor

$$4\ell - 2 + 2\sum_{i=1}^{\ell} h/r = O(d \cdot 2^d).$$

By Corollary 15, it also implies that we can efficiently find a solution for the connected k-diameter problem with disjoint clusters with approximation factor

$$(4\ell - 2) + h_1/r + 2\sum_{i=2}^{\ell} h_i/r = O(d \cdot 2^d).$$ ◀

4.2.3 Well-separated partitions in metrics with small doubling dimension

In this section, we study how to compute an r-well-separated partition if the metric has constant doubling dimension. This generalizes Lemma 19 for Euclidean spaces.

▶ **Definition 21** (doubling dimension)**.** *The* doubling constant *of a metric space $M = (X, d)$ is the smallest number k such that for all $x \in X$ and $r > 0$, the ball $B_r(x) := \{y \in X \mid d(x, y) \leq r\}$ can be covered by at most k balls of radius $r/2$, i.e.,*

$$\forall x \in X : \forall r > 0 : \exists Y \subseteq X, |Y| \leq k : B_r(x) \subseteq \bigcup_{y \in Y} B_{r/2}(y).$$

The doubling dimension *of M is defined as* $\dim(M) = \lceil \log_2 k \rceil$.

▶ **Lemma 22.** *For any metric $M = (X, d)$ with doubling dimension $\dim(M)$, an r-well-separated partition with $2^{3 \cdot \dim(M)}$ layers and parameters $(h, \dots, h)$ with $h = 2r$ can be computed in polynomial time.*

Proof. First we partition X greedily into balls of radius r: As long as not all points of X are covered, we choose arbitrarily an uncovered point x from X and put x into one group together with all uncovered points that have a distance of at most r from x. This way, we get a partition of X into groups with radius at most r.

Next, we try to reduce the number of groups by local improvements. We say that two groups are neighboring if the distance of their centers is at most $4r$. As long as there is a group that has at least $2^{3 \cdot \dim(M)}$ neighbors, we replace this group and its neighbors by $2^{3 \cdot \dim(M)}$ groups as follows: Let x be a center of a group that has at least $2^{3 \cdot \dim(M)}$ neighbors, and let the centers of the neighbors be $Y \subseteq X$. Since x has a distance of at most $4r$ from all centers in Y, we have

$$B_r(x) \cup_{y \in Y} B_r(y) \subseteq B_{5r}(x).$$

By definition of the doubling dimension, the ball $B_{5r}(x)$ can be covered by $2^{\dim(M)}$ balls of radius $5r/2$, each of these can be covered by $2^{\dim(M)}$ balls of radius $5r/4 < 2r$, and each of these can be covered by $2^{\dim(M)}$ balls of radius $5r/8 < r$. Hence, the points in

$B_r(x) \cup_{y \in Y} B_r(y)$ can be covered by $2^{3 \cdot \dim(M)}$ balls of radius r. In our partition, we replace the groups around x and around $y \in Y$ by the groups induced by these balls. Since this reduces the number of groups by at least one, after a linear number of these local improvements, no local improvement is possible anymore, i.e., every group has less than $2^{3 \cdot \dim(M)}$ neighbors.

We have obtained a partition of X into groups, where each group has a radius of at most r. Furthermore, each group has a center and two groups are neighbors if their centers have a distance of at most $4r$. Furthermore, every group has less than $2^{3 \cdot \dim(M)}$ neighbors. The groups will form the groups in the r-well-separated partition. Since points from groups that are not neighbored have a distance of more than $2r$, two groups that are not neighbored can be on the same layer of the partition without contradicting property (iii) from Definition 10. The diameter of each group is at most $h = 2r$. It remains to distribute the groups to the different layers of the partition. For this we find a coloring of the groups such that neighboring groups get different colors. The neighborhood defines implicitly a graph with the groups as vertices with degree at most $2^{3 \cdot \dim(M)} - 1$. Any such graph can be colored with $2^{3 \cdot \dim(M)}$ colors by a greedy algorithm. Now we assign the groups according to the colors to different layers, resulting in an r-well-separated partition with at most $2^{3 \cdot \dim(M)}$ layers. ◀

Based on Corollaries 12 and 15, it is now easy to prove the following theorem.

▶ **Theorem 23.** *For any metric $M = (X, d)$ with doubling dimension $\dim(M)$, there exists an $O(2^{3 \cdot \dim(M)})$-approximation algorithm for the connected k-center problem and for the connected k-diameter problem with disjoint clusters.*

Proof. According to Lemma 19, we can efficiently compute an r-well-separated partition with $2^{3 \cdot \dim(M)}$ layers and parameters $(h, \dots, h)$ for $h = 2r$.

By Corollary 12 this implies that we can efficiently find a solution for the connected k-center problem with disjoint clusters with approximation factor

$$4\ell - 2 + 2 \sum_{i=1}^{\ell} h/r = O(2^{3 \cdot \dim(M)}).$$

By Corollary 15, it also implies that we can efficiently find a solution for the connected k-diameter problem with disjoint clusters with approximation factor

$$4\ell - 2 + h_1/r + 2 \sum_{i=2}^{\ell} h_i/r = O(2^{3 \cdot \dim(M)}).$$ ◀

5 Conclusions

We studied the connected k-center and k-diameter problem and proved several new results on the approximability of different variants of these problems. In particular, we developed a general framework to obtain approximation algorithms for the disjoint versions of these problems that relies on the existence of well-separated partitions. While we obtain constant-factor approximations for L_p-metrics in constant dimension and metrics with constant doubling dimension, our general upper bound is $O(\log^2 k)$. Since all our lower bounds are constant, an obvious open question is to close the gaps between the upper and lower bounds. One possibility to approach this would be to derive better well-separated partitions. However, we also show that with our approach no bound better than $O(\log \log k)$ can be shown.

References

1. Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms (SODA)*, pages 642–651. ACM/SIAM, 2001.
2. Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. LP rounding for k-centers with non-uniform hard capacities. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 273–282. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.63.
3. Hu Ding and Jinhui Xu. A unified framework for clustering constrained data without locality property. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1471–1490. SIAM, 2015.
4. Rong Ge, Martin Ester, Byron J. Gao, Zengjian Hu, Binay K. Bhattacharya, and Boaz Ben-Moshe. Joint cluster analysis of attribute data and relationship data: The connected k-center problem, algorithms and applications. *ACM Trans. Knowl. Discov. Data*, 2(2):7:1–7:35, 2008. doi:10.1145/1376815.1376816.
5. Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.
6. Neelima Gupta, Aditya Pancholi, and Yogish Sabharwal. Clustering with internal connectedness. In *Proc. of 5th Intl. Workshop on Algorithms and Computation (WALCOM)*, volume 6552 of *Lecture Notes in Computer Science*, pages 158–169. Springer, 2011. doi:10.1007/978-3-642-19094-0_17.
7. Dorit S. Hochbaum. When are np-hard location problems easy? *Ann. Oper. Res.*, 1(3):201–214, 1984. doi:10.1007/BF01874389.
8. Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986.
9. Simon J. Holgate, Andrew Matthews, Philip L. Woodworth, Lesley J. Rickards, Mark E. Tamisiea, Elizabeth Bradshaw, Peter R. Foden, Kathleen M. Gordon, Svetlana Jevrejeva, and Jeff Pugh. New data systems and products at the permanent service for mean sea level. *Journal of Coastal Research*, 29:493–504, 2013.
10. Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215, 1979.
11. Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 646–659, 2018.
12. Zhung-Xun Liao and Wen-Chih Peng. Clustering spatial data with a geographic constraint: exploring local search. *Knowl. Inf. Syst.*, 31(1):153–170, 2012. doi:10.1007/s10115-011-0402-8.
13. Permanent Service for Mean Sea Level (PSMSL). Tide gauge data, retrieved on 03 February 2022 from http://www.psmsl.org/data/obtaining/.