Connected k-Median with Disjoint and Non-Disjoint Clusters

Jan Eube
University of Bonn, Germany

Kelin Luo
University at Buffalo, NY, USA

Dorian Reineccius
Deutsches Zentrum für Luft- und Raumfahrt, Bonn, Germany

Heiko Röglin
University of Bonn, Germany

Melanie Schmidt
Heinrich-Heine Universität Düsseldorf, Germany

Abstract

The connected k-median problem is a constrained clustering problem that combines distance-based k-clustering with connectivity information. The problem allows to input a metric space and an unweighted undirected connectivity graph that is completely unrelated to the metric space. The goal is to compute k centers and corresponding clusters such that each cluster forms a connected subgraph of G, and such that the k-median cost is minimized.

The problem has applications in very different fields like geodesy (particularly districting), social network analysis (especially community detection), or bioinformatics. We study a version with overlapping clusters where points can be part of multiple clusters which is natural for the use case of community detection. This problem variant is $\Omega(\log n)$-hard to approximate, and our main result is an $\mathcal{O}(k^2 \log n)$-approximation algorithm for the problem. We complement it with an $\Omega(n^{1-\epsilon})$-hardness result for the case of disjoint clusters without overlap with general connectivity graphs, as well as an exact algorithm in this setting if the connectivity graph is a tree.

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

Keywords and phrases Clustering, Connectivity constraints, Approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.63

Related Version *Full Version*: `https://arxiv.org/abs/2507.02774` [7]

Funding This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 390685813; 459420781 and by the Lamarr Institute for Machine Learning and Artificial Intelligence lamarr-institute.org.

1 Introduction

Clustering problems, like the k-center and the k-median problem, are classical optimization problems that have been widely studied both in theory and practice. In a typical center-based clustering problem, the input consists of a given set of data points in some metric space and a positive integer k. The goal is to partition the data points into k groups (clusters) and to find a center for each of these groups so as to optimize some objective: In the k-center problem, the goal is to minimize the maximal distance of any point to the center of its cluster; in the k-median problem, the goal is to minimize the sum of the distances of the points to their corresponding cluster centers.

Both the k-center and the k-median problem are NP-hard but various constant-factor approximation algorithms are known. For the k-center problem, there are 2-approximation algorithms [9, 13], and this is best possible [14]. The best known approximation algorithm for the k-median problem achieves a $(2.675 + \epsilon)$-approximation [2], while it is known that no approximation factor better than $1 + 2/e \approx 1.735$ can be achieved, unless P=NP [10].

Beyond these vanilla clustering problems, there has been a lot of interest in variants of clustering problems motivated by specific applications. These variants often introduce additional constraints. For example, in capacitated clustering problems, constraints are added that limit the number of data points that can be assigned to each cluster [17]. There are also models to address outliers in data sets [18], and in the last decade, different variants of fair clustering problems have been studied [4, 20, 21]. These are just a few examples of the numerous variants of clustering problems that have been studied to address diverse challenges in different application domains. Each of these variants requires specialized algorithms, often inspiring novel methodologies and insights in the broader field of clustering and data analysis.

In this paper, we study a *connectivity* constraint. The basic idea of this constraint is that clusters have to correspond to connected subgraphs of a given undirected and unweighted *connectivity graph*. This graph is independent of the metric that defines the distances between the input points.

This model has been studied in different contexts. One line of research, see Ge et al. [8], considers the connected k-center problem to allow for a joint analysis of two type of data: attribute data and relationship data. Attribute data can be used for clustering by finding a suitable metric distance function on the attribute vectors and then applying a popular clustering method like k-means or k-center. Relationship data arises in the context of social networks or other network structures and gives information on pairwise relationships. Clustering on relationship data can be done via methods like correlation clustering. Both types of data thus allow for well studied clustering methods, but the focus by Ge et al. [8] is a joint analysis. Connected clustering gives one option to combine relationship and attribute data by defining a constraint based on the relationships and then optimizing over the attribute vector distances.

More precisely, connected clustering expects as input a set of points in a metric space (stemming from the attribute data) and an unweighted graph G (representing the relationship data), and the goal is to cluster with respect to the metric distance function but under the constraint that clusters are connected in G. We call G the *connectivity graph* in the following.

Ge et al. [8] study the connected k-center problem. For this problem, the goal is to compute k centers $c_1, \ldots, c_k$ and corresponding clusters $C_1, \ldots, C_k$ such that all C_i are connected in the connectivity graph G, and such that the largest radius is minimized (the radius of C_i is $\max_{x \in C_i} d(x, c_i)$). They provide an optimal algorithm via dynamic programming for the special case of a tree connectivity graph which was rediscovered by Drexler et al. [6] in a later publication. Ge et al. also present algorithms for the general case with unknown approximation guarantee[1] and provide an experimental evaluation.

The best-known approximation guarantee for the connected k-center problem is a $O(\log^2 k)$-approximation given by Drexler et al. [6]. For Euclidean metrics with constant dimension and metrics with constant doubling dimension, they provide an $O(1)$-approximation algorithm. Their paper studies the problem from a very different context: Motivated by an application from geodesy and sea level analysis, they use the distance function to model differences in tide gauge measurements and the connectivity graph to model geographic closeness, and their modelling results in the same problem definition as Ge et al. [8].

[1] There is a proof in the paper that claims a guarantee of 6, but [6] gives a counterexample.

In this paper, we study the connected k-*median* problem, i.e., the related problem with the k-median objective instead of the k-center objective. We give a formal definition now.

▶ **Definition 1** (Connected k-Median Problem (disjoint))**.** *In an instance of the (disjoint) connected k-median problem, we are given a set V of points, a metric $d : V \times V \to \mathbb{R}_{\geq 0}$ on V, a positive integer $k \geq 2$, and an unweighted and undirected connectivity graph $G = (V, E)$. A feasible solution is a partition of V into k disjoint clusters $\mathcal{V} = \{V_1, \dots, V_k\}$ with corresponding centers $C = \{c_1, c_2, \dots, c_k\} \subseteq V$ which satisfies that for every $i \in [k] := \{1, \dots, k\}$ the subgraph of G induced by V_i is connected and $c_i \in V_i$. The* connected k-median problem *aims to find a feasible solution $(C, \mathcal{V})$ that minimizes*

$$\Phi_{med}(C, \mathcal{V}) := \sum_{i \in [k]} \sum_{v \in V_i} d(c_i, v).$$

The connected k-median has also appeared in different contexts. For example, Validi et al. [22] study connected k-median for a districting problem. Here the goal is to partition given administrative or geographic units into districts. The clusters should usually correspond to connected areas on the map which can be formulated via introducing centers for each unit and a graph to model which units are neighbors on the map. The k-median objective is one way to model compactness of districts. Validi et al. [22] study various districting methods, and connected k-median is one of them. They solve it via an integer linear programming approach, i.e., to optimality but with exponential worst-case running time. A related variant is the connected k-means problem, which for example has been considered in [19] where a heuristic approach to minimize it is employed.

Gupta et al. [12] study k-median/k-means from the point of view of approximation algorithms. They consider the special case where the number of clusters is $k = 2$ and the connectivity graph is star-like (there exists a point that is connected to all other points). The paper shows that even this case is $\Omega(\log n)$-hard to approximate for both problems and proposed an $O(\log n)$-approximation algorithm for this special case. There is nothing known about the general case, and our work is motivated by the following question:

Can the connected k-median problem be approximated for general connectivity graphs?

Unfortunately, we quickly arrive at the conclusion that no reasonable approximation guarantee is possible if $P \neq NP$ (see Thm 4). We do derive a dynamic programming based exact algorithm for trees that runs in time $O(n^2k^2)$ (see Thm 6), thus extending the landscape for the problem, but the answer to our main question is still negative.

However, our inapproximability proof depends on a subtlety of the problem definition: The clusters V_i have to be disjoint. This is a natural assumption in clustering, and in clustering without constraints, clusters can be assumed to be disjoint (adding a point to multiple clusters will in general only increase the objective). But here, if we allow for overlapping clusters, then satisfying connectivity can become easier because we can add critical vertices like articulation points to multiple clusters. This motivates the definition of *connected k-median with non-disjoint clusters.*

▶ **Definition 2** (Disjoint vs. non-disjoint clusters)**.** *In the* connected k-median problem with disjoint clusters*, we require the clusters $V_1, \dots, V_k$ to be pairwise disjoint. In the* connected k-median problem with non-disjoint clusters*, the clusters are allowed to overlap and we only require $V = \bigcup_{i \in [k]} V_i$.*

One may note that if a vertex appears in multiple clusters it will also contribute to the objective function multiple times. Thus also in the non-disjoint variant, we are incentivized to avoid assigning vertices unnecessarily often.

The non-disjoint version has also been studied previously. For example, Drexler et al. [6] study it for k-center and give a 2-approximation for this case. From the application side, it depends on the interpretation whether disjoint or non-disjoint clusters are preferred. In districting, units should certainly not be in multiple clusters. But for the application of community detection in social networks the modeling is natural since individuals can belong to different groups. We thus study the following question.

Can the connected k-median problem with non-disjoint clusters be approximated for general connectivity graphs?

The proof by Gupta et al. [12] gives a lower bound of $\Omega(\log n)$ for the disjoint case. For the sake of completeness, we provide in the full version of this paper an adapted version that also works if the clusters may overlap [7]. Our main result is a $O(k^2 \log n)$-approximation for the connected k-median problem with non-disjoint clusters.

Before we summarize our results, we need one more definition. For a clustering problem with constraints, the corresponding *assignment* version of the problem asks for assigning points to given centers while respecting the constraints and minimizing the cost. In our case, we get the following problem.

▶ **Definition 3** (Assignment version)**.** *In the* assignment version *of the connected k-median problem, we assume that k centers, denoted as $C \subseteq V$, are given and the goal is to find a feasible assignment of the points in V to the centers C such that the k-median costs are minimized. This version can be defined both for the disjoint and the non-disjoint variant of the connected k-median problem.*

Without a constraint, the assignment version of the k-median problem can be solved optimally by assigning every point to its closest center. For clustering with constraints, assignment problems can be difficult: For example, the assignment problem for fair k-center is NP-hard [1], and the connected k-center with disjoint clusters is also NP-hard [6] (both problems are even hard to approximate better than 3). We will establish an even higher lower bound for the assignment problem in the connected k-median setting.

1.1 Our results

Table 1 Overview of the results presented in this work.

(a) Our results in the non-disjoint setting

	Assignment version	Regular version
Approximation	$O(k \log(n))$ (Theorem 8)	$O(k^2 \log(n))$ (Theorem 9)
Hardness($k = 2$)	$\Omega(\log(n))$ (Theorem 7)	

(b) Our results in the disjoint setting. All results also hold for the assignment version.

	General Graph	Tree graph
Approximation	$O(n \log^2(k))$ (Theorem 5)	1 (Theorem 6)
Hardness($k = 2$)	$\Omega(n^{1-\epsilon})$ (Theorem 4)	–

We prove new approximation and hardness results for the connected k-median problem, both for the disjoint and the non-disjoint variant. Table 1 provides a brief overview of them. First we establish a strong hardness of approximation result for the connected k-median problem with disjoint clusters, which even holds in the special case that $k = 2$.

▶ **Theorem 4.** *For any $\epsilon > 0$, there is no polynomial-time $O(n^{1-\epsilon})$-approximation algorithm for the connected k-median problem with disjoint clusters even if $k = 2$, unless $P = NP$. For the assignment version, this lower bound also holds.*

For completeness we point out that one can almost match this lower bound using the connected k-center algorithm by Drexler et al. [6]:

▶ **Theorem 5.** *For the disjoint connected k-median problem, there is a polynomial-time algorithm with approximation factor $O(n \log^2 k)$.*

Another important special case that has also been studied for the connected k-center problem is the case that the connectivity graph is a tree. For this special case, we obtain a polynomial-time algorithm based on dynamic programming.

▶ **Theorem 6.** *When the connectivity graph G is a tree, then the connected k-median problem with disjoint clusters can be solved optimally in time $O(n^2k^2)$. This is true even if the distances are not a metric or the centers are given.*

Motivated by applications like community detection, in which the clusters do not necessarily have to be disjoint, we also study the connected k-median problem with non-disjoint clusters. This problem is $\Omega(\log n)$-hard to approximate even if $k = 2$.

▶ **Theorem 7.** *There is a constant $c > 0$ such that for all sufficiently large n, the non-disjoint connected k-median problem cannot be approximated within a factor of $c \cdot \log n$ unless P=NP, even when $k = 2$. The same is true for the assignment variant.*

Our main result are approximation algorithms for the clustering and the assignment problem whose approximation factors match this lower bound in terms of n.

▶ **Theorem 8.** *For the assignment version of the non-disjoint connected k-median problem, there is a polynomial-time algorithm with approximation factor $O(k \log n)$. This is true even if the distances are not a metric.*

▶ **Theorem 9.** *For the non-disjoint connected k-median problem, there is a polynomial-time algorithm with approximation factor $O(k^2 \log n)$.*

1.2 Outline

The remainder of this paper focuses on the non-disjoint variant. For the disjoint results we refer to the full version [7]. In Section 2.1 we present our algorithm for the assignment version and in Section 2.2 we give an overview on how the centers can be found. The full description and analysis of the latter algorithm as well as the hardness proof for this setting can again be found in the full version of this work.

2 Non-disjoint Connected k-Median Problem

2.1 Assignment Version

The high-level idea to solve the assignment version of the connected k-median problem is to first define an LP formulation for it and compute an optimum fractional solution for the LP relaxation in polynomial time. Using an observation that the connected clusters are similar to so called node weighted Steiner trees we can use this LP solution to obtain k instances of

the node weighted Steiner tree problem such that every node needs to be contained in at least one Steiner tree and there exist a fractional solution for each of these instances such that the total cost of these solutions is bounded by k times the optimum LP solution. Using a primal-dual algorithm by Klein and Ravi [15] we are able to find integral Steiner trees that cost at most $O(\log n)$ times as much as the respective fractional solutions and which directly form k connected clusters assigning each node to at least one center. In total, this gives us an approximation factor of $O(k \log n)$.

Now we will define the LP formulation. For that, we use the following definition of cuts:

▶ **Definition 10** (Vertex Cuts). *For two arbitrary nodes $v, w \in V$, a (v,w)-*cut *is a subset $S \subseteq V$ such that all paths from v to w in G contain at least one node in S. The set of all (v,w)-cuts is denoted by $\mathcal{S}_{v,w}$.*

Using this, we can reformulate the requirement that a node v must be connected to its assigned center c by ensuring that for any (v,c)-cut $S \in \mathcal{S}_{v,c}$, at least one node in S is also assigned to c. This gives us the following LP-formulation of the non-disjoint connected k-median problem:

$$\begin{aligned}
\min \quad & \sum_{c \in C, v \in V} d(v,c) x_v^c & & \\
\text{s.t.} \quad & \sum_{c \in C} x_v^c \geq 1, & & \forall v \in V, \\
& \sum_{v' \in S} x_{v'}^c \geq x_v^c, & & \forall v \in V, c \in C, S \in \mathcal{S}_{v,c}, \\
& x_v^c \in \{0,1\}, & & \forall c \in C, v \in V,
\end{aligned} \tag{1}$$

where for all $v \in V$, $c \in C$ the variable x_v^c indicates whether v is assigned to c ($x_v^c = 1$) or not ($x_v^c = 0$). While the first constraint ensures, that each point v must belong to at least one cluster, the second constraint ensures that the clusters are connected, i.e., if v is connected to center c, then in each (v,c)-cut, there must be at least one point assigned to c. To relax this problem we drop the integrality constraint and only require that $x_v^c \geq 0$ for all $v \in V$, $c \in C$.

It is worth noting that the linear program may have exponential size, as there can be exponentially many (v,c)-cuts for a given pair v, c. However, since the separation problem can be solved in polynomial time and there also exists an alternative flow-based formulation of the LP with polynomial size [7], it remains possible to find the optimum fractional LP-solution in polynomial time.

Let $\tilde{x}$ be such an optimum fractional solution. By Markov's inequality it is easy to verify that for any $v \in V$ there exists at least one $c \in C$ such that $\tilde{x}_v^c \geq \frac{1}{k}$. By multiplying all values with k, we get a new LP solution x with $x_v^c := k\tilde{x}_v^c$ for all $v \in V, c \in C$ where each node is assigned to at least one center in its entirety. But even if for a $v \in V, c \in C$ it holds that $x_v^c \geq 1$ there may still exist (v,c)-cuts not containing any nodes fully assigned to c. To deal with this problem we need some results shown for node weighted Steiner trees.

▶ **Definition 11** (Node Weighted Steiner Tree Problem). *In an instance of the* node weighted Steiner tree problem *we are given a graph $G = (V,E)$, a set of terminal nodes $T \subseteq V$ and a weight function $w : V \to \mathbb{R}_{\geq 0}$. The goal is to find a subset $S \subseteq V$ such that $T \subseteq S$, the nodes of S form a connected subgraph of G, and the sum $\sum_{v \in S} w(v)$ is minimized.*

Similar as for the connected k-median problem there also exists an LP relaxation for the node weighted Steiner tree problem:

$$\begin{aligned} \min \quad & \sum_{v \in V} w(v) x_v \\ \text{s.t.} \quad & \sum_{v' \in S} x_{v'} \geq 1, \quad \forall u, v \in T, S \in \mathcal{S}_{u,v}, \\ & x_v^c \geq 0, \quad \forall v \in V, \end{aligned} \tag{2}$$

One might note that the assignment version of the connected k-median problem is closely related to the node weighted Steiner tree problem as one could reinterpret the problem as finding k node weighted Steiner trees such that each node is contained in at least one Steiner tree. The distances of the nodes to the respective centers can in this context be interpreted as k different node weight functions for the different trees. Using the solution x we will now define for any center $c \in C$ the set $T_c := \{c\} \cup \{v \mid x_v^c \geq 1\}$. Then the solution x can be subdivided into k solutions for the Steiner tree LP using the respective sets $(T_c)_{c \in C}$ as terminals.

▶ **Lemma 12.** *Let $(x_v^c)_{v \in V, c \in C}$ be a solution for LP (1). Then for any center $c \in C$ the values $(x_v^c)_{v \in V}$ form a solution for LP (2) with $T := T_c$ and $w(v) := d(v, c)$.*

Proof. This lemma is proved by contradiction. Assume that $(x_v^c)_{v \in V, c \in C}$ is a solution for LP (1) that violates some constraint of LP (2). Then there exist two terminals $u, v \in T_c$ and an (u, v)-cut S such that

$$\sum_{v' \in S} x_{v'}^c < 1.$$

Now we distinguish two cases. In the first case let c be one of the nodes $\{u, v\}$, w.l.o.g. $u = c$. Then we can use the definition of LP (1) and T_c to obtain

$$\sum_{v' \in S} x_{v'}^c \geq x_v^c = 1,$$

which contradicts the assumption.

In the second case both $u \neq c$ and $v \neq c$. Then by the previous argument S can neither separate u nor v from c because then its weight would be at least 1. Thus there exists a path from u to c and from c to v not containing any nodes in S. But then there also exists a path from u to v not using any nodes in S and S would not be a (u, v)-cut. ◀

This lemma is crucial because for the node weighted Steiner tree problem there exists an $O(\log n)$-approximation algorithm by Klein and Ravi [15] for which Guha et al. [11] later proved that it is implicitly a primal-dual algorithm and that the cost of the solution is at most $O(\log n)$ times worse than any fractional solution of LP (2). Alternatively Könemann et al provided a more detailed primal-dual algorithm for a generalized version of the node weighted Steiner tree problem ensuring the same guarantees [16]. Using these algorithms and Lemma 12 we are able to subdivide x into k Steiner tree LP solutions that can each be transformed into a feasible cluster.

▶ **Theorem 8.** *For the assignment version of the non-disjoint connected k-median problem, there is a polynomial-time algorithm with approximation factor $O(k \log n)$. This is true even if the distances are not a metric.*

Proof. In the first step of the algorithm we will first calculate an optimum solution $\tilde{x}$ for the relaxed version of LP (1) which as argued above can be done in polynomial time. Since every assignment solution for the connected k-median problem is also an LP solution, the costs of $\tilde{x}$ are upper bounded by the optimum value OPT. After multiplying the values by k, the cost of solution x is then again bounded by $k \cdot \mathrm{OPT}$.

Consider T_c as defined above, by Markov's inequality for every node v there exists a center c such that $\tilde{x}_v^c \geq \frac{1}{k}$ which means that $v \in T_c$. Thus we have $\bigcup_{c \in C} T_c = V$. By Lemma 12 the fractional solution x can be split up into k solutions of the Steiner tree LP (2) with the respective terminal sets $(T_c)_{c \in C}$ such that the total sum of the objective value is exactly the cost of x.

By applying the algorithm by Klein and Ravi [15] with $T = T_c$ and $w(v) = d(v, c)$ for every $c \in C$ separately, one obtains a Steiner tree S_c with $T_c \subseteq S_c$ whose cost is upper bounded by $O(\log n)$ times the cost of the respective LP solution. Thus one obtains in polynomial time k Steiner trees each connecting the respective terminal set T_c and total cost bounded by $O(\log n)$ times the objective value of x which results in an upper bound of $O\left(k \log n\right) \mathrm{OPT}$. Since every node is contained in at least one of these Steiner trees, this directly yields the desired solution for the assignment version of the non-disjoint connected k-median problem.

This proof works even if the distances are not a metric because it does not use any of their properties. ◀

One may note that this approach works with any approximation algorithm for the node weighted Steiner tree problem whose cost is bounded by the optimum LP solution. For some restricted graph classes, including planar graphs, Demaine et al. [5] provided a primal dual algorithm calculating a constant approximation. Thus the approach can also be used to get an $O(k)$-approximation (and $O(k^2)$ if the centers are not given) for these kind of connectivity graphs.

2.2 Finding Centers

In this section we give an overview how suitable centers for the non-disjoint connected k-median problem can be found. For a more detailed description of the procedure and its analysis we refer to the full version of this paper [7]. First we will again provide a suitable ILP formulation:

$$
\begin{aligned}
\min \quad & \sum_{v,c \in V} d(v,c) x_v^c & & \\
\text{s.t.} \quad & \sum_{c \in V} x_v^c \geq 1, & & \forall v \in V, \\
& \sum_{c \in V} x_c^c \leq k, & & \\
& \sum_{v' \in S} x_{v'}^c \geq x_v^c, & & \forall v, c \in V, S \in \mathcal{S}_{v,c}, \\
& x_v^c \in \{0,1\}, & & \forall v, c \in V,
\end{aligned} \tag{3}
$$

where $\mathcal{S}_{v,c}$ denotes the set of vertex cuts, separating v and c, defined as in the previous section.

The main difference to the assignment variant is that we do now allow to assign nodes to any other nodes and not only to some predefined centers. However, by definition of $\mathcal{S}_{v,c}$, it holds that $\{c\} \in \mathcal{S}_{v,c}$ for any $v, c \in V$. Thus x_v^c will automatically be limited by x_c^c. This naturally leads to the property that nodes can only be assigned to c if $x_c^c = 1$. Thus limiting $\sum_{c \in V} x_c^c$ by k directly ensures that any solution of the ILP uses at most k centers.

If we relax the integrality constraint and only require that $x_v^c \geq 0$ for all $v, c \in V$ also LP (3) can be solved in polynomial time despite its exponential size. The drawback of this is that for arbitrarily many nodes $c' \in V$ it can hold that $x_{c'}^{c'} > 0$ if the respective values are sufficiently small. This means that other nodes can at least partially be assigned to them and we will call the value $x_{c'}^{c'}$ the *opening* of c'.

The main focus of this section is to move these openings to a limited set of k centers that will be determined by our algorithm such that reassigning the nodes accordingly is not too expensive. To do this, we follow a similar approach as the LP rounding algorithm for k-median by Charikar et al. [3]. In the first step, we will find a solution in which every node is opened either by at least $\frac{1}{2}$ or not opened at all (limiting the number of centers to $2k$). In the latter steps these openings will then be made integral by first finding possible replacements for some of the centers, which possibly allows us to close the respective centers, and then choosing a subset of the centers of size k such that for every center itself or its replacement is contained in this subset.

We will relax the connectivity constraint during the algorithm. We say for an $\alpha \in [0, 1]$ that an assignment $(x_v^c)_{v,c \in V}$ is an α-connected solution if $\sum_{c \in V} x_c^c \leq k$, $\forall v \in V : \sum_{c \in V} x_v^c \geq 1$ and for all $v, c \in V$ and all cuts $S \in \mathcal{S}_{v,c}$ separating v and c it holds that $\sum_{v' \in S} x_{v'}^c \geq \min(\alpha, x_v^c)$. To simplify notation, we will denote the weight of the minimum cut between two nodes v and c (with respect to the values $(x_{v'}^c)_{v' \in V}$) by $\mathrm{sep}^x(v, c) = \min_{S \in \mathcal{S}_{v,c}} \sum_{v' \in S} x_{v'}^c$. Then the last requirement can be reformulated as $\mathrm{sep}^x(v, c) \geq \min(\alpha, x_v^c)$. It is easy to verify that an assignment corresponds to a fractional solution for LP (3) if and only if it is 1-connected.

Now we give an overview over the different steps of the algorithm. Our main focus will lie on the first step, which ensures that all centers will be half opened afterwards. This is due to the fact that it introduces the largest amount of novel ideas and necessary modifications compared to the existing algorithm by Charikar et al.:

Step 1. To obtain half-opened centers, let an optimal (fractional) LP solution $\tilde{x}$ be given. For each v, let its mass be defined as $m_v = \sum_{c \in V} \tilde{x}_v^c$, its cost (or price) as $p_v = \sum_{c \in V} \tilde{x}_v^c \cdot d(v, c)$ and its (average) radius as $r_v = \frac{p_v}{m_v}$. We say that an assignment of a node v to another node c is *good* if $d(v, c) \leq 4r_v$. Since $m_v \geq 1$ for all nodes $v \in V$, we may observe that because of Markov's inequality at least $\frac{3}{4}$ of v gets assigned via good assignments, i.e., $\sum_{c \in V : d(v,c) \leq 4r_v} \tilde{x}_v^c \geq \frac{3}{4}$.

We will simultaneously maintain two assignments $(x_v^c)_{v,c \in V}$ and $(y_v^c)_{v,c \in V}$ such that for all $v, c \in V$ initially $x_v^c = \tilde{x}_v^c$ and $y_v^c = 0$ and that at the end of the algorithm $x_v^c = 0$. The final values of y_v^c will be the assignment using half opened centers. During the algorithm, the combination of x and y will always be a $\frac{1}{8k}$-connected solution, which means that the following inequalities stay satisfied throughout the entire process:

$$\sum_{c \in V} (x_v^c + y_v^c) \geq 1 \qquad \forall v \in V \tag{4}$$

$$\sum_{c \in V} (x_c^c + y_c^c) \leq k \tag{5}$$

$$\sum_{v' \in S} x_{v'}^c \geq x_v^c \qquad \forall v, c \in V, S \in \mathcal{S}_{v,c} \tag{6}$$

$$\sum_{v' \in S} y_{v'}^c \geq \min\left(\frac{1}{8k}, y_v^c\right) \qquad \forall v, c \in V, S \in \mathcal{S}_{v,c} \tag{7}$$

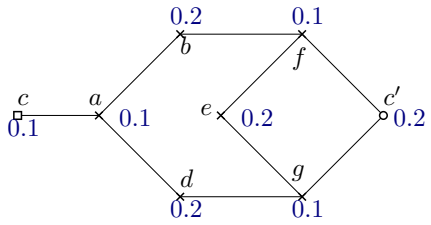

Figure 1 A sketch depicting a situation in which, at least if we require 1-connectivity, the total assignment needs to increase if we apply a shift from c' to c. The blue values denote the assignments of the respective nodes to c' before the shift, i.e., the $x_v^{v'}$-values. We assume that $x_v^c = 0$ for all nodes v. Now if we apply a shift from c' to c by $x_c^{c'} = 0.1$, both c' and c have an opening of 0.1. Thus the nodes b, d, e all need to be assigned to c and c' with value 0.1. If we would try to find assignments of a, f and g such that for each node its total assignment to c and c' is exactly 0.1 we would need to set $x_g^{c'} = x_f^{c'} = 0.1$ to ensure the connectivity of b and d to c'. However this would mean that $y_f^c = y_g^c = 0$ which would separate e from c. Thus to ensure connectivity, for at least one node the total assignment after the shift needs to be larger than the assignment before the shift.

The algorithm will now maintain an initially empty center set C and repeat in an alternating order the following two steps: First it decides for an additional node whether it gets added to the center set or not (in the latter case it will be added to a so called environment of an already existing center). Afterwards it shifts some openings in x (the x_c^c that are greater than 0) to already chosen centers c in C.

To get an intuition regarding the shifts, it can be beneficial to sometimes think about the connectivity constraints not from a cut-perspective but to rather interpret the assignments as flows. Using the max-flow min-cut theorem, it is easy to see that if we interpret the values $(x_{v'}^{c'})_{v' \in V}$ as node capacities then there exists a flow from a node v to a node c' of value f exactly if $\text{sep}^x(v, c') \geq f$. Thus our connectivity constraint implies that there always exists a flow from v to c' of value $x_v^{c'}$. By basically reverting this flow for a center $c \in C$ and a node $c' \in V$ we are able to shift an opening of value $x_c^{c'}$ from c' to c. One important question is now how the nodes that were previously assigned to c' get assigned. If $x_c^{c'}$ was equal to $x_{c'}^{c'}$, the entire opening of c' can be shifted and we can simply reassign all nodes to c by setting $y_v^c = y_v^c + x_v^{c'}$ and $x_v^{c'} = 0$ for any $v \in V$. However, if $x_c^{c'}$ was smaller than $x_{c'}^{c'}$ only a fraction of the opening of c' gets shifted and we basically have to split up the previous assignments to c' between c and c'. As Figure 1 shows this can lead to situations in which the total assignment to c and c' after this shift actually needs to be larger than the assignment before this shift. Since it is actually rather involved to determine the new values in this situation, a formal description will only be provided in the full version of this work. To get an intuition regarding the algorithm it is for now sufficient to only consider the first case.

Since either way also the assignments of nodes assigned to c' have to be shifted in this situation, we have to make sure that the distances of the respective shifts are not too large which the algorithm will ensure by only allowing shifts with a rather small distance between c and c' in the beginning and increasing this distance over time. More details will follow later.

While the shifts depend on the chosen centers C, the choice whether a node v will be added to C in turn depends on the shifts that have already been performed before v gets considered by the algorithm. The algorithm terminates once it has decided for every node whether it gets added to C and all openings in x have been shifted.

To determine which nodes should be added to C, we iterate over all nodes ordered by their radii in increasing order and check whether we can open them by at least $\frac{1}{2}$. To be more precise, a node v will be chosen as a center if the sum of the remaining good assignments $(x_v^{c'})_{c' \in V}$ is at least $\frac{1}{2}$. The construction of the algorithm will ensure that it will shift at least

all remaining openings belonging to good assignments of v to v itself which means that v will be at least half opened. When v is added to the center set we also create its environment $S_v = \{v\}$, which initially only contains v itself.

In the case that v does not get chosen as a center, we know that there were at least $\frac{1}{4}$ of good assignments of v that have already been shifted to other centers in C. Since every center in C gets at least half opened and the entire opening is limited by k, we know that $|C| \leq 2k$ and thus there exists at least one center to which $\frac{1}{8k}$ of the good assignments of v have been shifted. One critical property of the algorithm will be that before v gets considered, the distance between the source c' and target c of any shift will be at most $4r_v$. Thus for any good assignment to a node z whose opening has been shifted to a center c we have that $d(v,c) \leq d(v,z) + d(z,c) \leq 8r_v$. As a result we may conclude that there exists a center $c \in C$ such that $y_v^c \geq \frac{1}{8k}$ while $d(v,c) \leq 8r_v$ and we add v to the environment S_c of the closest center c fulfilling these requirements.

Adding v to this environment allows us to shift some openings of a node c' with $x_v^{c'} > 0$ over the node v to c even if there does not exist a direct flow from c to c' (i.e., c is not assigned to c'). The intuition behind this is that we first revert the flow from v to c' and thus are already shifting the opening to v. Now by combining $y_v^c \geq \frac{1}{8k}$ and Property (7) we know that already before this shift the minimum cut between v and c has value at least $\frac{1}{8k}$. Thus if we are moving the opening (and the assignments to it) to c instead of v itself we know that the solution still stays $\frac{1}{8k}$-connected. Furthermore since v and c are not too far apart, we know that the distances of these shifts does not increase too much compared to a direct shift to v. This will help us ensuring that if the openings of some nodes, to which v got assigned, get shifted to some other centers then the distance added by this shift will not be too large.

Every time after a new node v gets added to either C or to an environment S_c of a center $c \in C$ we will go over all possible shifts ordered by their distances in increasing order, and perform all shifts with distance at most $4r_{\text{next}}$, where r_{next} denotes the radius of the next node that will be considered by the algorithm. Note that any shift with radius $\leq 4_{r_v}$ can only result from the addition of v to S_c or C because any shift not involving v would already have been applied in the last iteration (as r_v is the previous value of r_{next}). As a result all those shifts will be performed before any shift not involving v will be executed. This directly guarantees that if v is added to C then all openings belonging to good assignments will be shifted to v, which ensures that v is half opened.

An important property of the shifts is that if we shift some opening of a node $c' \in V$ over a node $v \in V$ then afterwards $x_v^{c'} = 0$ which ensures that for all $v, c' \in V$ we will only shift some opening of c' over v at most once, limiting the number of shifts by $|V|^2$. Additionally after we have added v to C or a set S_c there always exists a possible shift while $x_v^{c'} > 0$ for a $c' \in V$. Thus, it is possible to execute the described algorithm for a polynomial time until for every $v, c' \in V$, $x_v^{c'} = 0$ and then return $(y_v^c)_{v \in V, c \in C}$. A pseudocode summarizing the structure of the described procedure can be found in Algorithm 1. The cost of the resulting $\frac{1}{8k}$-connected solution is bounded by $O(k)$ times the cost of the LP solution $\tilde{x}$.

Step 2. Now we would like to reduce the number of centers to at most k. However, if we want to reduce the opening of a center $c \in C$ to 0, we will need to find a suitable replacement, i.e., another center $\tilde{c}$ such that reassigning all nodes assigned to c to $\tilde{c}$ neither costs too much nor violates our connectivity constraints (which is further relaxed to $\frac{1}{16k}$-connectivity). To do this we will need two properties of Algorithm 1:

- For any node v we know that if a shift from another node c' to a center c caused v to be (partially) assigned to c then the radius r_c of c can be bounded by the radius r_v of v and the distance from v to c'. To be more precise $r_c \leq d(v,c') + 2r_v$. Intuitively, if this

Algorithm 1 The general structure of the first step.

```
Order v_1, ..., v_n according to their radii (r_v)_{v∈V} in increasing order
for v_1, ..., v_n do
    Decide whether v_i is added to the center set C or to an environment S_c of a
     center c ∈ C
    while A shift from a node v to a center c ∈ C with d(v, c) ≤ r_{v_{i+1}} is possible do
        Apply the cheapest shift
    end
end
```

was not the case, then the algorithm would have shifted some opening of c' either to v itself (if v gets chosen as a center) or over v to another center $\tilde{c} \in C$ with $v \in S_{\tilde{c}}$ before c was even added to the center set which would have prevented v from being assigned to c. The main consequence of this property is that if we can find another center $\tilde{c}$ that is only a constant times r_c away from c then reassigning all nodes that are currently assigned to c, to $\tilde{c}$ instead costs not too much.

- Additionally for any center c it holds that if c got assigned to another center $\tilde{c}$ because some opening got shifted from a node c' to c then it holds that $d(c, \tilde{c})$ is bounded by $d(c, c') + \max(4r_c, d(c, c'))$ because otherwise a shift from c' to c would have happened before the shift to $\tilde{c}$. Since the average length of the original assignments of c was r_c, one can use this to identify a center $\tilde{c}$ that is not too far apart from c.

Using these two properties, we are able to split up C into two sets C_1 and $C_{1/2}$ such that for every center $c \in C_{1/2}$ we have a successor $s(c) \in C$ such that the total cost that occurs if we replace any center in $C_{1/2}$ by its successor would not be too large and $|C_1| + \frac{1}{2}|C_{1/2}| \leq k$. Later we will then open all nodes in C_1 as well as at most half the nodes in $C_{1/2}$ to obtain a integral center set C^* such that for all nodes in $C_{1/2}$ either the node itself or its successor is contained in C^*.

To obtain these sets, we consider an arbitrary center $c \in C$ and define $a_c = 1 - y_c^c$. Then at least a total value of a_c of the openings corresponding to the initial assignments of c (the values $\tilde{x}_c^{c'}$ for any $c' \in V$) were shifted to other centers. If a_c has value at least $\frac{1}{4}$, we are able to use Markov's inequality together with the second property to identify another center $\tilde{c}$ such that $d(c, \tilde{c})$ lies within $O(r_c)$ and c is already sufficiently connected to $\tilde{c}$ (i.e., $y_c^{\tilde{c}} \geq \frac{1}{16k}$). We thus can set $s(c) = \tilde{c}$ and add c to $C_{1/2}$.

However, if a_c is rather small then it can be that all assignments of c to other centers $\tilde{c}$ only resulted from shifts of nodes c' to $\tilde{c}$ that are really far apart from c which means that $\tilde{c}$ can be even further away. Additionally the assignments $y_c^{\tilde{c}}$ might also be quite small which means that we would need to increase some existing assignments to $\tilde{c}$ to ensure $\frac{1}{16k}$-connectivity if we replace c by $\tilde{c}$. However, if there are multiple centers where the respective value a_c is small then we know that since $\sum_{c \in C}(1 - a_c) = \sum_{c \in C} y_c^c \leq k$ that if we add one of these centers to $C_{1/2}$ we can add multiple others to C_1. Since we do not need a replacement for the centers in C_1, we can use a careful potential argument to redistribute parts of the replacement cost of the centers added to $C_{1/2}$ to the centers in C_1 to bound the additional cost among all nodes.

Step 3. To decide which centers in $C_{1/2}$ can be closed we have a look at the directed Graph $G_{1/2} = (C_{1/2}, S)$, where $S = \{(c, s(c)) \mid c, s(c) \in C_{1/2}\}$ contains for any center in $C_{1/2}$ an edge to its successor if the latter is again contained in $C_{1/2}$. If this graph is bipartite we can simply take the smaller half of this partition and add this to the center set C^*. Then for every

edge one of its two endpoints is contained in C^* which ensures that for every center either itself or its successor is contained in C^*. However, the graph might not directly be bipartite because it can contain cycles of uneven length. But we can ensure by being a little bit more careful in step 2 that any of these cycles contains at most one center c with $a_c < \frac{1}{4}$. Using this, we can remove the respective cycles and thus obtain a bipartite graph. For all centers in $C_{1/2}$ not added to C^*, we reassign all nodes assigned to them to the respective successor $s(c)$. Since we bounded the cost of replacing all centers in $C_{1/2}$ and are only replacing a subset of them, the total cost of this is again bounded. The entire cost still stays in $O(k)$ times the cost of the fractional LP solution $\tilde{x}$.

Step 4. After step 3, we have an $\frac{1}{16k}$-connected solution only using integral centers. However the assignments might still be fractional. By multiplying the assignments by $16k$ the solution becomes 1-connected and similarly as for the assignment version in Section 2.1 we obtain that for every node v there exists at least one center c such that $x_v^c \geq 1$. Again applying the primal dual algorithm for node weighted Steiner trees, we obtain an entirely integral solution:

▶ **Theorem 9.** *For the non-disjoint connected k-median problem, there is a polynomial-time algorithm with approximation factor $O(k^2 \log n)$.*

References

1. Ioana Oriana Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R. Schmidt, and Melanie Schmidt. On the cost of essentially fair clusterings. In Dimitris Achlioptas and László A. Végh, editors, *APPROX/RANDOM 2019*, volume 145 of *LIPIcs*, pages 18:1–18:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.18.
2. Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Transactions on Algorithms (TALG)*, 13(2):1–31, 2017. doi:10.1145/2981561.
3. Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 1–10, 1999.
4. Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. *Advances in neural information processing systems*, 30, 2017.
5. Erik D Demaine, MohammadTaghi Hajiaghayi, and Philip N Klein. Node-weighted steiner tree and group steiner tree in planar graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 328–340. Springer, 2009. doi:10.1007/978-3-642-02927-1_28.
6. Lukas Drexler, Jan Eube, Kelin Luo, Dorian Reineccius, Heiko Röglin, Melanie Schmidt, and Julian Wargalla. Connected k-center and k-diameter clustering. *Algorithmica*, September 2024. doi:10.1007/s00453-024-01266-9.
7. Jan Eube, Kelin Luo, Dorian Reineccius, Heiko Röglin, and Melanie Schmidt. Connected k-median with disjoint and non-disjoint clusters, 2025. arXiv:2507.02774.
8. Rong Ge, Martin Ester, Byron J Gao, Zengjian Hu, Binay Bhattacharya, and Boaz Ben-Moshe. Joint cluster analysis of attribute data and relationship data: The connected k-center problem, algorithms and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(2):1–35, 2008. doi:10.1145/1376815.1376816.
9. Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.
10. Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. In Howard J. Karloff, editor, *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 25-27 January 1998, San Francisco, California, USA*, pages 649–657. ACM/SIAM, 1998. URL: http://dl.acm.org/citation.cfm?id=314613.315037.

11 Sudipto Guha, Anna Moss, Joseph (Seffi) Naor, and Baruch Schieber. Efficient recovery from power outage (extended abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, pages 574–582, New York, NY, USA, 1999. Association for Computing Machinery. doi:10.1145/301250.301406.

12 Neelima Gupta, Aditya Pancholi, and Yogish Sabharwal. Clustering with internal connectedness. In *WALCOM: Algorithms and Computation: 5th International Workshop, WALCOM 2011, New Delhi, India, February 18-20, 2011. Proceedings 5*, pages 158–169. Springer, 2011. doi:10.1007/978-3-642-19094-0_17.

13 Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Math. Oper. Res.*, 10(2):180–184, 1985. doi:10.1287/moor.10.2.180.

14 Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discret. Appl. Math.*, 1(3):209–215, 1979. doi:10.1016/0166-218X(79)90044-1.

15 P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *Journal of Algorithms*, 19(1):104–115, 1995. doi:10.1006/jagm.1995.1029.

16 Jochen Könemann, Sina Sadeghian, and Laura Sanita. An lmp o (log n)-approximation algorithm for node weighted prize collecting steiner tree. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 568–577. IEEE, 2013.

17 Shi Li. Approximating capacitated k-median with (1+ϵ) k open facilities. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 786–796. SIAM, 2016.

18 Shi Li and Xiangyu Guo. Distributed k-clustering for data with heavy noise. *Advances in Neural Information Processing Systems*, 31, 2018.

19 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.

20 Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Fair coresets and streaming algorithms for fair k-means. In *Approximation and Online Algorithms: 17th International Workshop, WAOA 2019, Munich, Germany, September 12–13, 2019, Revised Selected Papers 17*, pages 232–251. Springer, 2020. doi:10.1007/978-3-030-39479-0_16.

21 Ali Vakilian and Mustafa Yalciner. Improved approximation algorithms for individually fair clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 8758–8779. PMLR, 2022.

22 Hamidreza Validi, Austin Buchanan, and Eugene Lykhovyd. Imposing contiguity constraints in political districting models. *Oper. Res.*, 70(2):867–892, 2022. doi:10.1287/opre.2021.2141.