EPFL

Thèse n° 9818

Fairness and Explainability in Clustering Problems

Présentée le 11 août 2023

Faculté informatique et communications
Laboratoire de théorie du calcul 2
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Xinrui JIA

École
polytechnique
fédérale
de Lausanne

2023

The science of operations, as derived from mathematics more especially,
is a science of itself, and has its own abstract truth and value.
— Ada Lovelace

To my mom…

Acknowledgements

I would like to recognize the numerous people that have made my PhD journey possible — without them, it just wouldn't have been the same.

First and foremost, my advisor Ola Svensson has my gratitude and appreciation for being, without any hyperbole, the best advisor I could have hoped for. He has been extremely supportive and it is doubtless to me that I owe the successful completion of my PhD to him. From Ola I learned the importance of asking questions, the importance of perseverance to scientific research, and what it means to understand, really *understand*, a problem. His approach to problem solving will inspire me for years to come.

I would like to thank the members of my thesis committee: Prof. Friedrich Eisenbrand, Prof. Michael Kapralov, Dr. Vincent Cohen-Addad, and Prof. Christian Sohler, for reading my thesis and their interesting questions during the defense.

I would like to thank Chantal Schneeberger, who worked tirelessly behind the scenes to make day to day life in the lab run smoothly and also organized fun lab outings for special occasions.

This thesis would not have been possible without my coauthors: Kshiteej, Lars, Buddhima, Adam, and Weiqiang. A special thanks goes to Etienne for supplying the French translation of the abstract. I am grateful to the rest of the theory lab for informal discussions that I have learned a lot from. They also provided support and camaraderie for navigating the PhD, as did friends from other labs at EPFL. This part of the acknowledgement would not be complete without mentioning Phillip, who always believed in me even when I didn't believe in myself.

I would like to thank the professors from Waterloo who advised me on undertaking graduate studies in computer science: Prof. Ian Goulden, Prof. Prabhakar Ragde, and Prof. Eric Blais. I am also grateful for the opportunities to try research during my undergraduate studies, with Dr. Alfonso Cevallos and Prof. David Jao.

The start of this academic journey really began in high school when I became interested in mathematics. This is due to my teachers Mr. Ing and Mr. Mckenzie for always going above and beyond in enriching their students' education and generating excitement in mathematics and technology, despite the challenges posed by the school board. Another teacher whom I will never forget is Mr. White, who continued to be an inspiring educator into retirement and whose teaching methods in mathematics are unparalleled.

Last but not least, I have my parents and my brother Max to thank for their unconditional love and support. My parents gave up their careers and aspirations when starting their lives over in a new world, so that their children would have the opportunities they never had. Without their hard work and sacrifice, my life would look very different to what it is today.

Lausanne, June 27, 2023

Abstract

In this thesis we present and analyze approximation algorithms for three different clustering problems. The formulations of these problems are motivated by fairness and explainability considerations, two issues that have recently received attention in the algorithms and machine learning communities. In general, we are given a metric space and the task is to find a specified number of clusters to minimize some objective involving the centers of the clusters.

Our first problem is the *colorful k-center* problem. In the classic k-center with outliers problem, the objective is to minimize the maximum distance from any point to its nearest center, while a given number of points can be omitted from contributing to the maximum. That is, these points are not served by any center. Colorful k-center is a generalization of this problem. Instead of only serving enough points, each point belongs to some color class and a certain number of points of each color are required to be served in a solution. When this problem was first introduced by [Ban+19], a pseudo-approximation using $k+\omega-1$ centers was given for general metric spaces where ω is the number of color classes. We give the first true approximation algorithm with k centers that gives a 3-approximation for colorful k-center.

Next, we present our progress on the *non-uniform k-center* problem (NUkC or t-NUkC). In NUkC, we are given pairs (k_i, r_i), $1 \le i \le t$, $\sum_{i=1}^{t} k_i = k$, that represent k_i balls of radius r_i. The objective is to find $\sum_{i=1}^{t} k_i$ centers so that balls of radius αr_i around k_i of these centers cover all the points and α is minimized. NUkC was introduced by [CGK16], who conjectured that there exists a constant-factor approximation when t is constant. This is known as the NUkC conjecture. In a subsequent work, [CN21] gave a constant-factor approximation for $t=2$ with outliers (robust 2-NUkC). We make progress on the NUkC conjecture by presenting a simple reduction of t-NUkC to robust $(t-1)$-NUkC, which, in light of [CN21], gives the first constant-factor approximation to 3-NUkC.

Finally, we look at explainable algorithms for the k-medians and k-means objectives. Small decision trees are considered to be explainable models [Mur+19] for clustering, and in particular, when each node of the binary tree is a threshold on a single feature. Work by [Das+20] studied the theoretical guarantees obtainable from such a threshold tree, which they called *the price of explainability.* That is, the cost of an explainable clustering is compared to the cost of an optimal unrestricted clustering. We improve upon the results of [Das+20] to give nearly tight bounds on the price of explainability of the k-medians and k-means objectives, which also generalize to pth powers of ℓ_p norms.

Keywords: k-center clustering, k-medians clustering, k-means clustering, fairness, interpretability, explainability, approximation algorithms.

Résumé

Dans cette thèse nous présentons et analysons des algorithmes d'approximation pour trois problèmes différents de partitionnement. Les formulations de ces problèmes sont motivées par des considérations d'équité et d'explicabilité, deux questions qui ont récemment attiré l'attention des communautés d'algorithmique et d'apprentissage automatique. En général, nous disposons d'un espace métrique et la tâche consiste à trouver un nombre spécifié de groupes pour minimiser un objectif avec les centres des groupes.

Notre premier problème est le problème *k-centre coloré*. Dans le problème k-centre classique avec des données aberrantes, l'objectif est de minimiser la distance maximale de n'importe quel point à son centre le plus proche, exceptés pour un certain nombre de points qui peuvent être omis de contribuer au maximum. Autrement dit, ces points ne sont donc pas desservis par un centre. Le problème du k-centre coloré est une généralisation de ce problème. Au lieu de desservir suffisamment de points, chaque point appartient à une classe de couleur et un certain nombre de points de chaque couleur doivent être desservis dans une solution. Lorsque ce problème a été introduit pour la première fois par [Ban+19], une pseudo-approximation en utilisant $k+\omega-1$ centres a été donnée pour des espaces métriques généraux, où ω est le nombre de classes de couleurs. Nous donnons le premier algorithme d'approximation utilisant k centres avec un facteur d'approximation de 3 pour le problème du k-centre coloré.

Ensuite, nous présentons notre avancement sur le problème du *k-centre non-uniforme* (NUkC ou t-NUkC). Dans le NUkC, on nous donne des paires $(k_i, r_i), 1 \leq i \leq t$, $\sum_{i=1}^{t} k_i = k$, qui représentent k_i boules de rayon r_i. L'objectif est de trouver $\sum_{i=1}^{t} k_i$ centres de sorte que les boules de rayon αr_i autour de k_i de ces centres couvrent tous les points et α est minimisé. Le NUkC a été introduit par [CGK16], qui ont conjecturé qu'il existe une approximation de facteur constant lorsque t est constant. Ce problème est connu sous le nom de la conjecture NUkC. Un résultat ultérieur dû à [CN21] a donné une approximation de facteur constant pour $t = 2$ avec des données aberrantes (2-NUkC robuste). Nous progressons sur la conjecture NUkC en présentant une réduction simple de t-NUkC à $(t-1)$-NUkC robuste, qui, en raison de [CN21], donne la première approximation de facteur constant pour le problème 3-NUkC.

Enfin, nous étudions les algorithmes explicables pour les objectifs des k-médianes et des k-moyennes. Les petits arbres de décision sont considérés comme des modèles explicables [Mur+19] pour le partitionnement, en particulier lorsque chaque nœud de l'arbre binaire est une fonction seuil sur une seule caractéristique. Le travail de [Das+20] a étudié les garanties théoriques obtenues par ces arbres de seuil, ce qu'ils ont appelé *le prix de l'explicabilité*. Autrement dit, le coût d'un partitionnement explicable est comparé au coût d'un partitionnement optimal non restreint. Nous améliorons les résultats de [Das+20] pour donner des bornes presque optimales sur le prix de l'explicabilité des objectifs des k-médianes et des k-moyennes, qui se généralisent également aux puissances p-ièmes des normes ℓ_p.

Mot-clés : partitionnement k-centre, partitionnement k-médianes, partitionnement k-moyennes, équité, interprétabilité, explicabilité, algorithmes d'approximation.

List of Figures

Contents

1 Introduction

Clustering is a prototypical problem in theoretical computer science that is also implemented to solve practical problems in fields such as machine learning and operations research. In essence, clustering is the process of partitioning a set of objects into different groups. It can be seen as the process of splitting a data set based on similarity, or as a problem of picking locations for service centers. As a theoretical problem clustering has been well-studied for its interesting mathematical properties and yet there still remain open problems of technical interest.

Two topics that have been of interest recently in the machine learning community, and also in the algorithms community, are those of fairness and explainability. The aim in fair algorithms is to first modify a problem in such a way that reduces biases that may exist in optimal solutions to the original problem, and then find efficient solutions to the fair version of the problem. Designing fair algorithms becomes more important as processes and decisions become increasingly automated in modern society. For example, an algorithm that decides who is eligible for parole or decides which applications are approved for a loan carries heavy consequences for people. In this thesis, we look at two different clustering problems that can model fairness using the k-center objective.

Let us consider two examples of the type of fairness that can be modelled by our problems. For an application of colorful k-center, suppose a telecommunications company is choosing locations to build cellular towers and they need to provide coverage to 90% of the population. Some people live in dense urban regions and others live in sparse rural regions. It would be cheaper to only provide coverage to the urban regions, but this would be unfair to the people living in rural regions. Hence, coverage requirements for each individual group of clients in problems where outliers are allowed ensures that each group receives comparable treatment. In the second example, imagine that we have a certain number of facilities for ambulances and medical helicopters. Suppose that helicopters travel at twice the speed of ambulances, so that a client located at a certain distance from a helicopter actually receives service twice as quickly as a client located at the same distance from an ambulance. To ensure that service guarantees are fair, we can no longer measure the guarantee with the maximum distance from a client to a facility, but rather the distance to facilities that represent helicopters should be scaled by a factor of one-half. Representing different scaling of distances is the role that non-uniform radii play in the k-center problem.

Explainable algorithms are algorithms that are understandable and interpretable to humans by design. In order to trust the output of an algorithm, it may not be enough to rationalize the result of a black-box procedure, but rather the procedure itself should be designed to be explainable. This is important in applications where a high level of trust in the results is needed. For example, classification algorithms have been demonstrated to be useful in the medical field to assist in making diagnoses, which can improve efficiency and performance by making decisions faster and better than human practitioners can. It is vital that the results of such algorithms can also be understood and verified by humans. We study the *price of explainability* for a particular model of explainable clustering induced by decision-trees with the k-medians and k-means costs. That is, how much higher is the cost of this type of clustering compared to an optimal unrestricted clustering?

To illustrate an example of an explainable algorithm, consider a system that classifies people of being at risk of heart disease. A procedure that uses thresholds on single parameters such as BMI, blood pressure, age, etc is more understandable to a person than an algorithm that uses a linear combination of the features to make a classification. The algorithm that we study does exactly the former - the membership of a data point to a cluster is defined by whether each feature is above or below a threshold.

1.1 Overview of Contributions

In the remainder of this section, we summarize our contributions to each problem before giving a brief outline of the organization of this thesis.

1.1.1 Colorful k-center

Fairness in clustering problems can be modelled in various ways. In many of these models, each data point belongs to a particular group, and each group is represented by a color. Some examples of fairness constraints include requiring that the solution set of centers contains a certain number of points of each type, or that each cluster contains a certain proportion of points of each type [KAM19; Chi+17]. In Chapter 3, we use the model introduced by Bandyapadhyay, Inamdar, Pai, and Varadarajan [Ban+19] where we are given a certain number of points of each color that can remain uncovered as outliers. This is called the colorful k-center problem and represents a situation in which service is not guaranteed to everyone, but we ensure that a certain number from each group receives service. It is NP-hard to approximate to within a smaller factor than 2 since this is true of the original k-center problem [HN79]. In [Ban+19], a pseudo-approximation algorithm is given for colorful k-center in the general case. That is, $k+\omega-1$ centers of twice the optimal radius are opened, where ω is the number of color groups.

We study the colorful k-center problem to understand whether a pure approximation can be obtained by an efficient algorithm. This is based on a joint work with Kshiteej Sheth and Ola Svensson that was published in IPCO 2020 [JSS21a]. Our approach finds a way to eliminate the extra centers opened in the pseudo-approximation algorithm of [Ban+19] to obtain a true 3-approximation. We also present some negative results on natural linear programming (LP) strengthening techniques that fail to tighten the natural LP relaxation of the colorful k-center problem. Concurrent work [Ane+20] obtaining a 4-approximation via a different method was published at the same conference.

1.1.2 Non-uniform k-center

In the t-non-uniform k-center problem each of the k centers belongs to one of t different sizes of radii, and the approximation factor is the largest ratio between a point to its center and the specified radius. The objective is to cluster all the points with the least amount of radius expansion as possible. That is, the cost of the clustering comes from the k-center objective but the radii of the disks are not uniform. If we look at clustering from the lens of facility location then this problem models a situation in which different facilities provide different speeds of service. Hence, the quality of an approximation algorithm is the multiplicative expansion of the different radii. It is conjectured by Chakrabarty, Goyal, and Krishnaswamy [CGK16] that there is a constant-factor polynomial time approximation algorithm for constant t.

Chakrabarty et al. in [CGK16] give a bi-criteria approximation for NUkC which opens $c \cdot k_i$ balls of radius αr_i, where c and α are constants, as well as a constant-factor approximation for two different radii. Chakrabarty and Negahbani [CN21] advance the problem further by finding a constant-factor approximation for two radii with outliers. The paper that makes up Chapter 4, published in SOSA 2022 as joint work with Lars Rohwedder, Kshiteej Sheth, and Ola Svensson, obtains a constant-factor approximation for three radii by giving a reduction from t-NUkC to $(t-1)$-NUkC. In this paper we also describe the algorithm of [CN21] in a “bottom-up” approach, rather than the “top-down” approach of [CN21]. We believe that the resulting implementation is slightly simpler, since it avoids a nested ellipsoid algorithm.

Progress on NUkC has revealed unexpected connections to other k-center problems. In particular, a constant-factor approximation algorithm for $t = 4$ is now known [IV22], and the result makes use of colorful k-center clustering. Interestingly, the consideration of NUkC by [CGK16] also gave a tight approximation to k-center with outliers, which had been open for 15 years [Cha+01]. Note that k-center with outliers is a special case of NUkC, where we are given k balls of the optimal radius and m balls of radius 0, where m is the number of outliers permitted.

1.1.3 Explainable k-medians and k-means

In a recent paper published in ICML, Dasgupta, Frost, Moshkovitz, and Rashtchian [Das+20] study a model for explainable k-means and k-medians clustering from a theoretical point of view. Under this model, a clustering is explainable if the clusters are represented by leaves of a decision tree where each internal node of the tree is a split along an axis-aligned cut. Then the inclusion of a point to its cluster can be explained by the value of at most $k-1$ of the feature coordinates, depending on whether or not it is greater than the threshold cut value. In particular, the question studied in the aforementioned theoretical work was the *price of explainability*, that is, how much worse an explainable clustering is compared to an optimal unrestricted clustering. In general, Dasgupta et al. give an $O(k)$ approximation for the k-medians problem and an $O(k^2)$ approximation to k-means, while also showing that $\Omega(\log k)$ is a lower bound for both of these clustering problems

In a joint work with Buddhima Gamlath, Adam Polak, and Ola Svensson published in NeurIPS 2021, we provide a method of obtaining explainable clusterings with nearly tight guarantees. That is, our algorithms give an $O(\log^2 k)$ approximation for k-medians and $O(k \log^2 k)$ approximation for k-means. At the same time, we also improved the lower bound for k-means to $\Omega(k)$. For higher p-powers of ℓ_p-norms, we have an $O(k^{p-1} \log^2 k)$ approximation with an $\Omega(k^{p-1})$ lower bound. Our methods are oblivious to the data points in the sense that the algorithm only takes into account the position of the input centers of a general unrestricted clustering, so that our running time is independent of the number of data points. This work is presented in Chapter 5. Similar results were obtained around the same time by [MS21; CH22; EMN22]. Another work [LM21] by Laber and Murtinho studies the explainability of k-medians and k-means clustering in low dimensions, as well as two additional clustering objectives: k-center and maximum spacing. When clustering for maximum spacing, the objective to be maximized is the minimum distance between any two points that lie in different clusters.

1.2 Outline of the Thesis

The remainder of the thesis is organized as follows. In Chapter 2 we present common notation and concepts used throughout different sections, as well as formal definitions of the problems considered. We also present some background information, including fundamental results concerning the problems at hand.

We present the colorful k-center problem in Chapter 3. First, we describe the pseudo-approximation of [Ban+19] which we use as a subroutine in our 3-approximation algorithm. Then the main section of this chapter presents our algorithm, stated with two colors for notational simplicity. This is followed by a section that summarizes the algorithm in full generality. The last section discusses possibilities of natural ways of strengthening the linear programming description of the colorful k-center problem. In particular, we show that neither a constant number of rounds of the sum-of-squares hierarchy nor adding knapsack constraints succeeds in reducing the integrality gap.

We discuss the non-uniform k-center problem in Chapter 4. The first section contains a summary of the work of [CN21], followed by our reduction of t-NUkC to robust $(t-1)$-NUkC. The last section describes our alternative version of of the algorithm of [CN21].

In Chapter 5, we present our results on explainable clustering. We start with our algorithm and its analysis for explainable k-medians clustering, which also forms the intuition for general pth powers of ℓ_p-norms. In the next section we explain how to modify this algorithm for pth power of ℓ_p-norm clustering, which includes k-means. Finally, we present our lower-bound construction for pth power of ℓ_p-norms, for any $p \geq 1$.

Finally, in Chapter 6 we conclude with a discussion of future directions related to the work in this thesis.

2 Preliminaries

2.1 Notation and Definitions

Metric spaces
In Chapter 4 of this thesis we use (X, d) to denote a metric space where X is a set of n points and d is a metric on X. That is, d is a function $d : X \times X \to \mathbb{R}_{\geq 0}$ that satisfies

1. Non-negativity: $d(x, y) = 0$ if and only if $x = y$.
2. Symmetry: $d(x, y) = d(y, x)$.
3. Triangle inequality: $d(x, y) + d(y, z) \geq d(x, z)$.

Definition 2.1.1. *Let (X, d) be a metric space and $U \subseteq X$. Then*

$$B_U(v, r) := \{x \in U : d(x, v) \leq r\}.$$

By $B(v, r)$ we mean $B_X(v, r)$ and we refer to it as the ball of radius r centered at v. In k-center clustering problems, the cost of a solution is the maximum radius r of the balls that are used to cover the points.

For explainable clustering problems in Chapter 5 we use the ℓ_p-norm. Let $\mathbf{x} = (x_1, \ldots, x_n)$ and $p \geq 1$ an integer. Then

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}.$$

Note that the cost function in these problems is actually $\|\mathbf{x}\|_p^p$. We work in high dimensional Euclidean spaces, denoted $\mathbb{R}^d$ and here d denotes the dimension. We use boldface font to emphasize that our points are Cartesian coordinates. We write $[d] := \{1, 2, 3, \ldots, d\}$.

Center-based clustering
For k-center, k-medians, and k-means clustering, any set of centers defines a clustering. A

set of centers partitions the space into regions, where there is one region for each center consisting of all the points that are closer to that center than any other center (breaking ties arbitrarily). These regions are referred to in the literature as the *Voronoi regions* associated with the centers. Formally, let $C = \{c_1, \ldots, c_k\}$ denote the set of centers. Then the Voronoi region R_i associated with c_i is the set

$$R_i := \{x \in X : d(x, c_i) \leq d(x, c_j) \text{ for all } i \neq j\}.$$

In explainable clustering problems we use $\boldsymbol{\mu}$ to denote the individual centers to emphasize that they are cartesian coordinates.

Approximation algorithms
All of the problems considered in this thesis are NP-hard to solve optimally so we work with efficient approximation algorithms. That is, the algorithms run in time polynomial in the size of the input given in binary representation. Furthermore, since all the problems considered are minimization problems, an algorithm is an *α-approximation* if it is guaranteed to output a feasible solution with objective function value no more than α times the objective function value of an optimal solution. For some references on approximation algorithms see for example [WS11; Vaz01].

2.2 Problems Background

2.2.1 Colorful k-center

In the colorful k-center problem, we are given a set of n points X in a metric space (X, d) that are partitioned into ω color classes $\mathscr{C}_1, \ldots, \mathscr{C}_\omega$ with coverage requirements $p_1, \ldots, p_\omega$. The objective is to choose k of the points to be centers and $|\mathscr{C}_i| - p_i$ of the points of $\mathscr{C}_i$ to be outliers so that the maximum distance of a non-outlier point to its nearest center is minimized. A way to visualize this is to draw balls of radius r around each center so that a total of p_i points of $\mathscr{C}_i$ are covered by the balls. The objective is to choose the centers so that r is minimized.

Note that although it is NP-hard (proof below) to find a solution with the optimal radius we nevertheless can assume that the optimal radius is known to us. This is because there are only $O(n^2)$, or more precisely, at most $\binom{n}{2}$ possible distances for the optimal radius, which are the distances between every pair of points. So we can perform a binary search through all the possible radii with an algorithm that assumes the optimal radius is known, and take the smallest radius that produces a feasible solution.

Hardness of approximation
It has been known since 1979 that there does not exist an α-approximation algorithm for k-center with $\alpha < 2$, assuming P$\neq$NP [HN79]. It follows that finding a better-than-2 approximation algorithm for colorful k-center is also NP-hard. The hardness of k-center

follows from a reduction from the dominating set problem, a problem related to the independent set problem, both of which are NP-complete [Kar72; GJ90]. A dominating set of a graph is a set of vertices such that every vertex of the graph is either in the set or is neighboring a vertex in the set. The dominating set problem asks whether there exists a dominating set of a graph G of size less than or equal to k.

The reduction is as follows. Given a graph $G = (V, E)$ we create the following instance of the k-center problem with X being the set of points to be clustered: For each vertex $v \in V$ we have a point $p_v \in X$. If $(u, v) \in E$ then let $p_u, p_v \in X$ be 1 unit apart, and if $(u, v) \notin E$ then let p_u, p_v be 2 units apart. This forms a metric space since the distances satisfy the triangle inequality. It is easy to see that our k-center instance has a solution with optimal radius 2 if and only if G does not have a dominating set of size k. On the other hand, our k-center instance has a solution with optimal radius 1 if and only if G has a dominating set of size k. If we can find a solution to the k-center problem with approximation factor better than 2, then we can also answer whether G has a dominating set of size k.

A 2-approximation for k-center
A greedy algorithm that gives a 2-approximation for the k-center problem by T. F. Gonzalez [Gon85] has been known since 1985. This algorithm begins by selecting an arbitrary point to be a center, and then as long as the set of centers chosen is smaller than k, iteratively picks the point furthest away from any already chosen center to add to the set of centers. The claim is that the distance of any point to a nearest center is no more than 2 times the furthest such distance in an optimal solution.

To prove this, consider an optimal solution C^* of centers, which induces a set of Voronoi regions, $V_1^*, \ldots, V_k^*$, that partition the whole space. Let r^* denote the optimal radius. In the greedy selection of centers, if the center selected is in a Voronoi region that does not yet have a selected center, then all points in that Voronoi region are within $2r^*$ of the greedily selected center by the triangle inequality. Otherwise, consider the first time a greedy center is selected to be in the same Voronoi region as a previously selected center. These two centers are also within distance $2r^*$ by the triangle inequality. However, this means that all remaining points are also within distance $2r^*$ to some previous center since the new center was selected to be the point furthest away. This proves that this greedy algorithm gives a 2-approximation.

2.2.2 Non-uniform k-center

In the non-uniform k-center problem, we are given a set of n points X in a metric space along with radii $r_1 \geq r_2 \geq \cdots \geq r_t$ and positive integers $k_1, k_2, \ldots, k_t$ summing to k. The objective is to select $C_1 \cup C_2 \cup \cdots \cup C_t$ centers, where $|C_i| = k_i$, so that balls of radii αr_i around centers C_i cover all of the points, and α is minimized. In this thesis we give an approximation algorithm with constant $\alpha = 22$ for $t = 3$. More generally, we show that t-NUkC can be reduced to robust $(t-1)$-NUkC. It is conjectured that an efficient constant-factor approximation algorithm exists for constant t [CGK16], which is also known as the NUkC conjecture or the CGK conjecture.

A 3-approximation for k-center with outliers

Here we discuss a generalization of the k-center problem that is frequently used in the discussion of the NUkC problem, namely k-center with outliers. The NUkC problem with outliers is sometimes referred to as the *robust* version of NUkC. In k-center with outliers, an additional parameter is provided which indicates the total number of points that need to be serviced by a center, and all other points are omitted from consideration in the objective function. This models a situation in which it may be impractical to serve all points because an attempt to do so would either lead to a much higher service cost or require many more centers. This model was first introduced in [Cha+01] and the same paper provides a 3-approximation, which, similarly to the 2-approximation for k-center, also follows a greedy procedure. We describe this 3-approximation in the following.

Algorithm

Let r be the optimal radius. Consider the disk of radius r around each point in the input and add the point with the most number of points in its disk of radius r into the solution set of centers. Now we expand the radius to $3r$ by removing all points within distance $3r$ from the chosen center point (and assigning them to the chosen center). Repeat this process until k centers are chosen. If fewer than the required number of points are chosen return *False*, otherwise, *True*. Then we can perform a binary search on all possible inter-point distances to find the smallest radius that returns a feasible solution.

Proof of correctness

We prove that if an optimal solution uses balls of radius r then the above procedure returns *True*. Let $v_1, \ldots, v_k$ be the order of centers chosen by the algorithm and $B(v_i, r)$ and $B(v_i, 3r)$ be the disks of radius r and $3r$, respectively, around v_i. Let $o_1, \ldots, o_k$ be the centers in an optimal solution, and $B(o_i, r)$ the ball of radius r around o_i. The proof proceeds by induction to show that there is an ordering of o_i's such that each point covered by the first i balls of an optimal solution can be charged to a unique point covered by the first i balls of radius $3r$ of the solution of the algorithm.

To this end, suppose that points covered by the first i balls $B(o_1, r), \ldots, B(o_i, r)$ have already been charged to unique points covered by $B(v_1, 3r), \ldots, B(v_i, 3r)$. Consider $B(v_1, r) \cup B(v_2, r) \cup \cdots \cup B(v_{i+1}, r)$ intersected with the remaining optimal balls of radius r. If this intersection is non-empty then let o_i be an optimal center such that $B(o_i, r)$ has a non-empty intersection with the first $i+1$ balls of radius r selected by the algorithm. By the triangle inequality the first $i+1$ expanded balls of radius $3r$ picked by the algorithm cover $B(o_i, r)$ entirely. Hence, charge these points to themselves and note that points can be charged to themselves this way only once. If $\big(B(v_1, r) \cup \cdots \cup B(v_{i+1}, r)\big)$ has an empty intersection with the remaining optimal balls then let *uncovered* be the set of points in the instance that are not covered by $\big(B(v_1, 3r) \cup \cdots \cup B(v_i, 3r)\big)$. Let o_{i+1} be the center of the remaining optimal disks that cover the most points in *uncovered*. Charge the points in $B(o_{i+1}, r) \cap \big(B(v_1, 3r) \cup \cdots \cup B(v_i, 3r)\big)$ to themselves. By the greedy procedure, $B(v_{i+1}, r)$ covers as many points of *uncovered* as $B(o_{i+1}, r)$ does. Charge these uncovered points of $B(v_{i+1}, r)$ to the points of $B(o_{i+1}, r)$ in *uncovered*. To conclude the proof it suffices to note that i) $B(v_{i+1}, r)$ is disjoint from the remaining optimal balls and ii) these points are no longer in *uncovered* of future iterations, so these points will never be charged again.

A tight approximation for k-center with outliers
The 3-approximation described above was published in 2001 [Cha+01], and it is only recently that a tight 2-approximate algorithm was given for the k-center with outliers problem [CGK16]. Note that k-center with outliers is a special case of non-uniform k-center with 2 radii where the smaller radius is 0 and the number of balls of radius 0 is the number of outliers. In [CGK16], the authors give a constant-factor approximation for 2 radii, which also gives the tight 2-approximation in the case that the smaller radius is zero. In short, their algorithm opens k_1 balls of radius $2(r_1 + r_2)$ and k_2 balls of radius $2r_2$.

NUkC with 4 radii
Recently, Inamdar and Varadarajan [IV22] gave a constant-factor approximation for NUkC with four radii, using an algorithm that makes use of a colorful version of the problem. More details are given in Chapter 6.

2.2.3 Explainable k-medians and k-means

Here we work mainly with k-medians clustering and k-means clustering, although the latter is extended in our work to give results for more general clustering costs using pth powers of ℓ_p norms. In particular, a clustering of a set of points $\boldsymbol{X}$ in $\mathbb{R}^d$ is defined by its k centers $\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots, \boldsymbol{\mu}^k$ and the set of points in the ith cluster C^i are those that are closer to $\boldsymbol{\mu}^i$ than to any other center. The k-medians cost and k-means cost are, respectively,

$$\text{cost}_1(C^1, \ldots, C^k) = \sum_{j=1}^{k} \sum_{\boldsymbol{x} \in C^j} \|\boldsymbol{x} - \boldsymbol{\mu}^j\|_1$$

and

$$\text{cost}_2(C^1, \ldots, C^k) = \sum_{j=1}^{k} \sum_{\boldsymbol{x} \in C^j} \|\boldsymbol{x} - \boldsymbol{\mu}^j\|_2^2.$$

Higher p-norms are defined analogously.

In order to describe the model of explainability we are working with in this thesis we need the definition of a *threshold cut*, which is given by a dimension $i \in [d]$ and threshold value $\theta \in \mathbb{R}$. This threshold cut on set $\boldsymbol{X}$ divides the set into two subsets $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$ where $\boldsymbol{x} = [x_1, x_2, \ldots, x_d]$ is in $\boldsymbol{X}^1$ if $x_i \leq \theta$ and $\boldsymbol{x}$ is in $\boldsymbol{X}^2$ otherwise. A *threshold tree* is a decision tree that starts with the entire data set at the root node, and the children of each node are the subsets given by some threshold cut on the node. Finally, an explainable clustering is a clustering where the k clusters are given by the leaves of a decision tree with k leaves. That is, the inclusion of a point to a center is explained by the threshold cuts along the path from the root to the leaf node representing the center.

Our results in Chapter 5 are concerned with the *price of explainability*, that is, the ratio of the cost of an explainable clustering compared to an optimal unrestricted clustering of the same data set. Actually, since we are not concerned with constant factors in the approximation, we can compare the cost of our algorithm to any constant-factor clustering algorithm. Since there exist constant-factor approximation algorithms for both k-medians and k-means clustering, we can use as input a set of centers with cost that is a constant-factor of the optimal cost. Besides comparing the performance of our algorithm with an optimal unrestricted clustering, other possible objectives are discussed in Chapter 6 as part of future works.

Constant-factor approximations for k-medians and k-means
The first constant-factor approximation algorithm for the classic k-medians problem was given in [Cha+02] and obtained a ratio of $6\frac{2}{3}$. The best approximation ratio today in general metric spaces is $2.675+\epsilon$ [Byr+15] and was obtained through sophisticated LP-rounding techniques. Likewise, k-means also has a constant-factor approximation algorithm given by [Kan+04]. This algorithm has an approximation factor of $9+\epsilon$ and is based on local search. Since the work of [Kan+04], the approximation factor for general k-means has been improved to 6.357 [Ahm+20] and Euclidean k-means has been improved to 5.912 [Coh+22]. The same work [Coh+22] also improved Euclidean k-medians to 2.406.

Explainable clustering algorithm of [Das+20]
Here we give a brief summary of the main result of Dasgupta, Frost, Moshkovitz, and Rashtchian in [Das+20], which is an $O(k)$ algorithm for general explainable k-medians (and $O(k^2)$ for k-means). We only describe the algorithm in terms of k-medians, as the algorithm is the same for k-means and the analysis is similar.

The input to explainable k-medians clustering is a data set $X \subseteq \mathbb{R}^d$. The first step is to obtain a set of centers $\{\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^k\}$ from some standard constant-factor approximation algorithm for k-medians, and assign each data point to its closest center. Then the algorithm builds a decision tree top-down from the root node. At the beginning, all data points belong to the root node. The threshold cut (i, θ), where $i \in [d]$ is a dimension and $\theta \in \mathbb{R}$, is chosen so that the least number of points are split from their assigned centers and at least 2 centers are split. That is, all points x with $x_i \leq \theta$ are assigned to the left child of the root and all other points are assigned to the right child. This is repeated for all non-leaf nodes where the number of wrongly assigned centers (mistakes) is only counted among the points that belong to that node. A node is a leaf when it contains only a single center.

Main ingredients of the analysis
The analysis of the cost of the algorithm described above requires first rewriting the cost in terms of the *minimum* number of points that are classified to the wrong center. The cost of any clustering can also be related to the number of mistakes *required* by a clustering given by threshold cuts. With these ingredients, the cost of the algorithm can be related to the cost of an unrestricted clustering given by a set of centers. Intuitively, if the minimum number

of points classified to a wrong center by threshold cuts is high, then the cost of any optimal clustering is also high.

Let $\mathrm{cost}(\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^k)$ denote the cost of an unrestricted clustering given by the centers $\{\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^k\}$, where each point is assigned to its closest center. Let T be the tree produced by the mistake-minimizing algorithm described above. Let u be any internal node of T, and $B(u)$ be the bounding box of centers that end up in u. That is, for every dimension i the box $B(u)$ is delimited by an edge defined by the endpoints $\min_{\boldsymbol{\mu}\in u}\{\boldsymbol{\mu}_i\}$ and $\max_{\boldsymbol{\mu}\in u}\{\boldsymbol{\mu}_i\}$. The diameter of this box in the ℓ_1-norm, denoted $\mathrm{diam}(B(u))$, is just the sum of the side lengths of the box over all d dimensions. In the following we will not reproduce the formal proofs from [Das+20] but we will give a short explanation of how the proofs can be obtained.

Lemma 5.5 from [Das+20]. Let t_u be the number of mistakes made at node u in tree T. Then

$$\mathrm{cost}(T) \le \mathrm{cost}\left(\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^k\right) + \sum_{u\in T} t_u \cdot \mathrm{diam}(B(u)).$$

To see why this is true, note that only points that are assigned to the wrong center in T incur some cost additional to $\mathrm{cost}(\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^k)$. Indeed, this additional cost is no more than the diameter of the bounding box $B(u)$ where u is the node that the mistake for this point was made. The other lemma we need is as follows.

Lemma 5.6 from [Das+20]. Let H be the height of T. We have

$$\sum_{u\in T} t_u \cdot \mathrm{diam}(B(u)) \le 2H \cdot \mathrm{cost}\left(\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^k\right).$$

To prove this lemma, fix some non-leaf node u, some dimension i, and project the centers in u onto $B(u)$ along the ith dimension. If u contained m centers, then the edge of $B(u)$ in dimension i is divided into $m-1$ *segments* by the projections of the centers. The inequality above comes from considering how many times the length of each segment is included in $\mathrm{cost}(\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^k)$. Since the cut is chosen to minimize the number of mistakes, it must be that $\mathrm{cost}(\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^k)$ includes the edges of $B(u)$ at least t_u times. The factor 2 comes from supposing that cuts are only made at the midpoint of a segment, so that $\mathrm{cost}(\boldsymbol{\mu}^1,\ldots,\boldsymbol{\mu}^k)$ contains at least t_u-many half-segments. Finally, note that the algorithm can incur this cost along every node in a u-leaf path, so the total cost is multiplied by the height of the tree T.

With these two lemmas, we have that $\mathrm{cost}(T)$ is $O(k)$ since H can be linear in k: consider $d+1$ centers with a center at 0 and the d unit basis vectors as the remaining centers. Then each cut can only split away one center at a time. The $O(k)$ bound is therefore a barrier for the algorithm of [Das+20]. We show in this thesis that, perhaps surprisingly, an algorithm that uses random cuts can break this barrier.

3 A Constant-factor Approximation for Colorful k-Center Clustering

This chapter is based on joint work with Kshiteej Sheth and Ola Svensson [JSS21b]. It has been accepted to The 21st Conference on Integer Programming and Combinatorial Optimization (**IPCO 2020**) under the title

Fair Colorful k-Center Clustering.

It has also been accepted to the journal of *Mathematical Programming* vol. 192 under the same name.

3.1 Introduction

In the *colorful k-center* problem introduced in [Ban+19], we are given a set of n points X in a metric space partitioned into a set R of red points and a set B of blue points, along with parameters k, r, and b. The goal is to find a set of k centers $C \subseteq P$ that minimizes α so that balls of radius α around each point in C cover at least r red points and at least b blue points. More generally, the points can be partitioned into ω color classes $\mathscr{C}_1, \dots, \mathscr{C}_\omega$, with coverage requirements $p_1, \dots, p_\omega$. To keep the exposition of our ideas as clean as possible, we concentrate the bulk of our discussion to the version with two colors. In Section 3.3 we show how our algorithm can be generalized for ω color classes with an exponential dependence on ω in the running time in a rather straightforward way, thus getting a polynomial time algorithm for constant ω.

This generalization of the classic k-center problem has applications in situations where fairness is a concern. For example, if a telecommunications company is required to provide service to at least 90% of the people in a country, it would be cost effective to only provide service in densely populated areas. This is at odds with the ideal that at least some people in every community should receive service. In the absence of color classes, an approximation algorithm could be “unfair" to some groups by completely considering them as outliers. The inception of fairness in clustering can be found in the recent paper [CLV17] (see also [Bac+19;

Ana+19]), which uses a related but incomparable notion of fairness. Their notion of fairness requires *each individual cluster* to have a balanced number of points from each color class, which leads to very different algorithmic considerations and is motivated by other applications, such as “feature engineering”.

The other motive for studying the colorful k-center problem derives from the algorithmic challenges it poses. One can observe that it generalizes the *k-center problem with outliers*, which is equivalent to only having red points and needing to cover at least r of them. This outlier version is already more challenging than the classic k-center problem: only recent results give tight 2-approximation algorithms [CGK16; Har+19], improving upon the 3-approximation guarantee of [Cha+01]. In contrast, such algorithms for the classic k-center problem have been known since the ’80s [HS85; Gon85]. That the approximation guarantee of 2 is tight, even for classic k-center, was proved in [HN79].

At the same time, a subset-sum problem with polynomial-sized numbers is embedded within the colorful k-center problem. To see this, consider n numbers $a_1, \ldots, a_n$ and let $A = \sum_{i=1}^{n} a_i$. Construct an instance of the colorful k-center problem with $r = k \cdot A + A/2$, $b = k \cdot A - A/2$, and for every $i \in \{1, \ldots, n\}$, a ball of radius one containing $A + a_i$ red points and $A - a_i$ blue points. These balls are assumed to be far apart so that any single ball that covers two of these balls must have a very large radius. It is easy to see that the constructed colorful k-center instance has a solution of radius one if and only if there is a size k subset of the n numbers whose sum equals $A/2$.

We use this connection to subset-sum to show that the standard linear programming (LP) relaxation of the colorful k-center problem has an unbounded integrality gap even after a linear number of rounds of the powerful Lasserre/Sum-of-Squares hierarchy (see Section 3.4.1). We remark that the standard linear programming relaxation gives a 2-approximation algorithm for the outliers version even without applying lift-and-project methods. Another natural approach for strengthening the standard linear programming relaxation is to add flow-based inequalities specially designed to solve subset-sum problems. However, in Section 3.4.2, we prove that they do not improve the integrality gap due to the clustering feature of the problem. This shows that clustering and the subset-sum problem are intricately related in colorful k-center. This interplay makes the problem more complex and prior to our work only a randomized constant-factor approximation algorithm was known when the points are in $\mathbb{R}^2$ with an approximation guarantee greater than 6 [Ban+19].

Our main result overcomes these difficulties and we give a nearly tight approximation guarantee:

Theorem 1. *There is a* 3*-approximation algorithm for the colorful k-center problem.*

As aforementioned, our techniques can be easily extended to a constant number of color classes but we restrict the discussion here to two colors.

On a very high level, our algorithm manages to decouple the clustering and the subset-sum aspects. First, our algorithm guesses certain centers of the optimal solution that it then uses to partition the point set into a “dense” part X_d and a “sparse” part X_s. The dense part is clustered using a subset-sum instance while the sparse set is clustered using the techniques of Bandyapadhyay, Inamdar, Pai, and Varadarajan [Ban+19] (see Section 3.2.1). Specifically, we use the pseudo-approximation of [Ban+19] that satisfies the coverage requirements using $k+1$ balls of at most twice the optimal radius.

While our approximation guarantee is nearly tight, it remains an interesting open problem to give a 2-approximation algorithm or to show that the ratio 3 is tight. One possible direction is to understand the strength of the relaxation obtained by combining the Lasserre/Sum-of-Squares hierarchy with the flow constraints. While we show that individually they do not improve the integrality gap, we believe that their combination can lead to a strong relaxation.

Independent work. Independently and concurrently to our work, authors in [Ane+20] obtained a 4-approximation algorithm for the colorful k-center problem with $\omega = O(1)$ using different techniques than the ones described in this work. Furthermore they show that, assuming $P \neq NP$, if ω is allowed to be unbounded then the colorful k-center problem admits no algorithm guaranteeing a finite approximation. They also show that assuming the Exponential Time Hypothesis, colorful k-center is inapproximable if ω grows faster than $\log n$.

Organization. We begin by giving some notation and definitions and describing the pseudo-approximation algorithm in [Ban+19]. In fact, we then describe a 2-approximation algorithm on a certain class of instances that are *well-separated*, and the 3-approximation follows almost immediately. This 2-approximation proceeds in two phases: the first is dedicated to the guessing of certain centers, while the second processes the dense and sparse sets.

Section 3.3 explains the generalization to ω color classes. In Section 3.4 we present our integrality gaps under the Sum-of-Squares hierarchy and additional constraints deriving from a flow network to solve subset-sums.

LP1

$$\begin{aligned}
&\sum_{i\in B(j)} x_i \geq z_j, \quad \forall j \in X\\
&\sum_{i\in X} x_i \leq k\\
&\sum_{j\in R} z_j \geq r,\\
&\sum_{j\in B} z_j \geq b,\\
&z_j, x_i \in [0,1], \quad \forall i,j \in X.
\end{aligned}$$

LP2

$$\begin{aligned}
\text{maximize} \quad & \sum_{j\in S} r_j y_j\\
\text{subject to} \quad & \sum_{j\in S} b_j y_j \geq b,\\
& \sum_{j\in S} y_j \leq k,\\
& y_j \in [0,1] \qquad \forall j \in S.
\end{aligned}$$

Figure 3.1: The linear programs used in the pseudo-approximation algorithm.

3.2 A 3-Approximation Algorithm

In this section we present our 3-approximation algorithm. We briefly describe the pseudo-approximation algorithm of Bandhyapadhyay et al. [Ban+19] since we use it as a subroutine in our algorithm.

Notation: We assume that our problem instance is normalized to have an optimal radius of one and we refer to the set of centers in an optimal solution as OPT. The set of all points at distance at most α from a point j is denoted by $B(j,\alpha)$ and we refer to this set as a *ball of radius α at j*. We write $B(j)$ for $B(j,1)$. By a *ball of OPT* we mean $B(j)$ for some $j \in OPT$.

3.2.1 The pseudo-approximation algorithm

The algorithm of Bandhyapadhyay et al. [Ban+19] first guesses the optimal radius for the instance (there are at most $O(n^2)$ distinct values the optimal radius can take), which we assume by normalization to be one, and considers the natural LP relaxation LP1 depicted on the left in Figure 3.1. The variable x_i indicates how much point i is fractionally opened as a center and z_i indicates the amount that i is covered by centers.

Given a fractional solution to LP1, the algorithm of [Ban+19] finds a clustering of the points. The clusters that are produced are of radius two, and with a simple modification (details can be found in Appendix A.2), can be made to have a special structure that we call a flower:

Definition 3.2.1. *For $j \in X$, a **flower** centered at j is the set $\mathscr{F}(j) = \cup_{i\in B(j)} B(i)$.*

More specifically, given a fractional solution (x,z) to LP1, the clustering algorithm in [Ban+19] produces a set of points $S \subseteq X$ and a cluster $C_j \subseteq X$ for every $j \in S$ such that:

1. The set S is a subset of the points $\{j \in X : z_j > 0\}$ with positive z-values.

2. For each $j \in S$, we have $C_j \subseteq \mathscr{F}(j)$ and the clusters $\{C_j\}_{j\in S}$ are pairwise disjoint.

3. If we let $r_j = |C_j \cap R|$ and $b_j = |C_j \cap B|$ for $j \in S$, then the linear program LP2 (depicted on the right in Figure 3.1) has a feasible solution y of value at least r.

As LP2 has only two non-trivial constraints, any extreme point will have at most two variables attaining strictly fractional values. So at most $k+1$ variables of y are non-zero. The pseudo-approximation of [Ban+19] now simply takes those non-zero points as centers. Since each flower is of radius two, this gives a 2-approximation algorithm that opens at most $k+1$ centers. (Note that, as the clusters $\{C_j\}_{j\in S}$ are pairwise disjoint, at least b blue points are covered, and at least r red points are covered since the value of the solution is at least r.)

Obtaining a constant-factor approximation algorithm that only opens k centers turns out to be significantly more challenging. Nevertheless, the above techniques form an important subroutine in our algorithm. Given a fractional solution (x,z) to LP1, we proceed as above to find S and an extreme point to LP2 of value at least r. However, instead of selecting all points with positive y-value, we, in the case of two fractional values, only select the one whose cluster covers more blue points. This gives us a solution of at most k centers whose clusters cover at least b blue points. Furthermore, the number of red points that are covered is at least $r - \max_{j\in S} r_j$ since we disregarded at most one center. As $S \subseteq \{j : z_j > 0\}$ (see first property above) and $C_j \subseteq \mathscr{F}(j)$ (see second property above), we have $\max_{j\in S} r_j \le \max_{j:z_j>0} |\mathscr{F}(j) \cap R|$. We summarize the obtained properties in the following lemma.

Lemma 3.2.2. *Given a fractional solution (x,z) to LP1, there is a polynomial-time algorithm that outputs at most k clusters of radius two that cover at least b blue points and at least $r - \max_{j:z_j>0} |\mathscr{F}(j) \cap R|$ red points.*

We can thus find a 2-approximate solution that covers sufficiently many blue points but may cover fewer red points than necessary. The idea now is that, if the number of red points in any cluster is not too large, i.e., $\max_{j:z_j>0} |\mathscr{F}(j) \cap R|$ is "small", then we can hope to meet the coverage requirements for the red points by increasing the radius around some opened centers. Our algorithm builds on this intuition to get a 2-approximation algorithm using at most k centers for *well-separated* instances as defined below.

Definition 3.2.3. *An instance of colorful k-center is **well-separated** if there does not exist a ball of radius three that covers at least two balls of OPT.*

Our main result of this section can now be stated as follows:

Theorem 2. *There is a 2-approximation algorithm for well-separated instances.*

The above theorem immediately implies Theorem 1, i.e., the 3-approximation algorithm for general instances. Indeed, if the instance is not well-separated, we can find a ball of radius three that covers at least two balls of OPT by trying all n points and running the pseudo-approximation of [Ban+19] on the remaining uncovered points with $k-2$ centers. In the correct iteration, this gives us at most $k-1$ centers of radius two, which when combined with the ball of radius three that covers two balls of OPT, is a 3-approximation.

Remark 3.2.4. *We have an omission in the original submission in the case that some point that is covered by the ball of radius three needs to be opened as a center in the pseudo-approximation. The solution is to add x-variables for each point covered by the ball of radius three to the LP used by the pseudo-approximation, as x-variables indicate the amount that a point is opened as a center. We thank Chek-Manh Loi and Linda Kleist for pointing out this omission.*

Our algorithm for well-separated instances now proceeds in two phases with the objective of finding a subset of X on which the pseudo-approximation algorithm produces subsets of flowers containing not too many red points. In addition, we maintain a partial solution set of centers (some guessed in the first phase), so that we can expand the radius around these centers to recover the deficit of red points from closing one of the fractional centers.

3.2.2 Phase I

In this phase we will guess some balls of OPT that can be used to construct a bound on $\max_{j:z_j>0}|R\cap\mathscr{F}(j)|$. To achieve this, we define the notion of **Gain**(p,q) for any point $p\in X$ and $q\in B(p)$.

Definition 3.2.5. *For any $p\in X$ and $q\in B(p)$, let*

$$\textbf{\textit{Gain}}(p,q) := R\cap\big(\mathscr{F}(q)\setminus B(p)\big)$$

be the set of red *points added to $B(p)$ by forming a flower centered at q.*

Our algorithm in this phase proceeds by guessing three centers c_1, c_2, c_3 of the optimal solution OPT:

For $i=1,2,3$, guess the center c_i in OPT and calculate the point $q_i\in B(c_i)$ such that the number of red points in **Gain**$(c_i,q_i)\cap X_i$ is maximized over all possible c_i, where

$$\begin{aligned} X_1 &= X \\ X_i &= X_{i-1}\setminus\mathscr{F}(q_{i-1}) \quad \text{for } 2\le i\le 4. \end{aligned}$$

The time it takes to guess c_1, c_2, and c_3 is $O(n^3)$ and for each c_i we find the $q_i\in B(c_i)$ such that $|\textbf{Gain}(c_i,q_i)\cap X_i|$ is maximized by trying all points in $B(c_i)$ (at most n many).

For notation, define **Guess** $:=\cup_{i=1}^{3}B(c_i)$ and let

$$\tau = |\textbf{Gain}(c_3,q_3)\cap X_3|.$$

The important properties guaranteed by the first phase is summarized in the following lemma.

Lemma 3.2.6. *Assuming that c_1, c_2, and c_3 are guessed correctly, we have that*

1. *the $k-3$ balls of radius one in $OPT \setminus \{c_i\}_{i=1}^3$ are contained in X_4 and cover $b - |B \cap \textbf{Guess}|$ blue points and $r - |R \cap \textbf{Guess}|$ red points; and*

2. *the three clusters $\mathscr{F}(q_1)$, $\mathscr{F}(q_2)$, and $\mathscr{F}(q_3)$ are contained in $X \setminus X_4$ and cover at least $|B \cap \textbf{Guess}|$ blue points and at least $|R \cap \textbf{Guess}| + 3 \cdot \tau$ red points.*

Proof. 1) We claim that the intersection of any ball of $OPT \setminus \{c_i\}_{i=1}^3$ with $\mathscr{F}(q_i)$ in X is empty, for all $1 \le i \le 3$. Then the $k-3$ balls in $OPT \setminus \{c_i\}_{i=1}^3$ satisfy the statement of (1). To prove the claim, suppose that there is $p \in OPT \setminus \{c_i\}_{i=1}^3$ such that $B(p) \cap \mathscr{F}(q_i) \neq \emptyset$ for some $1 \le i \le 3$. Note that $\mathscr{F}(q_i) = \cup_{i \in B(q_i)} B(i)$, so this implies that $B(p) \cap B(q') \neq \emptyset$, for some $q' \in B(q_i)$. Hence, a ball of radius three around q' covers both $B(p)$ and $B(c_i)$ as $c_i \in B(q_i)$, which contradicts that the instance is well-separated.

2) Note that for $1 \le i \le 3$, $B(c_i) \cup \textbf{Gain}(c_i, q_i) \subseteq \mathscr{F}(q_i)$, and that $B(c_i)$ and $\textbf{Gain}(c_i, q_i)$ are disjoint. The balls $B(c_i)$ cover at least $|B \cap \textbf{Guess}|$ blue points and $|R \cap \textbf{Guess}|$ red points, while $\sum_{i=1}^3 |\textbf{Gain}(c_i, q_i) \cap X_i| \ge 3\tau$. □

3.2.3 Phase II

Throughout this section we assume c_1, c_2, and c_3 have been guessed correctly in Phase I so that the properties of Lemma 3.2.6 hold. Furthermore, by the selection and the definition of τ, we also have

$$|\textbf{Gain}(p, q) \cap X_4| \le \tau \qquad \text{for any } p \in X_4 \cap OPT \text{ and } q \in B(p) \cap X_4. \tag{3.1}$$

This implies that $\mathscr{F}(p) \setminus B(p)$ contains at most τ red points of X_4. However, to apply Lemma 3.2.2 we need that the number of red points of X_4 in the whole flower $\mathscr{F}(p)$ is bounded. To deal with balls with many more than τ red points, we will iteratively remove *dense* sets from X_4 to obtain a subset X_s of *sparse* points.

Definition 3.2.7. *When considering a subset of the points $X_s \subseteq X$, we say that a point $j \in X_s$ is* ***dense*** *if the ball $B(j)$ contains strictly more than $2 \cdot \tau$ red points of X_s. For a dense point j, we also let $I_j \subseteq X_s$ contain those points $i \in X_s$ whose intersection $B(i) \cap B(j)$ contains strictly more than τ red points of X_s.*

We remark that in the above definition, we have in particular that $j \in I_j$ for a dense point $j \in X_s$. Our iterative procedure now works as follows:

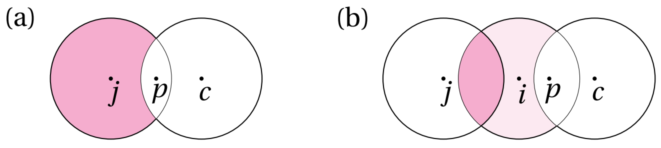

Figure 3.2: The shaded regions are subsets of **Gain**(c,p), which contain the darkly shaded regions that have $> \tau$ red points.

Initially, let $I = \emptyset$ and $X_s = X_4$. While there is a dense point $j \in X_s$:

- Add I_j to I and update X_s by removing the points $D_j = \cup_{i \in I_j} B(i) \cap X_s$.

Let $X_d = X_4 \setminus X_s$ denote those points that were removed from X_4. We will cluster the two sets X_s and X_d of points separately. Indeed, the following lemma says that a center in $OPT \setminus \{c_i\}_{i=1}^3$ either covers points in X_s or X_d but not points from both sets. Recall that D_j denotes the set of points that are removed from X_s in the iteration when j was selected and so $X_d = \cup_j D_j$.

Lemma 3.2.8. *For any $c \in OPT \setminus \{c_i\}_{i=1}^3$ and any $I_j \in I$, either $c \in I_j$ or $B(c) \cap D_j = \emptyset$.*

Proof. Let $c \in OPT \setminus \{c_i\}_{i=1}^3$, $I_j \in I$, and suppose $c \notin I_j$. If $B(c) \cap D_j \neq \emptyset$, there is a point p in the intersection $B(c) \cap B(i)$ for some $i \in I_j$. Suppose first that $B(c) \cap B(j) \neq \emptyset$. Then, since $c \notin I_j$, the intersection $B(c) \cap B(j)$ contains fewer than τ red points from D_j (recall that D_j contains the points of $B(j)$ in X_s at the time j was selected). But by the definition of dense clients, $B(j) \cap D_j$ has more than $2 \cdot \tau$ red points, so $(B(j) \setminus B(c)) \cap D_j$ has more than τ red points. This region is a subset of **Gain**$(c,p) \cap X_4$, which contradicts (3.1). This is shown in Figure 3.2(a). Now consider the second case when $B(c) \cap B(j) = \emptyset$ and there is a point p in the intersection $B(c) \cap B(i)$ for some $i \in I_j$ and $i \neq j$. Then, by the definition of I_j, $B(i) \cap B(j)$ has more than τ red points of D_j. However, this is also a subset of **Gain**$(c,p) \cap X_4$ so we reach the same contradiction. See Figure 3.2(b). □

Our algorithm now proceeds by guessing the number k_d of balls of $OPT \setminus \{c_i\}_{i=1}^3$ contained in X_d. We also guess the numbers r_d and b_d of red and blue points, respectively, that these balls cover in X_d. Note that after guessing k_d, we know that the number of balls in $OPT \setminus \{c_i\}_{i=1}^3$ contained in X_s equals $k_s = k - 3 - k_d$. Furthermore, by the first property of Lemma 3.2.6, these balls cover at least $b_s = b - |B \cap \textbf{Guess}| - b_d$ blue points in X_s and at least $r_s = r - |R \cap \textbf{Guess}| - r_d$ red points in X_s. As there are $O(n^3)$ possible values of k_d, b_d, and r_d (each can take a value between 0 and n) we can try all possibilities by increasing the running time by a multiplicative factor of $O(n^3)$. Henceforth, we therefore assume that we have guessed those parameters correctly. In that case, we show that we can recover an equally good solution for X_d and a solution for X_s that covers b_s blue points and almost r_s red points:

Lemma 3.2.9. *There exist two polynomial-time algorithms $\mathscr{A}_d$ and $\mathscr{A}_s$ such that if k_d, r_d, and b_d are guessed correctly then*

- *$\mathscr{A}_d$ returns k_d balls of radius one that cover b_d blue points of X_d and r_d red points of X_d;*
- *$\mathscr{A}_s$ returns k_s balls of radius two that cover at least b_s blue points of X_s and at least $r_s - 3\cdot\tau$ red points of X_s.*

Proof. We first describe and analyze the algorithm $\mathscr{A}_d$ followed by $\mathscr{A}_s$.

The algorithm $\mathscr{A}_d$ for the dense point set X_d. By Lemma 3.2.8, we have that all k_d balls in $OPT \setminus \{c_i\}_{i=1}^3$ that cover points in X_d are centered at points in $\cup_j I_j$. Furthermore, we have that each I_j contains at most one center of OPT. This is because every $i \in I_j$ is such that $B(i) \cap B(j) \neq \emptyset$ and so, by the triangle inequality, $B(j,3)$ contains all balls $\{B(i)\}_{i\in I_j}$. Hence, by the assumption that the instance is well-separated, the set I_j contains at most one center of OPT.

We now reduce our problem to a 3-dimensional subset-sum problem. For each $I_j \in I$, form a group consisting of an item for each $p \in I_j$. The item corresponding to $p \in I_j$ has the 3-dimensional value vector $(1, |B(p) \cap D_j \cap B|, |B(p) \cap D_j \cap R|)$. Our goal is to find k_d items such that at most one item per group is selected and their 3-dimensional vectors sum up to (k_d, b_d, r_d). Such a solution, if it exists, can be found by standard dynamic programming that has a table of size $O(n^4)$. For completeness, we provide the recurrence and precise details of this standard technique in Appendix A.1. Furthermore, since the D_j's are disjoint by definition, this gives k_d centers that cover b_d blue points and r_d red points in X_d, as required in the statement of the lemma.

It remains to show that such a solution exists. Let $o_1, o_2, \ldots, o_{k_d}$ denote the centers of the balls in $OPT \setminus \{c_i\}_{i=1}^3$ that cover points in X_d. Furthermore, let $I_{j_1}, \ldots, I_{j_{k_d}}$ be the sets in I such that $o_i \in I_{j_i}$ for $i \in \{1, \ldots, k_d\}$. Notice that by Lemma 3.2.8 we have that $B(o_i) \cap X_d$ is disjoint from $X_d \setminus D_{j_i}$ and contained in D_{j_i}. It follows that the 3-dimensional vector corresponding to an OPT center o_i equals $(1, |B(p) \cap X_d \cap B|, |B(p) \cap X_d \cap R|)$. Therefore, the sum of these vectors corresponding to $o_1, \ldots, o_{k_d}$ results in the vector (k_d, b_d, r_d), where we used that our guesses of k_d, b_d, and r_d were correct.

The algorithm $\mathscr{A}_s$ for the sparse point set X_s. Assuming that the guesses are correct we have that $OPT \setminus \{c_i\}_{i=1}^3$ contains k_s balls that cover b_s blue points of X_s and r_s red points of X_s. Hence, LP1 has a feasible solution (x,z) to the instance defined by the point set X_s, the number of balls k_s, and the constraints b_s and r_s on the number of blue and red points to be covered, respectively. Lemma 3.2.2 then says that we can in polynomial-time find k_s balls of radius two such that at least b_s blue balls of X_s are covered and at least

$$r_s - \max_{j: z_j > 0} |\mathscr{F}(j) \cap R|$$

red points of X_s are covered. Here, $\mathscr{F}(j)$ refers to the flower restricted to the point set X_s.

To prove the the second part of Lemma 3.2.9, it is thus sufficient to show that LP1 has a feasible solution where $z_j = 0$ for all $j \in X_s$ such that $|\mathscr{F}(j) \cap R| > 3 \cdot \tau$. In turn, this follows by showing that, for any such $j \in X_s$ with $|\mathscr{F}(j) \cap R| > 3 \cdot \tau$, no point in $B(j)$ is in OPT (since then $z_j = 0$ in the integral solution corresponding to OPT). Such a feasible solution can be found by adding $x_i = 0\ \forall i \in B(j)$ for all such j to LP1.

To see why this holds, suppose towards a contradiction that there is a $c \in OPT$ such that $c \in B(j)$. First, since there are no dense points in X_s, we have that the number of red points in $B(c) \cap X_s$ is at most $2 \cdot \tau$. Therefore the number of red points of X_s in $\mathscr{F}(j) \setminus B(c)$ is strictly more than τ. In other words, we have $\tau < |\mathbf{Gain}(c, j) \cap X_s| \le |\mathbf{Gain}(c, j) \cap X_4|$ which contradicts (3.1). □

Equipped with the above lemma we are now ready to finalize the proof of Theorem 2.

Proof of Theorem 2. Our algorithm guesses the optimal radius and the centers c_1, c_2, c_3 in Phase I, and k_d, r_d, b_d in Phase II. There are at most $\binom{n}{2}$ choices of the optimal radius, n choices for each c_i, and $n+1$ choices of k_d, r_d, b_d (ranging from 0 to n). We can thus try all these possibilities in polynomial time and, since all other steps in our algorithm run in polynomial time, the total running time will be polynomial. The algorithm tries all these guesses and outputs the best solution found over all choices. For the correct guesses, we output a solution with $3 + k_d + k_s = k$ balls of radius at most two. Furthermore, by the second property of Lemma 3.2.6 and the two properties of Lemma 3.2.9, we have that

- the number of blue points covered is at least $|B \cap \mathbf{Guess}| + b_d + b_s = b$; and
- the number of red points covered is at least $|R \cap \mathbf{Guess}| + 3\tau + r_d + r_s - 3\tau = r$.

We have thus given a polynomial-time algorithm that returns a solution where the balls are of radius at most twice the optimal radius. □

3.3 Constant Number of Colors

Our algorithm extends easily to a constant number ω of color classes $\mathscr{C}_1, \ldots, \mathscr{C}_\omega$ with coverage requirements $p_1, \ldots, p_\omega$. We use the LPs in Fig. 3.3 for a general number of colors, where $p_{j,i}$ in LP2(ω) indicates the number of points of color class i in cluster $j \in S$. S is the set of cluster centers obtained from modified clustering algorithm in Appendix A.2 to instances with ω color classes. LP2(ω) has only ω non-trivial constraints, so any extreme point has at most ω variables attaining strictly fractional values, and a feasible solution attaining objective value at least p_1 will have at most $k + \omega - 1$ positive values. By rounding up to 1 the fractional value of the center that contains the most number of points of $\mathscr{C}_\omega$, we can cover p_ω points of $\mathscr{C}_\omega$. We would like to be able to close the remaining fractional centers, so we apply an analogous procedure to the case with just two colors.

LP1(ω)

$$\begin{aligned}
&\sum_{m\in B(i)} x_m \geq z_i, \quad \forall i\in X\\
&\sum_{i\in X} x_i \leq k\\
&\sum_{i\in C_j} z_i \geq p_j, \quad \forall 1\leq j\leq \omega\\
&z_i, x_i \in [0,1], \quad \forall i \in X.
\end{aligned}$$

LP2(ω)

$$\begin{aligned}
\text{maximize} \quad & \sum_{i\in S} p_{1,i} y_i\\
\text{subject to} \quad & \sum_{i\in S} p_{j,i} y_i \geq p_j, \quad \forall 2\leq j\leq \omega\\
& \sum_{i\in S} y_i \leq k,\\
& y_i \in [0,1] \qquad \forall i \in S.
\end{aligned}$$

Figure 3.3: Linear programs for ω color classes.

We can guess $3(\omega-1)$ centers of OPT for each of the $\omega-1$ colors whose coverage requirements are to be satisfied. Then we bound the number of points of each color that may be found in a cluster, by removing dense sets that contain too many points of any one color and running a dynamic program on the removed sets. The final step is to run the clustering algorithm of [Ban+19] on the remaining points, and rounding to one the fractional center with the most number of points of $\mathscr{C}_1$, and closing all other fractional centers.

In particular, we get a running time with a factor of $n^{O(\omega^2)}$. The remainder of this section gives a formal description of the algorithm for ω color classes.

3.3.1 Formal algorithm for ω colors

The following is a natural generalization of Lemma 3.2.2 and summarizes the main properties of the clustering algorithm of Appendix A.2 for instances with ω color classes.

Lemma 3.2.2′. *Given a fractional solution (x,z) to LP1(ω), there is a polynomial-time algorithm that outputs at most k clusters of radius two that cover at least p_ω points of $\mathscr{C}_\omega$, and at least $p_i-(\omega-1)\max_{j:z_j>0}|\mathscr{F}(j)\cap\mathscr{C}_i|$ for $2\leq i\leq\omega$.*

Since we may not meet the coverage requirements for $\omega-1$ color classes, it is necessary to guess some balls of OPT for each of those colors, and for each fractional center. In total we guess $3(\omega-1)^2$ points of OPT as follows:

For $j=2,\dots,\omega$, for $i=1,2,\dots,3(\omega-1)$ guess the center $c_{j,i}$ in OPT and calculate the point $q_{j,i}\in B(c_{j,i})$ such that $|\mathscr{C}_j\cap\mathbf{Gain}(c_{j,i},q_{j,i})\cap X_{j,i}|$ is maximized over all possible $c_{j,i}$, where

$$\begin{aligned}
X_{j,1} &= X\\
X_{j,i} &= X_{j,i-1}\setminus\left(\mathscr{C}_i\cap\mathscr{F}(q_{j,i-1})\right) \quad \text{for } 2\leq i\leq 3(\omega-1)+1.
\end{aligned}$$

This guessing takes $O(n^{3(\omega-1)^2})$ rounds. It is possible that some $c_{j,i}$ coincide, but this does not affect the correctness of the algorithm. In fact, this can only improve the solution, in the sense that the coverage requirements will be met with fewer than k centers. Let k_c denote the number of distinct $c_{j,i}$ obtained in the correct guess. For notation, define

$$\begin{aligned}\textbf{Guess} &:= \cup_{j=2}^{\omega} \cup_{i=1}^{3(\omega-1)} B(c_{j,i})\\ \tau_j &= \left|\mathscr{C}_j \cap \textbf{Gain}(c_{j,3(\omega-1)}, q_{j,3(\omega-1)}) \cap X_{j,3(\omega-1)}\right|.\end{aligned}$$

To be consistent with previous notation, let

$$X_4 := X \setminus \cup_{j=2}^{\omega} \cup_{i=1}^{3(\omega-1)} \mathscr{F}(q_{j,i}).$$

The important properties guaranteed by the first phase can be summarized in the following lemma whose proof is the natural extension of Lemma 3.2.6.

Lemma 3.2.6′. *Assuming that $c_{j,i}$ are guessed correctly, we have that*

1. *the $k-3(\omega-1)^2$ balls of radius one in $OPT \setminus \cup_{j=2}^{\omega} \cup_{i=1}^{3(\omega-1)} \{c_{j,i}\}$ are contained in X_4 and cover $p_\omega - |\mathscr{C}_\omega \cap \textbf{Guess}|$ of points in $\mathscr{C}_\omega$ and $p_j - |\mathscr{C}_j \cap \textbf{Guess}|$ points of $\mathscr{C}_j$ for $j = 2,\dots,\omega$; and*
2. *the clusters $\mathscr{F}(q_{j,i})$ are contained in $X \setminus X_{3(\omega-1)+1}$ and cover at least $|\mathscr{C}_\omega \cap \textbf{Guess}|$ points of $\mathscr{C}_\omega$ and at least $|\mathscr{C}_j \cap \textbf{Guess}| + 3(\omega-1)\cdot\tau_j$ points of $\mathscr{C}_j$.*

Now we need to remove points which contain many points from any one of the color classes to partition the instance into dense and sparse parts which leads to the following generalized definition of dense points.

Definition 3.2.7′. *When considering a subset of the points $X_s \subseteq X$, we say that a point $p \in X_s$ is j-**dense** if $|\mathscr{C}_j \cap B(p) \cap X_s| > 2\tau_j$. For a j-dense point p, we also let $I_p \subseteq X_s$ contain those points $i \in X_s$ such that $|\mathscr{C}_j \cap B(i) \cap B(p) \cap X_s| > \tau_j$, **for every** $2 \le j \le \omega$.*

Now we perform a similar iterative procedure as for two colors:

Initially, let $I = \emptyset$ and $X_s = X_{3(\omega-1)}$. While there is a j-dense point $p \in X_s$ for any $2 \le j \le \omega$:

- Add I_p to I and update X_s by removing the points $D_p = \cup_{i\in I_p} B(i) \cap X_s$.

As in the case of two colors, set $X_d = X_{3(\omega-1)} \setminus X_s$. By naturally extending Lemma 3.2.8 and its proof, we can ensure that any ball of $OPT \setminus \cup_{j=2}^{\omega} \cup_{i=1}^{3(\omega-1)} \{c_{j,i}\}$ is completely contained in either X_d or X_s. We guess the number k_d of such balls of OPT contained in X_d, and guess the numbers $d_1,\dots,d_\omega$ of points of $\mathscr{C}_1,\dots,\mathscr{C}_\omega$ covered by these balls in X_d. There are $O(n^{\omega+1})$ possible values of $k_d, d_1,\dots,d_\omega$ and all the possibilities can be tried by increasing the running time by a multiplicative factor. The number of balls of $OPT \setminus \cup_{j=2}^{\omega} \cup_{i=1}^{3(\omega-1)} \{c_{j,i}\}$ contained in X_s is given by $k_s = k - k_c - k_d$ and these balls cover at least $s_j = p_j - |\mathscr{C}_j \cap \mathbf{Guess}_{all}| - d_j$ points of $\mathscr{C}_j$ in X_s, $1 \le j \le \omega$.

Assuming that the parameters are guessed correctly we can show, similar to Lemma 3.2.9, that the following holds.

Lemma 3.2.9′. *There exist two polynomial-time algorithms $\mathscr{A}'_d$ and $\mathscr{A}'_s$ such that if $k_d, d_1, \ldots d_\omega$ are guessed correctly then*

- *$\mathscr{A}'_d$ returns k_d balls of radius one that cover $d_1, \ldots, d_\omega$ points of $\mathscr{C}_1, \ldots, \mathscr{C}_\omega$ of X_d;*
- *$\mathscr{A}'_s$ returns k_s balls of radius two that cover at least s_1 points of $\mathscr{C}_1$ of X_s and at least $s_j - 3(\omega - 1) \cdot \tau_j$ points of $\mathscr{C}_j$ of X_s, $2 \le j \le \omega$.*

The algorithm $\mathscr{A}'_d$ proceeds as did $\mathscr{A}_d$, with the modification that the dynamic program is now $(\omega + 1)$-dimensional. Algorithm $\mathscr{A}'_s$, is also similar to $\mathscr{A}_s$, because LP1 has a feasible solution where $z_p = 0$ for all $p \in X_s$ such that $|\mathscr{F}(p) \cap \mathscr{C}_j| > 3\tau_j$ holds for any $2 \le j \le \omega$. Hence, we output a solution with $k_c + k_d + k_s = k$ balls of radius at most two, and

- the number of points of $\mathscr{C}_1$ covered is at least $|\mathscr{C}_1 \cap \mathbf{Guess}| + d_1 + s_1 = p_1$; and
- the number of points of $\mathscr{C}_j$ covered is at least $|\mathscr{C}_j \cap \mathbf{Guess}| + 3(\omega - 1)\tau_j + d_j + s_j - 3(\omega - 1)\tau_j = p_j$, for all $j = 2, \ldots, \omega$.

This is a polynomial-time algorithm for colorful k-center with a constant number of color classes.

3.4 LP Integrality Gaps

In this section, we present two natural ways to strengthen LP1 and show that they both fail to close the integrality gap, providing evidence that clustering and knapsack feasibility cannot be decoupled in the colorful k-center problem. On one hand, the Sum-of-Squares hierarchy is ineffective for knapsack problems, while on the other hand, adding knapsack constraints to LP1 is also insufficient due to the clustering aspect of this problem.

3.4.1 Sum-of-squares integrality gap

The Sum-of-Squares hierarchy (equivalently Lasserre [Las01a; Las01b]) is a method of strengthening linear programs that has been used in constraint satisfaction problems, set-cover, and graph coloring, to just name a few examples [AG11; CFG12; Tul09]. We use the same notation for the Sum-of-Squares hierarchy, abbreviated as SoS, as in Karlin et al.

[KMN11]. For a set V of variables, $\mathscr{P}(V)$ are the power sets of V and $\mathscr{P}_t(V)$ are the subsets of V of size at most t. Their succinct definition of the hierarchy makes use of the *shift operator*: for two vectors $x, y \in \mathbb{R}^{\mathscr{P}(V)}$ the **shift operator** is the vector $x * y \in \mathbb{R}^{\mathscr{P}(V)}$ such that

$$(x * y)_I = \sum_{J \subseteq V} x_J y_{I \cup J}.$$

Analogously, for a polynomial $g(x) = \sum_{I \subseteq V} a_I \prod_{i \in I} x_i$ we have $(g * y)_I = \sum_{J \subseteq V} a_J y_{I \cup J}$. In particular, we work with the linear inequalities $g_1, \dots, g_m$ so that the polytope to be lifted is

$$K = \{x \in [0,1]^n : g_\ell(x) \geq 0 \text{ for } \ell = 1, \dots, m\}.$$

Let $\mathscr{T}$ be a collection of subsets of V and y a vector in $\mathbb{R}^{\mathscr{T}}$. The matrix $M_{\mathscr{T}}(y)$ is indexed by elements of $\mathscr{T}$ such that

$$(M_{\mathscr{T}}(y))_{I,J} = y_{I \cup J}.$$

We can now define the t-th SoS lifted polytope.

Definition 3.4.1. *For any $1 \leq t \leq n$, the t-th SoS lifted polytope $SoS^t(K)$ is the set of vectors $y \in [0,1]^{\mathscr{P}_{2t}(V)}$ such that $y_\emptyset = 1$, $M_{\mathscr{P}_t(V)}(y) \succeq 0$, and $M_{\mathscr{P}_{t-1}(V)}(g_\ell * y) \succeq 0$ for all ℓ.*

A point $x \in [0,1]^n$ belongs to the t-th SoS polytope $SoS^t(K)$ if there exists $y \in SoS^t(K)$ such that $y_{\{i\}} = x_i$ for all $i \in V$.

We use a reduction from Grigoriev's SoS lower bound for knapsack [Gri01] to show that the following instance has a fractional solution with small radius that is valid for a linear number of rounds of SoS.

Theorem 3 (Grigoriev)**.** *At least $\min\{2\lfloor \min\{k/2, n-k/2\} \rfloor + 3, n\}$ rounds of SoS are required to recognize that the following polytope contains no integral solution for $k \in \mathbb{Z}$ odd.*

$$\sum_{i=1}^{n} 2w_i = k$$
$$w_i \in [0,1] \ \ \forall i.$$

Consider an instance of colorful k-center with two colors, $8n$ points, $k = n$, and $r = b = 2n$ where n is odd. Points $\{4i-3, 4i-2, 4i-1, 4i\} \forall i \in [2n]$ belong to cluster C_i of radius one. For odd i, C_i has three red points and one blue point and for even i, C_i has one red point and three blue points. A picture is shown in Figure 3.4. In an optimal integer solution, one center needs to cover at least 2 of these clusters while a fractional solution satisfying LP1 can open a center of $1/2$ around each cluster of radius 1. Hence, LP1 has an unbounded integrality gap since the clusters can be arbitrarily far apart. This instance takes an odd number of copies of the integrality gap example given in [Ban+19].

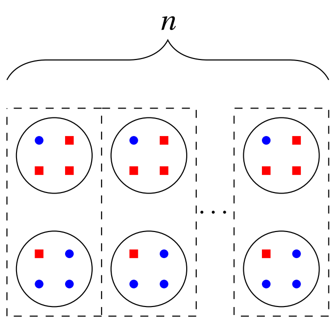

Figure 3.4: Integrality gap example for linear rounds of SoS

We can do a simple mapping from a feasible solution for the tth round of SoS on the system of equations in Theorem 3 to our variables in the tth round of SoS on LP1 for this instance to demonstrate that the infeasibility of balls of radius one is not recognized. More precisely, we assign a variable w_i to each pair of clusters of radius one as shown in Figure 3.4, corresponding to opening each cluster in the pair by w_i amount. Then a fractional opening of balls of radius one can be mapped to variables that satisfy the polytope in Theorem 3. The remainder of this subsection is dedicated to formally describing the reduction from Theorem 3.
Let W denote the set of variables used in the polytope defined in Theorem 3. Let w be in the t-th round of SoS applied to the system in Theorem 3 so that w is indexed by subsets of W of size at most t. Let $V = V_x \cup V_z$, where $V_x = \{x_1, \dots, x_{8n}\}$ and $V_z = \{z_1, \dots, z_{8n}\}$, be the set of variables used in LP1 for the instance shown in Figure 3.4. We define vector y with entries indexed by subsets of V, and show that y is in the t-th SoS lifting of LP1. In each ball we pick a representative x_i, $i \equiv 1 \mod 4$, to indicate how much the ball is opened, so we set $y_I = 0$ if $x_j \in I$, $j \not\equiv 1 \mod 4$. Otherwise, we set $y_I = w_{\pi(I)}$ where

$$\pi(I) = \{w_i : x_{8i-3} \text{ or } x_{8i-7} \text{ or } z_{8i-j} \in I, \text{ for some } i \in [n], j \in [7]\}.$$

We have $M_{\mathscr{P}_t(W)}(w) \succeq 0$, and for $g_1 = -n + \sum_{i=1}^n 2x_i$ and $g_2 = n - \sum_{i=1}^n 2x_i$, $M_{\mathscr{P}_{t-1}(W)}(g_\ell * w) \succeq 0$ for $\ell = 1, 2$ since w satisfies the t-th round of SoS. This implies that $M_{\mathscr{P}_{t-1}(W)}(g_\ell * w)$ is the zero matrix.

To show that $M_{\mathscr{P}_t(V)}(y) \succeq 0$, we start with $M_{\mathscr{P}_t(W)}(w)$ and construct a sequence of matrices such that the semidefiniteness of one implies the semidefiniteness of the next, until we arrive at a matrix that is $M_{\mathscr{P}_t(V)}(y)$ with rows and columns permuted, i.e. $M_{\mathscr{P}_t(V)}(y)$ multiplied on the left and right by a permutation matrix and its transpose. Since the eigenvalues of a matrix are invariant under this operation, $M_{\mathscr{P}_t(W)}(w) \succeq 0$ implies that $M_{\mathscr{P}_t(V)}(y) \succeq 0$.

Lemma 3.4.2. *There exists a sequence of square matrices $M_{\mathscr{P}_t(W)}(w) := M_0, M_1, M_2, \dots, M_p$, such that the rank of M_i is the same as the rank of M_{i+1}, M_i is the leading principal submatrix of M_{i+1} of dimension one less, and M_p is $M_{\mathscr{P}_t(V)}(y)$ with rows and columns permuted.*

Proof. We claim that this sequence of matrices exists with the following description. Firstly, the matrix M_{i+1} has one extra row and column than M_i, and is the same on the leading principal submatrix of size M_i. Then there are two possibilities:

(a) The last row and column of M_{i+1} are all zeroes, or

(b) for some j, the last row of M_{i+1} is a copy of the jth row of M_i, the last column is a copy of the jth column of M_i, and the last entry is $(M_i)_{j,j}$.

Either way, the rank of M_{i+1} would be the same as the rank of M_i.

To prove this claim, it suffices to consider a sequence of indices of the matrix $M_{\mathscr{P}_t(V)}(y)$. The matrix M_0 in our sequence will be the submatrix of $M_{\mathscr{P}_t(V)}(y)$ indexed by the first k indices, where k is the dimension of $M_{\mathscr{P}_t(W)}(w)$, i.e. the number of subsets of W of size at most t. Each subsequent matrix M_i will be the submatrix of $M_{\mathscr{P}_t(V)}(y)$ indexed by the first $k+i$ indices. Note that the rows/columns of $M_{\mathscr{P}_t(V)}(y)$ can be considered to be indexed by all the subsets of V of size at most t. With this in mind, consider a sequence of subsets of V of size at most t with the following properties:

1. All subsets of $\{x_{8i-7} : i \in [n]\}$ of size at most t form a prefix of our sequence.

2. Each set index after the first has exactly one more element than some set index that came earlier in the sequence.

It is clear that it is possible to arrange all the subsets of V of size at most t in a sequence to satisfy these properties. It only remains to show that this sequence produces the desired construction for $M_0, M_1, \ldots, M_p$.

We have

$$\left(M_{\mathscr{P}_t(y)}\right)_{I,J} = y_{I\cup J} = w_{\pi(I\cup J)} = w_{\pi(I),\pi(J)}$$

so property (1) guarantees that we begin with M_0 being $M_{\mathscr{P}_t(W)}(w)$, up to the correct permutation of subsets of $\{x_{8i-7} : i \in [n]\}$. Now consider some k'th index in the sequence, $k' > k$ where k is the dimension of $M_{\mathscr{P}_t(W)}(w)$. By property (2), it is of the form $J \cup \{x\}$, where J is one of the first $k'-1$ indices, and $x \in V$. There are two cases:

- If x is some x_i with $i \not\equiv 1 \mod 4$, then $y_{I_\ell \cup J} = 0$ for all $\ell \leq k'$.

- Otherwise, $\pi(J \cup \{x\}) = \pi(J)$.

In the first case, the matrix constructed from the first k' indices will have property (a), and in the second, property (b). Finally, it is clear that at each step the dimension of the matrices increases by one, and that it is the leading principal submatrix of the following matrix in the sequence, until we end up with $M_{\mathscr{P}_t(V)}(y)$ (up to some permutation of its rows and columns). □

By the rank-nullity theorem, M_{i+1} has one more 0-eigenvalue than M_i, so we can apply the following theorem.

Theorem 4 (Cauchy's Interlace Theorem)**.** *Let A be a symmetric $n \times n$ matrix and B be a principal submatrix of A of dimension $(n-1) \times (n-1)$. If the eigenvalues of A are $\alpha_1 \geq \cdots \geq \alpha_n$ and the eigenvalues of B are $\beta_1 \geq \cdots \geq \beta_{n-1}$ then $\alpha_1 \geq \beta_1 \geq \alpha_2 \geq \beta_2 \geq \cdots \geq \alpha_{n-1} \geq \beta_{n-1} \geq \alpha_n$.*

With $M_{i+1} = A$ and $M_i = B$ as in Theorem 4 we have that $\alpha_n = 0$ (since M_{i+1} and M_i have the same eigenvalues but the dimension of the zero eigenspace of M_{i+1} is one greater than that of M_i). Hence, M_{i+1} has no negative eigenvalues if M_i has no negative eigenvalues. This is sufficient to show that each matrix in the sequence constructed is positive semidefinite, and concludes the proof that $M_{\mathscr{P}_t(V)}(y) \succeq 0$.

It remains to show that the matrices arising from the shift operator between y and the linear constraints of our polytope are positive semidefinite. Let h_i denote the linear inequalities in LP1. In essence, the corresponding moment matrices $M_{\mathscr{P}_{t-1}(V)}(h_i * y)$ are zero matrices since all h_i are tight for the example in Figure 3.4. Formally, we have

Lemma 3.4.3. *Matrices $M_{\mathscr{P}_{t-1}(V)}(h_\ell * y)$ are the zero matrix, for each h_ℓ a linear constraint from LP1.*

Proof. Let $h_{1,j}$ be the linear polynomial that corresponds to the first inequality of LP1 for $j \in X$. First, if $i \not\equiv 1 \mod 4$, then $y_{I \cup \{x_i\}} = 0$ for any $I \subseteq V$. Otherwise, we have

$$\begin{aligned}(M_{\mathscr{P}_{t-1}}(h_{1j} * y))_{I,J} &= \left(\sum_{i \in B(j,1)} y_{I \cup J \cup \{x_i\}}\right) - y_{I \cup J \cup \{z_j\}} \\ &= w_{\pi(I \cup J) \cup \pi(x_i)} - w_{\pi(I \cup J) \cup \pi(z_j)} = 0\end{aligned}$$

since $\pi(\{x_i\}) = \pi(z_j)$ for $i \in B(j,1)$, $i \equiv 1 \mod 4$. For the remaining inequalities of LP1: h_2, h_3, and h_4, we have that $M_{\mathscr{P}_{t-1}(V)}(h_\ell * y)$ is the zero matrix because of how we defined the projection onto w:

$$\begin{aligned}(M_{\mathscr{P}_{t-1}}(h_2 * y))_{I,J} &= n y_{I \cup J} - \sum_{x_j \in V_x} y_{I \cup J \cup \{x_j\}} \\ &= n w_{\pi(I \cup J)} - \sum_{j=1}^{n} 2 w_{\pi(I \cup J \cup \{w_j\})} \\ &= (M_{\mathscr{P}_{t-1}}(g_2 * w))_{\pi(I),\pi(J)} = 0 \\ M_{\mathscr{P}_{t-1}}(h_3 * y))_{I,J} &= M_{\mathscr{P}_{t-1}}(h_4 * y))_{I,J}\end{aligned}$$

$$
\begin{aligned}
&= \left(\sum_{j \in R} y_{I \cup J \cup \{z_j\}} \right) - 2n y_{I \cup J} \\
&= \left(\sum_{i=1}^{n} 4 w_{\pi(I \cup J) \cup \{w_i\}} \right) - 2n w_{\pi(I \cup J)} \\
&= 2(M_{\mathscr{P}_{t-1}}(g_1 * w))_{\pi(I),\pi(J)} = 0.
\end{aligned}
$$

□

This concludes the formal proof of the following theorem.

Theorem 5. *The integrality gap of LP1 with* $8n$ *points persists up to* $\Omega(n)$ *rounds of Sum-of-Squares.* □

3.4.2 Flow constraints

In this section we add additional constraints based on standard techniques to LP1. These incorporate knapsack constraints for the fractional centers produced in the hope of obtaining a better clustering and show that this fails to reduce the integrality gap.

We define an instance of a knapsack problem with multiple objectives. Each point $p \in X$ corresponds to an item with three dimensions: a dimension of size one to restrict the number of centers, $|B \cap B(p)|$, and $|R \cap B(p)|$. We set up a flow network with an $(n+1) \times n \times n \times k$ grid of nodes and we name the nodes with the coordinate (w, x, y, z) of its position. The source s is located at $(0,0,0,0)$ and we add an extra node t for the sink. Assign an arbitrary order to the points in X. For the item corresponding to $i \in X$, for each $x \in [n]$, $y \in [n]$, $z \in [k]$:

1. Add an edge from (i, x, y, z) to $(i+1, x, y, z)$ with flow variable $e_{i,x,y,z}$.
2. With $b_i := |B \cap B(i)|$ and $r_i := |R \cap B(i)|$, if $z < k$ add an edge from (i, x, y, z) to $(i+1, \min\{x + b_i, n\}, \min\{y + b_i, n\}, z+1)$ with flow variable $f_{i,x,y,z}$.

For each $x \in [b, n]$, $y \in [r, n]$:

3. Add an edge from $(n+1, x, y, k)$ to t with flow variable $g_{x,y}$.

Set the capacities of all edges to one. In addition to the usual flow constraints, add to LP1 the constraints

$$x_i = \sum_{x,y \in [n], z \in [k]} f_{i,x,y,z} \quad \text{for all } i \in X \tag{3.2}$$

$$1 - x_i = \sum_{x,y \in [n], z \in [k]} e_{i,x,y,z} \quad \text{for all } i \in X. \tag{3.3}$$

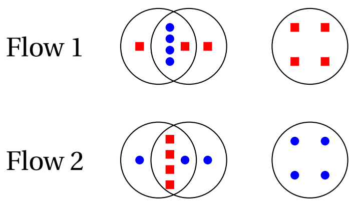

Figure 3.5: $k = 3$, $r = b = 8$

We refer to the resulting linear program as LP3. Notice that an integral solution to LP1 defines a path from s to t through which one unit of flow can be sent; hence LP3 is a valid relaxation. On the other hand, any path P from s to t defines a set C_P of at most k centers by taking those points c for which $f_{c,x,y,z} \in P$ for some x, y, and z. Moreover, as t can only be reached from a coordinate with $x \geq b$ and $y \geq r$ we have that $\sum_{c \in C_p} |B(c) \cap B| \geq b$ and $\sum_{c \in C_p} |B(c) \cap R| \geq r$. It follows that C_P forms a solution to the problem of radius one *if the balls are disjoint*. In particular, our integrality gap instances for the Sum-of-Squares hierarchy do not fool LP3.

The example in Figure 3.5 shows that in an instance where balls overlap, the integrality gap remains large. Here, the fractional assignment of open centers is 1/2 for each of the six balls and this gives a fractional covering of 8 red and 8 blue points as required. This assignment also satisfies the flow constraints because the three balls at the top of the diagram define a path disjoint from the three at the bottom. By double counting the five points in the intersection of two balls we cover 8 red and 8 blue points with each set of three balls. Hence, we can send flow along each path. However, this does not give a feasible integral solution with three centers as any set of three clusters does not contain enough points. In fact, the four clusters can be placed arbitrarily far from each other and in this way we have an unbounded integrality gap since one ball needs to cover two clusters.

4 The Non-Uniform k-Center Problem with Three Types of Radii

This chapter is based on joint work with Lars Rohwedder, Kshiteej Sheth, and Ola Svensson [Jia+22]. It has been accepted to the SIAM Symposium on Simplicity in Algorithms (**SOSA 2022**) under the title

Towards Non-Uniform k-Center with Constant Types of Radii.

4.1 Introduction

Clustering is a classic topic in algorithms and theoretical computer science. The k-center problem [Gon85; HS85] is a well-studied formulation of clustering, where one wants to cover points in a metric space with balls of minimum radius around k of them. This problem has been investigated under multiple generalizations such as with outliers [Cha+01] and multiple color classes [Ban+19; JSS21a; Ane+21]. From the viewpoint of k-center being a location and routing problem, classic k-center represents minimizing the maximum service time, assuming the speed of service is uniform at all locations. Chakrabarty, Goyal, and Krishnaswamy (CGK) [CGK16] introduce the *non-uniform k-center* problem to capture varying speeds at different locations. In other words, the k balls come with radii of varying sizes. The formal definition is as follows.

Definition 4.1.1 (The t-non-uniform k-center problem (t-NUkC))**.** The input is a metric space (X, d) and radii $r_1 \geq r_2 \geq \cdots \geq r_t$ with k_i balls of radius r_i. The objective is to find k_i centers $C_i \subseteq X$, $i = 1, \dots, t$, so that balls of radius αr_i around C_i, $i = 1, \dots, t$, cover all of the points in X and α is minimized.

The *robust* t-NUkC problem is a generalization of t-NUkC to incorporate the case of outliers, i.e. where one needs to cover only a certain number of points.

Definition 4.1.2 (Robust t-NUkC)**.** This problem is the same as the t-NUkC problem except for an extra parameter m and one needs to cover only m many of the points in X.

It is easy to observe that robust $(t-1)$-NUkC is a special case of t-NUkC with $|X|-m$ balls of radius 0. In [CGK16], the authors gave a $(1+\sqrt{5})$-approximation for 2-NUkC and a 2-approximation for robust 1-NUkC. They also showed that no constant-factor approximation is possible when t is part of the input, assuming $\mathbf{P} \neq \mathbf{NP}$. Further, the authors conjectured an $O(1)$-approximation to be possible when $t = O(1)$. Recently, Chakrabarty and Negahbani (CN) [CN21] obtained an important result making progress towards this conjecture. They obtained a 10-approximation for robust 2-NUkC, which is a special case of 3-NUkC. However, their techniques do not seem to extend for 3-NUkC and they state in their paper that new ideas would be needed to make further progress. We show a simple reduction in this paper from t-NUkC to robust $(t-1)$-NUkC for all t that loses only a constant factor in the approximation guarantee. This together with the algorithm of [CN21] for robust 2-NUkC implies a simple constant-approximation for 3-NUkC.

Theorem 6. *If there is an α-approximation for robust $(t-1)$-NUkC, then there is a $(2\alpha+2)$-approximation for t-NUkC.*

Thus, the 10-approximate algorithm of [CN21] for robust 2-NUkC implies a 22-approximate algorithm for 3-NUkC. Since no constant approximation was known when $t \geq 3$, this makes further progress on the conjecture of [CGK16]. We also note that using the 2-approximation algorithm of [CGK16] or [Ban+19] for k-center with outliers, Theorem 6 also gives a simpler alternate 6-approximation algorithm for 2-NUkC as compared to the algorithm of [CGK16].

Comparison of previous work and our approach. We briefly describe the approach of [CN21] and compare it with ours. The 10-approximation algorithm of [CN21] for robust 2-NUkC uses a connection to the *firefighter on trees* problem initially developed in [CGK16] and employs a multi-layered *round-or-cut* procedure using the ellipsoid algorithm. Given a fractional solution x to an instance of robust 2-NUkC, [CN21] obtains an instance of a 2-layered firefighter problem with a corresponding fractional solution y. This firefighter instance has the property that an integral solution to it would imply an approximate solution to the initial robust 2-NUkC instance. The top layer in the firefighter instance corresponds to potential centers for balls of the larger radius and the bottom layer corresponds to potential centers for balls of the smaller radius. They show that if y does not put too much mass on vertices of the top layer then one can easily obtain an integral solution to the firefighter instance, which in turn gives an approximate solution for the original robust 2-NUkC instance. In the other case, they show that if there is an integral solution that puts a lot of mass on the top layer, then one can reduce the original instance to a *well-separated* instance with respect to to balls of the larger radii, that is, balls of the larger radii are only allowed to be placed on a specified set of points that have large pairwise distance. If this instance is infeasible, then it implies that every integral solution of the original instance does not put too much mass on the top layer of the firefighter instance. This can be used to obtain a linear inequality violated by x but satisfied by every integral solution of the original instance, which is then fed back to the ellipsoid algorithm in the round-or-cut framework. [CN21] then designs an algorithm that either returns an approximate solution for a well-separated instance with respect to the larger radius or proves that the instance is infeasible, by exploiting the fact that balls of the larger radius do not intersect and interact as they are only allowed to be centered on points that have large pairwise distance. This algorithm is again based on the round-or-cut framework.

The algorithm of [CN21] proceeds in a top-down fashion in the following sense: They first reduce a general instance to a well-separated one with respect to the larger radius and then proceed by solving such an instance. On the contrary, our reduction that gives Theorem 6 is bottom-up. Given an instance of t-NUkC, we greedily partition the metric space into clusters of radius two times the smallest radius. These clusters are disjoint and thus the centers of these clusters are well-separated with respect to the smaller radius. This allows us to throw away information about the smallest radius and all points except the centers of these clusters to obtain an instance of robust $(t-1)$-NUkC, i.e., one type of radius is eliminated. In Section 4.3, we show how we can also obtain a 10-approximation for robust 2-NUkC that works in a bottom-up fashion as compared to the top-down approach of [CN21]. We also use a multi-layered round-or-cut approach. In our outer layer of the round-or-cut framework using the ellipsoid algorithm we reduce a general instance to an instance where balls of the smaller radius do not interact. Then using another layer of round-or-cut we reduce such a structured instance to a well-separated instance with respect to the larger radius. We observe that such an extremely structured instance is a *laminar instance* that can be solved using standard dynamic programming techniques. Our approach can be viewed as a bottom-up implementation of the algorithm of [CN21] with a simpler analysis. In particular, we do not need to prove Lemma 4 in [CN21], which essentially argues that if the firefighter instance obtained from a well-separated robust 2-NUkC instance with respect to the larger radius does not have an integral solution, then a certain linear inequality serves as a separating inequality.

Using the bottom-up view rather than a top-down one, we are able to obtain a simple reduction from t-NUkC to robust $(t-1)$-NUkC which in turn implies a simple constant approximation for 3-NUkC, thus making progress on the conjecture of [CGK16]. Secondly, with this view we are also able to design a bottom-up implementation of the 10-approximation algorithm of [CN21] for robust 2-NUkC that has a simpler analysis.

Preliminaries and Notation. For any vector $x \in \mathbb{R}^{|X|}$ and set $S \subseteq X$ we write $x(S) = \sum_{v \in S} x_v$. We will work with the approximate feasibility versions of the problems defined in Definitions 4.1.1 and 4.1.2. Our algorithms for these problems will either output that the input instance is infeasible, that is there is no solution with $\alpha = 1$, or output a feasible solution with some $\alpha \leq \alpha^*$. Using binary search, such an algorithm would imply an α^*-approximation for the t-NUkC and robust $(t-1)$-NUkC. Thus in this paper, by a *feasible instance* of t-NUkC and robust $(t-1)$-NUkC we mean an instance that has a feasible solution with $\alpha = 1$.

4.2 Reducing t-NUkC to Robust $(t-1)$-NUkC

In this section we present our simple reduction of t-NUkC to robust $(t-1)$-NUkC and its analysis, which will imply Theorem 6. Using Theorem 6 and the recent 10-approximation algorithm for robust 2-NUkC obtained in [CN21], we obtain the following corollary.

Theorem 7. *There is a 22-approximation algorithm for 3-NUkC.*

We first present the algorithm for performing the reduction and then follow it with the main statement of the reduction. Then we show how to prove Theorem 6 using the reduction and we conclude the section by proving correctness of the reduction.

Algorithm 1: RadiiCompression

```
1 Input: $\mathscr{I} = ((X,d),(k_1,r_1),\dots,(k_t,r_t))$ ,$r_1 \geq \dots \geq r_t$;
2 Init: Set $L = \emptyset$, $U = X$ ;
3 while $U \neq \emptyset$ do
4     Let $v$ be an arbitrary point in $U$;
5     $L \leftarrow L \cup \{v\}$;
6     Child$(v) := \mathscr{B}_U(v, 2r_t)$;
7     $U \leftarrow U \setminus \mathscr{B}_U(v, 2r_t)$;
8 end
9 Return: $(L, \{\text{Child}(v)\}_{v \in L})$
```

The crucial property of the reduction is summarized in the following lemma.

Lemma 4.2.1. *Given a feasible instance $\mathscr{I} = ((X,d),(k_1,r_1),\dots,(k_t,r_t))$ of t-NUkC, RadiiCompression($\mathscr{I}$) returns $(L, \{Child(v)\}_{v\in L})$ where $L \subseteq X$ and $\{Child(v)\}_{v\in L}$ partitions X such that*

- *if $|L| \leq k_t$, then k_t balls of radius $2r_t$ around points in L cover X,*
- *otherwise, $\mathscr{I}_{reduced} = ((L,d),(k_1,2r_1),\dots,(k_{t-1},2r_{t-1}), m = |L| - k_t)$ is a feasible instance of robust $(t-1)$-NUkC.*

Before we prove this lemma, let us see how it implies Theorem 6.

Proof of Theorem 6. We can solve the original feasible t-NUkC instance $\mathscr{I}$ as follows. We first run RadiiCompression($\mathscr{I}$) to obtain $(L, \{\text{Child}(v)\}_{v\in L})$. Then by applying Lemma 4.2.1 we either obtain a 2-approximation to $\mathscr{I}$ (if $|L| \leq k_t$) or a feasible instance $\mathscr{I}_{reduced}$ of robust $(t-1)$-NUkC which then is solved using the α-approximation algorithm assumed to exist. The algorithm returns an α-approximate solution, i.e. sets $C_1,\dots,C_{t-1} \subseteq L$ with $|C_i| \leq k_i$ such that k_i balls of radius $2\alpha r_i$ around points in C_i for all $1 \leq i \leq t-1$ cover at least $|L| - k_t$ points of L. We increase the radius of each of these balls from $2\alpha r_i$ to $2\alpha r_i + 2r_t$. We also open at most k_t balls of radius $2r_t$ around the points in L not covered. Since each point in X is at distance at most $2r_t$ from some point in L and every point in L is either an open center or is at distance at most $2\alpha r_i$ from an open center in C_i for some $1 \le i \le t-1$, by the triangle inequality all points in X are covered. We used at most k_i balls of radius $2\alpha r_i + 2r_t \le (2\alpha+2)r_i$ for each $1 \le i \le t-1$ and at most k_t balls of radius $2r_t$. Thus, we get a $(2\alpha+2)$-approximation algorithm for t-NUkC, assuming there is an α-approximation algorithm for robust $(t-1)$-NUkC. $\square$

Now we present the proof of Lemma 4.2.1.

Proof of Lemma 4.2.1. Clearly $\{\text{Child}(v)\}_{v \in L}$ partitions X, as otherwise the while loop would not have terminated. If $|L| \le k_t$, then since $\{\text{Child}(v)\}_{v \in L}$ partitions X and every point in $\text{Child}(v)$ is at distance at most $2r_t$ from v for all $v \in L$, we can open $|L| \le k_t$ balls of radius $2r_t$ around points in L and cover all points in X.
For the rest of the proof we assume that $|L| > k_t$. Consider a feasible solution $C_1, \dots, C_t \subseteq X$ where $|C_i| \le k_i$ for all $1 \le i \le t$ of $\mathcal{I}$, i.e. k_i balls of radius r_i around points in C_i for all $1 \le i \le t$ cover X. Let $L_i \subseteq L$ be the points of L that are covered by C_i. If a point is covered by C_i and C_j where $i < j$ then we only include it in L_i. Since each point in L is covered, $\{L_i\}_{i=1}^t$ partitions L. Note that each ball of radius r_t can cover at most one point in L_t as the pairwise distance between any two points in $L_t \subseteq L$ is strictly more than $2r_t$. Hence, $|L_t| \le |C_t| \le k_t$.

By "slightly" moving the centers $C_1, \dots, C_{t-1}$ we will exhibit a solution that covers all points in $L \setminus L_t$. To this end, consider some ball of radius r_i centered at a point $p \in C_i$ that covers a point $v \in L_i$. Then for all $u \in L_i$ also covered by the ball around p we have $d(u,v) \le d(u,p) + d(p,v) \le 2r_i$. Thus if we move this ball to be centered at v instead of p and increase the radius to $2r_i$, it will cover all the points in L_i that were covered by it previously when it was centered at p. Repeating this procedure for every $p \in C_i$ and every $1 \le i \le t-1$, we obtain new centers $C'_1 \subseteq L_1, \dots, C'_{t-1} \subseteq L_{t-1}$. It follows that balls of radius $2r_i$ around the centers in C'_i, $1 \le i \le t-1$, cover at least $\sum_{i=1}^{t-1} |L_i| = |L| - |L_t| \ge |L| - k_t$ points of L. This exhibits feasibility of the robust $(t-1)$-NUkC instance $\mathcal{I}_{reduced}$. $\square$

4.3 A Bottom-up Algorithm for Robust 2-NUkC

The main result in [CN21] is a 10-approximation for robust 2-NUkC. In this section we present an alternative, bottom-up implementation of the algorithm of [CN21] as briefly discussed in Section 1. The main theorem we will show in this section is the following.

Theorem 8. *There is a 10-approximation for robust 2-NUkC.*

Linear programming relaxation. The input consists of an instance $\mathcal{I} = ((X,d),$
$(k_1, r_1), (k_2, r_2), m)$ of robust 2-NUkC with $r_1 \ge r_2$. We will be working with the following natural

LP formulation for the problem that we refer to as LP1.

$$\begin{aligned}
\mathrm{cov}_1(v) &\le \sum_{u\in\mathcal{B}(v,r_1)} x_{u,1} \quad \forall v \in X\\
\mathrm{cov}_2(v) &\le \sum_{u\in\mathcal{B}(v,r_2)} x_{u,2} \quad \forall v \in X\\
\mathrm{cov}(v) :=& \mathrm{cov}_1(v) + \mathrm{cov}_2(v) \le 1 \quad \forall v \in X\\
\sum_{v\in X} x_{v,1} &\le k_1, \quad \sum_{v\in X} x_{v,2} \le k_2\\
\sum_{v\in X} &\mathrm{cov}(v) \ge m\\
x_{v,i} &\ge 0, \quad \forall v \in X, i \in \{1,2\}.\\
\mathrm{cov}_i(v) &\ge 0, \quad \forall v \in X, i \in \{1,2\}.
\end{aligned}$$

For every $v \in X$, $\mathrm{cov}_i(v)$ denotes the (fractional) amount that v is covered by balls of radius r_i and $x_{v,i}$ denotes the (fractional) amount that a ball of radius r_i centered at v is open, for $i \in \{1,2\}$. For the instance $\mathcal{I}$, we denote by $\mathcal{P}_{\mathcal{I}}$ the convex hull of all possible integral coverages $\{(\mathrm{cov}_1(v), \mathrm{cov}_2(v))\}_{v\in X}$ induced by integral feasible solutions of $\mathcal{I}$.

We now proceed to the proof of Theorem 8. We will use the round-or-cut method on $\mathcal{P}_{\mathcal{I}}$ via the ellipsoid algorithm to solve this problem. This method was first used in [Car+00] in the context of the minimum knapsack problem and later had been used as a successful technique in designing approximation algorithms for other problems such as clustering [ASS17; Li17; Li16; CN19; Ane+21] and network design [Cha+15]. We now explain the round-or-cut method in our context. In this iterative method, we are given fractional coverages $\{(\mathrm{cov}_1(v), \mathrm{cov}_2(v))\}_{v\in X}$ in each iteration. Using these coverages we will either generate a 10-approximate solution to $\mathcal{I}$, or find a linear inequality violated by these coverages but satisfied by each point in $\mathcal{P}_{\mathcal{I}}$. This inequality is then fed back to the ellipsoid algorithm, which computes a new fractional solution to be used in the next iteration. The separating hyperplanes we output will have encoding length bounded by a polynomial in $|X|$, so this procedure will terminate in a polynomial number of iterations and output an approximate solution along the way or prove that the instance is infeasible.

Recall that $\mathrm{cov}(v) = \mathrm{cov}_1(v) + \mathrm{cov}_2(v)$ for all $v \in X$. First we check if $\sum_{v\in X} \mathrm{cov}(v) \ge m$ as otherwise this inequality itself acts as a separating inequality. Now given these fractional coverages, we run the classic Hochbaum and Shmoys (HS) subroutine [HS85] on $\mathcal{I}$. This subroutine greedily partitions X into clusters of radius r specified in the input. This is done by picking the point with the highest fractional coverage, removing a ball of radius r around it, and repeating. A pseudocode description of this can be found in Algorithm 2. We set

$$(L_2, \{\mathrm{Child}_2(v)\}_{v\in L_2}) = \mathrm{HS}((X,d), \{\mathrm{cov}(v)\}_{v\in X}, 2r_2),$$

where $\{\mathrm{Child}(v)\}_{v\in L_2}$ partitions X and $\mathrm{Child}(v)$ has radius $2r_2$ for all $v \in L_2$. Let $w(v) = |\mathrm{Child}(v)|$. The HS subroutine guarantees that $\mathrm{cov}(v) \ge \mathrm{cov}(u)$ for all

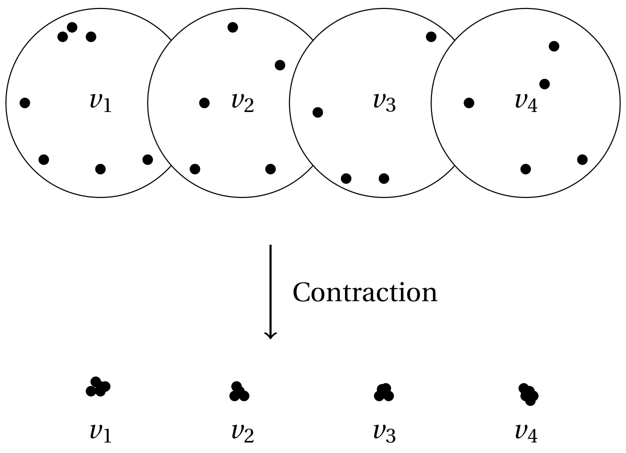

Figure 4.1: Example of a contraction procedure to get $\mathscr{I}_{\text{contracted}}$. v_1, v_2, v_3, and v_4 are points of L_2, and points inside the circle centered at v_i make up Child(v_i).

$v \in L_2, u \in \text{Child}(v)$. Hence the coverages satisfy

$$\sum_{v\in L_2} w(v)\text{cov}(v) = \sum_{v\in L_2}\sum_{u\in \text{Child}(v)} \text{cov}(v) \geq \sum_{v\in L_2}\sum_{u\in \text{Child}(v)} \text{cov}(u) = \sum_{v\in X} \text{cov}(v) \geq m.$$

Algorithm 2: HS

```
10 Input: $(X,d)$, $\{\text{cov}(v)\}_{v\in X}$, $r$;
11 Let $L=\emptyset$, $U=X$ ;
12 while $U\neq\emptyset$ do
13     Let $v = \underset{u\in U}{\text{argmax}}\,\text{cov}(u)$;
14     $L \leftarrow L\cup\{v\}$;
15     Child$(v) = \mathscr{B}_U(v,r)$;
16     $U \leftarrow U\setminus\mathscr{B}_U(v,r)$;
17 end
18 Return $(L,\{\text{Child}(v)\}_{v\in L})$
```

We will now show that if there is an integral solution satisfying $\sum_{v\in L_2} w(v)\text{cov}(v) \geq m$, we can reduce the problem to a more structured instance

$$\mathscr{I}_{\text{contracted}} = ((X', d'), (k_1, 2r_1), (k_2, 0), m).$$

The metric (X', d') is obtained by contracting each Child(v), i.e. by co-locating each point in Child(v) with v, for all $v \in L_2$. Thus, the number of points co-located with each $v \in L_2$ is $w(v) = |\text{Child}(v)|$. We refer the reader to Figure 4.1 for an illustration of this contraction.

Remark 4.3.1. *Note that we will use the convention that a ball of radius 0 around any point v covers all points co-located with v while solving $\mathscr{I}_{\text{contracted}}$.*

We now state the formal lemmas about the feasibility of $\mathscr{I}_{\text{contracted}}$ and about approximately solving $\mathscr{I}_{\text{contracted}}$ or determining its infeasibility.

Lemma 4.3.2. *Either the robust 2-NUkC instance $\mathscr{I}_{\text{contracted}}$ is feasible, or the inequality $\sum_{v\in L_2} w(v)\text{cov}(v) < m$ separates $\{(\text{cov}_1(v), \text{cov}_2(v))\}_{v\in X}$ from $\mathscr{P}_{\mathscr{I}}$.*

We summarize the key properties of $I_{\text{contracted}}$ in the following definition.

Definition 4.3.3. An instance $\mathscr{I} = ((X,d),(k_1,r_1),(k_2,r_2),m)$ of robust 2-NUkC is called a *contracted* instance if we are given an additional set $L \subseteq X$ in the input and $\mathscr{I}$ and L satisfy the following properties.

1. $r_2 = 0$.
2. For every point $u \in X$ there is a point $v \in L$ such that $d(u,v) = 0$. Furthermore, $d(v,v') > 0$ for every $v, v' \in L$.

Lemma 4.3.4. *There is a polynomial time 4-approximation algorithm for contracted instances.*

Before we prove these lemmas, let us see how they imply Theorem 8. In the current iteration of round-or-cut we have constructed $\mathscr{I}_{\text{contracted}}$ using the fractional coverages as described above. Then we apply Lemma 4.3.4 to $\mathscr{I}_{\text{contracted}}$ as it is a contracted instance with $L = L_2$: If we obtain a 4-approximate solution C' for $\mathscr{I}_{\text{contracted}}$ then we can increase the radius of each ball in C' by $2r_2$ and get a solution for $\mathscr{I}$ that covers at least m points. This is because the radius of Child(v) for any $v \in L_2$ was $2r_2$ before the contraction procedure. Thus, we use k_1 balls of radius $4\cdot 2r_1 + 2r_2 \le 10r_1$ and k_2 balls of radius $0 + 2r_2 = 2r_2$ to cover at least m points. This results in a 10-approximate solution for $\mathscr{I}$. Otherwise, the algorithm of Lemma 4.3.4 outputs that $\mathscr{I}_{\text{contracted}}$ is infeasible, and by Lemma 4.3.2 we know that $\sum_{v\in L_2} w(v)\text{cov}(v) < m$ acts as a separating inequality which is then fed back to the ellipsoid algorithm. This concludes the proof of Theorem 8.

Now we present the proof of Lemma 4.3.2. In the next subsection we present the algorithm of Lemma 4.3.4 and its analysis.

Proof of Lemma 4.3.2. We know that $\{(\text{cov}_1(v), \text{cov}_2(v))\}_{v\in X}$ satisfies $\sum_{v\in L_2} w(v)\text{cov}(v) \ge m$. If $\{(\text{cov}_1(v), \text{cov}_2(v))\}_{v\in X}$ is in $\mathscr{P}_{\mathscr{I}}$ then there must be an integral solution $C = (C_1, C_2)$ of $\mathscr{I}$, i.e. k_i balls of radius r_i around points in C_i for all $1 \le i \le 2$ cover m points of X, satisfying $\sum_{v\in L_2} w(v)\mathbb{1}_{\{v \text{ covered by } C\}} \ge m$. We construct a solution for $\mathscr{I}_{\text{contracted}}$ as follows: Move each ball of radius r_1 centered at some point in C_1 to be centered at any point in L_2 that it covers and increase its radius to $2r_1$. Thus, each such ball still covers all the points of L_2 it covered before. Let $C_1' \subseteq L_2$ be the set of centers of balls of radius $2r_1$ obtained by this procedure. Similarly, we obtain $C_2' \subseteq L_2$ by applying this procedure to C_2. However, since the pairwise distance between points of L_2 was strictly more than $2r_2$, each such ball of radius r_2 can cover at most one point of L_2. We can decrease its radius to 0 and it still covers all the points of L_2 it covered before. Let $C' = (C'_1, C'_2)$. From the construction it follows that since C satisfies $\sum_{v\in L_2} w(v)\mathbb{1}_{\{v \text{ covered by C}\}} \geq m$, C' also satisfies $\sum_{v\in L_2} w(v)\mathbb{1}_{\{v \text{ covered by } C'\}} \geq m$. Thus, k_1 balls of radius $2r_1$ around points in C'_1 and k_2 balls of radius 0 around points in C'_2 cover at least m points in the instance $\mathscr{I}_{\text{contracted}}$, as for any $v \in L_2$, all the $w(v) = |\text{Child}(v)|$ -many points are co-located with v in $\mathscr{I}_{\text{contracted}}$. This finishes the proof of the lemma. □

4.3.1 Algorithm for contracted instances

We now present the proof of Lemma 4.3.4.

Proof of Lemma 4.3.4. Recall that we are given a contracted instance of robust 2-NUkC $\mathscr{I} = ((X,d),(k_1,r_1),(k_2,r_2),m)$ and a set $L \subseteq X$ that satisfy the following properties.

1. $r_2 = 0$.
2. For every point $u \in X$ there is a point $v \in L$ such that $d(u,v) = 0$. Furthermore, $d(v,v') > 0$ for every $v, v' \in L$.

We again use the round-or-cut method on $\mathscr{P}_{\mathscr{I}}$ using the ellipsoid algorithm. This time, when we are given fractional coverages $\{(\text{cov}_1(v), \text{cov}_2(v))\}_{v\in X}$ in each round we either generate a 4-approximate solution to $\mathscr{I}$, or we find an inequality separating these coverages from $\mathscr{P}_{\mathscr{I}}$ which is then fed back to the ellipsoid algorithm.

The algorithm in [CN21], as well as our method, uses a standard greedy partitioning scheme $\mathscr{I}$ called CGK [CGK16], which is just the HS procedure applied twice. Since this is a standard technique, we will not present the proofs associated with it but rather just state its guarantees in the form of a lemma that will be useful for our algorithm. CGK uses the fractional coverages $\{(\text{cov}_1(v), \text{cov}_2(v))\}_{v\in X}$ to obtain a tree-structured instance with properties summarized as follows.

Lemma 4.3.5 ([CGK16; CN21])**.** *Given an instance $\mathscr{I} = ((X,d),(k_1,r_1),(k_2,r_2),m)$ of robust 2-NUkC, parameters $\alpha_1, \alpha_2 \geq 2$ and $\{(\text{cov}_1(v), \text{cov}_2(v)\}_{v\in X}$ satisfying $\sum_{v\in X} \text{cov}(v) \geq m$, there is a polynomial time algorithm (CGK) that returns the following:*

1. *sets $L_1, L_2 \subseteq X$ that satisfy $d(v,v') > \alpha_i r_i$ for all $v, v' \in L_i$ for $i \in \{1,2\}$; and*
2. *sets $\text{Child}_1(v) \subseteq L_2$ for all $v \in L_1$ that partition L_2. Furthermore, $d(u,v) \leq \alpha_1 r_1$ for all $u \in \text{Child}_1(v)$ and $v \in L_1$; and*
3. *sets $\text{Child}_2(v) \subseteq X$ for all $v \in L_2$ that partition X. Furthermore, $d(u,v) \leq \alpha_2 r_2$ for all $u \in \text{Child}_2(v)$ and $v \in L_2$.*

4. *If $\sum_{v\in L_1} \mathrm{cov}_1(v) \le k_1 - x$ for some $x \ge 0$ and $\sum_{v\in L_2} \mathrm{cov}_2(v) \le k_2$, then one can obtain a solution for $\mathscr{I}$ that uses $k_1 + 2 - x$ balls of radius $(\alpha_1 + \alpha_2) r_1$ and k_2 balls of radius $\alpha_2 r_2$.*

Furthermore, if $\{(\mathrm{cov}_1(v), \mathrm{cov}_2(v))\}_{v\in X}$ is feasible for LP1 then $\sum_{v\in L_i} \mathrm{cov}_i(v) \le k_i$ is satisfied for $i \in \{1,2\}$.

First, we check whether $\sum_{v\in X} \mathrm{cov}(v) \ge m$. If this is not true, then this inequality itself serves as a separating inequality. Next, we run CGK on $\mathscr{I}$ using these coverages with $\alpha_1 = 4$ and $\alpha_2 = 2$ to obtain L_1, L_2, and $\{\mathrm{Child}_1(v)\}_{v\in L_1}, \{\mathrm{Child}_2(v)\}_{v\in L_2}$. We check if $\sum_{v\in L_i} \mathrm{cov}_i(v) \le k_i$ for $i \in \{1,2\}$. If not, then by Lemma 4.3.5 these coverages are not feasible for LP1 and thus whichever inequality is violated will serve as a separating inequality. Now we branch into two cases. First, assume that $\sum_{v\in L_1} \mathrm{cov}_1(v) \le k_1 - 2$. In this case we can apply (4) of Lemma 4.3.5 to get a solution for $\mathscr{I}$ that opens k_1 balls of radius $\alpha_1 r_1 + \alpha_2 r_2 = 4r_1$ and k_2 balls of radius $\alpha_2 r_2 = 0$, which is a 4-approximate solution to $\mathscr{I}$.

For the remainder of the proof we will assume that $\sum_{v\in L_1} \mathrm{cov}_1(v) > k_1 - 2$. If the fractional coverages $\{(\mathrm{cov}_1(v), \mathrm{cov}_2(v))\}_{v\in X}$ are in $\mathscr{P}_{\mathscr{I}}$, then there must be an integral solution (C_1, C_2) of $\mathscr{I}$ such that balls of radius r_1 centered around points of S_1 cover at least $k_1 - 1$ points of L_1. Moreover, since the pairwise distance between points of L_1 is strictly more than $\alpha_1 r_1 = 4r_1$, each ball of radius r_1 of C_1 can cover at most one point in L_1. Thus, there is at most one ball centered at some point $v_1 \in C_1$ that does not cover a point of L_1 and we can guess v_1 by enumerating over all possibilities. By opening balls of radius $2r_1$ around points of L_1 covered by balls centered at points in C_1, and a ball of radius r_1 around v_1, we can cover all points that are covered by balls centered at points in C_1. We remove the ball of radius r_1 around v_1 from the metric X, update m by subtracting the number of points in this ball around v_1, and reduce k_1 by 1. For simplicity of notation, we still refer to these *updated quantities* as X, m and k_1. This now implies there is a feasible solution to the following question: Is it possible to open k_1 balls of radius $2r_1$ only centered at points in L_1, and k_2 balls of radius 0 only centered at points of L (this is without loss of generality as every point in X is co-located with a point in L by the definition of a contracted instance) to cover at least m points. We claim that this is a *laminar instance* according to the following definition.

Definition 4.3.6. An instance of robust 2-NUkC, $\mathscr{I} = ((X,d),(k_1,r_1),(k_2,r_2),m)$ is said to be *laminar* if we are given sets $L_1, L_2 \subseteq X$ such that the following are satisfied.

1. The k_i balls of radius r_i are only allowed to be centered at points in L_i, $i \in \{1,2\}$;
2. $\mathscr{B}(u,r_i) \cap \mathscr{B}(v,r_i) = \emptyset$ for all $u, v \in L_i$, $i \in \{1,2\}$;
3. $C(v) \cap C(v') = \emptyset$ for all $v, v' \in L_1$, where $C(v) = \{u \in L_2 : \mathscr{B}(v,r_1) \cap \mathscr{B}(u,r_2) \ne \emptyset\}$ are the *children* of v.

We refer the reader to Figure 4.2 for an illustration of a laminar instance. Due to this laminar

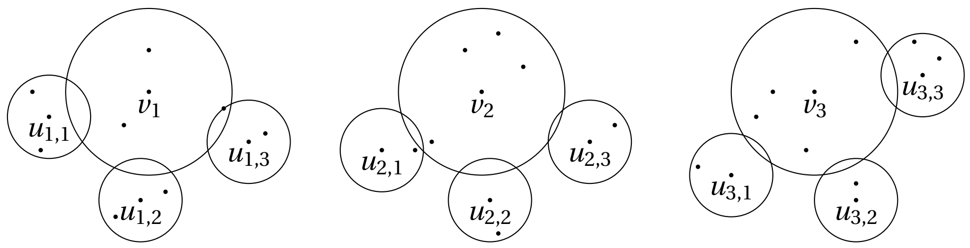

Figure 4.2: Example of a laminar instance with $L_1 = \{v_1, v_2, v_3\}$ and $\mathrm{C}(v_i) = \{u_{i,1}, u_{i,2}, u_{i,3}\}$ for all $i \in \{1,2,3\}$

structure, we can easily solve laminar instances using standard dynamic programming techniques.

Lemma 4.3.7. *There is a polynomial time algorithm based on dynamic programming that exactly solves a given laminar robust 2-NUkC instance* $\mathscr{I} = ((X,d),(k_1,r_1),(k_2,r_2),m)$ *with sets of candidate centers* L_1, L_2.

The precise details of the dynamic programming algorithm and the proof of Lemma 4.3.7 are given in Appendix A. Equipped with this definition and Lemma 4.3.7, we now formally show why our updated contracted instance is laminar, and also how to solve it.

Claim 4.3.8. *The* updated instance $((X,d),(k_1,2r_1),(k_2,r_2),m)$ *where* $r_2 = 0$ *with sets* L_1 *and* L *of candidate centers for balls of radius* $2r_1$ *and* 0 *respectively, is a laminar instance.*

Proof. For every $v \in L_1$, let $C(v) = \{p \in L_2 : \mathscr{B}(v,2r_1) \cap \mathscr{B}(p,0) \neq \emptyset\}$. We claim that for any $u,v \in L_1$, $C(v) \cap C(u) = \emptyset$. Suppose for a contradiction that there are $u,v \in \widehat{L}_1$ such that $C(v) \cap C(u) \neq \emptyset$ and let p be a point in the intersection. Then $d(u,v) \le d(u,p) + d(p,v) \le 4r_1$ by triangle inequality, but we know that the pairwise distance of points in L_1 is strictly more than $4r_1$, a contradiction. This also shows that $\mathscr{B}(u,2r_1) \cap \mathscr{B}(v,2r_1) = \emptyset\ \forall u,v \in L_1$. Also, since $d(u,v) > 0$, $\mathscr{B}(u,0) \cap \mathscr{B}(v,0) = \emptyset$ for any $u,v \in L$ as per the definition of a contracted instance. We add all points p in L that do not appear in $C(v)$ for any $v \in L_1$ to the set $C(v)$ of an arbitrary $v \in L_1$. □

Thus, we can find a solution to this instance using Lemma 4.3.7. A solution to this problem will either result in a 2-approximation for $\mathscr{I}$, or, if the algorithm of Lemma 4.3.7 returns infeasible for each guess of v_1, then it implies that there is no integral solution to $\mathscr{I}$ in which balls of radius r_1 cover at least $k_1 - 1$ points of L_1. Hence, $\sum_{v \in L_1} \mathrm{cov}_1(v) \le k_1 - 2$ separates $\{(\mathrm{cov}_1(v), \mathrm{cov}_2(v))\}_{v \in X}$ from $\mathscr{P}_{\mathscr{I}}$ which is then fed back to the ellipsoid algorithm. This finishes the proof of Lemma 4.3.4. □

4.4 Conclusion

In this paper we developed a bottom-up framework for NUkC that allowed us to reduce from t-NUkC to robust $(t-1)$-NUkC. A constant approximation for 3-NUkC follows as a corollary from the work of CN [CN21]. This bottom-up approach when applied to robust 2-NUkC a gives an alternate presentation of the algorithm of [CN21] with the same guarantees but has a simpler analysis. Thus, further progress is made towards obtaining a constant approximation for t-NUkC when t is a constant. We believe that this bottom-up approach is a promising approach for proving this conjecture in full generality, which remains an exciting open problem.

5 Nearly Tight and Oblivious Algorithms for Explainable Clustering

This chapter is based on joint work with Buddhima Gamlath, Adam Polak, and Ola Svensson [Gam+21]. It has been accepted to The 35th Conference on Neural Information Processing Systems (**NeurIPS 2021**) under the title

Nearly Tight and Oblivious Algorithms for Explainable Clustering.

5.1 Introduction

An important topic in current machine learning research is understanding how models actually make their decisions. For a recent overview on the subject of explainability and interpretability, see, e.g., [Mol19; Mur+19]. Many good methods exist (e.g. [RSG16]) for interpreting black-box models, so called *post-modeling explainability*, but this approach has been criticized [Rud19] for providing little insight into the data. Currently, there is a shift towards designing models that are interpretable by design.

Clustering is a fundamental problem in unsupervised learning. A common approach to clustering is to minimize the k-medians or k-means objectives, e.g., with the celebrated Lloyd's [Llo82] or k-means++ [AV07] algorithms. Both objectives are also widely studied from a theoretical perspective, and, in particular, they admit constant-factor approximation algorithms running in polynomial time [Cha+02; Byr+15; Kan+04; Ahm+20].

In their recent paper [Das+20], Dasgupta et al. were the first to study provable guarantees for explainable clustering. They define a k-clustering to be *explainable* if it is given by a decision tree, where each internal node splits data points with a *threshold cut* in a single dimension (feature), and each of the k leaves corresponds to a cluster (see Figure 5.4).

This definition is motivated by the desire to have a concise and easy-to-explain reasoning behind how the model chooses data points that form a cluster. See the original paper [Das+20] for an extensive discussion of motivations and a survey of previous (empirical) approaches to explainable clustering.

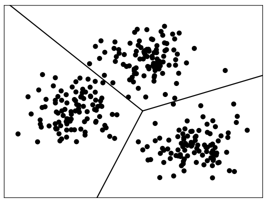
Figure 5.1: Non-explainable clustering

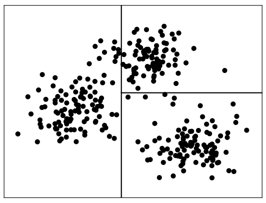
Figure 5.2: Explainable clustering

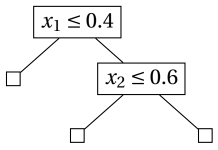

Figure 5.3: Threshold tree

Figure 5.4: Examples of an optimal non-explainable and a costlier explainable clustering of the same set of points in $\mathbb{R}^2$, together with the threshold tree defining the explainable clustering.

The central question to study in this setting is that of the *price of explainability*: How much do we have to lose—in terms of a given objective, e.g., k-medians or k-means—compared to an optimal unconstrained clustering, if we insist on an explainable clustering, and can we efficiently construct such a clustering?

Dasgupta et al. [Das+20] proposed an algorithm that, given an unconstrained (non-explainable) *reference clustering* [1], produces an explainable clustering losing at most a multiplicative factor of $O(k)$ for the k-medians objective and $O(k^2)$ for k-means, compared to the reference clustering. They also gave a lower bound showing that an $\Omega(\log k)$ loss is unavoidable, both for the k-medians and k-means objective. Later, Laber and Murtinho [LM21] improved over the upper bounds in a low-dimensional regime $d \leq k/\log(k)$, giving an $O(d \log k)$-approximation algorithm for explainable k-medians and an $O(dk \log k)$-approximation algorithm for explainable k-means.

5.1.1 Our contributions

Improved clustering cost. We present a randomized algorithm that, given k centers defining a reference clustering and a number $p \geq 1$, constructs a threshold tree that defines an explainable clustering that is, in expectation, worse than the reference clustering by at most a factor of $O(k^{p-1} \log^2 k)$ for the objective given by the ℓ_p-norm. That is $O(\log^2 k)$ for k-medians and $O(k \log^2 k)$ for k-means.

Simple and oblivious algorithm. Our algorithm is remarkably simple. It samples threshold cuts uniformly at random (for k-medians; k-means and higher ℓ_p-norms need slightly fancier distributions) until all centers are separated from each other. In particular, the input to the algorithm includes only the centers of a reference clustering and not the data points.

[1]A reference clustering can be obtained, e.g., by running a constant-factor approximation algorithm for a given objective function. Then, the asymptotic upper bounds of the explainable clustering cost compared to the reference clustering translate identically to the bounds when compared to an optimal clustering.

Table 5.1: **Algorithms and lower bounds for explainable k-clustering in $\mathbb{R}^d$.** For a given objective function, how large a multiplicative factor do we have to lose, compared to an optimal unconstrained clustering, if we insist on an explainable clustering?

	k-medians	k-means	ℓ_p-norm	
Algorithms	$O(k)$	$O(k^2)$		Dasgupta et al. [Das+20]
	$O(d \log k)$	$O(kd \log k)$		Laber and Murtinho [LM21]
	$O(\log^2 k)$	$O(k \log^2 k)$	$O(k^{p-1} \log^2 k)$	**This paper**
	$O(\log k \log\log k)$	$O(k \log k \log\log k)$		Makarychev and Shan [MS21]
	$O(\log k \log\log k)$	$O(k \log k)$		Esfandiari et al. [EMN22]
	$O(d \log^2 d)$			Esfandiari et al. [EMN22]
		$O(k^{1-2/d} \operatorname{polylog} k)$		Charikar and Hu [CH22]
Lower bounds	$\Omega(\log k)$	$\Omega(\log k)$		Dasgupta et al. [Das+20]
		$\Omega(k)$	$\Omega(k^{p-1})$	**This paper**
		$\Omega(k/\log k)$		Makarychev and Shan [MS21]
	$\Omega(\min(d, \log k))$	$\Omega(k)$		Esfandiari et al. [EMN22]
		$\Omega(k^{1-2/d} / \operatorname{polylog} k)$		Charikar and Hu [CH22]

As a consequence, the algorithm cannot overfit the data (any more than the reference clustering possibly already does), and the same expected cost guarantees hold for any future data points not known at the time of the clustering construction. Besides, the algorithm is fast; its running time does not depend on the number of data points n. A naive implementation runs in time $O(dk^2)$, and in Section 5.3.3, we show how to improve it to $O(dk \log^2 k)$ time, which is near-linear in the input size dk of the k reference centers.

Nearly-tight bounds. We complement our results with a lower bound. We show how to construct instances of the clustering problem such that any explainable clustering must be at least $\Omega(k^{p-1})$ times worse than an optimal clustering for the ℓ_p-norm objective. In particular, this improves the previous $\Omega(\log k)$ lower bound for k-means [Das+20] to $\Omega(k)$.

In consequence, we give a nearly-tight answer to the question of the price of explainability. We leave a $\log(k)$ gap for k-medians, and a $\log^2(k)$ gap for k-means and higher ℓ_p-norm objectives. See Table 5.1 for a summary of the upper and lower bounds discussed above and recent independent works discussed in Section 5.1.3.

5.1.2 Technical overview

The theoretical guarantees obtained by Dasgupta et al. [Das+20] depend on the number of clusters k and the height of the threshold tree obtained H. Their algorithm loses, compared to the input reference clustering, an $O(H)$ factor for the k-medians cost and $O(Hk)$ for k-means. These approximations are achieved by selecting a threshold cut that separates some two centers and minimizes the number of points that get separated from their centers in the reference clustering. This creates two children of a tree node, and the threshold tree is created by recursing on each of the children. The height of the tree H may need to be $k-1$. For example, consider the data set in $\mathbb{R}^k$ consisting of the k standard basis vectors (see Figure 5.5). Laber and Murtinho [LM21] replace the dependence on H with d, the dimension of the data set, by first constructing optimal search trees for each dimension and then carefully using them to guide the construction of the threshold tree. In our work, we obtain improved guarantees by using randomized cuts that are oblivious to the data points and depend only on the reference centers, in contrast to the above-mentioned two prior approaches, which selected cuts based on the data points.

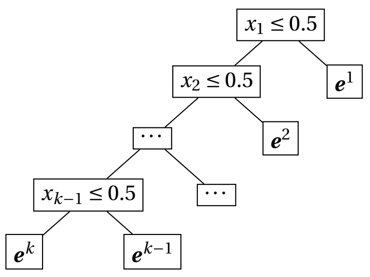

Figure 5.5: An optimal threshold tree for the k standard basis vectors in $\mathbb{R}^k$. Any optimal threshold tree on this data set has height $k-1$.

There are two components to achieving our improved guarantees that correspond with two aspects of the minimum cut algorithm of [Das+20]: the use of the minimum cut, and the height of the threshold tree produced. The first observation is that, for the ℓ_1-norm, we do not lose in the analysis by taking a cut uniformly at random compared to always using the minimum cut. (The corresponding distribution for higher ℓ_p-norms is proportional to the p-th power of the distance to the closest center.) Indeed, using a random cut makes us robust against specifically engineered examples, such as the one that fools the minimum cut algorithm of [Das+20] (see Section 5.6) In that example we add dimensions in which a cut is minimum, but these minimum cuts produce a tree of height $\Omega(k)$ whose cost is $\Omega(k)$ times larger than the optimum.

However, threshold trees of height $\Omega(k)$ are unavoidable in certain instances as seen in the example with k standard basis vectors (Figure 5.5). This leads to our second observation that it is necessary to use a tighter upper bound on the cost of reassigning a point since any height $k-1$ threshold tree produced on this example is actually optimal. Using the diameter definition in [Das+20], the cost of each cut is upper bounded by k while the actual distance between any two centers is at most 2, which is also a valid upper bound for the reassignment cost. Hence, we use the maximum distance between any two centers to upper bound the cost of misclassifying a point.

Limitations and further work.

Remark 5.1.1. *The conjecture made below has recently been resolved in the positive. See Chapter 6.*

We conjecture that our k-medians algorithm is asymptotically optimal. In particular, we believe the actual approximation ratio of our algorithm is $1 + H_{k-1}$, where H_n is the n-th harmonic number (recall that $\ln(n) \le H_n \le 1 + \ln(n)$). There are two potential barriers in our current analysis that prevent us from demonstrating this optimality. The first is that our upper bound on the cost increase of assigning a single point to a wrong center is not tight, and secondly, our analysis may include the cost of the same point multiple times. Despite the further developments mentioned in Section 5.1.3, it still remains to fully resolve the correct asymptotic price of explainability.

Some potential directions to expanding our work include parallelizations, generalizing the notion of explainability, and defining natural clusterability assumptions under which the price of explainability is reduced. Constructing a threshold tree seems inherently sequential; it would be interesting to explore parallelizations for faster implementation. Another direction would be to allow each node to be a hyperplane in a chosen number of dimensions instead of only splitting along one feature. Finally, it seems a non-trivial question to find a right clusterability assumption on the data points distribution—that would allow us to overcome the existing lower bounds—because these lower bounds are in fact very “clusterable” instances, in the traditional usage of this notion.

5.1.3 Independent work

We note independent further developments by Makarychev and Shan [MS21]; Esfandiari, Mirrokni, and Narayanan [EMN22]; and Charikar and Hu [CH22].

Makarychev and Shan [MS21] showed $O(\log k \log\log k)$ and $O(k \log k \log\log k)$ upper bounds for k-medians and k-means, respectively, thus improving over our bounds by a factor of $\log k/\log\log k$. Their k-medians algorithm is essentially the same as our *modified* Algorithm 3 (see Section 5.3.2), but they provide a tighter analysis. Their k-means upper bound follows from combining their k-medians algorithm with their insightful reduction from k-means to k-medians that loses a factor of $O(k)$. However, the k-means algorithm resulting from that combination is essentially the same as our Algorithm 4. They also provide an $\Omega(k/\log k)$ lower bound for k-means, which is slightly worse than ours. Finally, they study the explainable k-medoids problem (i.e., k-medians with ℓ_2 norm), and provide an $O(\log^{3/2} k)$ upper bound and an $\Omega(\log k)$ lower bound.

Esfandiari, Mirrokni, and Narayanan [EMN22] also give an $O(\log k \log\log k)$ upper bound for k-medians. Their algorithm is essentially the same as our (unmodified) Algorithm 3, and, again, they provide a tighter analysis. They also give an $O(k \log k)$ upper bound for k-means, improving over the result of Makarychev and Shan by a factor of $\log\log k$. Their k-means algorithm is similar to our Algorithm 4 but samples cuts from a different distribution. They also match our $\Omega(k)$ lower bound for k-means, and improve the k-medians lower bound of Dasgupta et al. [Das+20] to $\Omega(\min(d, \log k))$.

Charikar and Hu [CH22] focus on explainable k-means and present an

$O(k^{1-2/d}\,\mathrm{poly}(d\log k))$-approximation algorithm, which is better than any previous algorithm when $d = O(\log k/\log\log k)$ (in particular, for any constant dimension d). Resorting to an $O(k\,\mathrm{polylog}\,k)$-approximation algorithm (e.g., [MS21]) when this is not the case, they obtain an $O(k^{1-2/d}\,\mathrm{polylog}\,k)$ upper bound. They match it with an $\Omega(k^{1-2/d}/\,\mathrm{polylog}\,k)$ lower bound, which is tight up to polylogarithmic factors.

5.2 Preliminaries

Following the notation of [Das+20], we use bold variables for vector values and corresponding non-bold indexed variables for scalar coordinates. Intuitively, a clustering is explainable because the inclusion of a data point $\boldsymbol{x} = [x_1, \dots, x_d]$ to a particular cluster is "easily explained" by whether or not $\boldsymbol{x}$ satisfies a series of inequalities of the form $x_i \le \theta$. These inequalities are called ***threshold cuts***, defined by a coordinate $i \in [d]$ (denoting the set $\{1, 2, \dots, d\}$) and a threshold $\theta \in \mathbb{R}$. More precisely, a ***threshold tree*** is a binary tree where each non-leaf node is a threshold cut (i, θ) which assigns the point $\boldsymbol{x}$ of that node into the left child if $x_i \le \theta$ and the right child otherwise. A clustering is ***explainable*** if the clusters are in bijection to the leaves of a threshold tree with exactly k leaves that started with all the data points at the root.

Given a set of points $\mathcal{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \dots, \boldsymbol{x}^n\} \subseteq \mathbb{R}^d$ and its clustering $\{C^1, \dots, C^k\}$, $\bigcup_{j=1}^k C^j = \mathcal{X}$, the k***-medians cost*** of the clustering is defined in [Das+20] as

$$\mathrm{cost}_1(C^1, \dots, C^k) = \sum_{j=1}^k \min_{\boldsymbol{\mu} \in \mathbb{R}^d} \sum_{\boldsymbol{x} \in C^j} \|\boldsymbol{x} - \boldsymbol{\mu}\|_1 = \sum_{j=1}^k \sum_{\boldsymbol{x} \in C^j} \|\boldsymbol{x} - \mathrm{median}(C^j)\|_1.$$

The k***-means*** cost is defined analogously with the square of the ℓ_2 distance of every point to $\mathrm{mean}(C^j)$.

For a set of centers $\mathcal{U} = \{\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^k\} \subseteq \mathbb{R}^d$, a non-explainable clustering $\{\widetilde{C}^1, \dots, \widetilde{C}^k\}$ of $\mathcal{X}$ is given by $\widetilde{C}^j = \{\boldsymbol{x} \in \mathcal{X} \mid \boldsymbol{\mu}^j = \arg\min_{\boldsymbol{\mu} \in \mathcal{U}} \|\boldsymbol{x} - \boldsymbol{\mu}\|_1\}$, and we write $\mathrm{cost}_1(\mathcal{U}) = \mathrm{cost}_1(\widetilde{C}^1, \dots, \widetilde{C}^k)$. Note that $\mathrm{cost}_1(\mathcal{U}) = \sum_{\boldsymbol{x} \in \mathcal{X}} \min_{\boldsymbol{\mu} \in \mathcal{U}} \|\boldsymbol{x} - \boldsymbol{\mu}\|_1$.

Given a threshold tree T, the leaves of T induce an explainable clustering $\{\widehat{C}^1, \dots, \widehat{C}^k\}$, and we write $\mathrm{cost}_1(T) = \mathrm{cost}_1(\widehat{C}^1, \dots, \widehat{C}^k)$. In the analyses, however, we often upper bound the cost of each explainable cluster $\widehat{C}^j$ using the corresponding reference center $\boldsymbol{\mu}^j$:

$$\mathrm{cost}_1(T) \le \sum_{j=1}^k \sum_{\boldsymbol{x} \in \widehat{C}^j} \|\boldsymbol{x} - \boldsymbol{\mu}^j\|_1.$$

These may not be optimal center locations, yet we are still able to obtain guarantees that are polylog away from being tight.

We generalize the above to higher ℓ_p-norms, $p \geq 1$, as follows

$$\text{cost}_p(C^1,\dots,C^k) = \sum_{j=1}^{k} \min_{\boldsymbol{\mu}\in\mathbb{R}^d} \sum_{\boldsymbol{x}\in C^j} \|\boldsymbol{x}-\boldsymbol{\mu}\|_p^p, \qquad \text{cost}_p(T) \leq \sum_{j=1}^{k} \sum_{\boldsymbol{x}\in \widehat{C}^j} \|\boldsymbol{x}-\boldsymbol{\mu}^j\|_p^p.$$

5.3 Explainable k-Medians Clustering

In this section we present our algorithm for explainable k-medians and its analysis. Recall that our algorithm is oblivious to the data points: It determines the threshold tree using only the center locations. The algorithm simply samples a sequence of cuts until it defines a threshold tree with each center belonging to exactly one leaf. In what follows, we elaborate on this process in detail.

The algorithm's input is a set of centers $\mathcal{U} = \{\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \dots, \boldsymbol{\mu}^k\} \subset \mathbb{R}^d$. We consider cuts that intersect the bounding box of $\mathcal{U}$. Letting $\mathrm{I}_i = [\min_{j\in[k]} \mu_i^j, \max_{j\in[k]} \mu_i^j]$ be the interval between the minimum and maximum i-coordinate of centers, the set of all possible cuts that intersect the bounding box of $\mathcal{U}$ is AllCuts $= \{(i,\theta) : i \in [d], \theta \in \mathrm{I}_i\}$. Our algorithm uses a stream of independent uniformly random cuts from AllCuts. In particular, the probability density function of $(i,\theta) \in$ AllCuts is $1/L$ where $L = \sum_{i\in[d]} |I_i|$ is the sum of the side lengths of the bounding box of $\mathcal{U}$.

The algorithm simply takes cuts from this stream until it produces a threshold tree. To this end, it maintains a tentative set of tree leaves, each identified by a subset of centers, and continues until it has k leaves of singleton sets. We say a cut *splits* a leaf if the cut properly intersects with the bounding box of the corresponding subset of centers. In other words, a cut (i,θ) splits a leaf B if and only if the two sets $B^- = \{\boldsymbol{\mu} \in B : \mu_i \leq \theta\}$ and $B^+ = \{\boldsymbol{\mu} \in B : \mu_i > \theta\}$ are both non-empty. At the beginning, the algorithm starts with a single leaf identified by $\mathcal{U}$, the set of all centers. It then samples a cut (i,θ) and checks if it splits any existing leaf. If so, it saves the cut, and for each leaf B that gets split by the cut into B^- and B^+, adds B^- and B^+ as two new leaves rooted at B. These saved cuts define the output threshold tree. We present the pseudo-code as Algorithm 3.

Algorithm 3: Explainable k-medians algorithm.

```
19 Input: A collection of $k$ centers $\mathcal{U} = \{\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \dots, \boldsymbol{\mu}^k\} \subset \mathbb{R}^d$.
20 Output: A threshold tree with $k$ leaves.
21 Leaves ← $\{\mathcal{U}\}$
22 while |Leaves| < $k$ do
23     Sample $(i,\theta)$ uniformly at random from AllCuts.
24     for each $B \in$ Leaves that are split by $(i,\theta)$ do
25         Split $B$ into $B^-$ and $B^+$ and add them as left and right children of $B$.
26         Update Leaves.
27 return the threshold tree defined by all cuts that separated some $B$.
```

5.3.1 Cost analysis

We show that Algorithm 3 satisfies the following guarantees.

Theorem 9. *Given reference centers $\mathcal{U} = \{\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots, \boldsymbol{\mu}^k\}$, Algorithm 3 outputs a threshold tree T whose expected cost satisfies*

$$\mathbb{E}[\mathrm{cost}_1(T)] \le O(\log(k) \cdot (1 + \log(c_{\max}/c_{\min}))) \cdot \mathrm{cost}_1(\mathcal{U}),$$

where $c_{\max}$ and $c_{\min}$ denote the maximum and minimum pairwise distance between two centers in $\mathcal{U}$, respectively. Furthermore, with probability at least $1 - 1/k$, Algorithm 3 outputs a threshold tree T from a distribution with $\mathbb{E}[\mathrm{cost}_1(T)] \le O(\log^2 k) \cdot \mathrm{cost}_1(\mathcal{U})$.

The (more involved) proof of the furthermore statement is given in Section 5.3.2. We remark that the success probability $1 - 1/k$ can be made larger by only slightly increasing the hidden constant in the cost guarantee. Furthermore, in Section 5.3.2, we give a slight adaptation of the above algorithm that has an expected cost bounded by $O(\log^2 k) \cdot \mathrm{cost}_1(\mathcal{U})$. The remaining part of this section is devoted to proving the upper bound on the expected cost of Algorithm 3.

Proof outline. First, in Lemma 5.3.1, we show that a random cut in expectation separates $\mathrm{cost}_1(\mathcal{U})/L$ points from their closest centers. Indeed, note that the probability of separating a point $\boldsymbol{x}$ from its center $\pi(\boldsymbol{x})$ is at most $\|\boldsymbol{x} - \pi(\boldsymbol{x})\|_1/L$, and on the other hand, $\mathrm{cost}_1(\mathcal{U}) = \sum_{x \in \mathcal{X}} \|\boldsymbol{x} - \pi(\boldsymbol{x})\|_1$, hence the bound follows from linearity of expectation. Each such separated point incurs a cost of at most $c_{\max}$. Next, in Lemma 5.3.2, we show that with good probability $O(\log(k) \cdot L/c_{\max})$ random cuts separate all pairs of centers that are at distance at least $c_{\max}/2$ from each other. Morally, the cost of halving $c_{\max}$, which we will need to perform $1 + \log(c_{\max}/c_{\min})$ many times, is therefore $\mathrm{cost}_1(\mathcal{U})/L \cdot c_{\max} \cdot O(\log(k) \cdot L/c_{\max})$ $= O(\log(k)) \cdot \mathrm{cost}_1(\mathcal{U})$, and the bound follows (see Lemma 5.3.3).

Formal analysis of the expected cost. We first bound the number of points that are separated from their closest center by a random cut. This quantity is important as it upper bounds the number of points whose cost is increased in the final tree due to the considered cut. Recall that $L = \sum_{i=1}^{d} |I_i|$ denotes the total side lengths of the bounding box of the input centers $\mathcal{U}$. We also let $f_i(\theta)$ be the number of points separated from their closest center by the cut (i, θ).

Lemma 5.3.1. *We have $\mathbb{E}_{(i,\theta)}[f_i(\theta)] \le \mathrm{cost}_1(\mathcal{U})/L$ where the expectation is over a uniformly random threshold cut $(i, \theta) \in \mathrm{AllCuts}$.*

Proof. For a point $\boldsymbol{x} \in \mathcal{X}$ let $\pi(\boldsymbol{x})$ denote the closest center in $\mathcal{U}$. Then by definition,

$$\mathrm{cost}_1(\mathcal{U}) = \sum_{x \in \mathcal{X}} \|\boldsymbol{x} - \pi(\boldsymbol{x})\|_1 = \sum_{i=1}^{d} \sum_{\boldsymbol{x} \in X} |x_i - \pi(\boldsymbol{x})_i|.$$

Moreover, if we let $f_i(\theta)$ be the number of points separated from their closest center by the cut

(i,θ), we can rewrite the cost of a fixed dimension i as follows:

$$\sum_{\boldsymbol{x}\in X} |x_i - \pi(\boldsymbol{x})_i| = \sum_{\boldsymbol{x}\in X} \int_{-\infty}^{\infty} \mathbb{1}[\theta \text{ between } x_i \text{ and } \pi(\boldsymbol{x})_i]d\theta = \int_{-\infty}^{\infty} f_i(\theta)d\theta\,.$$

We thus have $\mathrm{cost}_1(\mathcal{U}) = \sum_{i=1}^d \int_{-\infty}^{\infty} f_i(\theta)d\theta$.

At the same time, if we let $[a_i, b_i]$ denote the interval I_i, then $\frac{1}{|I_i|}\int_{a_i}^{b_i} f_i(\theta)d\theta$ equals the number of points separated from their closest center by a uniformly random threshold cut $(i,\theta) : \theta \in I_i$ along dimension i. Thus the expected number of points separated from their closest center by a uniformly random threshold cut in AllCuts is

$$\sum_{i=1}^d \frac{|I_i|}{L}\cdot\frac{1}{|I_i|}\int_{a_i}^{b_i} f_i(\theta)d\theta = \frac{1}{L}\sum_{i=1}^d \int_{a_i}^{b_i} f_i(\theta)d\theta \le \frac{1}{L}\sum_{i=1}^d \int_{-\infty}^{\infty} f_i(\theta)d\theta = \mathrm{cost}_1(\mathcal{U})/L,$$

where we used $f_i(\theta) \ge 0$ for the inequality. □

The above lemma upper bounds the expected number of points whose cost increases from a uniformly random threshold cut. We proceed to analyze how much this increase is, in expectation. Let Leaves(t) denote the state of Leaves at the beginning of the t-th iteration of the while loop of Algorithm 3 and let $c_{\max}(t) = \max_{B\in \mathrm{Leaves}(t)} \max_{\boldsymbol{\mu}^i,\boldsymbol{\mu}^j\in B} \|\boldsymbol{\mu}^i - \boldsymbol{\mu}^j\|_1$ denote the maximum distance between two centers that belong to the same leaf at the beginning of the t-th iteration. With this notation we have that Leaves$(1) = \{\mathcal{U}\}$ and that $c_{\max}(1)$ equals the $c_{\max}$ in the statement of Theorem 9. Observe that $c_{\max}(t) \ge c_{\max}(t+1)$ and $c_{\max}(t) = 0$ if $|$Leaves$| = k$ (i.e., when each leaf contains exactly one center). Understanding the rate at which $c_{\max}(t)$ decreases is crucial for our analysis because of the following observation: Consider a leaf $B \in$ Leaves(t) and a point $\boldsymbol{x} \in \mathcal{X}$ that has not yet been separated from its closest center $\pi(\boldsymbol{x}) \in B$. If the threshold cut selected in the t-th iteration separates $\boldsymbol{x}$ from $\pi(\boldsymbol{x})$ then the cost of $\boldsymbol{x}$ in the final threshold tree is upper bounded by $\max_{\boldsymbol{\mu}\in B} \|\boldsymbol{x} - \boldsymbol{\mu}\|_1$, which, by the triangle inequality, is at most

$$\max_{\boldsymbol{\mu}\in B} \|\boldsymbol{x} - \pi(\boldsymbol{x})\|_1 + \|\pi(\boldsymbol{x}) - \mu\|_1 \le \|\boldsymbol{x} - \pi(\boldsymbol{x})\|_1 + c_{\max}(t)\,. \tag{5.1}$$

In other words, a point that is first separated from its closest center by the threshold cut selected in the t-th iteration has a cost increase of at most $c_{\max}(t)$.

Lemma 5.3.2. *Fix the the threshold cuts selected by Algorithm 3 during the first $t-1$ iterations (this determines the random variable* Leaves(t) *and thus $c_{\max}(t)$). Let $M = 3\ln(k)\cdot 2L/c_{\max}(t)$. Then*

$$\Pr[c_{\max}(t+M) \le c_{\max}(t)/2] \ge 1 - 1/k,$$

where the probability is over the random cuts selected in iterations $t, t+1, \ldots, t+M-1$.

Proof. Consider two centers $\boldsymbol{\mu}^i$ and $\boldsymbol{\mu}^j$ that belong to the same leaf in Leaves(t). The probability that a uniformly random threshold cut from AllCuts separates these two centers equals $\|\boldsymbol{\mu}^i - \boldsymbol{\mu}^j\|_1/L$. Thus if the centers are at distance at least $c_{\max}(t)/2$, the probability that they are *not* separated by any of M independently chosen cuts is at most

$$\left(1 - \frac{c_{\max}(t)/2}{L}\right)^M = \left(1 - \frac{c_{\max}(t)}{2L}\right)^{3\ln(k)\cdot 2L/c_{\max}(t)} \le (1/e)^{3\ln(k)} = 1/k^3 .$$

There are at most $\binom{k}{2}$ pairs of centers in the leaves of Leaves(t) at distance at least $c_{\max}(t)/2$. By the union bound, we thus have, with probability at least $1-1/k$, that each of these pairs are separated by at least one of the cuts selected in iterations $t, t+1, \ldots, t+M-1$. In that case, any two centers in the same leaf of Leaves$(t+M)$ are at distance at most $c_{\max}(t)/2$ and so $c_{\max}(t+M) \le c_{\max}(t)/2$. □

Equipped with the above lemmas we are ready to analyze the expected cost of the tree output by Algorithm 3. Let (i_t, θ_t) denote the cut selected by Algorithm 3 in the t-th iteration. As argued above in (5.1), $c_{\max}(t)$ upper bounds the cost increase of the points first separated from their closest center by the t-th threshold cut. Hence,

$$\mathbb{E}[\mathrm{cost}_1(T)] \le \mathrm{cost}_1(\mathscr{U}) + \mathbb{E}\left[\sum_t c_{\max}(t) f_{i_t}(\theta_t)\right],$$

where the sum is over the iterations of Algorithm 3 (and recall that $f_i(\theta)$ denotes the number of points separated from their closest center by the cut (i,θ)). We remark that the right-hand side is an upper bound (and not an exact formula of the cost) for two reasons: first, not every separated point may experience a cost increase of $c_{\max}(t)$, and second, the right-hand side adds a cost increase every time a cut separates a point from its closest center and not only the first time. Nevertheless, we show that this upper bound yields the stated guarantee. We do so by analyzing the expected cost increase of the cuts until $c_{\max}(t)$ has halved. Specifically, let

$$\text{cost-increase}(r) = \sum_{t:\, c_{\max}(t) \in (c_{\max}/2^{r+1}, c_{\max}/2^r]} c_{\max}(t) f_{i_t}(\theta_t)$$

be the random variable that upper bounds the cost increase caused by the cuts selected during the iterations t when $c_{\max}/2^{r+1} < c_{\max}(t) \le c_{\max}/2^r$. Then

$$\mathbb{E}[\mathrm{cost}_1(T)] \le \mathrm{cost}_1(\mathscr{U}) + \sum_r \mathbb{E}[\text{cost-increase}(r)],$$

where the sum is over r from 0 to $1 + \lfloor \log_2(c_{\max}/c_{\min}) \rfloor$. The bound on the expected cost therefore follows from the following lemma.

Lemma 5.3.3. *For every r, $\mathbb{E}[\text{cost-increase}(r)] \le 12\ln(k)\cdot \mathrm{cost}_1(\mathscr{U})$.*

Proof. First, Let $M = 3\ln(k)\cdot 2L/c_{\max}(t)$ as in Lemma 5.3.2. Using Lemma 5.3.1, one can upper bound the expected cost of M uniformly random cuts in iterations $t, t+1, \ldots, t+M-1$ by $6\ln(k)\cdot \mathrm{cost}_1(\mathscr{U})$ and M cuts is very likely to halve $c_{\max}(t)$ as in Lemma 5.3.2. Intuitively, it is thus very likely that the cost of these M cuts upper bounds cost-increase(r). The additional constant factor of 2 in the statement of the lemma arises by considering the small "failure" probability of such a trial.

For a formal proof, let t be the first iteration when $c_{\max}(t) \le c_{\max}/2^r$ and let $M = 3\ln(k)\cdot 2L/c_{\max}(t)$ as in Lemma 5.3.2. In the following, we use cost_M to denote the random variable that equals the cost increase caused by adding M uniformly random cuts after the t-th iteration. Then

$$\mathbb{E}[\mathrm{cost}_M] \le M\cdot c_{\max}(t)\cdot \mathbb{E}_{(i,\theta)}[f_i(\theta)] \le M\cdot c_{\max}(t)\cdot \mathrm{cost}_1(\mathscr{U})/L = 6\ln(k)\cdot \mathrm{cost}_1(\mathscr{U}),$$

where the first inequality holds because $c_{\max}(t)$ is monotonically decreasing and the second inequality is by Lemma 5.3.1. At the same time, if we let H denote the event that $c_{\max}(t)$ has halved after adding these M cuts, i.e., that $c_{\max}(t+M) \le c_{\max}(t)/2$, then $\Pr[H] \ge 1-1/k$ by Lemma 5.3.2. We now upper bound the expectation of cost-increase(r) by considering "trials" of M cuts until one of these succeeds in halving $c_{\max}(t)$. Indeed, split the sequence of random cuts selected by the algorithm after iteration t into such trials $A_1,\ldots,A_\ell$ where each A_j consist of M cuts, and A_ℓ is the first successful trial in the sense that selecting (only) those cuts after iteration t would cause $c_{\max}(t)$ to halve. Then we must have that $c_{\max}(t)$ has halved also after adding all the cuts in the ℓ trials. It follows that cost-increase(r) is upper bounded by the cost increase caused by the cuts in $A_1, A_2, \ldots, A_\ell$. We can thus upper bound $\mathbb{E}[\text{cost-increase}(r)]$ by the expected cost of these trials until one succeeds:

$$\sum_{i=0}^{\infty} \Pr[H]\cdot \Pr[\neg H]^i \cdot \left(\mathbb{E}\left[\mathrm{cost}_M \mid H\right] + i\cdot \mathbb{E}\left[\mathrm{cost}_M \mid \neg H\right]\right),$$

where we use $\mathbb{E}[\mathrm{cost}_M \mid H]$ and $\mathbb{E}[\mathrm{cost}_M \mid \neg H]$ for the expected costs of a successful and unsuccessful trials, respectively. By standard calculations (as for the geometric distribution), this upper bound simplifies to $\mathbb{E}\left[\mathrm{cost}_M \mid H\right] + \frac{\Pr[\neg H]}{\Pr[H]}\mathbb{E}\left[\mathrm{cost}_M \mid \neg H\right]$. This can be further rewritten as

$$\frac{1}{\Pr[H]}\cdot\left(\Pr[H]\cdot\mathbb{E}[\mathrm{cost}_M \mid H] + \Pr[\neg H]\cdot \mathbb{E}[\mathrm{cost}_M \mid \neg H]\right) = \frac{\mathbb{E}[\mathrm{cost}_M]}{\Pr[H]} \le 12\ln(k)\cdot \mathrm{cost}_1(\mathscr{U}),$$

where we used that $\Pr[H] \ge 1-1/k \ge 1/2$. □

5.3.2 Upper bounding cost by a factor of $O(\log^2 k)$

Observe that our analysis of Algorithm 3 implies that the expected cost of the output tree is at most $O(\log^2(k)\cdot \mathrm{cost}_1(\mathscr{U}))$ whenever $c_{\max}$ and $c_{\min}$ do not differ by more than a polynomial factor in k. However, our current techniques fail to upper bound this expectation by a factor $o(k)$ for general $c_{\max}$ and $c_{\min}$. To illustrate this point, consider the k-dimensional instance with a single point $\boldsymbol{x}$ at the origin and k centers where the i-th center $\boldsymbol{\mu}^i$ is located at the i-th standard basis vector scaled by the factor 2^i. In our analysis, we upper bound the cost of $\boldsymbol{x}$ with its *maximum* distance to those centers that remain in the same leaf whenever $\boldsymbol{x}$ is separated from its closest center $\boldsymbol{\mu}^1$. This yields the following upper bound on the expected cost of $\boldsymbol{x}$ in the final tree

$$\sum_{i=2}^{k} 2^i \Pr[\boldsymbol{x} \text{ is separated from } \boldsymbol{\mu}^1 \text{ and } \boldsymbol{\mu}^i \text{ is the farthest remaining center}]\,.$$

Due to the exponentially increasing distances, this is lower bounded by

$$\sum_{i=2}^{k} 2^{i-1} \Pr[\boldsymbol{x} \text{ is separated from } \boldsymbol{\mu}^1 \text{ and } \boldsymbol{\mu}^i \text{ is a remaining center}].$$

Now note that the probability in this last sum equals

$$\Pr[\boldsymbol{x} \text{ is separated from } \boldsymbol{\mu}^1 \text{ before } \boldsymbol{x} \text{ is separated from } \boldsymbol{\mu}^i\ ] = 2/(2+2^i).$$

It follows that any analysis of Algorithm 3 that simply upper bounds the reassignment cost of a point with the maximum distance to a remaining center cannot do better than a factor of $\Omega(k)$.

We overcome this obstacle by analyzing a slight modification of Algorithm 3 that avoids these problematic cuts that separate very close centers. Recall the notation used in the previous section: Leaves(t) denotes the state of Leaves at the beginning of the t-th iteration of the while loop and $c_{\max}(t) = \max_{B \in \text{Leaves}(t)} \max_{\boldsymbol{\mu}^i, \boldsymbol{\mu}^j \in B} \|\boldsymbol{\mu}^i - \boldsymbol{\mu}^j\|_1$ denotes the maximum distance between two centers that belong to the same leaf at the beginning of the t-th iteration. We now modify Algorithm 3 by replacing Line 5 "Sample (i,θ) uniformly at random from AllCuts" by

> Sample (i,θ) uniformly at random from those cuts in AllCuts that do *not* separate two centers that are within distance at most $c_{\max}(t)/k^4$.

This modification allows us to prove a nearly tight guarantee on the expected cost.

Theorem 10. *Given reference centers* $\mathcal{U} = \{\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \ldots, \boldsymbol{\mu}^k\}$, modified *Algorithm 3 outputs a threshold tree T whose expected cost satisfies* $\mathbb{E}[\text{cost}_1(T)] \le O(\log^2 k) \cdot \text{cost}_1(\mathcal{U})$.

Before we give the proof of the above theorem, we explain how it implies the furthermore statement of Theorem 9. Recall that the difference between Algorithm 3 and the modified version is that Algorithm 3 samples cuts uniformly at random whereas the modified version only adds a random cut if it does *not* separate two centers that are within distance $c_{\max}(t)/k^4$.

Algorithm 3 adds $k-1$ cuts to its tree. We now argue that these $k-1$ cuts are with probability at least $1 - 1/k$ sampled from the same distribution as the $k-1$ cuts added by the modified version. This then implies the furthermore statement of Theorem 9 since Theorem 10 says that the expected cost of the modified algorithm is $O(\log^2 k) \cdot \text{cost}_1(\mathcal{U})$. To this end, consider the i-th such cut and let t be the iteration when the $(i-1)$-th cut was added to the tree. Then when the i-th cut is added there must be two centers in the same leaf at distance $c_{\max}(t)$.

So the probability that two centers within distance $c_{\max}(t)/k^4$ are separated by the i-th cut (which is a uniformly random cut among all cuts that would separate at least two centers in the same leaf) is at most $1/k^4$. There can be at most $\binom{k}{2}$ such pairs and so by the union bound, we can conclude that, with probability at least $1-1/k^2$, the i-th cut of Algorithm 3 does not separate any such nearby centers. We can thus view the distribution from which Algorithm 3 samples the i-th cut as follows: With probability $p \le 1/k^2$ it samples a uniformly random cut that separates two centers within distance $c_{\max}(t)/k^4$ and with remaining probability it samples a uniformly random cut that does not separate any such centers, i.e., from the same distribution that the modified algorithm samples the i-th cut from. Applying the union bound over the $k-1$ cuts then yields the furthermore statement of Theorem 9. Finally, we remark that the same arguments imply a larger success probability if applied to the modified algorithm that only adds cut that do not separate centers within distance $c_{\max}(t)/k^{\ell}$ for some $\ell \ge 4$.

In the remainder of this section, we prove Theorem 10, i.e., that modified Algorithm 3 returns a threshold tree whose expected cost is $O(\log^2 k)\cdot \mathrm{cost}_1(\mathcal{U})$. The proof is similar to the cost analysis in Section 5.3.1 with the main difference being that here we are more careful in bounding the cost when considering different "rounds" of the algorithm. In the analysis it will be convenient to take the following viewpoint of the modified algorithm: it samples a uniformly random cut and then discards it if it separates two centers within distance $c_{\max}(t)/k^4$. While the number of iterations may increase with this viewpoint, the output distribution is the same as the modified algorithm in that, in each iteration, a cut is sampled uniformly at random among those that do not separate any centers within distance $c_{\max}(t)/k^4$. In the following, we refer to this as the *sample-discard algorithm* and we prove Theorem 10 by showing that the sample-discard algorithm outputs a tree whose expected cost is $O(\log^2 k)\cdot \mathrm{cost}_1(\mathcal{U})$.

Let (i_t, θ_t) denote the (uniformly random) cut selected in the t-th iteration of the sample-discard algorithm and recall the following notation: Leaves(t) denotes the state of Leaves at the beginning of the t-th iteration of the while-loop and $c_{\max}(t) = \max_{B\in \mathrm{Leaves}(t)} \max_{\boldsymbol{\mu}^i,\boldsymbol{\mu}^j\in B} \|\boldsymbol{\mu}^i - \boldsymbol{\mu}^j\|_1$ denotes the maximum distance between two centers that belong to the same leaf at the beginning of the t-th iteration. We start by observing that Lemma 5.3.2 readily generalizes to the modified version.

Lemma 5.3.4. *Fix the the threshold cuts selected by the sample-discard algorithm during the first $t-1$ iterations (this determines the random variable* Leaves(t) *and thus $c_{\max}(t)$). Let $M = 3\ln(k)\cdot 4L/c_{\max}(t)$. Then*

$$\Pr[c_{\max}(t+M) \le c_{\max}(t)/2] \ge 1 - 1/k,$$

where the probability is over the random cuts selected in iterations $t, t+1, \ldots, t+M-1$.

Proof. The proof is similar to that of Lemma 5.3.2 but some care has to be taken as certain cuts are now discarded.

Consider two centers $\boldsymbol{\mu}^i$ and $\boldsymbol{\mu}^j$ that belong to the same leaf in Leaves(t). Further suppose that

$c_{\max}(t)/2 \le \|\boldsymbol{\mu}^p - \boldsymbol{\mu}^q\|_1 \le c_{\max}(t)$. We have that any cut (i,θ) that separates these two centers is considered (i.e., not discarded) by the sample-discard algorithm after the t-th iteration unless (i,θ) also separates two centers within distance $c_{\max}(t)/k^4$. Here we used that $c_{\max}(\cdot)$ is monotonically decreasing and so the set of cuts that are discarded if sampled is only decreasing in later iterations. We can thus obtain the lower bound $c_{\max}(t)/(4L)$ on the probability that a uniformly random cut separates $\boldsymbol{\mu}^p$ and $\boldsymbol{\mu}^q$ by subtracting

$$\frac{1}{L}\sum_{i=1}^{d}\int_{-\infty}^{\infty} \mathbb{1}\left[\theta \text{ separates two centers within distance } c_{\max}(t)/k^4\right] d\theta \le \frac{1}{L}\binom{k}{2}\frac{c_{\max}(t)}{k^4}$$

from

$$\frac{1}{L}\sum_{i=1}^{d}\int_{-\infty}^{\infty} \mathbb{1}[\theta \text{ between } \boldsymbol{\mu}_i^p \text{ and } \boldsymbol{\mu}_i^q]\, d\theta \ge \frac{1}{L}c_{\max}(t)/2\,.$$

The proof now proceeds in the exact same way as that of Lemma 5.3.2. Indeed, if the centers are at distance at least $c_{\max}(t)/2$, the probability that they are *not* separated by any of M independently chosen cuts is at most

$$\left(1-\frac{c_{\max}(t)}{4L}\right)^M = \left(1-\frac{c_{\max}(t)}{4L}\right)^{3\ln(k)\cdot 4L/c_{\max}(t)} \le (1/e)^{3\ln(k)} = 1/k^3\,.$$

There are at most $\binom{k}{2}$ pairs of centers in the leaves of Leaves(t) at distance at least $c_{\max}(t)/2$. By the union bound, we thus have, with probability at least $1-1/k$, that each of these pairs are separated by at least one of the cuts selected in iterations $t, t+1, \ldots, t+M-1$. In that case, any two centers in the same leaf of Leaves$(t+M)$ are at distance at most $c_{\max}(t)/2$ and so $c_{\max}(t+M) \le c_{\max}(t)/2$. □

Now, similar to Section 5.3.1, we can upper bound the expected cost of the constructed threshold tree T by

$$\mathbb{E}[\mathrm{cost}_1(T)] \le \mathrm{cost}_1(\mathcal{U}) + \mathbb{E}\left[\sum_t c_{\max}(t) f_{i_t}(\theta_t)\mathbb{1}[(i_t,\theta_t) \text{ was added to the tree}]\right],$$

where the sum is over the iterations. We remark that, in contrast to Section 5.3.1, we have strengthened the upper bound by only considering those cuts that were actually added to the threshold tree by the modified algorithm. This refinement is necessary for obtaining the improved guarantee. We now analyze the sum in the expectation by partitioning it into $1+\lfloor\log_2(c_{\max}/c_{\min})\rfloor$ rounds. Specifically for $r \in \{0,1\ldots,\lfloor\log_2(c_{\max}/c_{\min})\rfloor\}$, we let

$$\text{cost-increase}'(r) = \sum_{t: c_{\max}(t)\in(c_{\max}/2^{r+1}, c_{\max}/2^r]} c_{\max}(t) f_{i_t}(\theta_t)\mathbb{1}[(i_t,\theta_t) \text{ was added to the tree}]$$

be the cost of the cuts selected during the iterations t when $c_{\max}/2^{r+1} < c_{\max}(t) \le c_{\max}/2^r$.

To upper bound $\mathbb{E}[\text{cost-increase}'(r)]$ we use $\text{active}_r(i,\theta) \in \{0,1\}$ to denote the indicator variable of those cuts that separate two centers within distance $c_{\max}/2^r$ and do not separate any two centers within distance $c_{\max}/(2^{r+1}k^4)$.

Lemma 5.3.5. *For a round r,*

$$\mathbb{E}[\text{cost-increase}'(r)] \leq 24\ln(k) \cdot \sum_{i=1}^{d} \int_{-\infty}^{\infty} f_i(\theta)\, \text{active}_r(i,\theta)\, d\theta\,.$$

Before giving the proof of this lemma, let us see how it implies Theorem 10. For this, note that a cut (i,θ) only has $\text{active}_r(i,\theta) = 1$ for at most $O(\log(k^4))$ many values of r. Indeed, let c be the distance between the closest centers that (i,θ) separates. Then any round r for which $\text{active}_r(i,\theta) = 1$ must satisfy $c_{max}/(2^{r+1}k^4) \leq c \leq c_{max}/2^r$. Hence, we have

$$\begin{aligned}
\text{cost}_1(T) &\leq \text{cost}_1(\mathcal{U}) + \sum_r \mathbb{E}[\text{cost-increase}'(r)] \\
&\leq \text{cost}_1(\mathcal{U}) + \sum_r 24\ln(k) \cdot \sum_{i=1}^{d} \int_{-\infty}^{\infty} f_i(\theta)\, \text{active}_r(i,\theta)\, d\theta \\
&\leq \text{cost}_1(\mathcal{U}) + O(\log^2 k) \cdot \sum_{i=1}^{d} \int_{-\infty}^{\infty} f_i(\theta)\, d\theta \\
&= O(\log^2 k) \cdot \text{cost}_1(\mathcal{U})\,.
\end{aligned}$$

In other words, we proved that the sample-discard algorithm outputs a tree T with $\mathbb{E}[\text{cost}_1(T)] \leq O(\log^2 k) \cdot \text{cost}_1(\mathcal{U})$, which implies Theorem 10 since modifed Algorithm 3 and the sample-discard algorithm have the same output distribution. It remains to prove the lemma.

Proof of Lemma 5.3.5. Consider the first iteration t such that $c_{\max}(t) \leq c_{\max}/2^r$. Further suppose that $c_{\max}(t) > c_{\max}/2^{r+1}$ since otherwise $\text{cost-increase}'(r) = 0$ and the statement is trivial. We proceed to upper bound $\mathbb{E}[\text{cost-increase}'(r)]$ as follows. First note that the cost of a random cut sampled in an iteration t' such that $c_{\max}/2^{r+1} \leq c_{\max}(t') \leq c_{\max}(t)$ equals

$$\frac{c_{\max}(t')}{L} \sum_{i=1}^{d} \int_{-\infty}^{\infty} f_i(\theta)\, \mathbb{1}[(i,\theta) \text{ was added to the tree}]\, d\theta\,.$$

The cut (i,θ) can be added to the tree only if it does not separate any centers within distance $c_{\max}(t')/k^4 \geq c_{\max}/(2^{r+1}k^4)$ and it must separate two centers within distance at most $c_{\max}(t') \leq c_{\max}/2^r$. In other words, any cut that is added to the tree must have $\text{active}_r(i,\theta) = 1$. We can thus upper bound the above cost of a single cut by

$$\frac{c_{\max}(t)}{L} \sum_{i=1}^{d} \int_{-\infty}^{\infty} f_i(\theta)\, \text{active}_r(i,\theta)\, d\theta\,, \tag{5.2}$$

where we also used that $c_{\max}(t') \leq c_{\max}(t)$.

The proof now follows arguments that are again similar to those in Section 5.3.1. Select $M = 12\ln(k)\cdot L/c_{\max}(t)$ as in Lemma 5.3.4. We upper bound $\mathbb{E}[\text{cost-increase}(r)]$ by adding "trials" of M cuts until $c_{\max}(\cdot)$ goes below $c_{\max}/2^{r+1}$. (Strictly speaking this may not happen after a multiple of M cuts but considering more cuts may only increase the cost of our upper bound.) Let H be the event that the following M cuts causes $c_{\max}(\cdot)$ to drop below $c_{\max}/2^{r+1}$. By Lemma 5.3.4, $\Pr[H] \geq 1 - 1/k$. Furthemore, the expected cost of M cuts is M times (5.2) which equals

$$12\ln(k)\cdot\sum_{i=1}^{d}\int_{-\infty}^{\infty} f_i(\theta)\,\text{active}_r(i,\theta)\,d\theta\,.$$

The statement now follows from the same "geometric distribution" calculations as in the proof of Lemma 5.3.3. □

5.3.3 Implementation details

Since we only use cuts that split at least one leaf, the algorithm in fact only needs to sample cuts conditioned on this event. Note that if we sample only the cuts that split at least one leaf, the while loop in Line 22 of Algorithm 3 runs for at most $k-1$ iterations. We now explain how to efficiently sample cuts (Line 23), find the leaves split by a given cut (Line 24), and implement the split operation (Lines 25–26).

We first show how to efficiently implement the split operation. For a cut (i,θ) that splits a given leaf B into B^- and B^+, the split operation can be implemented in $O(d\cdot\min(|B^-|,|B^+|)\cdot\log|B|)$ time as follows: In each leaf B, we maintain d binary search trees $T_1^B,\ldots,T_d^B$ where T_i^B stores the i-th coordinate of the centers in B. Now, given a cut (i,θ) and a leaf B that gets split by (i,θ), we can find the number of centers in B that have a smaller or equal i-th coordinate than θ using T_i^B in $O(\log|B|)$ time. Let b^- be this number, and let $b^+ = |B| - b^-$. Suppose that $b^- \leq b^+$. For the other case, the implementation is analogous. In this case, we construct B^- by initializing it with d empty binary search trees and inserting the centers whose i-th coordinate is at most θ to each of them. This takes $O(d\cdot b^-\cdot\log|B|)$ time. For B^+, we just reuse the binary search trees of B after removing the centers that belong to B^-. This also takes $O(d\cdot b^-\cdot\log|B|)$ time. Let $\tau(k)$ denote the running time of all splitting operations performed by the algorithm when starting with a single leaf with k leaves. Then, $\tau(k) = \tau(k-k')+\tau(k')+O(d\cdot\min(k-k',k')\cdot\log k)$ and by induction, we conclude that $\tau(k) = O(dk\log^2 k)$.

To find the leaves that get separated by a cut, we employ the following data structure. For each dimension i, we maintain a balanced interval tree T_i^{int}. For each tentative leaf node with centers B, we store the interval indicating the range of the i-th coordinate of B in T_i^{int}. We update the corresponding interval trees after each split operation, which amounts to removing at most one interval and adding at most two intervals per node that gets split. Note that the added and removed intervals for a single split operation for a fixed dimension can be computed in $O(\log k)$ time using the previously described node binary search trees. Moreover, adding and removing intervals to and from an interval tree with at most k intervals also takes

$O(\log k)$ time. As we have $O(k)$ split operations in total, the time to maintain the interval trees is $O(dk\log k)$. Now, given a cut (i,θ), we can retrieve all the leaves that get separated by (i,θ) in $O(\log(k_1)+k_2)$ time where k_1 is the number of tentative leaves and k_2 is the number of tentative leaves that get separated by the cut. To retrieve such leaves, we query the i-th interval tree to find all intervals that contain the value θ. Since we sample at most $k-1$ cuts from the conditioned distribution, the total time for this operation over all cuts and all dimensions is $O(dk\log k)$.

What remains is to show that we can efficiently sample a uniform cut conditioned on the event that it splits at least one leaf. To this end, in the interval trees described above, we also maintain the lengths of the unions of intervals in each subtree. This length information can be updated in $O(\log k')$ time where $k' \le k$ is the number of intervals in an interval tree. Then in $O(d)$ time, one can sample a dimension i and in $O(\log k')$ time, sample a suitable θ value.

5.4 Explainable k-Means and General ℓ_p-norm Clustering

In this section, we generalize Theorem 10 to the explainable k-clustering problems with assignment cost defined in terms of the ℓ_p-norm, which includes the explainable k-means ($p=2$) problem.

Recall that in Section 5.3, we sample cuts from the uniform distribution over AllCuts, and consequently, the probability that a point $\boldsymbol{x} \in \mathcal{X}$ is separated from its closest center $\pi(\boldsymbol{x})$ is proportional to the ℓ_1 distance between $\boldsymbol{x}$ and $\pi(\boldsymbol{x})$. However, selecting cuts according to the uniform distribution can be arbitrarily bad for higher p-norms even in one-dimensional space. For example, consider the k-means (i.e. $p=2$) problem with $d=1$ where the cost of assigning a point $\boldsymbol{x}$ to a center $\boldsymbol{\mu}$ is defined as $\|\boldsymbol{x}-\boldsymbol{\mu}\|_2^2$. Suppose we have two centers $\boldsymbol{\mu}^1=-1$ and $\boldsymbol{\mu}^2=D>1$, and fix one data point $\boldsymbol{x}=\mathbf{0}$. The closest center to $\boldsymbol{x}$ is $\boldsymbol{\mu}^1$ and hence the original cost is 1. However, the expected cost of a uniformly random cut is $((D\cdot 1^2+1\cdot D^2)/(1+D)=D$ which can be arbitrarily large.

To avoid such drastic costs, we sample cuts from a generalized distribution. Ideally, we would like to sample cuts analogously to the case of k-medians so that the probability that we separate a point $\boldsymbol{x}$ from its closest center $\pi(\boldsymbol{x})$ is proportional to $\|\boldsymbol{x}-\pi(\boldsymbol{x})\|_p^p$. However, sampling from such a distribution seems very complicated if at all possible. Instead, we sample from a slightly different distribution: Namely, for a p-norm where the cost of assigning a point x to a center y is $\|x-y\|_p^p$, we sample cuts (i,θ) from the distribution where the probability density function of (i,θ) is proportional to $\min_{j\in[k]}|\mu_i^j-\theta|^{p-1}$, the $(p-1)$-th power of the minimum distance to a center *along the i-th dimension*. We call this distribution $\mathcal{D}_p$.

Using samples from $\mathcal{D}_p$ with a modified version of Algorithm 3 yields Theorem 11.

Theorem 11. *For every $p \ge 1$, there exists a randomized algorithm that when given input* *centers $\mathcal{U} = \{\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \dots, \boldsymbol{\mu}^k\}$, outputs a threshold tree T whose expected cost satisfies*

$$\mathbb{E}[\mathrm{cost}_p(T)] \le O(k^{p-1}\log^2 k) \cdot \mathrm{cost}_p(\mathcal{U}).$$

Proof. To prove this, we consider a generalized version of Algorithm 3 where we sample threshold cuts from the distribution $\mathcal{D}_p$ introduced in Section 5.4. Recall that $\mathcal{D}_p$ is defined such that the p.d.f. of a cut (i,θ) is proportional to the $(p-1)$-th power of the minimum distance from θ to a projection of a center in the i-th dimension.

We start by introducing some notation and making the definition of $\mathcal{D}_p$ precise. For a dimension $i \in [d]$, let $\mu_i^- = \min_{j\in[k]} \mu_i^j$ and $\mu_i^+ = \max_{j\in[k]} \mu_i^j$. For a dimension $i \in [d]$ and two coordinates $x, y \in \mathbb{R}$, let $\mathcal{I}_i(x,y)$ be the set of consecutive intervals along the i-th dimension delimited by the coordinates x and y themselves and the projections of the centers in $\mathcal{U}$ that lie between x and y. For example, consider the 2-dimensional instance with four centers $\boldsymbol{\mu}^1, \dots, \boldsymbol{\mu}^4$ shown in Figure 5.6. On the horizontal axis, two coordinates x and y are marked along with the projections of the four centers $\mu_1^1, \mu_1^2, \mu_1^3$, and μ_1^4. Here, $\mathcal{I}_1(x,y)$ consists of the three consecutive intervals $[x, \mu_1^4]$, $[\mu_1^4, \mu_1^2]$, and $[\mu_1^2, y]$.

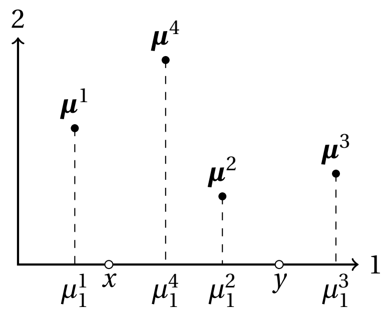

Figure 5.6: Intervals defined by projecting points onto a coordinate axis.

Observe that, by the definition of $\mathcal{I}_i(x,y)$, we have $|x-y| = \sum_{[a,b]\in\mathcal{I}_i(x,y)} |b-a|$.

Let

$$\mathcal{I}_{\mathrm{all}} = \bigcup_{i\in[d]} \left\{(i,[a,b]) : [a,b] \in \mathcal{I}_i(\mu_i^-, \mu_i^+)\right\}$$

denote the collection of all dimension–interval pairs that are delimited by the projections of the centers onto the respective dimensions. We define

$$L_p = \sum_{(i,[a,b])\in\mathcal{I}_{\mathrm{all}}} |b-a|^p.$$

With the introduced notation, the distribution $\mathcal{D}_p$ can be formally described as follows: We first select a dimension i and an interval $[a,b] \in \mathcal{I}_i(\mu_i^-, \mu_i^+)$ along with dimension i (i.e., we select a dimension–interval pair $(i,[a,b]) \in \mathscr{I}_{\text{all}})$ with probability $|b-a|^p/L_p$. Then we pick $\theta \in [a,b]$ randomly such that the p.d.f. θ is

$$P_{a,b}(\theta) := \frac{p \cdot 2^{p-1}}{(b-a)^p} \min(\theta - a, b-\theta)^{p-1}.$$

Another key component of the design and analysis of the generalized algorithm is a *pseudo-distance* function. For two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, Let

$$\mathscr{I}(\boldsymbol{x},\boldsymbol{y}) = \bigcup_{i \in [d]} \{(i,[a,b]) : [a,b] \in \mathscr{I}_i(x_i,y_i)\}.$$

We then define the pseudo-distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ as

$$d_p(\boldsymbol{x},\boldsymbol{y}) = \sum_{(i,[a,b]) \in \mathscr{I}(\boldsymbol{x},\boldsymbol{y})} |b-a|^p.$$

Note that the p-th power of the ℓ_p distance, $\|\boldsymbol{x}-\boldsymbol{y}\|_p^p$, between two points $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as $\sum_{i \in [d]} |x_i - y_i|^p$. It is easy to see that $\|\boldsymbol{x}-\boldsymbol{y}\|_p^p \geq d_p(\boldsymbol{x},\boldsymbol{y})$ since

$$|x_i - y_i|^p = \left(\sum_{[a,b] \in \mathscr{I}_i(x_i,y_i)} |a-b| \right)^p \geq \sum_{[a,b] \in \mathscr{I}_i(x_i,y_i)} |a-b|^p$$

for each dimension i. For $p=1$, we have equality.

A key observation now is that, if we sample a cut from $\mathscr{D}_p$, the probability that it separates two centers $\boldsymbol{\mu}^g$ and $\boldsymbol{\mu}^h$ is *proportional to their pseudo-distance* $d_p(\boldsymbol{\mu}^g,\boldsymbol{\mu}^h)$.

□

5.4.1 The algorithm for ℓ_p-norms with $p \geq 1$.

We now present the generalized algorithm. The only difference from the modified version of Algorithm 3 is how we sample random cuts at Line 23. Recall from Section 5.3 that we defined Leaves(t) to denote the state of Leaves at the beginning of the t-th iteration. We define $c'_{p,\max}(t)$ as the maximum pseudo-distance between any pair of centers in a leaf in Leaves(t). Formally, $c'_{p,\max}(t) = \max_{B \in \text{Leaves}(t)} \max_{\boldsymbol{\mu}^i,\boldsymbol{\mu}^j \in B} d_p(\boldsymbol{\mu}^i,\boldsymbol{\mu}^j)$. Let $c'_{p,\max} = c'_{p,\max}(1)$.

Now, in the sampling step (Line 32), we draw samples from $\mathscr{D}_p$. However, we discard the cut if it separates any two centers in a leaf whose pseudo-distance is at most $c'_{p,\max}(t)/k^4$. Note that this is a generalization of the sample-discard algorithm from the proof of Theorem 10. We present the pseudo-code in Algorithm 4.

Following the lines of the proof of Theorem 10, we now upper bound the expected cost of Algorithm 4.

Algorithm 4: Generalized explainable clustering algorithm for higher ℓ_p-norms.

```
Input: A collection of $k$ centers $\mathcal{U} = \{\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \dots, \boldsymbol{\mu}^k\} \subset \mathbb{R}^d$.
Output: A threshold tree with $k$-leaves.
Leaves ← $\{\mathcal{U}\}$
while |Leaves| < $k$ do
    Sample a cut $(i, \theta)$ from $\mathcal{D}_p$
    if $(i, \theta)$ separates two centers that are closer than $c'_{p,\max}(\cdot)/k^4$ in pseudo-distance then
        Discard the cut.
    else
        for each $B \in$ Leaves that are split by $(i, \theta)$ do
            Split $B$ into $B^-$ and $B^+$ and add them as left and right children of $B$.
            Update Leaves.
return the threshold tree defined by all cuts that separated some $B$.
```

Lemma 5.4.1. *Fix the threshold cuts selected by Algorithm 4 during the first $t-1$ iterations. Let $M = 3 \cdot 4 \cdot \ln(k) \cdot L_p / c'_{p,max}(t)$. Then*

$$\Pr[c'_{p,\max}(t+M) \le c'_{p,\max}(t)/2] \ge 1 - 1/k,$$

where the probability is over the random cuts selected in iterations $t, t+1, \dots, t+M-1$.

Proof. We begin by introducing a few more notations that are useful in the analysis. For an iteration t, let TooClose(t) be the set of pairs of centers $(\boldsymbol{\mu}^g, \boldsymbol{\mu}^h)$ that satisfy $d_p(\boldsymbol{\mu}^g, \boldsymbol{\mu}^h) \le c'_{p,\max}(t)/k^4$. In other words, TooClose(t) contains pairs of centers that the algorithm is not allowed to separate at the t-th iteration. Note that for any $(\boldsymbol{\mu}^g, \boldsymbol{\mu}^h) \in$ TooClose(t), both $\boldsymbol{\mu}^g$ and $\boldsymbol{\mu}^h$ will be in the same leaf in Leaves(t). Let

$$\mathcal{I}_{\text{bad}}(t) = \bigcup_{(\boldsymbol{\mu}^g, \boldsymbol{\mu}^h) \in \text{TooClose}(t)} \mathcal{I}(\boldsymbol{\mu}^g, \boldsymbol{\mu}^h)$$

be the set of dimension–interval pairs $(i, [a,b])$ such that making a cut in interval $[a,b]$ along dimension i will separate a pair of centers in TooClose. Observe that a cut that is made outside of $\mathcal{I}_{\text{bad}}(t)$ will not separate any pair of centers in TooClose.

Consider a leaf $B \in$ Leaves(t) and two centers $\boldsymbol{\mu}^g$ and $\boldsymbol{\mu}^h$ in B such that $c'_{p,\max}(t)/2 \le d_p(\boldsymbol{\mu}^g, \boldsymbol{\mu}^h) \le c'_{p,\max}(t)$.

Note that

$$\sum_{[a,b] \in \mathcal{I}_{\text{bad}}(t)} |b-a|^p \le \sum_{(\boldsymbol{\mu}^{g'}, \boldsymbol{\mu}^{h'}) \in \text{TooClose}(t)} \sum_{(i,[a,b]) \in \mathcal{I}(\boldsymbol{\mu}^{g'}, \boldsymbol{\mu}^{h'})} |b-a|^p$$

$$= \sum_{(i,[a,b])\in\mathscr{I}(\boldsymbol{\mu}^{g'},\boldsymbol{\mu}^{h'})} d_p(\boldsymbol{\mu}^{g'},\boldsymbol{\mu}^{h'})$$
$$\le \binom{k}{2}\frac{c'_{p,\max}(t)}{k^4} \le \frac{c'_{p,\max}(t)}{4}.$$

In the last inequality, we use that $k \ge 2$.

Hence, the probability that a cut selected at the t-th iteration separates $\boldsymbol{\mu}^g$ and $\boldsymbol{\mu}^h$ is at least

$$\frac{d_p(\boldsymbol{\mu}^g,\boldsymbol{\mu}^h)}{L_p} - \frac{\sum_{[a,b]\in\mathscr{I}_{\text{bad}}(t)}|b-a|^p}{L_p} \ge \frac{c'_{p,\max}(t)}{2L_p} - \frac{c'_{p,\max}(t)}{4L_p} \ge \frac{c'_{p,\max}(t)}{4L_p}.$$

The proof now follows by replacing $c_{\max}(t)$ with $c'_{p,\max}(t)$ and L with L_p in the remaining part of the proof of Lemma 5.3.4. □

In the following analysis, we use the Hölder's inequality stated below:

Lemma 5.4.2 (Hölder's inequality)**.** *For two real numbers u and v such that $1/u + 1/v = 1$ and two positive real number sequences $y_1,\dots,y_m$ and $z_1,\dots,z_m$, it holds that*

$$\sum_{i\in[m]} y_i z_i \le \left(\sum_{i\in[m]} y_i^u\right)^{1/u}\left(\sum_{i\in[m]} z_i^v\right)^{1/v}.$$

In particular, setting $y_1 = y_2 = \dots = y_m = 1$, $u = p/(p-1)$ and $v = p$ for some p, and taking the p-th power on both sides, it holds that

$$\left(\sum_{i\in[m]} z_i\right)^p \le m^{p-1}\sum_{i\in[m]} z_i^p.$$

We now upper bound the expected cost. Recall that $\pi(\boldsymbol{x})$ denotes the closest center in $\mathscr{U}$ to a point $\boldsymbol{x}\in\mathscr{X}$ and that $\text{cost}_p(\mathscr{U})$ is defined as

$$\text{cost}_p(\mathscr{U}) = \sum_{\boldsymbol{x}\in\mathscr{X}} \|\boldsymbol{x}-\pi(\boldsymbol{x})\|_p^p = \sum_{\boldsymbol{x}\in\mathscr{X}}\sum_{i\in[d]} |x_i - \pi(\boldsymbol{x})_i|^p.$$

To bound the cost of the output clustering in the k-medians setting, we used the triangle inequality. For general p-th power of p-norms, we use the following generalized triangle inequality:

Lemma 5.4.3. *Consider three points $\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\in\mathbb{R}^d$. We have $\|\boldsymbol{z}-\boldsymbol{x}\|_p^p \le 2^{p-1}\left(\|\boldsymbol{z}-\boldsymbol{y}\|_p^p + \|\boldsymbol{y}-\boldsymbol{x}\|_p^p\right)$.*

Proof. Expanding $\|\cdot\|_p^p$ as a summation over d dimensions, it is sufficient to prove that for any three real numbers $x,y,z\in\mathbb{R}$, $|z-x|^p \le 2^{p-1}(|z-y|^p + |y-x|^p)$. Without loss of generality, assume that $z \ge x$. If $y \le x$ or $y \ge z$, the proof follows trivially because we have $|z-x| \le |z-y|$

or $|z-x| \le |y-z|$, respectively. Now suppose that $x \le y \le z$. Let $a = y-x$ and $b = z-y$. Since $a+b = z-x$, we simply need to prove that $(a+b)^p \le 2^{p-1}(a^p + b^p)$ which follows from Hölder's inequality. $\square$

Recall that we defined $c'_{p,\max}(t)$ and $c'_{p,\max}$ earlier using the pseudo-distance function d_p. We now define $c_{p,\max}(t)$ and $c_{p,\max}$ similarly, but using the p-th power of the ℓ_p-norm: Namely, $c_{p,\max}(t) = \max_{B\in\text{Leaves}(t)} \max_{\boldsymbol{\mu}^i,\boldsymbol{\mu}^j\in B} \|\boldsymbol{\mu}^i - \boldsymbol{\mu}^j\|_p^p$ and $c_{p,\max} = c_{p,\max}(1)$. We again use (i_t, θ_t) to denote the cut selected by 4 in the t-th iteration and $f_i(\theta)$ to denote the number of points $\boldsymbol{x} \in \mathscr{X}$ that are separated from $\pi(\boldsymbol{x})$ by a cut (i, θ).

For a point $\boldsymbol{x} \in \mathscr{X}$, suppose that it is assigned to some center $\boldsymbol{\mu}$ in the final threshold tree. If $\boldsymbol{\mu} = \pi(\boldsymbol{x})$, the cost contribution of $\boldsymbol{x}$ in the final clustering is the same as that in the original clustering. Suppose $\boldsymbol{\mu} \ne \pi(\boldsymbol{x})$ and suppose that $\boldsymbol{x}$ was separated from $\pi(\boldsymbol{x})$ at iteration t. Then, using Lemma 5.4.3, we conclude that the cost of assigning $\boldsymbol{x}$ to $\boldsymbol{\mu}$, i.e., $\|\boldsymbol{x} - \boldsymbol{\mu}\|_p^p$, is upper bounded by

$$2^{p-1}\left(\|\boldsymbol{x} - \pi(\boldsymbol{x})\|_p^p + \|\pi(\boldsymbol{x}) - \mu\|_p^p\right) \le 2^{p-1}\left(\|\boldsymbol{x} - \pi(\boldsymbol{x})\|_p^p + c_{p,\max}(t)\right).$$

Let UsedCuts be the set of cuts used to split some leaf in Line 37 of Algorithm 4. Now using the above observation, we can upper bound the expected cost of the output tree, $\mathbb{E}[\text{cost}_p(T)]$, by

$$\text{cost}_p(T) \le 2^{p-1}\left(\text{cost}_p(\mathscr{U}) + \sum_{r=0}^{\infty} \mathbb{E}[\text{cost-increase}(r)]\right)$$

where

$$\text{cost-increase}(r) = \sum_{t:\frac{c'_{p,\max}}{2^{r+1}} \le c'_{p,\max}(t) \le \frac{c'_{p,\max}}{2^r}} c_{p,\max}(t) \cdot f_{i_t}(\theta_t) \cdot \mathbb{1}[(i_t, \theta_t) \in \text{UsedCuts}].$$

Note that in the last expression, the summed terms use $c_{p,\max}(t)$ whereas the condition of the summation uses $c'_{p,\max}(t)$. Note that

$$\begin{aligned}\text{cost-increase}(r) &\le \sum_{t:\frac{c'_{p,\max}}{2^{r+1}} \le c'_{p,\max}(t) \le \frac{c'_{p,\max}}{2^r}} k^{p-1} c'_{p,\max}(t) \cdot f_{i_t}(\theta_t) \cdot \mathbb{1}[(i_t, \theta_t) \in \text{UsedCuts}] \\ &\le k^{p-1}\frac{c'_{p,\max}}{2^r} \cdot \sum_{t:\frac{c'_{p,\max}}{2^{r+1}} \le c'_{p,\max}(t) \le \frac{c'_{p,\max}}{2^r}} f_{i_t}(\theta_t) \cdot \mathbb{1}[(i_t, \theta_t) \in \text{UsedCuts}].\end{aligned}$$

The first inequality is by Hölder's inequality (applied independently in each dimension in the computation of respective d_p and $\|\cdot\|_p^p$ values). The second inequality simply uses the condition of the summation.

We now upper bound the expected value of cost-increase(r). Let $\mathscr{I}_{\text{act}}(r)$ be the set of dimension–interval pairs in $\mathscr{I}$ that do not separate any pair of centers that are closer than $c'_{p,\max}/(k^4 2^{r+1})$ in pseudo-distance but separate at least one pair of centers that are closer than $c'_{p,\max}/2^r$ in pseudo-distance. We prove the following lemma which is analogous to Lemma 5.3.5 in the proof of Theorem 10.

Lemma 5.4.4. *For a round r, $\mathbb{E}[\text{cost-increase}(r)]$ is*

$$O\left(k^{p-1}\cdot\log k\right)\cdot\left(\sum_{(i,[a,b])\in\mathscr{I}_{\text{act}}(r)}|b-a|^p\int_a^b P_{a,b}(\theta)\,f_i(\theta)\,d\theta\right).$$

Proof. We consider "trials" of M consecutive iterations in round r where

$$M = 12\cdot 2^{r+1}\ln(k)\cdot L_p/c'_{p,max}.$$

We perform independent trials until $c'_{p,\max}(\cdot)$ at the end of a trial goes below $c'_{p,\max}/2^{r+1}$.

Consider one trial and let s be the starting iteration of the trial. Note that we have

$$M \geq 3\cdot 4\cdot\ln(k)\cdot L_p/c'_{p,\max}(s)$$

since $c'_{p,\max}(s)\geq c'_{p,\max}/2^{r+1}$. Thus, by Lemma 5.4.1, after M iterations, round r ends with probability at least $1-1/k\geq 1/2$. (Note that round r may end before all M iterations of a trial are completed. In such trials, we assume that we discard the additional cuts that are made after the round ends.)

Let $\text{UB}_\ell = \sum_{s=t'}^{t'+M-1} f_{i_s}(\theta_s)\cdot\mathbb{1}[(i_s,\theta_s)\in\text{UsedCuts}]$ and observe that

$$\text{cost-increase}(r)\leq k^{p-1}\frac{c'_{p,\max}}{2^r}\cdot\sum_\ell \text{UB}_\ell \tag{5.3}$$

where the sum is over all trials we perform in round r.

We first upper bound each term UB_ℓ and then use the expectation of a geometric random variable to upper bound the expected value of $\sum_\ell \text{UB}_\ell$. We have

$$\begin{aligned}\mathbb{E}[\text{UB}_\ell] &\leq \sum_{s=t'}^{t'+M-1}\mathbb{E}\left[f_{i_s}(\theta_s)\cdot\mathbb{1}[(i_s,\theta_s)\in\text{UsedCuts}]\right]\\ &\leq \sum_{s=t'}^{t'+M-1}\frac{1}{L_p}\sum_{(i,[a,b])\in\mathscr{I}_{\text{act}}(r)}|b-a|^p\int_a^b P_{a,b}(\theta)\,f_i(\theta)\,d\theta. \end{aligned}\tag{5.4}$$

Note that we only sum over dimension–interval pairs in $\mathscr{I}_{\text{act}}(r)$ as cuts made outside of this set will be discarded. To elaborate, the dimension–interval pair in which a cut (i,θ) is made can be outside of $\mathscr{I}_{\text{act}}(r)$ for two reasons:

1. Because it separates two centers that are closer than $c'_{p,\max}/(k^4 2^{r+1})\leq c'_{p,\max}(t')/k^4$. Then it will get discarded in Line 34.

2. Because it does not separate two centers that are closer than $c'_{p,\max}/2^r$. Such a cut will not split any leaves in Line 36.

Consequently, for all $(i_s, \theta_s) \in \text{UsedCuts}$, we have $\theta_s \in [a,b]$ for some interval $[a,b]$ such that $(i_s,[a,b]) \in \mathscr{I}_{\text{act}}(r)$.

Now, since the summed terms in (5.4) no longer depend on the summed index s, we now have

$$\begin{aligned}
\mathbb{E}[\text{UB}_\ell] &\le \frac{M}{L_p} \sum_{(i,[a,b])\in\mathscr{I}_{\text{act}}} |b-a|^p \int_a^b P_{a,b}(\theta)\, f_i(\theta)\, d\theta \\
&= \frac{12\cdot 2^{r+1}\ln(k)}{c'_{p,\max}} \sum_{(i,[a,b])\in\mathscr{I}_{\text{act}}} |b-a|^p \int_a^b P_{a,b}(\theta)\, f_i(\theta)\, d\theta.
\end{aligned}$$

Now, considering that round r ends after a trial with probability at least $1/2$, using the expected value of a geometric distribution, we conclude that

$$\mathbb{E}\left[\sum_\ell \text{UB}_\ell\right] \le \frac{24\cdot 2^{r+1}\ln(k)}{c'_{p,\max}} \sum_{(i,[a,b])\in\mathscr{I}_{\text{act}}(r)} |b-a|^p \int_a^b P_{a,b}(\theta)\, f_i(\theta)\, d\theta.$$

The proof of the lemma then follows by combining this with the bound in (5.3). □

With Lemma 5.4.4 in hand, we now prove Theorem 11.

Proof of Theorem 11. Using Lemma 5.4.4, we can upper bound $\sum_{r=0}^{\infty}\mathbb{E}[\text{cost-increase}(r)]$ as follows:

$$\begin{aligned}
&\sum_{r=0}^{\infty} \mathbb{E}[\text{cost-increase}(r)] \\
&\quad = O\big(k^{p-1}\cdot\log k\big)\cdot \sum_{r=0}^{\infty} \sum_{(i,[a,b])\in\mathscr{I}_{\text{act}}(r)} |b-a|^p \int_a^b P_{a,b}(\theta)\, f_i(\theta)\, d\theta \\
&\quad = O\big(k^{p-1}\cdot\log k\big)\cdot \sum_{(i,[a,b])\in\mathscr{I}} \sum_{r=0}^{\infty} \mathbb{1}[(i,[a,b])\in\mathscr{I}_{\text{act}}(r)]\cdot\left(|b-a|^p \int_a^b P_{a,b}(\theta)\, f_i(\theta)\, d\theta\right) \\
&\quad = O\big(k^{p-1}\cdot\log k\big)\cdot \sum_{(i,[a,b])\in\mathscr{I}} |b-a|^p \int_a^b P_{a,b}(\theta)\, f_i(\theta)\, d\theta \cdot\left(\sum_{r=0}^{\infty} \mathbb{1}[(i,[a,b])\in\mathscr{I}_{\text{act}}(r)]\right).
\end{aligned}$$

We now claim that for any dimension–interval pair in $(i,[a,b]) \in \mathscr{I}$

$$\sum_{r=0}^{\infty} \mathbb{1}[(i,[a,b])\in\mathscr{I}_{\text{act}}(r)] = \big|\{r : (i,[a,b])\in\mathscr{I}_{\text{act}}(r)\}\big| = O(\log k). \tag{5.5}$$

Namely, fix some dimension–interval pair $(i,[a,b]) \in \mathscr{I}$. Let c be the smallest pseudo-distance between any pair of centers that are separated if a cut (i,θ) such that $\theta \in [a,b]$ is made. Then $(i,[a,b])$ is in $\mathscr{I}_{\text{act}}(r)$ only if $c \le c'_{p,\max}/2^r$ and $c'_{p,\max}/2^{r+1} \le k^4 c$, or equivalently,

$\log(c'_{p,\max}/2ck^4) \le r \le \log(c'_{p,\max}/c)$ which yields (5.5). Thus we have

$$\begin{aligned}
&\sum_{r=0}^{\infty} \mathbb{E}[\text{cost-increase}(r)] \\
&= O\left(k^{p-1}\cdot\log^2 k\right)\cdot \sum_{(i,[a,b])\in\mathscr{I}} \left(|b-a|^p \int_a^b P_{a,b}(\theta)\, f_i(\theta)d\theta\right) \\
&= O\left(k^{p-1}\cdot\log^2 k\right)\cdot \sum_{(i,[a,b])\in\mathscr{I}} \left(|b-a|^p \int_a^b p2^{p-1}\frac{\min(\theta-a,b-\theta)^{p-1}}{|b-a|^p} f_i(\theta)d\theta\right) \\
&= O\left(k^{p-1}\cdot\log^2 k\right)\cdot(p2^{p-1})\cdot \sum_{(i,[a,b])\in\mathscr{I}} \int_a^b \min(\theta-a,b-\theta)^{p-1} f_i(\theta)d\theta. \qquad (5.6)
\end{aligned}$$

Now to conclude the proof of Theorem 11, let $\text{cost}'_p(\mathscr{U}) = \sum_{\boldsymbol{x}\in\mathscr{X}} d_p(\boldsymbol{x},\pi(\boldsymbol{x}))$ which is the cost of $\mathscr{U}$ defined in terms of the pseudo-distances. Recall that $\|\boldsymbol{x}-\boldsymbol{y}\|_p^p \ge d_p(\boldsymbol{x},\boldsymbol{y})$ and hence we have $\text{cost}_p(\mathscr{U}) \ge \text{cost}'_p(\mathscr{U})$ where the equality holds if $p=1$. We then have

$$\begin{aligned}
\text{cost}_p(\mathscr{U}) \ge \text{cost}'_p(\mathscr{U}) &= \sum_{\boldsymbol{x}\in\mathscr{X}} d_p(\boldsymbol{x},\pi(\boldsymbol{x})) = \sum_{\boldsymbol{x}\in\mathscr{X}}\sum_{i\in[d]}\sum_{[a,b]\in\mathscr{I}_i(x_i,\pi(x)_i)} |a-b|^p \\
&= \sum_{\boldsymbol{x}\in\mathscr{X}}\sum_{i\in[d]}\sum_{[a,b]\in\mathscr{I}_i(x_i,\pi(x)_i)} \int_a^b p(\theta-a)^{p-1}d\theta \\
&\ge \sum_{\boldsymbol{x}\in\mathscr{X}}\sum_{(i,[a,b])\in\mathscr{I}} \int_a^b p\cdot\min(\theta-a,b-\theta)^{p-1}\cdot\mathbb{1}[\theta \text{ is between } x_i \text{ and } \pi(x)_i]\,d\theta \qquad (5.7) \\
&= p\sum_{(i,[a,b])\in\mathscr{I}} \int_a^b \min(\theta-a,b-\theta)^{p-1} f_i(\theta)\,d\theta. \qquad (5.8)
\end{aligned}$$

The inequality in (5.7) above needs an explanation. First notice that, by the definition of $\mathscr{I}$, summing over $(i,[a,b])\in\mathscr{I}$ is the same as summing over $i\in[d]$ and $[a,b]\in\mathscr{I}_i(\mu_i^-,\mu_i^+)$. Fix some point $\boldsymbol{x}$ and dimension i, and w.l.o.g. assume that $x_i \le \pi(\boldsymbol{x})_i$. Then each interval $[a,b]\in\mathscr{I}_i(\mu_i^-,\mu_i^+)$ falls into one of the following categories:

1. $b \le x_i$ or $\pi(\boldsymbol{x}))_i \le a$. The contribution from such intervals are zero in both sides of the inequality in (5.7).

2. $x_i \le a \le b \le \pi(\boldsymbol{x})_i$. The contribution from such intervals to the left hand side of (5.7) is at least as the contribution to the right hand side because of the min function.

3. $a < x_i < b$. Note that $\int_a^b(\theta-a)^{p-1}d\theta = \int_a^b(b-\theta)^{p-1}d\theta$. Hence, in this case, the contributions to both sides of (5.7) are equal if $x_i \ge (a+b)/2$. Otherwise, the contribution to the L.H.S. is higher.

4. $a < \pi(\boldsymbol{x}) < b$. This case is analogous to Item 3.

The inequality in (5.7) follows by applying this observation to each interval in $\mathscr{I}$ and each point in $\mathscr{X}$.

The theorem statement then follows by combining bounds (5.6) and (5.8). $\square$

5.4.2 Implementation details

Note that Algorithm 4 differs from Algorithm 3 in the sampling step and the new sample discarding step. Recall that the implementation details of Algorithm 3 are presented in Section 5.3.3. In this section, assuming that we can sample a $\theta \in [a,b]$ with p.d.f. $P_{a,b}(\theta)$ from a given interval $[a,b]$ in $\Theta(1)$ time, we show how to efficiently implement the sampling step of Algorithm 4. In particular, we show how to select the dimension–interval pair $(i,[a,b])$ with probability proportional to $|b-a|^p$. Once the cut is sampled, the discarding step can be implemented by simulating the splitting operation and ignoring the cut if it separates two centers that are too close.

Suppose that for each dimension i, we maintain a data structure S_i that stores the intervals in $\mathscr{I}_i(\mu_i^-,\mu_i^+)$ that are not yet split by a cut. There are $k-1$ disjoint intervals in $\mathscr{I}_i(\mu_i^-,\mu_i^+)$, and we assume they are ordered by the left coordinate and indexed $[a_1,b_1],\ldots,[a_{k-1},b_{k-1}]$. Additionally, each S_i supports the following operations:

1. Initialize with all intervals in $\mathscr{I}_i(\mu_i^-,\mu_i^+)$ in $O(k\log k)$ time.

2. Remove an interval in $\mathscr{I}_i(\mu_i^-,\mu_i^+)$ in $O(\log k)$ time.

3. Given two indices $\ell, r \in [k-1]$, answer the query for $\sum_{j=\ell}^{r} |b_j - a_j|^p \mathbb{1}[(a_j,b_j) \in S_i]$ in $O(\log k)$ time.

We can implement S_i as a segment tree.

Now we can sample an interval from $S_1,\ldots,S_d$ as follows in $O(d\log k)$ time: We first query $(1,k-1)$ in each tree, aggregate the results, and pick a dimension i with the correct probability. Then we select an interval from S_i with the correct probability by employing a binary-search like algorithm. To elaborate, we first query it for $(1,\lfloor (k-1)/2\rfloor)$ and $(\lfloor (k-1)/2\rfloor)+1,k-1)$ and use the results to randomly decide if the index of the sampled interval should be in the sub-range $\{1,\ldots,\lfloor (k-1)/2\rfloor\}$ or $\{\lfloor (k-1)/2\rfloor)+1,\ldots,k-1\}$. Then we recursively apply the same procedure on the selected sub-range of indices until we end up with only one interval. A crude runtime analysis gives $O(\log^2 k)$ running time for the recursive sampling as there are $O(\log k)$ queries and each query takes $O(\log k)$ time. However we can modify the segment tree such that the partial sums maintained in the segment tree coincide with our queries so that each query can be answered in constant time.

5.5 Lower Bound

In this section we show how to construct an instance of the clustering problem such that any explainable clustering has cost at least $\Omega(k^{p-1})$ times larger than the optimal non-explainable clustering for the objective function given by the ℓ_p-norm, for every $p \geq 1$. In particular, for $p = 2$, this entails an $\Omega(k)$ lower bound for (explainable) k-means.

Let m be a prime. Our hard instance is in $\mathbb{R}^d$ for $d = m \cdot (m-1)$ and the set of dimensions corresponds to the set of all linear functions over $\mathbb{Z}_m$ with non-zero slope. That is, we associate the i-th dimension with the function $f_i : x \mapsto (a_i x + b_i) \bmod m$, where $a_i = 1 + \lfloor i/m \rfloor$ and $b_i = i \bmod m$. Consider $k = m$ centers $\boldsymbol{\mu}^1, \ldots, \boldsymbol{\mu}^k$ such that the i-th coordinate of the j-th center is given by $\mu_i^j = f_i(j)$. For each center $\boldsymbol{\mu}^j$ we create a set of $2d$ points B_j, each point differing from the center in exactly one dimension by either -1 or $+1$, i.e., $B_j = \{\boldsymbol{\mu}^j + c \cdot \boldsymbol{e}^i \mid c \in \{-1, 1\}, i \in [d]\}$, where $\boldsymbol{e}^i$ denotes the standard basis vector in the i-th dimension. Then, our hard instance is just $\bigcup_{j \in [k]} B_j$.

Since every point is at distance exactly 1 from its closest center, the cost of the optimal clustering OPT is equal to the total number of points $n = 2dk$ (regardless of the ℓ_p-norm). We will prove that:

Claim 1. Any two centers are at the same distance $\delta = \Theta(d^{1/p} k)$ from each other.

Claim 2. Any nontrivial threshold cut, i.e., one that separates some two centers, separates also some two points from the same B_j.

It follows that, in any explainable clustering, already the first threshold cut (from the decision tree's root) forces some two points from the same set B_j to eventually end up in two different leaves, and hence at least one of the k leaves has to contain two points from two different B_j's. The distance between these two points, by the triangle inequality, is at least $\delta - 2$, and therefore the cost of the explainable clustering is at least $\Omega(\delta^p) = \Omega(dk^p)$, which is $\Omega(k^{p-1}) \cdot \mathrm{OPT}$.

Proof of Claim 1. Fix two different centers $\boldsymbol{\mu}^{j_1}$, $\boldsymbol{\mu}^{j_2}$, $j_1 \neq j_2$. Their distance δ satisfies

$$\delta^p = \sum_{i \in [d]} \left| f_i(j_1) - f_i(j_2) \right|^p = \sum_{a=1}^{m-1} \sum_{b=0}^{m-1} \left| (aj_1 + b) \bmod m - (aj_2 + b) \bmod m \right|^p.$$

For $a \in \{1, \ldots, p-1\}$, let $x(a) = (aj_1 - aj_2) \bmod m$. Observe that

$$\left| (aj_1 + b) \bmod m - (aj_2 + b) \bmod m \right| \in \{x(a), m - x(a)\},$$

and whether it is $x(a)$ or $m - x(a)$ depends on b, with it being $x(a)$ for exactly $m - x(a)$ values of b and $m - x(a)$ for the remaining $x(a)$ values of b. Hence,

$$\delta^p = \sum_{a=1}^{m-1} (m - x(a)) \cdot x(a)^p + x(a) \cdot (m - x(a))^p.$$

Since $j_1 \not\equiv j_2 \pmod m$, we have $\{x(a) \mid a \in \{1,\dots,m-1\}\} = \{1,\dots,m-1\}$, and

$$\delta^p = \sum_{i=1}^{m-1}(m-i)\cdot i^p + i\cdot(m-i)^p = 2\cdot\sum_{i=1}^{m-1}(m-i)\cdot i^p = \Theta(m^{p+2}) = \Theta(dk^p).$$

□

Proof of Claim 2. Let the cut be (i,θ). It must be that $0 \le \theta < m-1$, because otherwise the cut would not separate any two centers. Note that there exists a center $\boldsymbol{\mu}^j$ with $\boldsymbol{\mu}^j_i = \lfloor\theta\rfloor$. Indeed, consider $j = (\lfloor\theta\rfloor - b_i)\cdot a_i^{-1} \bmod m$, using the fact that a_i and m are coprime. To finish the proof observe that the cut separates point $(\boldsymbol{\mu}^j + \boldsymbol{e}^i) \in B_j$ from all other points in B_j. □

5.6 The Minimum Cut Algorithm Loses $\Omega(k)$ Factor for k-Medians

We give an example where the minimum cut algorithm of [Das+20] produces a threshold tree with cost $\Omega(k)$ times the cost of an optimal clustering in the ℓ_1-norm. The idea is to start with the lower bound example in Section 5.5 since any two centers are "far apart". By adding a dimension for each center in which fewer edges are cut, the minimum cut will make linearly many cuts that split only one center. Combined with the large distance to reassign a point to the wrong center, the result is the minimum cut algorithm losing an $\Omega(k)$ factor. In the ℓ_1-norm, it suffices to map half of the coordinate values to -1 and the other half to +1 and still maintain the "large" distance between centers. The remainder of this section is a formal description of the instance.

Take the lower bound example from Section 5.5 and increase the dimension by k. Now the points are in $\mathbb{R}^{d+k}$ with $d+k$ coordinates (recall that $d = m(m-1)$ and $k = m$ with m prime). First, we describe the k centers $\mathcal{U}' = \{\boldsymbol{\mu}'^1,\dots,\boldsymbol{\mu}'^k\}$ as a mapping from the centers $\mathcal{U} = \{\boldsymbol{\mu}^1,\dots,\boldsymbol{\mu}^k\}$ in Section 5.5. For the first d coordinates, $\mu'^j_i = \mu^j_i \mod 2$. For the last k coordinates, center $\boldsymbol{\mu}'^j$ has a 0 in every coordinate $d+i$, $1 \le i \le k$, except coordinate $d+j$ which is a 1.

The reasoning behind this mapping is that the family of functions f_i in Section 5.5 is the standard construction of a family of pairwise independent hash functions [FKS84]. In particular, if $f_{a,b}(x) = (ax+b) \bmod m$ and $h_{a,b}(x) = f_{a,b}(x) \bmod 2$, then for $x \neq y$, $h_{a,b}(x) = h_{a,b}(y)$ with probability at most $1/2$ when a and b are chosen uniformly at random from $\{0,1,\dots,m-1\}$, $a \neq 0$. Recall that $f_i(x) = (a_i x + b_i) \bmod m$ where a_i, b_i range over all elements in $\{1,2,\dots,m-1\}$, $\{0,\dots,m-1\}$, respectively, and $\mu^j_i = f_i(j)$. Fix any pair of centers $\boldsymbol{\mu}^{j_1},\boldsymbol{\mu}^{j_2}$ where $j_1 \neq j_2$. Note that picking $i \in [d]$ uniformly at random is equivalent to picking $a_i, b_i \in \{0,1,\dots,m-1\}$, $a \neq 0$, uniformly at random due to the definition of a_i, b_i. We have $\mu'^{j_1}_i = (\mu^{j_1}_i \bmod 2) = (\mu^{j_2}_i \bmod 2) = \mu'^{j_2}_i$ with probability at most $1/2$ over the uniformly random choice of i, so any pair of centers are the same on at most $1/2$ of the coordinates. Hence, our new centers $\mathcal{U}'$ are at pairwise distance $\Theta(d)$.

Now we define the remaining points. Let $\boldsymbol{e}^i$ be the standard $(d+k)$-dimensional i-th basis vector. Similar to Section 5.5, we have a set B'_j for each center $\boldsymbol{\mu}'^j$ with $2d$ points where each point differs from $\boldsymbol{\mu}'^j$ on one of the first d coordinates by ± 1. Additionally, we want $(k-1)/2$ points to differ on one of the last k coordinates. To this end, define $B'_j = \{\boldsymbol{\mu}'^j + c \cdot \boldsymbol{e}^i \mid c \in \{-1,1\}, i \in [d]\} \cup \{\boldsymbol{\mu}'^j - \boldsymbol{e}^{d+j}\}^{(k-1)/2}$, where $\{\boldsymbol{\mu}'^j - \boldsymbol{e}^{d+j}\}^{(k-1)/2}$ denotes a multiset of $(k-1)/2$ copies of the point $\boldsymbol{\mu}'^j - \boldsymbol{e}^{d+j}$.

In particular, our construction has the following properties:

1. The distance between any pair of centers is $\Theta(d)$.
2. A cut (i,θ) in any dimension i, $1 \le i \le d$, and $\theta \in (0,1)$ splits some two centers and the number of points separated is equal to the number of centers.
3. A cut (i,θ) in any dimension i, $d+1 \le i \le d+k$, and $\theta \in (0,1)$ splits some two centers and separates $\approx k/2$ points.

Property (2) holds because for any dimension $1 \le i \le d$ and for each center $\boldsymbol{c}$, c_i is either 1, in which case there is exactly one point at $\boldsymbol{c} - \boldsymbol{e}^i$, or 0, in which case there is exactly one point at $\boldsymbol{c} + \boldsymbol{e}^i$. Note that this further implies that, when (after separating some centers) x centers are remaining, the number of points separated by a cut of type (2) will be equal to x. Then the cuts in (3) will be minimum for $\approx k/2$ cuts of all minimum cuts required to separate all centers since each cut in (3) separates exactly one center from the remaining centers. Hence we have that the minimum cut algorithm of [Das+20] will construct a threshold tree with $\Omega(k)$ height by making some $\Omega(k)$ cuts in dimensions $d+1$ through $d+k$.

To see that the minimum cut algorithm loses a $\Omega(k)$ factor, note that the optimal clustering has a value of $2dk + (k-1)k/2 = \Theta(k^3)$. The first term in the sum is because each of $2d$ points in each cluster differs from the center by ± 1 in exactly one of the first d coordinates and the second term is because $(k-1)/2$ of the points in each cluster differ by -1 from the center in one of the last k coordinates. On the other hand, an algorithm that always makes a minimum cut incurs a cost of $\Theta(dk^2)$ to reassign $\approx k/2$ points to the wrong center for $\approx k/2$ centers, just for those cuts of type (2). This gives an overall cost of $\Omega(dk^2)$ for the threshold tree produced. Since $d = \Theta(k^2)$ we have that the minimum cut algorithm is $\Omega(k)$ away from the cost of an optimal clustering.

6 Future Work

In spite of its simple mathematical formulation, clustering is a classic problem that has produced many algorithmic techniques. Moreover, it is a problem relevant to practical applications in statistics, machine learning, and operations research. In this thesis we presented some theoretical frameworks for studying clustering problems that were motivated by fairness and explainability. Here, we discuss some open questions related to the problems we have explored and possible avenues for making progress. We also briefly summarize related follow-up works that were completed after the publication of the manuscripts contained in this thesis.

Colorful k-Center Clustering

In Chapter 3 we described a 3-approximation for colorful k-center clustering. In fact, it is actually a 2-approximation except in instances that are not *well-separated*; that is, when there exists a ball of radius three times the optimal radius that can cover two balls in an optimal solution. An obvious open question is whether it is possible to obtain a 2-approximation for colorful k-center in all instances, as such an approximation exists for k-center and k-center with outliers. On the other hand, any reduction proving that a 3-approximation is tight for colorful k-center with two colors would need to start with a problem that is no longer hard when the parameter corresponding to k is increased to $k+1$. This is in light of the result in [Ban+19] that gives a solution using at most $k+\omega-1$ centers with radius twice that of the optimal solution for ω colors.

Let us now discuss what may be required to obtain a 2-approximation for the colorful k-center problem, should such a solution be possible. Although it may appear that we are not too far from such an approximation, our 2-approximation algorithm in well-separated instances relies crucially on our definition of well-separated and the structure that such an instance presents. In particular, we do not know how many extra points can be obtained through radius expansion if the balls of an optimal solution are not well-separated. Expanding to gain more points is the main idea that allows us to remove the extra centers from the pseudo-approximation. It is possible that significantly different ideas will be needed here, comparable to how the 2-approximation of k-center with outliers was obtained through considering non-uniform k-center [IV22].

Non-Uniform k-Center Clustering

In Chapter 4 we presented a constant-factor approximation for 3-NUkC. However, the conjecture in [CGK16] that there exists a constant-factor approximation for t-NUkC when t is a constant remains wide open. A recent publication by Inamdar and Varadarajan [IV22] gave a constant-factor approximation for four different types of radii. Interestingly, they make use of colorful k-center (with two color classes) in order to “remove” the smallest radius type. Essentially, an instance of $(t+1)$-NUkC with outliers can be reduced to $O(n)$ instances of colorful t-NUkC where n is the number of points in the instance. By finding a constant-factor approximation to the colorful version of 2-NUkC, a constant-factor approximation can then be obtained for robust 3-NUkC and finally 4-NUkC. Here we give a brief summary of the reduction from robust $(t+1)$-NUkC to colorful t-NUkC.

Let us begin with an instance of *well-separated* robust t-NUkC since reducing from $(t+1)$-NUkC to such an instance was an approach already used in [CN21] and [Jia+22]. The first step is to use the greedy clustering algorithm from [Cha+01] to reduce the smallest radius to zero, while increasing the other radii by a constant. The result is that points in this new instance are weighted since multiple points can be clustered to be co-located at a single point, and this gives an ordering of the points by non-increasing weight. The x heaviest points are colored red while the remaining are colored blue and the $O(n)$ instances are created by guessing the value of $0 \leq x \leq n$. Finally, the weight of the colored points and the coverage requirements are set in such a way that there is a feasible solution for at least one value of x. Given such a solution, it is then possible to obtain a feasible solution for the original robust t-NUkC instance.

It remains to approximate the colorful 2-NUkC problem to obtain an approximation for 4-NUkC. First, a colorful t-NUkC problem can be reduced to one in which the smallest radius is 0, and this can be done for any number of radii. However, a constant-factor approximation is only known for the version of this problem with just two radii (recall that an algorithm exists to reduce the smallest radius to 0). This is because when the larger radius balls satisfy the well-separated condition and the smaller radius is 0, the clusters form a laminar family so a solution can be found using dynamic programming.

At this point, however, we may have reached the limits of what this technique can do. For example, we do not know if we can use more than two colors to further reduce the number of radii in robust t-NUkC. The reason we can reduce the number of radii by one is as follows. If we consider all the (multi-)points ordered in non-increasing order by weight, we can greedily move the points covered by the smallest radius, which has been reduced to zero, to a prefix of the heaviest points that are either outliers or covered by the smallest radius in an optimal solution. This is the meaning of the coloring—we ensure that there is a solution where balls of the smallest radius in robust t-NUkC are only placed at red points. If we naively try to apply the same technique to more than one radius we immediately run into a problem since we do not know how to color the points so that a correct number is allocated to each radius. In effect, this paper develops many clever techniques to extract a solution for an additional radius, but we do not yet have enough techniques to fully resolve the NUkC conjecture.

The non-uniform k-center problem has been a fruitful line of research. Many connections between different problems have been discovered while designing a constant-factor approximation for small values of t. For example, the paper [CGK16] that defined the original problem discussed connections to the firefighter problem on trees and also gave a tight approximation factor of 2 for k-center with outliers. A 3-approximate algorithm was known since 2001 [Cha+01] but it was only fifteen years later that [CGK16] obtained a tight 2-approximation while looking at a far more general version of the clustering problem. It is possible that a tight approximation algorithm for colorful k center would also require looking at the problem from a significantly different point of view.

Explainable Clustering

In the main chapter following colorful k-center we presented nearly-tight approximation algorithms for explainable clustering that were oblivious to the data points. Our lower bound for k-means (and higher pth power of ℓ_p-norms) also only leaves a polylogarithmic gap from the guarantees of the approximation algorithm. One problem left open by our work was a tight lower bound on the price of explainability for the ℓ_1-norm. Very recent results [Gup+23; MS23] resolve this question and show that the price of explainability for the ℓ_1-norm is at most $1 + H_{k-1}$ and that the algorithm presented in this thesis that takes random cuts achieves this approximation factor. Furthermore, [Gup+23] showed that no polynomial time algorithm can approximate explainable k-medians (and also k-means) better than $(1/2 - o(1))\ln k$ unless $P = NP$. The upper bound for the price of explainability for k-means was also improved to $O(k \ln\ln k)$, but the tight approximation factor is still open as the lower bound is $\Omega(k)$.

Markarychev and Shan published a work [MS22] that gives an $\tilde{O}(1/\delta \log^2 k)$ bi-criteria approximation for k-means clustering that opens $(1+\delta)k$ clusters. The motivation behind opening more centers was the observation that while there exists an $O(\log^2 k)$ algorithm for explainable k-medians, the lower bound for explainable k-means is $\Omega(k)$, which is exponentially worse. For large data sets this can be a quite poor approximation. However, experiments indicate that the approximations are much better in practice. This bi-criteria approximation provides an explanation to this phenomenon since for a fixed data set and for larger values of k, the optimal clustering with $k+1$ centers is not much better than an optimal k-clustering. We can also interpret this result as relaxing the explainability requirement because the resulting clustering is not entirely split along axis-aligned cuts, but rather some points of some clusters can lie on the other side of a cut.

Another direction to pursue with explainable clustering also involves relaxing the explainability condition: removing the restriction that cuts must be axis-aligned and instead allowing general hyperplane cuts. For a slightly more constrained problem we can require the number of dimensions used for each hyperplane to be fixed or upper bounded. More data features can be incorporated into an “explanation” by using general hyperplane cuts. This is in contrast to the original formulation of explainability where there can be at most $k-1$ nodes from a leaf to the root, and hence every cluster is explained by at most $k-1$ features. It would be interesting to develop other explainable clustering methods that incorporate more than $k-1$ features.

So far, in all of these problems, we have only discussed the price of explainability, that is, the approximation factor when comparing to the best general clustering. It is also an interesting problem to consider efficient approximation algorithms that are compared to the best possible explainable clustering.

A Supplementary Material for Chapter 3

A.1 Dynamic Program for Dense Points

In this section we describe the dynamic programming algorithm discussed in Lemma 3.2.9. As stated in the proof of Lemma 3.2.9, given $I = \cup_j I_j$ and correct guesses for k_d, b_d, r_d, we need to find k_d balls of radius one centered at points in I covering b_d blue and r_d red points with at most one point from each $I_j \in I$ picked as a center. To do this, we first order the sets in I arbitrarily as $I = \{I_{j_1}, \ldots, I_{j_m}\}, m = |I|$. We create a 4-dimensional table T of dimension (m, b_d, r_d, k_d). $T[m', b', r', k']$ stores whether there is a set of k' balls in the first m' sets of I covering b' blue and r' red points. The recurrence relation for T is

$$\begin{aligned}
T[0,0,0,0] &= \text{True} \\
T[0,b',r',k'] &= \text{False}, \quad \text{for any } b',r',k' \neq 0 \\
T[m',b',r',k'] &= \begin{cases} \text{True} & \text{if } T[m'-1,b',r',k'] = \text{True} \\ \text{True} & \text{if } \exists c \in I_{j_{m'}} \text{ s.t. } T[m'-1,b'',r'',k'-1] = \text{True, for} \\ & b'' = b' - |B(c) \cap B|, r'' = r' - |B(c) \cap R| \\ \text{False} & \text{otherwise} \end{cases} .
\end{aligned}$$

The table T has size $O((m+1)\cdot(n+1)\cdot(n+1)\cdot(n+1)) = O(n^4)$ since the first parameter has range from 0 to m, and the other parameters can have value 0 up to at most n. Moreover, since $|I_j| \leq n$ for all $I_j \in I$, we can compute the the whole table in time $O(n^5)$ using e.g. the bottom-up approach. We can also remember the choices in a separate table and so we can find a solution in time $O(n^5)$ if it exists.

A.2 The Clustering Algorithm

In this section we present the clustering algorithm used in [Ban+19] with a simple modification. The algorithm is described in pseudo-code in Algorithm 5.

Algorithm 5: Clustering Algorithm

```
40 $S \leftarrow \emptyset$, $X' \leftarrow X$
41 while $X' \neq \emptyset$ and $\max_{j \in X'} z_j > 0$ do
42     $j \in X'$ be a point with maximum $z_j$: let $S \leftarrow S \cup \{j\}$
43     $y_j \leftarrow \min\{1, \sum_{i \in B(j)} x_i\}$; $\tilde{z}_j \leftarrow y_j$
44     $C_j \leftarrow \mathscr{F}(j) \cap X'$
45     For all $j' \neq j \in C_j$, set $y_{j'} \leftarrow 0$, $\tilde{z}_{j'} \leftarrow \tilde{z}_j$
46     $X' \leftarrow X' \setminus C_j$
47 end
```

Now we state the theorem which states the properties of this clustering algorithm used in Section 3.2.1.

Theorem 12. *Given a feasible fractional solution (x, z) to LP1, the set of points $S \subseteq X$ and clusters $C_j \subseteq X$ for every $j \in S$ produced by Algorithm 5 satisfy:*

1. *The set S is a subset of the points $\{j \in X : z_j > 0\}$ with positive z-values.*
2. *For each $j \in S$, we have $C_j \subseteq \mathscr{F}(j)$ and the clusters $\{C_j\}_{j \in S}$ are pairwise disjoint.*

Moreover, if we let $R_j = C_j \cap R$ and $B_j = C_j \cap B$ with $r_j = |R_j|$ and $b_j = |B_j|$ for $j \in S$, then y is a feasible solution to LP2 (depicted on the right in Fig.3.1) with objective value at least r.

Proof. The proof of the first statement is clear from the condition in the while loop of the algorithm.
For the second statement, observe that, by the definition of C_j as stated in the algorithm, $C_j \subseteq \bigcup_{i \in B(j)} B(i) = \mathscr{F}(j)$. Since in each iteration, the cluster is removed from X', the clusters are clearly disjoint.
In order to prove that y is feasible this we first state some useful observations.

- First, for any $i \in X$ there is at most one $j \in S$ such that $d(i, j) \leq 1$. This is true because if there were $j, j' \in S$ such that both $j, j' \in B(j)$ then, assuming w.l.o.g. j was considered before in the while loop, $j' \in C_j$ and thus j' cannot be in S which is a contradiction.
- Second, note that for any $j_1 \in X$ such that $j_1 \in C_j$ for some j, then $\tilde{z}_j = \tilde{z}_{j_1} \geq z_{j_1}$. This is trivially true if $\tilde{z}_j = 1$, otherwise $\tilde{z}_j = \sum_{i \in B(j)} x_i \geq z_j \geq z_{j_1}$ where the first inequality follows from LP1 constraints and second inequality from the fact that when C_j was removed, z_j had the highest z value.

Now we show that y is feasible for LP2 with objective value at least r. First, we show that

$\sum_{j\in S} r_j y_j \geq r$. To see this,

$$\begin{aligned}
\sum_{j\in S} r_j y_j &= \sum_{j\in S} |R_j| y_j \\
&= \sum_{j\in S}\sum_{j'\in R_j} \tilde{z}_j \qquad (y_j = \tilde{z}_j \text{ for any } j \in S) \\
&\geq \sum_{j\in S}\sum_{j'\in R_j} z_{j'} \qquad (\text{from second observation, } \tilde{z}_j \geq z_{j'} \text{ for any } j' \in C_j) \\
&= \sum_{j'\in R: z_{j'}>0} z_{j'} \qquad (\text{since } C_j\text{'s are disjoint and contain all } j \text{ s.t. } z_j > 0) \\
&= \sum_{j'\in R} z_{j'} \geq r \qquad (\text{since } z \text{ satisfies LP1}))
\end{aligned}$$

Similarly $\sum_{j\in S} b_j y_j \geq b$. Finally we will show that $\sum_{j\in S} y_j \leq k$,

$$\begin{aligned}
\sum_{j\in S} y_j &\leq \sum_{j\in S}\sum_{j'\in B(j)} x_{j'} \qquad (\text{since } y_j \leq \sum_{j'\in B(j)} x_{j'}) \\
&\leq \sum_{j'\in X} x_{j'} \qquad (\text{from the first observation}) \\
&\leq k \qquad (\text{since } x \text{ satisfies LP1})
\end{aligned}$$

This concludes the proof of the claim that y is a feasible solution to LP2 with objective value at least r. □

B Supplementary Material for Chapter 4

B.1 Dynamic Program for Laminar Instances

In this section we describe the details of the dynamic programming algorithm that can be used to solve a laminar robust 2-NUkC instance. Recall that a robust 2-NUkC instance $\mathcal{I} = ((X,d),(k_1,r_1),(k_2,r_2),m)$ is said to be *laminar* if we are given sets $L_1, L_2 \subseteq X$ such that the following are satisfied.

1. The k_i balls of radius r_i are only allowed to be centered at points in L_i, $i \in \{1,2\}$;
2. $\mathcal{B}(u,r_i) \cap \mathcal{B}(v,r_i) = \emptyset$ for all $u,v \in L_i$, $i \in \{1,2\}$;
3. $C(v) \cap C(v') = \emptyset$ for all $v,v' \in L_1$, where $C(v) = \{u \in L_2 : \mathcal{B}(v,r_1) \cap \mathcal{B}(u,r_2) \neq \emptyset\}$.

For any $v \in L_1$ we refer to the set $C(v)$ as the children of v. Let $v_1, v_2, \ldots, v_{|L_1|}$ be an enumeration of points in L_1, and for each v_i let $u_{1,i}, u_{2,i}, \ldots, u_{|C(v_i)|,i}$ be an enumeration of the points in $C(v_i)$. See Figure 4.2 for an illustration of a laminar instance. In our dynamic programming algorithm, we will have a local table and a global table. Since the sub-instance $\mathcal{B}(v_i,r_1) \cup (\cup_{j=1}^{|C(v_i)|} \mathcal{B}(u_{j,i},r_2))$ is completely disjoint from the rest of the instance for all i, we will have a local table that computes solutions to this sub instance. The global table will then be used to combine solutions to these sub instances.

The local table LOCAL will be indexed by i, bit, j, k'_2, m'. We set LOCAL$[i,bit,j,k'_2,m'] = 1$ if and only if there is a solution that opens k'_2 balls of radius r_2 centered at points in $\{u_{1,i},\ldots,u_{j,i}\}$ and these balls cover at least m' points, assuming a ball of radius r_1 is open at v_i if and only if $bit = 1$. We will compute this table bottom-up from $j = 1$ to $|C(v_i)|$ for a fixed i and bit, each time computing the table entry for all values of k'_2, m' for each j. Now by laminarity of the instance, balls of radius r_2 centered at points in $C(v_i)$ are disjoint. Thus we can express LOCAL$[i,bit,j,\ldots]$ in terms of LOCAL$[i,bit,j-1,\ldots]$ easily by enumerating whether there is a ball of radius r_2 open at $u_{j,i}$ or not, and if yes then reducing the budget of points needed to be covered by the amount we gain by a ball of radius r_2 centered at $u_{j,i}$. We formalize this and state the recurrence with the base cases below.

$$\textsc{local}[i, bit, 0, 0, 0] = 1$$

$$\textsc{local}[i, bit, 0, k_2', m'] = 0, \quad \text{for any other } m', k_2' \neq 0$$

$$\textsc{local}[i, bit, j, k_2', m'] = \begin{cases} \textsc{local}[i, bit, j-1, k_2', m'] \text{ if } u_{j,i} \text{ not taken,} \\ \\ \textsc{local}[i, 1, j-1, k_2'-1, m' - |(\mathscr{B}(u_{j,i}, r_2) \setminus \mathscr{B}(v_i, r_1)| \\ \qquad\qquad\qquad \text{if } bit = 1 \text{ and } u_{j,i} \text{ taken,} \\ \\ \textsc{local}[i, 0, j-1, k_2'-1, m' - |\mathscr{B}(u_{j,i}, r_2)|] \\ \qquad\qquad\qquad \text{if } bit = 0 \text{ and } u_{j,i} \text{ taken.} \end{cases}$$

The recurrence for the global table GLOBAL is described next. This table is indexed by i, k_1', k_2', m' as follows. $\textsc{global}[i, k_1', k_2', m'] = 1$ if there is a solution that opens k_1' balls of radius r_1 centered at points in $\{v_j\}_{j=1}^{i}$ and k_2' balls of radius r_2 centered at points in $\cup_{j=1}^{i} C(v_j)$ and covers m' points.

Now by laminarity of the instance, balls of radius r_1 around points in L_1 are disjoint, $C(v)$ for all $v \in L_1$ are disjoint, and balls of radius r_2 around points in $\cup_{j=1}^{i} C(v_j)$ are disjoint as well. Thus we can express $\textsc{global}[i, k_1', k_2', m']$ in terms of $\textsc{global}[i-1, \ldots]$ and $\textsc{local}[v_i, |C(v_i)|, \ldots]$ by enumerating how many balls out of k_1' are open in $\{v_j\}_{j=1}^{i-1}$, how many in v_i, and similarly enumerate how the other parameters k_2', m' are divided. Formally, we set $\textsc{global}[i, k_1', k_2', m']$ to 1 if any one of the following are 1, otherwise we set it to 0.

- $\textsc{local}[v_1, 1, |C(v_i)|, k_2', m' - |\mathscr{B}(v_i, r_1)|]$ if $i = 1$ and $k_1' = 1$
- $\textsc{local}[v_1, 0, |C(v_i)|, k_2', m']$ if $i = 1$ and $k_1' = 0$
- $\bigvee_{k_2'', m''} \left[\textsc{global}[i-1, k_1'-1, k_2'-k_2'', m'-m''] \wedge \textsc{local}[v_i, 1, |C(v_i)|, k_2'', m'' - |\mathscr{B}(v_i, r_1)|] \right]$ $(v_i$ taken)
- $\bigvee_{k_2'', m''} \left[\textsc{global}[i-1, k_1', k_2'-k_2'', m'-m''] \wedge \textsc{local}[v_i, 0, |C(v_i)|, k_2'', m''] \right]$ $(v_i$ not taken)

After we have computed the LOCAL table for all parameter values, we can compute the GLOBAL table in a bottom-up fashion from $i = 1$ to $|L_1|$, each time computing $\textsc{global}[i, \ldots]$ for all values of the other parameters. If the instance is feasible then consider a feasible solution of the problem. Let k_{1i}, k_{2i} be the number of balls of radius r_1 and r_2 in this solution centered at points in $\{v_j\}_{j=1}^{i}$ and $\cup_{j=1}^{i} C(v_j)$ respectively and let m_i be the number of points covered by them. Then the global table value $\textsc{global}[i, k_{1i}, k_{2i}, m_i]$ will be feasible and set to 1 for all $1 \leq i \leq |L|$. Thus $\textsc{global}[|L_1|, k_1, k_2, m]$ will be set to 1. We can also remember the choices made while computing both the local and global tables bottom-up to also find a solution to the problem, if it exists.

Bibliography

[AG11] S. Arora and R. Ge. “New tools for graph coloring”. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2011, pp. 1–12.

[Ahm+20] S. Ahmadian et al. “Better Guarantees for k-Means and Euclidean k-Median by Primal-Dual Algorithms”. In: *SIAM Journal on Computing* 49.4 (2020). DOI: 10.1137/18M1171321.

[Ana+19] A. Anagnostopoulos et al. “Principal Fairness: Removing Bias via Projections”. In: *CoRR* abs/1905.13651 (2019).

[Ane+20] G. Anegg et al. “A Technique for Obtaining True Approximations for k-Center with Covering Constraints”. In: *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 2020, pp. 52–65.

[Ane+21] G. Anegg et al. “A technique for obtaining true approximations for k-center with covering constraints”. In: *Mathematical Programming* (2021), pp. 1–25.

[ASS17] H.-C. An, M. Singh, and O. Svensson. “LP-based algorithms for capacitated facility location”. In: *SIAM Journal on Computing* 46.1 (2017), pp. 272–306.

[AV07] D. Arthur and S. Vassilvitskii. “k-means++: the advantages of careful seeding”. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007*. SIAM, 2007, pp. 1027–1035.

[Bac+19] A. Backurs et al. “Scalable Fair Clustering”. In: *Proceedings of the 36th International Conference on Machine Learning, ICML*. 2019, pp. 405–413.

[Ban+19] S. Bandyapadhyay et al. “A Constant Approximation for Colorful k-Center”. In: *27th Annual European Symposium on Algorithms, ESA*. 2019, 12:1–12:14.

[Byr+15] J. Byrka et al. “An Improved Approximation for k-median, and Positive Correlation in Budgeted Optimization”. In: *Proceedings of the 2015 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2015, pp. 737–756. DOI: 10.1137/1.9781611973730.50.

[Car+00] R. D. Carr et al. “Strengthening Integrality Gaps for Capacitated Network Design and Covering Problems”. In: *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’00. San Francisco, California, USA: Society for Industrial and Applied Mathematics, 2000, pp. 106–115.

[CFG12] E. Chlamtac, Z. Friggstad, and K. Georgiou. “Understanding Set Cover: Sub-exponential Time Approximations and Lift-and-Project Methods”. In: *CoRR* abs/1204.5489 (2012).

[CGK16] D. Chakrabarty, P. Goyal, and R. Krishnaswamy. “The Non-Uniform k-Center Problem”. In: *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Vol. 55. 2016, p. 67.

[CH22] M. Charikar and L. Hu. “Near-Optimal Explainable k-Means for All Dimensions”. In: *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*. To appear. SIAM, 2022.

[Cha+01] M. Charikar et al. “Algorithms for Facility Location Problems with Outliers”. In: *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’01. Washington, D.C., USA: Society for Industrial and Applied Mathematics, 2001, pp. 642–651.

[Cha+02] M. Charikar et al. “A Constant-Factor Approximation Algorithm for the k-Median Problem”. In: *Journal of Computer and System Sciences* 65.1 (2002), pp. 129–149. DOI: 10.1006/jcss.2002.1882.

[Cha+15] D. Chakrabarty et al. “Approximability of capacitated network design”. In: *Algorithmica* 72.2 (2015), pp. 493–514.

[Chi+17] F. Chierichetti et al. “Fair Clustering Through Fairlets”. In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[CLV17] R. Chierichetti F.and Kumar, S. Lattanzi, and S. Vassilvitskii. “Fair clustering through fairlets”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2017, pp. 5029–5037.

[CN19] D. Chakrabarty and M. Negahbani. “Generalized center problems with outliers”. In: *ACM Transactions on Algorithms (TALG)* 15.3 (2019), pp. 1–14.

[CN21] D. Chakrabarty and M. Negahbani. “Robust k-Center with Two Types of Radii”. In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer, 2021, pp. 268–282.

[Coh+22] V. Cohen-Addad et al. “Improved Approximations for Euclidean K-Means and k-Median, via Nested Quasi-Independent Sets”. In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2022. Rome, Italy: Association for Computing Machinery, 2022, pp. 1621–1628. DOI: 10.1145/3519935.3520011.

[Das+20] S. Dasgupta et al. “Explainable k-Means and k-Medians Clustering”. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 7055–7065.

[EMN22] H. Esfandiari, V. Mirrokni, and S. Narayanan. “Almost Tight Approximation Algorithms for Explainable Clustering”. In: *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022.* To appear. SIAM, 2022.

[FKS84] M. L. Fredman, J. Komlós, and E. Szemerédi. “Storing a Sparse Table with O(1) Worst Case Access Time”. In: *Journal of the ACM* 31.3 (1984), pp. 538–544. DOI: 10.1145/828.1884.

[Gam+21] B. Gamlath et al. “Nearly-Tight and Oblivious Algorithms for Explainable Clustering”. In: *Thirty-Fifth Conference on Neural Information Processing Systems* (2021).

[GJ90] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* USA: W. H. Freeman & Co., 1990.

[Gon85] T. F. Gonzalez. “Clustering to minimize the maximum intercluster distance”. In: *Theoretical Computer Science* 38 (1985), pp. 293–306.

[Gri01] D. Grigoriev. “Complexity of Positivstellensatz proofs for the knapsack”. In: *Computational Complexity* 10.2 (2001), pp. 139–154.

[Gup+23] A. Gupta et al. *The Price of Explainability for Clustering.* https://arxiv.org/abs/2304.09743, 2023.

[Har+19] D. G. Harris et al. “A Lottery Model for Center-Type Problems With Outliers”. In: *ACM Trans. Algorithms* 15.3 (2019), 36:1–36:25.

[HN79] W.-L. Hsu and G. L. Nemhauser. “Easy and hard bottleneck location problems”. In: *Discrete Applied Mathematics* 1.3 (1979), pp. 209–215.

[HS85] D. S. Hochbaum and D. B. Shmoys. “A best possible heuristic for the k-center problem”. In: *Mathematics of operations research* 10.2 (1985), pp. 180–184.

[IV22] T. Inamdar and K. R. Varadarajan. “Non-Uniform k-Center and Greedy Clustering”. In: *SWAT.* 2022.

[Jia+22] X. Jia et al. “Towards Non-Uniform k-Center with Constant Types of Radii”. In: Jan. 2022, pp. 228–237. DOI: 10.1137/1.9781611977066.16.

[JSS21a] X. Jia, K. Sheth, and O. Svensson. “Fair colorful k-center clustering”. In: *Mathematical Programming* (2021), pp. 1–22.

[JSS21b] X. Jia, K. Sheth, and O. Svensson. “Fair colorful k-center clustering”. In: *Mathematical Programming* 192 (July 2021), pp. 1–22. DOI: 10.1007/s10107-021-01674-7.

[KAM19] M. Kleindessner, P. Awasthi, and J. Morgenstern. *Fair k-Center Clustering for Data Summarization.* 2019.

[Kan+04] T. Kanungo et al. “A local search approximation algorithm for k-means clustering”. In: *Computational Geometry* 28.2 (2004). Special Issue on the 18th Annual Symposium on Computational Geometry - SoCG2002, pp. 89–112. DOI: https://doi.org/10.1016/j.comgeo.2004.03.003.

[Kar72] R. M. Karp. “Reducibility among Combinatorial Problems”. In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations.* Ed. by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9.

[KMN11] A. R. Karlin, C. Mathieu, and C. T. Nguyen. “Integrality Gaps of Linear and Semi-Definite Programming Relaxations for Knapsack”. In: *Integer Programming and Combinatoral Optimization IPCO.* 2011, pp. 301–314.

[Las01a] J. B. Lasserre. “An explicit exact SDP relaxation for nonlinear 0-1 programs”. In: *International Conference on Integer Programming and Combinatorial Optimization (IPCO).* 2001, pp. 293–303.

[Las01b] J. B. Lasserre. “Global optimization with polynomials and the problem of moments”. In: *SIAM Journal on optimization* 11.3 (2001), pp. 796–817.

[Li16] S. Li. “Approximating capacitated k-median with $(1+\varepsilon)$ k open facilities”. In: *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms.* SIAM. 2016, pp. 786–796.

[Li17] S. Li. “On uniform capacitated k-median beyond the natural LP relaxation”. In: *ACM Transactions on Algorithms (TALG)* 13.2 (2017), pp. 1–18.

[Llo82] S. P. Lloyd. “Least squares quantization in PCM”. In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–136. DOI: 10.1109/TIT.1982.1056489.

[LM21] E. S. Laber and L. Murtinho. “On the price of explainability for some clustering problems”. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021.* Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 5915–5925.

[Mol19] C. Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable.* https://christophm.github.io/interpretable-ml-book/. 2019.

[MS21] K. Makarychev and L. Shan. “Near-Optimal Algorithms for Explainable k-Medians and k-Means”. In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021.* Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 7358–7367.

[MS22] K. Makarychev and L. Shan. “Explainable K-Means: Don’t Be Greedy, Plant Bigger Trees!” In: STOC 2022. Rome, Italy: Association for Computing Machinery, 2022, pp. 1629–1642.

[MS23] K. Makarychev and L. Shan. *Random Cuts are Optimal for Explainable k-Medians.* https://arxiv.org/abs/2304.09743, 2023.

[Mur+19] W. J. Murdoch et al. “Definitions, methods, and applications in interpretable machine learning”. In: *Proceedings of the National Academy of Sciences* 116.44 (Oct. 2019), pp. 22071–22080. DOI: 10.1073/pnas.1900654116.

[RSG16] M. T. Ribeiro, S. Singh, and C. Guestrin. “"Why Should I Trust You?": Explaining the Predictions of Any Classifier”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016*. ACM, 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778.

[Rud19] C. Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215. DOI: 10.1038/s42256-019-0048-x.

[Tul09] M. Tulsiani. “CSP gaps and reductions in the lasserre hierarchy”. In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC*. 2009, pp. 303–312.

[Vaz01] V. V. Vazirani. *Approximation Algorithms*. Springer Berlin, Heidelberg, 2001. DOI: 10.1007/978-3-662-04565-7.

[WS11] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. DOI: 10.1017/CBO9780511921735.

Xinrui JIA

(shin-ray)

xinrui.321@gmail.com xinruij.github.io
+41 78 334 23 63 Avenue du Temple 6, 1020 Renens VD, CH

EDUCATION

École Polytechnique Fédérale de Lausanne (EPFL) — *09.2018 - Present*
PhD in Computer Science — **Lausanne, Switzerland**
Theory of Computation Lab, research area: algorithms

University of Waterloo — *09.2014 - 06.2018*
Bachelor's in Mathematics *with Distinction - Dean's Honours List* — **Waterloo, Canada**
Majors: Combinatorics and Optimization, Pure Mathematics

RESEARCH INTERESTS

Combinatorial optimization, discrete algorithms, operations research.

SKILLS

Programming: Experience with Python, SQL, C/C++, Java, Matlab, Gurobi.
Optimization: Discrete optimization, algorithm design, modelling and problem solving.

PROFESSIONAL EXPERIENCE

Apple Inc., *Optimization Intern* — 06.2021 - 12.2021
Zurich, Switzerland
Conducted applied research in AI/ML team. Designed techniques for optimization of system-wide operations and implemented methods in Python. *Details proprietary.*

Private Tutor, *Self-employed* — 06.2020 - 05.2021
Remote
Tutored University of Waterloo classes over video conferences: Analysis, Intro to Optimization, Intro to Combinatorics, Graph Theory.

Grand River School, *Math Class Teacher* — 09.2016 - 04.2018
Waterloo, Canada
Designed and taught competition math class to students at non-profit organization.

University of Waterloo, *Residence Advisor* — 09.2016 - 04.2017
Waterloo, Canada
Organized and advertised residence activities, mediated conflicts, and assisted in emergencies.

HAA Analytics, *Backend Application Developer* — 05.2015 - 08.2015
Toronto, Canada
Used Python, NumPy, Pandas, SQL to build reinsurance analytics web app at start-up. Collaborated with team of 6 programmers in every part of software development cycle. Product successfully demoed 3 months into internship.

ACADEMIC EXPERIENCE

Simons Institute at UC Berkeley, *Visiting Student Researcher* — 08.2022 - 09.2022
Berkeley, USA
Participated in program Data-Driven Decision Processes.

EPFL, *Teaching Assistant* — 02.2019 - 12.2022
Lausanne, Switzerland
Conducted exercise sessions for the classes Theory of Computation and Algorithms. Experience as head TA, responsible for coordinating teaching duties and communication between professor and team of student TAs.

University of Waterloo, *Teaching Assistant* 09.2016 - 04.2018
Tutored classes in calculus, algebra, and linear algebra in residence tutorial center. **Waterloo, Canada**

EPFL Discrete Optimization Lab, *Research Intern* 01.2017 - 04.2017
Proved guarantees for two algorithms for max dispersion problem and explored applications to image recognition. Presented results with visualizations in PyGame. **Lausanne, Switzerland**

University of Waterloo, *Research Intern* 05.2016 - 08.2016
Implemented key-exchange protocol using C++ and SageMath. Developed cryptanalysis attack for proposed post-quantum Diffie-Hellman key-exchange protocol. **Waterloo, Canada**

LANGUAGES

English: Native
French: Intermediate
German: Beginner

COMPETITIONS AND AWARDS

University of Waterloo Mathematics Scholarships ($31 000 cad)	*2014 - 2018*
Highline Family Scholarship ($8 000 cad)	*2014 - 2017*
Ontario Music Festivals (provincial competition, grade 10 violin)	*2013*
Windsor Symphony Youth Orchestra (concertmaster)	*2012-2013*
Canada National Math Camp (invitation only top 21 students nationally)	*2012*
Western University High School Debate Competition (quarter finalist)	*2012*

ACTIVITIES AND HOBBIES

Rock climbing, hiking, violin, chamber music, choir.

CONFERENCE REVIEWING

Contributed as a reviewer/sub-reviewer in the following

- conferences: ICALP, SODA, ISAAC
- journals: Algorithmica, Theoretical Computer Science, Computer and System Sciences, Machine Learning Research

PUBLICATIONS

1. Xinrui Jia, Ola Svensson, and Weiqiang Yuan. The Exact Bipartite Matching Polytope Has Exponential Extension Complexity. In *Symposium on Discrete Algorithms* (SODA 2023).
2. Xinrui Jia, Lars Rohwedder, Kshiteej Sheth, and Ola Svensson. Towards Non-Uniform k-Center with Constant Types of Radii. In *Symposium on Simplicity in Algorithms* (SOSA 2022).
3. Buddhima Gamlath, Xinrui Jia, Adam Polak, and Ola Svensson. Nearly-tight and Oblivious Algorithms for Explainable Clustering. In *Conference on Neural Information Processing Systems* (NeurIPS 2021).
4. Xinrui Jia, Kshiteej Sheth, and Ola Svensson. Fair Colorful k-Center Clustering. In *Proceedings of the 21st International Conference on Integer Programming and Combinatorial Optimization* (IPCO 2020).