Algorithmic Complexity of Clustering and Low-Rank Approximation Problems

Kirill Simonov

Thesis for the degree of Philosophiae Doctor (PhD)
University of Bergen, Norway
2021

UNIVERSITY OF BERGEN

Algorithmic Complexity of Clustering and Low-Rank Approximation Problems

Kirill Simonov

Thesis for the degree of Philosophiae Doctor (PhD)
at the University of Bergen

Date of defense: 29.03.2021

Year: 2021

Title: Algorithmic Complexity of Clustering and Low-Rank Approximation Problems

Name: Kirill Simonov

Print: Skipnes Kommunikasjon / University of Bergen

Acknowledgements

My first and foremost gratitude goes to my advisor, Professor Fedor V. Fomin, whose constant stream of ideas and support was key to the successful completion of this work. I have been very fortunate to receive such a great deal of your time in fruitful discussions, co-working over paper drafts, and just checking on my progress. Thanks a lot for being the best guide in this complicated world of research I could possibly hope for. I would also like to thank my co-advisor, Professor Saket Saurabh, for providing thoughtful advice, filling everything around with energy and enthusiasm, and for your amazing courses that I had the opportunity to take.

Needless to say, I am thankful to the evaluation committee, Professor Cristina Bazgan, Vincent Cohen-Addad, and Pål Grønås Drange, for taking the time to read and evaluate this thesis.

No man is an island, and the results presented here would not have been achieved without collaboration with other amazing researchers. First of all, thank you William Lochet and Eduard Eiben for being my academic big brothers. Apart from our cool research projects, I learned a lot from you two during the numerous research discussions in the office corridors, as well as in coffee machine bantering. I am grateful to Petr A. Golovach for partaking in nearly all my scientific results, always being utterly productive and reliable. Thanks Danil Sagunov for the truly unstoppable problem solving capabilities that culminated in one of the coolest research projects I could partly entitle myself to. I would also like to express gratitude to the rest of my co-authors Sayan Bandyapadhyay, Fahad Panolan, Yogesh Dahiya; I very much enjoyed working with all of you.

It has been very fortunate for me to be a part of the diverse UiB algorithms group that always provided a motivating and friendly atmosphere. I am grateful to all members of the group for the whirlwind of nice ideas and all the pleasant interactions at the office, seminars, schools and workshops. Among the people I have met there or at other scientific events, I wish to specifically thank those with whom we have also spent a lot of good time outside of work: Erlend, for so many deep conversations and board game nights; Paloma, for being the coolest office mate, with you, it was easy to have both pleasant talks and a productive environment; Lars, Benjamin, Christophe, Olav, Arindam, Vasily, Liana. Additionally, I am thankful to everyone on the administrative side of the department, whose efficient work made my PhD experience so smooth. I would also like to thank the UiBdoc initiative for bringing together and supporting PhD students and postdocs across the whole university.

I would not be able to go through the PhD hardships, were it not for my dear friends who have made even the hardest research times so colorful. Have we met in the recent years or ages ago, I am grateful to you all. Thank you Alexey and Anastasia for everything that could be shared with you two, be it research, games or saunas. Zoya, for your incessant care and support even from the distant places. Anastasia,

for all the fun times we had, from Saint Petersburg to London. I am thankful as well to everyone from back home, Sasha, Alexander, Maksim, Kirill, Alexey, Anton, for numerous cheerful meetings we had both offline and online. Thanks to the SPbU Mathematics gang, Nikolay, Eugène and the rest. I am no less grateful for the friends I have made in Bergen; you helped me greatly to not feel isolated while working here. Thank you Belén, for all the trips and walks and talks. Mats, for the Russian-Norwegian collaboration, together with the exciting ASF stuff and the numerous beers. Marius, for so many cinematography pieces we enjoyed. Last but certainly not least, thank you Gin for coming into my life at just the right moment.

Finally, I would like to express how grateful I am to my family for always providing support no matter how far apart we were. Thanks Nikita and Roma for all the wonderful experiences we shared over these years; not only as brothers, but also as very good friends. Most of all, thank you Irina and Alexey for being the most reliable source of care and support throughout my research career, education, and my whole life. I am happy to dedicate this work to you.

Abstract

The two most popular unsupervised learning problems are k-Clustering and Low-Rank Approximation. Consider a set of n datapoints, in the k-Clustering problem, the objective is to partition these points into k clusters and select k centers such that each cluster is represented well by its center. In the Low-Rank Approximation problem, the task is to find an r-dimensional subspace that minimizes the sum of deviations from each point to the subspace. Both problems are of utmost importance for the modern data-driven applications, and both can be thought of as structured representation problems.

In this thesis, we provide a thorough study of the multivariate complexity of k-Clustering and Low-Rank Approximation. We focus on extensions of these problems, such as robust and constrained versions, that reach beyond the well-studied standard setting. The main body of the thesis is divided into three parts. In the first part, we study parameterized complexity of exact algorithms, where the parameter is the total cost of the clustering. Problems that constitute the main focus of this part are k-Clustering in L_p-norm for $p \in [0, \infty]$, and Categorical Clustering with Row/Column Outliers. We provide a number of fixed-parameter tractable algorithms based on hypergraph enumeration, and a number of hardness results. The second part can be summarized as employing sampling methods to provide $(1 + \varepsilon)$-approximation in FPT time. We show a space-efficient coreset for the Fair Clustering problem, and an FPT approximation scheme for the problem of clustering points with missing entries, where the number of missing entries in each point is bounded. Finally, in the third part we deal with the Low-Rank Approximation problem, and its robust variant, Low-Rank Approximation with Outliers. For the latter, we employ algebraic geometry methods to provide an $n^{\mathcal{O}(rd)}$ exact algorithm that is nearly tight even for arbitrary-factor approximation, and its dimensionality reduction-based improvements. We also present a PTAS for Low-Rank Approximation of binary matrices in column-sum norm.

Contents

1

Introduction

The modern world is dominated by data. The sheer spread of powerful computational devices allows for collection of megabytes of information from each particular individual. In turn, this data is used to determine a multitude of things in our life, ranging from our social network feed and streaming service suggestions, to credit card approval and our credibility in the eyes of law enforcement. All these automatic decisions are achieved by machine learning solutions. By now, there exists a vast multitude of machine learning primitives and approaches that skillful engineers pick, calibrate and combine with each other into convoluted pipelines to maximize the utility of decisions with respect to underlying data. As the power and the outreach of such systems grows, together with their internal complexity, so does the concern about explainability and transparency of these systems. One side is assessing the credibility of a particular solution on the market as a whole, which is a very general affair, requiring deep domain knowledge extending to legal and ethic aspects. However, this work is in the line of the opposite approach, where the main objective is to shed more light on the basic building blocks of data analysis, which can be rigorously studied as mathematical objects. Specifically, the main goal of this thesis is to study computational complexity and performance guarantees of the key unsupervised learning primitives.

To be more concrete, let us consider a particular problem. Suppose we have a set of n data points in $\mathbb{R}^d$, that we want to partition into k clusters while minimizing the dissimilarity of points in the same cluster. Denote the resulting clusters by C_1, $\ldots$, C_k and let the objective in this context be

$$\sum_{i=1}^{k} \min_{\mathbf{c}_i \in \mathbb{R}^d} \sum_{\mathbf{x} \in C_i} ||\mathbf{x} - \mathbf{c}_i||_2^2,$$

that is, minimize the sum of the squared Euclidean distances from the points of the cluster to its optimal center, summed over all clusters. This problem is the renowned

k-Means, and it is arguably the most prevalent unsupervised learning primitive. The k-Means problem also showcases perfectly the discrepancy between the practical usage and the theoretical analysis. In practice, the common thing is to apply a heuristic algorithm to obtain some locally optimal clustering. Consider for instance the most classical Lloyd's heuristic [147], that often produces good clusterings on real datasets, yet in general the cost of the clustering can be arbitrarily bad compared to the optimal value. There also exists a multitude of other heuristic algorithms for clustering, that further improve the performance in a number of ways. However, from the classical complexity viewpoint, the situation is quite baffling: k-Means is NP-hard even for $k = 2$ [9], and even for $d = 2$ [151]. Moreover, it is also known to be NP-hard to approximate the k-Means objective better than a certain constant factor [14].

Seeing such a gap between the theoretical complexity and the practical performance, it is natural to go for "beyond the worst-case" kind of analysis. Therefore, in this work we adopt the idea of *multivariate analysis* for the problems in question. The philosophy in this case, derived from the field of parameterized complexity, is to consider a secondary measure of the input apart from the input size, and express the running time of algorithms in terms of both measures. In particular, a central concept is a *fixed-parameter tractable* algorithm, that has the running time of $f(k)n^c$ for some function f of k and some constant c, where n is the size of the input, and k is the selected parameter. While f can be exponential or in fact any function, for any fixed value of k the running time is the same polynomial of n. Thus, the general-case NP-hardness might be circumvented for structured instances where the value of the parameter is small, without any loss in the quality of the solution. In particular, as a part of this thesis, we conduct an extensive study of the complexity of clustering problems depending on the parameter D, the cost of the optimal clustering[1].

However, the classical parameterized complexity approach is still very limiting, as finding exact solutions often turns out to be hard even in the multivariate setting. This is highlighted by the results on the parameter D mentioned above, that are often negative even for such a humble parameterization. On the other hand, for the continuous kind of problems like clustering it is even less principal to have an exact solution, as opposed to a very good approximate one. Note that some discretization and approximation necessary occurs even by representing real-valued data on a fixed word-size machine. Thus for the next part of the thesis we move to FPT *approximation schemes*, that is, $(1 + \varepsilon)$-approximation algorithms with running time $f(k, \varepsilon)n^c$ for some function f of the parameter k and ε, and some constant c. FPT approximation schemes combine the power of approximation and parameterization, and a vast body of results of this form has been already formed for clustering and low-rank approximation problems. Techniques such as dimensionality reduction and sampling in order to obtain good approximate centers or coresets, are closely tied to this area

[1] Provided that the coordinates of the points are integral, otherwise the magnitude of D plays no role because of scaling.

of research, as they allow to reduce the search space to something that depends only on the number of centers k and ε. Moreover, both dimensionality reduction and coresets are also of independent interest, as they lead to more efficient data processing in a variety of settings, be it streaming or parallel computations, and for variety of purposes, from theoretical algorithms with provable guarantees to practical routines that reduce storage or simplify datasets. We extend the outreach of this approaches by showing small-sized coresets for a variety of clustering problems, where clusters have to satisfy certain size constraints with respect to a predetermined coloring of the points. In particular, this holds for (α, β)-Fair Clustering, where the goal is to compute optimal clustering of the given points subject to a fairness condition, i.e. each of the given protected groups must not be underrepresented in any of the resulting clusters. Studying the (α, β)-Fair Clustering problem is also very much in line with the general goal of increasing accountability of machine learning primitives, as it has been consistently shown that bluntly optimizing the performance of a decision-making system can make it biased or discriminatory towards traditionally underrepresented groups [13, 73, 98].

Going back to designing FPT approximation schemes, one of the most renowned examples is the algorithm of Kumar, Sabharwal, and Sen [137] for the already mentioned k-Means. This algorithm provides $(1+\varepsilon)$ approximation in time $2^{(k/\varepsilon)^{\mathcal{O}(1)}} n^{\mathcal{O}(1)}$, and is crucially based on a sampling lemma ensuring that a uniform sample of size $\mathcal{O}(1/\varepsilon)$ from a cluster approximates its optimal center well, with good probability. This lemma is very powerful tool for tackling k-Means, yet also very specific for this objective. Take, however, a similar problem where the objects to cluster are not points in $\mathbb{R}^d$, but axis-parallel subspaces of small dimension. Or, equivalently, points in $\mathbb{R}^d$ with few missing entries, that could be completed arbitrarily. This corresponds to a very natural problem of clustering a dataset where some entries are not known: imagine a movie rating database where not every user has a rating for every movie. The approach of [137] is not directly applicable in this case, as neither the sampling lemma holds, nor even the triangle inequality for distances between such objects, making the problem geometrically very different from k-Means. In fact, for years a comparable algorithm for this problem eluded researchers. We take a step forward and present an FPT approximation scheme for clustering of points with few missing entries, parameterized by both the number of clusters k and the maximum number Δ of missing entries per point.

For the final example, consider the problem of *low-rank approximation*, where the task is to construct a matrix of given rank that approximates the given matrix best. Reducing in this fashion the rank of the dataset, also known as principal component analysis (PCA), is a ubiquitous routine for compressing and simplifying data given in matrix form. Computationally, it is not as challenging as the previous examples, since there exists a polynomial-time algorithm for computing the optimal low-rank approximation, that proceeds via singular value decomposition (SVD). However, one weakness of PCA is that it is highly susceptible to outliers: even one corrupted row

can arbitrarily change the approximation subspace. Motivated by this, we introduce the PCA with Outliers problem, where the task it to remove a given number of outliers in such a way that allows for the best possible low-rank approximation for the remaining matrix. In contrast to the usual approaches that deal with outliers by, for instance, placing strong assumptions on the distribution of outliers, an algorithm for PCA with Outliers always finds the best possible set of outliers in terms of the cost. This makes our formulation very flexible with respect to arbitrary models of what is considered to be an outlier, and in particular resistant to any adversarially placed outliers. However, the price for such flexibility is the increased computational complexity of the problem, as we show that there cannot exist an $n^{o(d)}$-time algorithm solving PCA with Outliers, where n is the number of rows and d the number of columns in the matrix, under a suitable complexity assumption. In fact, even designing an algorithm matching this lower bound is not trivial, and we employ involved results from computational algebraic geometry in order to design an $n^{\mathcal{O}(rd)}$ algorithm for PCA with Outliers, where r is the target rank. Together with some additional results on dimensionality reduction, and an approximation scheme for binary low-rank approximation, this comprises the final technical part of this thesis. It is worthy to note that these results follow the general approach of studying *robustness* of data-driven algorithms, that is, how robust is the algorithm in question when run on ill-generated or adversarial data. The study of robustness can be seen as yet another step towards algorithmic accountability, as the highly desired outcome is that an algorithm in charge of making decisions could not be failed by accident or fooled by purpose with a small portion of unexpected data.

All in all, we believe that the main contribution of this thesis is providing novel algorithms with provable guarantees for clustering and low-rank approximation problems parameterized by one or several structural characteristics of the input, and the corresponding conditional lower bounds. Moreover, we emphasize on conjunction with other means of achieving accountability of algorithms, such as fairness and robustness. Next, we outline the structure of the thesis.

Outline of the Thesis. In Part I, we start with introducing fundamental concepts and previous work relevant for our technical results. In Chapter 2 we define common preliminaries and notation for the following parts, and introduce some standard background in algorithmic complexity. Then in Chapter 3 we introduce formally our main problems of interest, and in Chapter 4 we give basics on the key tools from the literature that we rely on. In both Chapter 3 and Chapter 4 we give also a brief survey of the relevant previous works. The rest of the thesis is divided into three main parts.

In Part II, we consider exact parameterized complexity of clustering problems parameterized by the cost. Intuitively, this is a "structural editing" point of view on the problem, where we assume that the instance is already close to be clustered, and thus the cost of clustering is reasonably small. In Chapter 5, we consider a generalization of k-Means to L_p-distances, and draw a wide complexity landscape with

respect to the cost parameterization for various values of p. This chapter is based on the work [FGS21]. We move to robust clustering of categorical data in Chapter 6, that is, datapoints with entries from a fixed-size set equipped with the Hamming distance. We show an interesting dichotomy between the following two robust versions: removing column outliers, corresponding to corrupted datapoints, and row outliers, corresponding to corrupted features, or dimensions. The algorithm for the latter extends to robust variants of various other problems, such as binary low-rank approximation, and these results are featured in [BFGS21].

We focus on FPT approximation schemes in Part III. In Chapter 7, based on the work [BFS20], we present an efficient coreset construction for clustering problems with color-sized constraints, and a variety of novel algorithms based on that coreset, in particular for (α, β)-Fair Clustering. In Chapter 8, we show the approximation scheme for clustering of points with missing entries, this result is published as [EFG+21].

In Part IV, we deal with low-rank approximation problems. The main focus is PCA with Outliers and the algebraic geometry approach, that we first introduce in Chapter 9, together with the $n^{\mathcal{O}(rd)}$-time algorithm and the $n^{o(d)}$ lower bound. Then we proceed to improve this approach by designing specific dimensionality reduction methods in Chapter 10 that allow to reduce the running time for sublinear number of outliers, and we also provide a general polynomial-time approximation scheme when the target rank is fixed. These two chapters are based on the articles [SFGP19, DFPS21]. In Chapter 11, we make one further step in the study of the low-rank approximation problem, and show the first polynomial-time approximation scheme for binary matrices equipped with GF(2)-rank and column-sum norm. This setting is radically different from the usual Euclidean space and rank, and thus requires developing a very special pipeline. This result is published in [FGPS20].

Finally, we conclude with future research directions in Part V. Next we give the list of publications to which the author of this thesis has contributed. Out of these, the publications [FGS21, FGPS20, EFG+21, SFGP19] and the manuscripts [BFGS21, BFS20, DFPS21] serve as the basis for the main body of this work.

List of publications

[BFGS21] Sayan Bandyapadhyay, Fedor V. Fomin, Petr A. Golovach, and Kirill Simonov. Parameterized complexity of robust categorical data clustering. Unpublished manuscript, 2021.

[BFS20] Sayan Bandyapadhyay, Fedor V. Fomin, and Kirill Simonov. On coresets for fair clustering in metric and Euclidean spaces and their applications. Preprint, arXiv: 2007.10137, 2020.

[DFPS21] Yogesh Dahiya, Fedor V. Fomin, Fahad Panolan, and Kirill Simonov. Multivariate complexity of pca with outliers. Unpublished manuscript, 2021.

[EFG+21] Eduard Eiben, Fedor V. Fomin, Petr A. Golovach, William Lochet, Fahad Panolan, and Kirill Simonov. EPTAS for k-means clustering of affine subspaces. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*. To appear, 2021.

[EFPS20] Eduard Eiben, Fedor V. Fomin, Fahad Panolan, and Kirill Simonov. Manipulating districts to win elections: Fine-grained complexity. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1902–1909. AAAI Press, 2020.

[FGPS20] Fedor V. Fomin, Petr A. Golovach, Fahad Panolan, and Kirill Simonov. Low-Rank Binary Matrix Approximation in Column-Sum Norm. In Jarosław Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, volume 176 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 32:1–32:18, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[FGS21] Fedor V. Fomin, Petr A. Golovach, and Kirill Simonov. Parameterized k-clustering: Tractability island. *Journal of Computer and System Sciences*, 117:50 – 74, 2021.

[FSS20] Fedor V. Fomin, Danil Sagunov, and Kirill Simonov. Building Large k-Cores from Sparse Graphs. In Javier Esparza and Daniel Kráľ, editors, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, volume 170 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[SFGP19] Kirill Simonov, Fedor Fomin, Petr Golovach, and Fahad Panolan. Refined complexity of PCA with outliers. volume 97 of *Proceedings of Machine Learning Research*, pages 5818–5826, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Part I

Basic Concepts

2

Preliminaries

In this chapter we fix some basic notation and terminology, concerning numbers and sets, linear algebra, graphs, and probability. We also recall the fundamental concepts of several aspects of algorithmic complexity.

2.1 Notation

Numbers and sets. For a set S, we denote the size of S by $|S|$, and the set of all subsets of S by 2^S (called also the power set of S). For an integer $k \leq |S|$, we denote by $\binom{S}{k}$ the set of all k-sized subsets of S. For two sets A and B we use $A \setminus B$ and $A - B$ interchangeably to denote the set difference of A and B, e.g. the set of elements of A that are not present in B.

For any integer $n > 0$, we denote be $[n]$ the set of integers between 1 and n, that is, $\{1, \ldots, n\}$. We denote the set of real numbers by $\mathbb{R}$, the set of integers by $\mathbb{Z}$, and the set of natural numbers by $\mathbb{N}$. By $\mathbb{R}_{\geq 0}$, we denote the set of nonnegative real numbers, and by $\mathbb{R}_{>0}$ the set of positive real numbers. Respectively, $\mathbb{Z}_{\geq 0}$ denotes the set of nonnegative integers, that is, $\mathbb{Z}_{\geq 0} = \mathbb{N} \cup \{0\}$. For non-negative real numbers a and b, we use the notation $a = (1 \pm \varepsilon)b$ if $a \in [(1-\varepsilon)b, (1+\varepsilon)b]$.

Linear algebra. Throughout the thesis, we use extensively matrix notations. We use bold lowercase, e.g. $\mathbf{x} = (\mathbf{x}[i])$, to denote a vector and bold uppercase, e.g. $\mathbf{A} = (\mathbf{A}_{ij})$, to denote a matrix. For an element $\mathbf{x}$ of d-dimensional space Σ^d, we denote its coordinates by $\mathbf{x} = (\mathbf{x}[1], \ldots, \mathbf{x}[d])$, where Σ is the underlying field or alphabet. For a matrix $\mathbf{A}$, we conventionally denote its i-th row by $\mathbf{a}_i$, and the element at the intersection of the i-th row and the j-th column by $\mathbf{A}_{ij}$. By $\mathbf{A}^\top$, we denote the transpose of a matrix $\mathbf{A}$. We denote the identity matrix of size $n \times n$ by $\mathbf{I}_n$, and the zero vector by $\mathbf{0}$. For a matrix $\mathbf{A} \in \Sigma^{n \times d_1}$ and a matrix $\mathbf{B} \in \Sigma^{n \times d_2}$, $[\mathbf{A}|\mathbf{B}] \in \Sigma^{n \times (d_1 + d_2)}$ denotes the column-wise matrix concatenation of $\mathbf{A}$ and $\mathbf{B}$. For

a matrix $\mathbf{A} \in \Sigma^{n\times d}$ and a set of row indices $I \subset [n]$, let $\mathbf{A}[I :]$ denote the matrix composed by selecting the rows of $\mathbf{A}$ corresponding to the indices in I.

For a vector $\mathbf{x} \in \mathbb{R}^d$, $\|\mathbf{x}\|_2$ denotes the Euclidean norm of $\mathbf{x}$, that is,

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^{d} x[i]}.$$

Respectively, for two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $\|\mathbf{x} - \mathbf{y}\|_2$ denotes the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$. For a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, its Frobenius norm is given by

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{d}(\mathbf{A}_{ij})^2}.$$

For two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, we denote their dot product by

$$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{d} \mathbf{x}[i] \cdot \mathbf{y}[i].$$

For a vector $\mathbf{x} \in \mathbb{R}^d$ and an r-dimensional linear subspace $\mathcal{U}$ of $\mathbb{R}^d$, where the basis of $\mathcal{U}$ is given by the rows of $\mathbf{U} \in R^{r\times d}$, we denote by $\text{proj}_{\mathcal{U}}\, \mathbf{x}$ the orthogonal projection of $\mathbf{x}$ on $\mathcal{U}$, that is, the vector

$$\text{proj}_{\mathcal{U}}\, \mathbf{x} = \sum_{i=1}^{r} (\mathbf{x} \cdot \mathbf{u}_i)\mathbf{u}_i,$$

where $\mathbf{u}_1, \ldots, \mathbf{u}_r$ are the rows of $\mathbf{U}$, assuming that the rows of $\mathbf{U}$ are pairwise orthogonal and of length one. For a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ and an r-dimensional linear subspace $\mathcal{U}$ of $\mathbb{R}^d$, given by $\mathbf{U} \in R^{r\times d}$, let $\text{dist}^2(\mathbf{A}, \mathbf{U})$ (and $\text{dist}^2(\mathbf{A}, \mathcal{U})$) represent the sum of squares of the Euclidean distances of the rows of $\mathbf{A}$ from $\mathcal{U}$, which is equal to $\min_{\mathbf{X}\in R^{n\times r}} \|\mathbf{A} - \mathbf{X}\mathbf{U}\|_F^2$. For $\mathbf{A} \in \mathbb{R}^{n\times d}$, the column and row spaces of a matrix are denoted by $\text{col}(\mathbf{A})$ and $\text{row}(\mathbf{A})$ respectively.

Graphs. In this work, we use standard graph notations. For a graph G, $V(G)$ denotes its vertex set, and $E(G) \subset \binom{V(G)}{2}$ denotes its edge set. We often associate $V(G)$ with the set of numbers $[V(G)]$, and use the name of a vertex interchangeably with its number. Unless stated otherwise, a graph is finite, undirected and simple. For a vertex set $X \subset V(G)$, $G[X]$ is the subgraph of G induced by X. For a vertex $v \in V(G)$, we denote its degree by $\deg_G(v)$, its open neighborhood by $N_G(v)$, and its closed neighborhood by $N_G[v]$.

Probability. For an event A, we denote its probability by $\Pr[A]$. We denote random variables by uppercase letters, e.g. X. For a random variable X, we denote its expectation by $\mathrm{E}[X]$. Throughout the thesis, we use versions of Chernoff bound for probability estimates. Unless stated otherwise, by "Chernoff bound" we mean the following statement.

Proposition 2.1 ([12]). *Let $X_1, X_2, \ldots, X_n$ be independent 0-1 random variables. Denote $X = \sum_{i=1}^{n} X_i$ and $\mu = E[X]$. Then for $0 < \beta \leq 1$,*

$$P[X \leq (1-\beta)\mu] \leq \exp(-\beta^2\mu/2),$$
$$P[X \geq (1+\beta)\mu] \leq \exp(-\beta^2\mu/3).$$

Real computations. Since we deal with the problem concerning real-valued matrices, we express the running time of algorithms in terms of number of operations over the reals. However, the results for exact algorithms over integers or categorical data, and approximation algorithms, hold in the bit complexity model as well.

2.2 Complexity

In this section, we recall the necessary complexity background, discussing parameterized complexity and hardness assumptions commonly used to show lower bounds, such as Exponential Time Hypothesis, as well as some basic definitions regarding approximation algorithms.

2.2.1 Parameterized Complexity

Consider an NP-hard combinatorial problem, such as Vertex Cover, where the input is a graph G and an integer k, and the task is to decide whether there is a subset C of vertices such that each edge in G has at least one endpoint in C. Unless $\mathsf{P} = \mathsf{NP}$, there is generally no polynomial-time algorithm for Vertex Cover; however, when k is bounded by a constant, a trivial algorithm that in time $\mathcal{O}(|V(G)|^k|E(G)|)$ tries all possible k-sized subsets C of $V(G)$ and checks whether each subset is a vertex cover, runs in polynomial time. There is another, smarter, algorithm that starts with an empty C, and recursively picks an arbitrary edge e that is yet uncovered by C. Then the algorithm tries to put each of its endpoints into C, branching into two new subproblems and processing each of them recursively. When the size of C is already k, the algorithm checks whether there is any other uncovered edge, and if not, returns C as the vertex cover. Clearly, the algorithm follows a recursive binary tree of depth k, thus its running time is bounded by $\mathcal{O}(2^k|E(G)|)$. Now compare the two algorithms, from the classical complexity point of view, they have the same properties: both are not polynomial-time in the general case, and when k is constant, the running time of both becomes polynomial. However, the running times of these two algorithms grow

very differently as k and $|V(G)|$ increase: while the first one would be completely unfeasible on a moderately large graph even for very small values of k, the second could easily find a vertex cover of size a couple of dozens in a graph with thousand vertices. Moreover, in the running time of the second algorithm, k and $|V(G)|$ are in a sense separated: as k grows, the dependence on the size of the graph is always the *same* polynomial. In short, the primary objective of parameterized complexity is to distinguish problems that allow these two kinds of algorithms, allowing for a more fine-grained study of the complexity of an NP-complete problem in the regime where a certain parameter of the input is small. Now we give basics of the field formally, while for the detailed introduction to parameterized complexity, we refer to the books [70, 78].

A *parameterized problem* is a language $Q \subseteq \Sigma^* \times \mathbb{N}$ where Σ^* is the set of strings over a finite alphabet Σ. Respectively, an input of Q is a pair (I, k) where $I \subseteq \Sigma^*$ and $k \in \mathbb{N}$; k is the *parameter* of the problem. A parameterized problem Q is *fixed-parameter tractable* (FPT) if it can be decided whether $(I, k) \in Q$ in time $f(k) \cdot |I|^{\mathcal{O}(1)}$ for some function f that depends of the parameter k only. Respectively, the parameterized complexity class FPT is composed of fixed-parameter tractable problems. On the other hand, the class XP contains problems solvable in time $|I|^{g(k)}$, for some function g of k. Going back to the Vertex Cover example, the trivial enumeration algorithm with running time $\mathcal{O}(|V(G)|^k|E(G)|)$ is thus called an XP algorithm, while the recursive $\mathcal{O}(2^k|E(G)|)$-time algorithm is an FPT algorithm. One of the main goals of the field of parameterized complexity is to identify parameterized problems that allow FPT algorithms.

Naturally, the theory of parameterized complexity admits also a way to conclude the parameterized hardness of a problem. Similarly to the polynomial hierarchy, a collection of parameterized complexity classes is defined, called the W-hierarchy. We omit the formal definitions here, as they are rather technical, and refer to the book [70] instead. The following relation is known amongst the classes in the W-hierarchy: $\mathsf{FPT} = \mathsf{W}[0] \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \ldots \subseteq \mathsf{W}[P]$. It is widely believed that $\mathsf{FPT} \neq \mathsf{W}[1]$, playing a role as the working hypothesis of parameterized complexity, similarly to $\mathsf{P} \neq \mathsf{NP}$. Hence if a problem is hard for the class $\mathsf{W}[i]$ (for any $i \geq 1$) then it is considered to be fixed-parameter intractable, i.e. it is unlikely that the problem admits an FPT algorithm. For the purpose of this work, it suffices to know that Clique is W[1]-hard. Here Clique is the problem of finding a k-clique in the graph: given a graph G, and an integer k, the task is to determine whether G has a clique of size k as a subgraph. The hardness of other problems may be concluded via a *parameterized reduction* from a problem known to be $\mathsf{W}[i]$-hard. Parameterized reductions provide an analogue of polynomial-time reductions which also do not blow up the parameter by too much.

Definition 2.2 (Parameterized Reduction [70])**.** *A* parameterized reduction *from a parameterized problem Q to a parameterized problem P is an algorithm that transforms each instance (I, k) of Q into an instance (I', k') of P such that*

(i) (I, k) *is a yes-instance of* Q *if and only if* (I', k') *is a yes-instance of* P*,*

(ii) $k' \leq g(k)$ *for some computable function* g *of* k*, and*

(iii) the running time of the algorihtm is $f(k) \cdot |I|^{\mathcal{O}(1)}$ *for some computable function* f *of* k*.*

We observe that the definition of a parameterized reduction above works indeed as intended. First, if there is a parameterized reduction from Q to P, and P admits an FPT algorithm, then Q also admits an FPT algorithm. To see this, given an instance (I, k) of Q, run the reduction algorithm to compute an equivalent instance (I', k') of P, then run the FPT algorithm of P on (I', k') and return its output. The property (i) of a parameterized reduction ensures that the algorithm above correctly identifies yes- and no-instances of Q. The property (iii) ensures that the reduction runs in FPT time, and the property (ii) ensures that the new parameter k' is bounded by a function of k, thus the FPT algorithm for P runs in FPT time in terms of k as well.

On the other hand, if there is a parameterized reduction from Q to P, and Q is W[i]-hard, then P is also W[i]-hard; in fact, the classes of the W-hierarchy are defined as closed under parameterized reductions. All in all, this allows us to transfer hardness from a problem known to be W[i]-hard, like Clique. This is the primary method of deriving parameterized hardness, and we employ it in this work extensively as well.

Finally we note that all definitions above may be easily extended to the case of several parameters, by introducing a single parameter that is the sum of the parameters of interest. Thus, we say that a problem is FPT/XP/W[i]-hard parameterized by x and y if it is FPT/XP/W[i]-hard parameterized by $x + y$.

2.2.2 Approximation Algorithms

Another way to deal with NP-hardness of a problem lies in designing approximation algorithms. Intuitively, while a classical exact algorithm has to correctly determine whether an input is a yes- or no-instance of a decision problem, an approximation algorithm is allowed some slack in the objective function. In this section we introduce the standard terminology for approximation algorithms that we use extensively in the text. For the in-depth introduction to the area of approximation algorithms, we refer to the classical book by Vazirani [184], and the more recent textbook by Williamson and Shmoys [187].

Formally, for a minimization problem Q and a value α, an *α-approximation algorithm* outputs a solution that has an objective value of at most $\alpha \cdot \mathsf{OPT}$, where OPT is the objective value of the optimal solution. Symmetrically, for a maximization problem an α-approximation algorithm outputs a solution with the objective value at least $\alpha \cdot \mathsf{OPT}$. One important direction of research is identifying polynomial-time

constant-factor approximation for various NP-hard problems. The complexity class that contains problems admitting such algorithms is called APX.

For some problems in APX, it is possible to design polynomial-time algorithms with approximation factor arbitrarily close to one. Formally, for a minimization problem Q, a *polynomial-time approximation scheme (PTAS)* is a family of polynomial-time algorithms $\{\mathcal{A}_\varepsilon\}$ such that for each $\varepsilon > 0$ there is an algorithm $\mathcal{A}_\varepsilon$ that is a $(1+\varepsilon)$-approximation algorithm for Q. We measure the running time of a PTAS as a function of n and ε, and for each $\varepsilon > 0$ we put the running time of the respective algorithm $\mathcal{A}_\varepsilon$. Note that the running time of a PTAS may have an arbitrary dependence on ε, i.e a PTAS in general has a running time of $|I|^{g(\varepsilon)}$ where g is some function of ε, similarly to XP algorithms in parameterized complexity. Just as in the case of the FPT–XP dichotomy, one can define an efficient version of PTAS. An *efficient polynomial-time approximation scheme (EPTAS)* is a PTAS that has the running time of $f(\varepsilon)|I|^{\mathcal{O}(1)}$ for some function f of ε. In other words, in the running time of an EPTAS, ε influences only the constant factor, but not the degree of the polynomial.

An equivalent of NP-hardness or W[1]-hardness for approximation problems is APX-hardness. An optimization problem is APX-hard if there is a PTAS reduction from every problem in APX to that problem. Here, a PTAS reduction is, intuitively, a natural object preserving the approximation factor. A PTAS reduction from Q to P consists of three maps f, g, and α, such that f transforms an instance I of A to an instance of B, and g lifts the solution from B back to A, i.e. if a solution S to $f(I)$ is $(1+\alpha(\varepsilon))$-approximate, then $g(I, S, \varepsilon)$ is an $(1+\varepsilon)$-approximate solution to I. Under the $\mathsf{P} \neq \mathsf{NP}$ assumption, any APX-hard problem does not admit a PTAS.

The ideas of parameterized complexity and approximation algorithms can be applied simultaneously: we can measure the running time of a $(1+\varepsilon)$-approximate algorithm both in terms of the error parameter ε and a structural parameter k, defined for the particular instance. For the purpose of this work, we denote by an FPT *approximation scheme (FPT-AS)* a PTAS that has the running time of $f(k,\varepsilon)n^{\mathcal{O}(1)}$. Recently, this kind of results for clustering problems became very common. To name a few, Kumar, Sabharwal, and Sen [137] gave an $(1+\varepsilon)$-approximation algorithm for Euclidean k-median and k-means that runs in time $2^{(k/\varepsilon)^{\mathcal{O}(1)}}nd$. Fomin et al. [88] designed an FPT approximation scheme for a variety of low-rank approximation and clustering problems on binary matrices. Another notable approach is to employ FPT time to design approximation algorithms with improved approximation factors, compared to the best known polynomial-time approximation. For instance, Cohen-Addad et al. [64] showed a $(1+2/e+\varepsilon)$-factor approximation algorithm for k-median clustering in general metric spaces that runs in time $f(k,\varepsilon)n^{\mathcal{O}(1)}$ for some function f. We give a more detailed overview of the relevant literature in Chapter 3. Part III of this work is dedicated to FPT-time approximation for variants of clustering problems.

2.2.3 Exponential Time Hypothesis

One source of "coarseness" in the theory of parameterized complexity is the fact that technically the dependence on the parameter k can be *any* computable function, be it a slow-growing quasipolynomial or a very fast-growing tower of exponents like $2^{2^{2^k}}$. A way to obtain more precise lower bounds is to replace the working hypothesis $\mathsf{FPT} \neq \mathsf{W}[1]$ by a stronger one, and in this section we discuss the widely-believed Exponential Time Hypothesis (ETH), along with its stronger variants.

First, recall that in the classical CNF-SAT problem, given a Boolean formula in conjunctive normal form (CNF), the task is to determine whether the formula has any satisfying assignment. For a positive integer q, the q-SAT problem is the restriction of CNF-SAT to CNF-formulas where clauses contain at most q literals. By the famous Cook–Levin theorem [67] from 1971, CNF-SAT, and even the special case 3-SAT, is NP-hard. In 2001, Impagliazzo, Paturi, and Zane [117] introduced two new conjectures about the complexity of q-SAT, formalizing a barrier to devising better algorithms for q-SAT that researchers were unable to bypass for so many years since the dawn of the complexity theory. We state these conjectures next; for an in-depth introduction to the topic we refer to Chapter 14 of [70].

Conjecture 2.3 (Exponential Time Hypothesis (ETH) [117])**.** *There is a positive real number ε such that* 3-SAT *with n variables and m clauses cannot be solved in time $2^{\varepsilon n}(n+m)^{\mathcal{O}(1)}$.*

Conjecture 2.4 (Strongly Exponential Time Hypothesis (SETH) [117])**.** *For every positive real number ε, there is an integer $q = \mathcal{O}(1)$ such that q-*SAT *with n variables and m clauses cannot be solved in time $(2-\varepsilon)^n(n+m)^{\mathcal{O}(1)}$.*

We note that SETH is strictly stronger than ETH, for the proof we refer to [70]. Out of the two, ETH bears the crucial importance for this work and parameterized complexity in general. The main usage for us is through the following theorem due to Chen et al. [51].

Theorem 2.5. *Assuming ETH, there is no $f(k)n^{o(k)}$-time algorithm for* Clique *for any computable function f of k.*

By Theorem 2.5, any parameterized reduction from Clique to another parameterized problem not only shows that the latter is W[1]-hard. Such a reduction also provides a more precise algorithmic lower bound assuming ETH, that there is no $f(k)n^{o(g^{-1}(k))}$-time algorithm for the target problem, where g is the function from the property (ii) in Definition 2.2. In particular, if the reduction transforms the parameter k into $\mathcal{O}(k)$, then the bound of Theorem 2.5 holds for the target problem as well, i.e. there is no $f(k)n^{o(k)}$-time algorithm. In this way, we state most of the algorithmic lower bounds in this work as ETH-lower bounds.

For the design of lower bounds for approximation algorithms, the following strengthening of ETH is often used, due to [76]. Intuitively, this variant conjectures that it

is not only hard to solve 3-SAT in subexponential time, but hard even to determine whether nearly all clauses can be satisfied simultaneously.

Conjecture 2.6 (Gap-ETH [76])**.** *There are positive real numbers ε and D such that any algorithm that, given a* 3-SAT *formula Φ with n variables and at most Dn clauses, can distinguish between the events "there is an assignment satisfying all clauses of Φ" and "no assignment can satisfy a $9/10$ fraction of the clauses of Φ", must run in time at least time $2^{\varepsilon n}$.*

Such an extension of ETH creates a gap that can be transferred to the target problem by a reduction, showing that it is hard to approximate the objective function in subexponential time.

3

Problem Models

Here we define our main problems of interest, clustering and low-rank approximation, together with their variants that we consider in this work, and give an overview of the current state of research concerning these problems.

3.1 Clustering

In this section, we discuss basic definitions and results concerning clustering problems. In its most general version, the unrestricted k-Clustering problem is defined as follows. We are given a set of points P in a space $\mathcal{X}$ equipped with a distance measure $\text{dist} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$, where dist is not necessarily a metric. Additionally, we are given a description of a set $F \subset \mathcal{X}$ of possible cluster centers, and an integer k. The goal is to partition the set P into k clusters $C_1, \ldots, C_k$ and select k centers $c_1, \ldots, c_k$ from F such that

$$\sum_{i=1}^{k} \sum_{p \in C_i} \text{dist}(p, c_i)$$

is minimized. Observe that it suffices to pick only the set of centers $C = \{c_1, \ldots, c_k\}$, since to minimize the quantity above, it is always better to assign a point to the closest center. Then the objective becomes to minimize

$$\sum_{p \in P} \text{dist}(p, C),$$

where we use the notation $\text{dist}(p, C)$ for $\min_{c \in C} \text{dist}(p, c)$.

A multitude of settings for k-Clustering is studied in the literature. In this work, our primary interest lies in the following cases.

- In the discrete metric k-Median, $(\mathcal{X}, \text{dist})$ is a finite metric space and both sets P and F are given in the input. We assume that dist is given as an oracle

that in constant time returns $\mathrm{dist}(x, y)$ for any $x, y \in \mathcal{X}$. Recall that a function $\mathrm{dist} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a metric when it satisfies the following three properties hold for any $x, y \in \mathcal{X}$:

(i) $\mathrm{dist}(x, y) = 0$ if and only if $x = y$,

(ii) $\mathrm{dist}(x, y) = \mathrm{dist}(y, x)$,

(iii) triangle inequality, that is, for any point $z \in \mathcal{X}$

$$\mathrm{dist}(x, y) \leq \mathrm{dist}(x, z) + \mathrm{dist}(z, y).$$

The discrete metric k-Means is defined analogously, except that the objective to minimize is

$$\sum_{p \in P} \mathrm{dist}^2(p, C),$$

that is, the sum of squared distances from each point to the closest center.

· In the Euclidean k-Median problem, $\mathcal{X}$ is the d-dimensional Euclidean space $\mathbb{R}^d$, dist is the Euclidean distance, also denoted by $\|\mathbf{x} - \mathbf{y}\|_2$, and F is the whole space $\mathbb{R}^d$. Recall that

$$\|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{\sum_{i=1}^{d} (\mathbf{x}[i] - \mathbf{y}[i])^2}.$$

The Euclidean k-Means problem is defined analogously, only the objective to minimize is

$$\sum_{\mathbf{p} \in P} \min_{\mathbf{c} \in C} \|\mathbf{p} - \mathbf{c}\|_2^2,$$

In other words, in the case of k-Means, $\mathrm{dist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2$.

· We consider also a generalization of the Euclidean k-Means to L_p distances that we call L_p-k-Clustering. Here, the space $\mathcal{X}$ is still $\mathbb{R}^d$, but dist is defined by the p-th power of the Minkowski L_p-norm, that is

$$\mathrm{dist}(\mathbf{x}, \mathbf{y}) = \mathrm{dist}_p(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_p^p = \sum_{i=1}^{d} |\mathbf{x}[i] - \mathbf{y}[i]|^p.$$

Clearly, for $p = 2$ we get k-Means, and for $p = 1$ we get k-Median with the rectilinear (Manhattan) distance. The definition above holds for $p \in (0, \infty)$, we also use the natural definitions in the extreme cases,

$$\mathrm{dist}_0(\mathbf{x}, \mathbf{y}) = d_H(\mathbf{x}, \mathbf{y}) = |\{i \in \{1, \ldots, d\} \mid \mathbf{x}[i] \neq \mathbf{y}[i]\}|$$

Figure 3.1: Optimal clusterings of the same set of vectors with different distances: dist_1 in the left subfigure, $\text{dist}_{1/4}$ in the right subfigure. Shapes denote clusters, crosses denote cluster centers.

for $p = 0$, also called the Hamming distance, and

$$\text{dist}_\infty(\mathbf{x}, \mathbf{y}) = \max_{i \in \{1,\dots,d\}} |\mathbf{x}[i] - \mathbf{y}[i]|$$

for $p = \infty$. We define the L_0-k-Clustering and L_∞-k-Clustering problems as k-Clustering on $\mathbb{R}^d$ with dist_0 and dist_∞, respectively. In Figure 3.1, we present an example of optimal clusterings with respect to different values of p. Note that increasing p penalizes less the number of coordinates where points differ, and more the absolute value of the difference. Smaller values of p, on the other hand, are less susceptible to distortion from outlier points, especially when the "noisy" part is restricted to few columns, leading to a more robust clustering. Thus, the choice of p allows to pick the right trade-off of this form, motivating the study of L_p-k-Clustering.

- We consider also a variation of L_0-k-Clustering that we call Categorical k-Clustering, where the metric space is the set of strings of length d over alphabet Σ, equipped with the Hamming distance. The problem is trivially equivalent to L_0-k-Clustering when Σ is of size at least n, as the Hamming distance cares only about whether values are equal or distinct for each coordinate. Viewing points as strings is however very natural with respect to d_H, and specially interesting is the case where Σ is small, for example binary. We refer to the binary version of Categorical k-Clustering by the name Binary k-Clustering.

  The name Categorical k-Clustering stems from statistics, where a categorical variable is a variable that can admit a fixed number of possible values, as well as the motivation to study this problem. A categorical variable could be gender, blood type, political orientation, etc. A prominent example of categorical data is binary data where the points are vectors each of whose coordinates can take value either 0 or 1. Binary data arise in several important applications. For example, in electronic commerce, each transaction can be modeled as

a binary vector (known as market basket data), where each of the coordinates denotes whether a particular item is purchased or not [193, 146].

· Another generalization of k-Means is the following problem of clustering points with missing entries. Instead of $\mathbb{R}^d$ consider the space $\mathbb{H}^d$, the set of d-dimensional vectors that in each coordinate take either an arbitrary real value or the special value "?". The only difference to the normal k-Means with Euclidean distance is that values "?" do not contribute to the distance, i.e. for any $\mathbf{x}, \mathbf{y} \in \mathbb{H}^d$,

$$\operatorname{dist}(\mathbf{x}, \mathbf{y}) = \sum_{\substack{i \in [d] \\ \mathbf{x}[i] \neq ?, \mathbf{y}[i] \neq ?}} (\mathbf{x}[i] - \mathbf{y}[i])^2.$$

This problem models the situation of a dataset with incomplete or corrupted entries, commonly occurring in practice. Imagine a movie rating database, where each datapoint corresponds to a user and each coordinate to a movie; clearly it is infeasible that the rating for each user-movie pair is known, however the known ratings might still contain a lot of useful information. A number of methods in various domains have been proposed to deal with missing data. We refer to the books [8], [185], and the Wikipedia entry[1] for an introduction to the topic.

The k-Clustering problem defined above covers most of the variants of k-median and k-means studied in the literature. Another problem that is traditionally receiving a lot of attention is the k-center problem, where the minimization objective is

$$\max_{p \in P} \operatorname{dist}(p, C),$$

over all possible sets of k centers C. However, the results presented in this thesis are falling under the k-Clustering formulation above, so we mention k-center problems primarily as a point of reference when discussing relevant work.

3.1.1 Cluster Notations

Here we fix some common notations related to clustering problems. Throughout the text, we use n to denote the size of the given point set P, while in the discrete metric case we denote $n = |P \cup F|$. By a *cluster* we always mean a subset of the input points P. Note that in vector spaces such as $\mathbb{R}^d$, the input set might contain repeated vectors, i.e. P might be a multiset in $\mathbb{R}^d$. The *cost* of a given cluster C with respect to a center $c \in F$ is the total distance from all vectors in the cluster to c,

$$\operatorname{cost}(C, c) = \sum_{x \in C} \operatorname{dist}(x, c).$$

[1] https://en.wikipedia.org/wiki/Missing_data

An *optimal* cluster center for a given cluster C is any $c \in F$ minimizing $\sum_{x \in C} \mathrm{dist}(x, c)$, we also use $\mathrm{cost}(C)$ to denote the cost of the cluster C with respect to an optimal cluster center for C, i.e.

$$\mathrm{cost}(C) = \min_{c \in F} \sum_{x \in C} \mathrm{dist}(x, c).$$

For a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ of X, we say that the cost of the clustering is the sum of the costs of each cluster,

$$\mathrm{cost}(\mathcal{C}) = \sum_{i=1}^{k} \mathrm{cost}(C_i).$$

For a number of spaces that we consider, the optimal cluster center has a natural characterization. Notably, for Euclidean k-Means, the optimal cluster center is the coordinate-wise mean of the cluster points, called also the center of mass. Moreover, the following well-known observation holds.

Proposition 3.1 (Claim 3.2, [137])**.** *For the Euclidean k-Means problem, given a cluster $C \subset P$ and a vector $\mathbf{c} \in \mathbb{R}^d$ it holds that*

$$\mathrm{cost}(C, \mathbf{c}) = \mathrm{cost}(C, \mu(C)) + |C| \cdot \|\mathbf{c} - \mu(C)\|_2^2,$$

where $\mu(C)$ is the point $\frac{\sum_{\mathbf{p} \in C} \mathbf{p}}{|C|}$.

For Categorical k-Clustering/L_0-k-Clustering, it can be trivially noted that the optimal center can be computed as the coordinate-wise majority of the cluster points, and for L_1-k-Clustering it is the coordinate-wise median. We prove these facts, and a number of other statements about optimal centers for L_p-k-Clustering in Chapter 5.

Matrix notations. Observe also that in the spaces where the elements are represented by a vector of coordinates, k-Clustering can be naturally seen as a matrix approximation problem. For example, consider an instance of k-Means in the d-dimensional Euclidean space $\mathbb{R}^d$. We associate the input set of points $P = \{p_1, \ldots, p_n\}$ with the matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ such that its rows are the coordinate vectors of the respective points. Specifically, the rows of $\mathbf{A}$ are $\mathbf{a}_1, \ldots, \mathbf{a}_n$, and for each $i \in [n], j \in [d]$, $\mathbf{a}_i[j]$ is a real number representing the j-th coordinate of the point p_i in the standard basis of the space $\mathbb{R}^d$. Thus, k-Means can be phrased as the following optimization problem.

Euclidean k-MEANS

Input: A matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, and a positive integer k.
Task: Find the matrix $\mathbf{B} \in \mathbb{R}^{n \times d}$ with at most k distinct rows that minimizes

$$\|\mathbf{A} - \mathbf{B}\|_F^2 .$$

Recall that the Frobenius norm of a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ is the sum of its squared entries, that is,

$$\|\mathbf{A}\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{d} \mathbf{A}_{i,j},$$

where $\mathbf{A}_{i,j}$ is the entry of $\mathbf{A}$ at the intersection of the i-th row and the j-th column. The equivalence to the Euclidean k-MEANS problem defined above holds because of the following. Any set of centers $C = \{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ and a clustering $C_1, \ldots, C_k$ of the given points P correspond naturally to a matrix $\mathbf{B}$ that has the rows from C and gives the same value of $\|\mathbf{A} - \mathbf{B}\|_F^2$ as the cost of the clustering. For each $i \in [n]$, suppose that the point $p_i \in P$ gets assigned into the cluster C_t, for some $t \in [k]$, then the i-th row of $\mathbf{B}$ is the vector $\mathbf{c}_t$. Then the cost of the clustering can be expressed as

$$\sum_{t=1}^{k} \sum_{\substack{i \in [n] \text{ s.t.} \\ p_i \in C_t}} \|\mathbf{a}_i - \mathbf{c}_t\|_2^2 = \sum_{t=1}^{k} \sum_{p_i \in C_t} \sum_{j=1}^{d} (\mathbf{a}_i[j] - \mathbf{c}_t[j])^2 = \sum_{i=1}^{n} \sum_{j=1}^{d} (\mathbf{A}_{ij} - \mathbf{B}_{ij})^2 = \|\mathbf{A} - \mathbf{B}\|_F^2 .$$

Going backwards, we also obtain naturally the clustering that corresponds to a given matrix $\mathbf{B}$ with at most k distinct rows, having the same cost as $\|\mathbf{A} - \mathbf{B}\|_F^2$.

3.1.2 Previous Work

Having listed the various spaces that k-CLUSTERING can be defined on, we move to discussing the current state of research of these problems. Clustering problems, and especially Euclidean k-MEANS, are among the most prevalent problems occurring in virtually every subarea of data science. We refer to the survey of Jain [121] for an extensive overview. While in practice the most common approaches to clustering are based on different variations of Lloyd's heuristic [147], the problem is interesting from the theoretical perspective as well. In particular, there is a vast amount of literature on approximation algorithms for k-CLUSTERING whose behavior can be analyzed rigorously. One long line of results concerns designing polynomial-time constant-factor approximation algorithms for various clustering problems. As polynomial-time approximation is not the main focus of this work, we only mention briefly the key results. For a Euclidean space of arbitrary dimension, a 6.357-approximation and

a 2.633-approximation is the current state-of-the-art for k-Means and k-Median respectively [4]. On the other hand, starting from APX-hardness [14], the best currently known lower bound is that k-Means and k-Median are hard to approximate within a factor of 1.17 and 1.06 under Unique Games Conjecture, respectively [65]. In general metric spaces, the best known algorithms have approximation factors of 9 for k-Means and 2.67 for k-Median [40, 4], while the corresponding NP-hardness results hold for approximation up to a factor of 3.94 and 1.73, respectively [106].

In the recent years, there have also appeared a number of results providing FPT-time $(1+\varepsilon)$-approximation when parameterized by the number of clusters k. From this line of work, we first mention the seminal paper of Kumar, Sabharwal, and Sen [137]. In that work, Kumar et al. gave an $(1+\varepsilon)$-approximation algorithm for Euclidean k-Median and k-Means that runs in time $2^{(k/\varepsilon)^{\mathcal{O}(1)}}nd$. In Chapter 8, we provide a natural extension of this result to clustering points with missing entries, when the number of missing entries per point is bounded by a parameter. For another problem of our interest, Binary k-Clustering, Ostrovsky and Rabani [165] gave a randomized $(nd)^{f(k,\varepsilon)}$-time algorithm for a certain function f of k and ε. This result is recently improved to running time of the form $g(k,\varepsilon)nd$ in [88, 17], for some function g of k and ε. Approximation schemes for non-binary Categorical k-Clustering are given by Ban et al. in [17]. For k-Median and k-Means in general metric spaces, analogous results are not possible, however FPT time still allows for a better approximation factor. Recently, Cohen-Addad et al. [64] showed a $(1+2/e+\varepsilon)$-factor approximation (respectively, $(1+8/e+\varepsilon)$-factor) in FPT time when parameterized by k. Note that this matches NP-hardness lower bounds of [106], and this hardness was further strengthened in [64] to the following. There exists a function f of ε such that any $(1+2/e-\varepsilon)$-approximation algorithm for k-Median in general metric spaces (respectively, $(1+8/e-\varepsilon)$-approximation for k-Means) must run in time at least $n^{k^{f(\varepsilon)}}$, assuming Gap-ETH.

For approximation algorithms that perform well when the number of clusters k is small, tools that received immense attention and usage recently, are dimensionality reduction and coresets. In short, they allow to replace the original dataset by a smaller one, while preserving approximately the clustering objective. Respectively, the number of dimensions is reduced to a certain function of k and ε, or the number of distinct points. We discuss more about this concepts and the body of work behind them later in the respective sections.

When it comes to exact solutions, we observe the following phenomena. While heuristic algorithms for k-Clustering work surprisingly well in practice, and there exist powerful approximation algorithms, from the perspective of exact parameterized complexity, k-Clustering is practically intractable for all previously studied parameterizations. The k-Clustering problem is naturally "multivariate": in addition to the number of points n, there are also parameters like space dimension d, number of clusters k or the cost of clustering D. Even the special cases, Euclidean k-Means and Binary k-Clustering, are known to be NP-complete for $k=2$ [9, 82]

and for $d = 2$ [157, 151]. By the classical work of Inaba et al. [118], in the case when both d and k are constants, Euclidean k-Means is solvable in polynomial time $\mathcal{O}(n^{dk+1})$. It is a long-standing open problem whether k-Means is FPT parameterized by $d + k$. Under ETH, the lower bound of $n^{\Omega(k)}$, even when $d = 4$, was shown by Cohen-Addad et al. in [63] for the settings where the set of potential candidate centers is explicitly given as input. However the lower bound of Cohen-Addad et al. does not generalize to the general continuous setting, where any point in Euclidean space can serve as a center. In the case of Categorical k-Clustering on binary vectors, the problem is solvable in time $2^{\mathcal{O}(D \log D)}(nd)^{\mathcal{O}(1)}$ [90], where D is the optimal cost of the clustering. In fact, the work [90] is the departure point for our study of the parameterized complexity of L_p-k-Clustering, presented in Chapter 5. For some L_p-norms we show an analogue of the FPT algorithm of [90], while for others we show negative results.

3.1.3 Robustness

One extension of k-Clustering that we study in this work is motivated by the following. A major drawback of k-Median and k-Means is that beyond capturing the most basic scenarios, none of them is suitable for modeling complex applications. For example, all of the algorithms for these problems fail to retrieve the natural clustering in the presence of noise or outliers. To address the issue of finding a generic model of clustering, researchers have taken the approach of studying the basic clustering problems with additional constraints. For example, to deal with the issue of noisy data, the most common model used is robust clustering [52, 94, 87, 49, 136]. In robust clustering, the clustering objective remains the same, however in addition it is allowed to discard a subset of data points from the input. The idea is that ignoring the discarded points (also called outliers) would allow the clustering algorithm to focus on correctly partitioning the remaining uncorrupted data.

Another considerable challenge to k-means (or k-median) clustering approaches is the high dimensionality of modern datasets. When the data contains many irrelevant features (or attributes), an application of cluster analysis with a complete set of features could significantly decrease the solution's quality. The typical approach to overcome this challenge in practice is *feature selection*. The method is based on selecting a small subset of relevant features from the data and applying the clustering algorithm only on the selected features. The survey of Alelyani et al. [7] provides a comprehensive overview on methods for feature selection in clustering. Due to the significance of feature selection, there is a multitude of heuristic methods addressing the problem. However, very few provably correct methods are known [34, 35, 60].

Kim et al. [130] introduced a model of feature selection in the context of k-means clustering. We use their motivating example here. Decision-making based on market surveys is a pragmatic marketing strategy used by manufacturers to increase customer satisfaction. The respondents of a survey are segmented into similar-interest groups so that each group of customers can be treated in a similar way. Consider

such a market survey data that typically contains responses of customers to a set of questions regarding their demographic and psychographic information, shopping experience, attitude towards new products and expectations from the business. The standard practice used by market managers to segment customers is to apply clustering techniques w.r.t. the whole set of features. However, depending on the application, responses corresponding to some of the features might not be relevant to find the target set of market segments. Also, some of the responses might contain incomplete or spurious information. To address this issue, Kim et al. [130] considered several quality criteria to return *pareto* optimal (or non-dominated) solutions that optimize one or more criteria. One such solution removes a suitable subset of features and cluster the data w.r.t. the remaining features.

In Chapter 6, we study a theoretical model of feature selection and cluster analysis with respect to Categorical k-Clustering. In particular, we define the best subset of features (of a given size) as the subset that minimizes the corresponding cost of clustering. The goal is again to compute such a subset and the respective clusters. We provide the first parameterized algorithmic and complexity results for cluster analyses of categorical data in two situations: (i) a subset of data points (outliers) is to be removed, and (ii) a subset of features is to be removed. To compare with the first case of clustering involving traditional (row) outliers, in the first case, we refer to the irrelevant features as column outliers. Thus we arrive to the definitions of the following two problems. First, we define the extension of Categorical k-Clustering with row outliers, where it is allowed to remove a certain number of points from the dataset to minimize the cost of the clustering. We state the problem in the matrix form and as a decision problem, as we will use this definition extensively in Chapter 6.

Categorical k-Clustering with Row Outliers

Input: An alphabet Σ, an $n \times d$ matrix $\boldsymbol{A}$ with rows $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$ such that $\boldsymbol{a}_i \in \Sigma^d$ for all $1 \leq i \leq n$, a positive integer k, non-negative integers D and ℓ.

Task: Decide whether there is a subset $O \subset \{1, 2, \ldots, n\}$ of size at most ℓ, a partition of $\{1, 2, \ldots, n\} \setminus O$ into k sets $\{I_1, I_2, \ldots, I_k\}$ called clusters, and vectors $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k \in \Sigma^m$ such that the cost of clustering is at most D, that is,

$$\sum_{t=1}^{k} \sum_{i \in I_i} d_H(\boldsymbol{a}_i, \boldsymbol{c}_t) \leq D.$$

Now we define the feature selection variant, and for that we need the following notation. For a set of indices $S \subset \{1, 2, \ldots, d\}$ and an $n \times d$ matrix $\boldsymbol{A}$, let $\boldsymbol{A}^{-S}$ be the matrix obtained from $\boldsymbol{A}$ by removing the columns with indices in S. We denote

the rows of $\boldsymbol{A}^{-S}$ by $\boldsymbol{a}_i^{-S}$ for $1 \leq i \leq n$.

Categorical k-Clustering with Column Outliers

Input: An alphabet Σ, an $n \times d$ matrix $\boldsymbol{A}$ with rows $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n$ such that $\boldsymbol{a}_j \in \Sigma^d$ for all $1 \leq i \leq n$, a positive integer k, non-negative integers D and ℓ.

Task: Decide whether there is a subset $O \subset \{1, 2, \ldots, d\}$ of size at most ℓ, a partition $\{I_1, I_2, \ldots, I_k\}$ of $\{1, 2, \ldots, n\}$, and vectors $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k \in \Sigma^{d-|O|}$ such that

$$\sum_{t=1}^{k} \sum_{i \in I_i} d_H(\boldsymbol{a}_i^{-O}, \boldsymbol{c}_t) \leq D.$$

In other words, we want to identify $m - \ell$ features that are suited best for the clustering objective. Even though the cost function in our problem is basically the k-median cost function, we call our problem by a different name. This is because categorical data is in general unordered and hence the median of a set of points is not defined in our case. While Categorical k-Clustering with Column Outliers looks very similar to the one with row outliers, it appears that the former is computationally much more challenging.

3.1.4 Constrained Clustering

Another weakness of the basic model of k-Clustering is that it allows no control over the composition of the clusters, apart from the global cost minimization. This spurred researchers to consider various variants of constrained clustering, where in addition to the clustering objective there is also a constraint on which clusterings are allowed. More formally, for any fixed version of the k-Clustering problem, its $\mathcal{C}$-constrained version is defined as follows, given a family $\mathcal{C}$ of the allowed partitions of the points. The objective is again to pick k centers $c_1, \ldots, c_k$ and partition the input points P into k clusters $C_1, \ldots, C_k$ in order to minimize

$$\sum_{i=1}^{k} \sum_{p \in C_i} \text{dist}(p, c_i),$$

but only the partitions belonging to $\mathcal{C}$ are allowed, i.e. the tuple $(C_1, \ldots, C_k)$ must belong to the family $\mathcal{C}$. One of the most studied problems of this form is capacitated clustering that we use here as an example. In the simplest variant of capacitated clustering, in addition to the clustering input, a uniform upper bound U on the cluster size is given, and the task is to find the clustering that has at most U points in each

cluster, and has the lowest cost among those. Observe that even such a simple constrained version of k-Clustering is already very different from the unconstrained version, and the existing algorithms for k-Clustering cannot be applied immediately. For instance, the property of k-Clustering that each point $p \in P$ is assigned to the closest center does not hold, as such an assignment could violate the constraint arbitrarily. This alone is a crucial requirement for a number of unconstrained clustering algorithms. For the capacitated version of k-Median and k-Means in general metric spaces, no polynomial-time $\mathcal{O}(1)$-approximation is known, in contrast to the unconstrained variants. However, bicriteria constant-approximations are known that violate either the capacity constraints or the constraint on the number of clusters, by an $\mathcal{O}(1)$ factor [41, 39, 48, 58, 74, 145]. Recently, Cohen-Addad and Li [66] designed $(3+\varepsilon)$- and $(9+\varepsilon)$-approximation in FPT time parameterized by k for the capacitated version of k-Median and k-Means, respectively.

For Euclidean k-Means and FPT-time $(1+\varepsilon)$-approximation, the situation is more fortunate. Ding and Xu [75] gave a unified framework with running time $2^{\mathrm{poly}(k/\varepsilon)}(\log n)^{k+1}nd$ that generates a collection of $2^{\tilde{\mathcal{O}}(k/\varepsilon)}$ candidate sets of centers[2], such that for any partition of points at least one of these sets provides $(1+\varepsilon)$ approximation to the cost of the respective partition. Thus, this algorithm allows to solve Euclidean k-Means with any given constraint $\mathcal{C}$, provided that a good assignment to a given set of centers can be found reasonably fast. Subsequently, Bhattacharya et al. [27] improved this result to the running time of $2^{\tilde{\mathcal{O}}(k/\varepsilon)}nd$, where $\tilde{\mathcal{O}}$ hides factors logarithmic in k and ε^{-1}, and Feng et al. [86] later improved the constant under $\tilde{\mathcal{O}}$. All in all, this makes the case of Euclidean k-Means with constraints similar to the unconstrained version where the FPT approximation scheme of Kumar et al. works [137]. Note that the latter algorithm does not extend to the constrained case automatically as it relies heavily on the fact that a point is assigned to the closest center, and thus the framework discussed above is a highly non-trivial generalization of [137].

Fair Clustering

Apart from capacitated clustering, many other clustering constraints have been studied in the literature, e.g. lower bounds [179], matroid [50], fault tolerance [129], chromatic clustering [75] and diversity [143]. We introduce some of these problems later, for now, let us focus on clustering problems with *fairness* constraints, which received a lot of attention recently, and is central to Chapter 7 of this work. Clustering with fairness constraints or fair clustering was introduced by Chierichetti et al. [56] in a seminal work. The notion became widely popular within a short period triggering a large body of new work [173, 23, 24, 115, 15, 30, 54, 3, 134]. The idea of fair clustering is to enforce additional (fairness) constraints to remove the inherent bias or discrimination from vanilla (unconstrained) clustering. For example, suppose we

[2] $\tilde{\mathcal{O}}$ hides factors polylogarithmic in k and ε^{-1}.

have a sensitive feature (e.g, race or gender). We want to find a clustering where the fraction of points from a traditionally underrepresented group in every cluster is more or less equal to the fraction of points from this group in the dataset. Indeed, the work of Chierichetti et al. [56] shows that clustering computed by classical vanilla algorithms can lead to widely varied ratios for a particular group, especially when the number of clusters is large enough.

There are many settings where machine learning algorithms, trained on datasets of past instances, play a crucial role in decision-making [77, 128, 154, 169]. These algorithms are sophisticated and time-efficient and produce accurate results most of the time. However, there has been a growing concern that these algorithms are biased or discriminatory towards traditionally underrepresented groups [13, 73, 98]. One example that stands out and has generated substantial controversy in recent years is concerning the COMPAS risk tool, which is a widely used statistical method for assigning risk scores in the criminal justice system. Angwin et al. argued that this tool was biased against African-American defendants [13, 140]. Most of the automated decision-making systems are highly influenced by human players, especially during the training procedure. Importantly, clustering also plays a crucial role in this training part. For example, a widely used technique called feature engineering [124, 101] labels samples with their cluster id to enhance the expressive power of learning methods. Hence, the study of biases and discriminatory practices in the context of clustering is well-motivated.

Over the past few years, researchers have put a lot of effort into understanding and resolving the issues of biases in machine learning. This research has led towards different notions of fairness [42, 69, 79]. Kleinberg et al. [132] formalized three fairness conditions and showed that it is not possible to satisfy them simultaneously, except in very special cases (see also [57] for a similar treatment). The notion of fairness studied by Chierichetti et al. [56] is based on the concept of *disparate impact* (DI) [85]. Roughly, the DI doctrine articulates that the protected attributes should not be explicitly used in decision-making, and the decisions taken should not be disproportionately different for members in different protected groups.

Following the DI doctrine, Chierichetti et al. [56] considered the model where there is a single sensitive or protected attribute called color that can take only two values: red and blue. The coordinates of the points are unprotected; that is, they do not take part in the fairness constraints. For any integer $t \geq 1$, Chierichetti et al. defined the (t, k)-*fair clustering* problem where in each cluster the ratio of the number of red points to the number of blue points must be at most t and at least $1/t$. Thus in their case, the notion of fairness is captured by the balance parameter t.

Rösner and Schmidt [171] studied a multicolored version of the above problem, where a clustering is fair if the ratios between points of different colors are the same in every cluster. Subsequently, Bercea et al. [24] and Bera et al. [23] independently formulated a model generalizing the problems studied in [56] and [171]. In this model,

we are given ℓ groups $P_1, \ldots, P_\ell$ of points in a metric space and balance parameters $\alpha_i, \beta_i \in [0, 1]$ for each group $1 \leq i \leq \ell$. A clustering is *fair* if the fraction of points from group i in every cluster is at least β_i and at most α_i. Additionally, in [23], the groups are allowed to overlap, i.e, a point can belong to multiple protected classes. Note that this assumption is needed to model many applications, e.g, consider clustering of individuals where a subset of the individuals are African-American women. In fact, the experiments in [23] show that imposing fairness concerning one sensitive attribute (say gender) might lead to unfairness to another (say race) if not protected. We refer to the fair clustering problem with overlapping groups as (α, β)-*fair clustering*. We note that this is the most general version of fair clustering considered in the literature, and this is the notion of fairness we adopt in Chapter 7. Both [24] and [23] obtain polynomial time $\mathcal{O}(1)$-approximation for this problem that violates the fairness constraints by at most small additive factors that depend on Γ. Here Γ is the number of distinct collections of groups to which a point may belong. If all the groups are disjoint, then $\Gamma = \ell$. Note that if a point can belong to at most Λ groups, then Γ is at most ℓ^Λ. As noted in [23] and [115], while Λ can very well be more than 1, it is usually a constant in most of the applications. Thus, in this case, $\Gamma = \ell^{\mathcal{O}(1)}$, which is expected to be much smaller compared to n, the total number of points in the union of the groups. Apart from the above, a number of works related to fair clustering were devoted to scalability and coresets, as well as our main result of Chapter 7, we discuss this more later in this chapter. As well as the number of clusters k and the error parameter ε, the sizes of the coresets typically depend on Γ.

Other Color-constrained Problems

In fact, our results hold in a more general setting, for any kind of constraints that can be stated in terms of how many points from each group are assigned to each cluster, which we call *color constraints*. Note that any kind of constraint can be expressed in this way if every point belongs to its own group, however we are interested in the case where the number of colors/groups is small compared to the input size. Both capacitated clustering, which is a special case where all points belong to a single group, and (α, β)-fair clustering belong to this category, as well as several other well-studied problems that we introduce next.

- In the lower-bounded clustering problem, we are given a lower bound parameter L and the size of each cluster must be at least L. This constraint is essentially a counterpart of the capacitated clustering, and falls also into the class of color constraints, where all points belong to a single group. Polynomial time constant-approximations for lower-bounded k-Median in general metric spaces follow from [179, 5]. Also, an FPT $\mathcal{O}(1)$-approximation with parameter k is known for this problem [23].
- In the clustering with diversity problem, $P = \cup_{i=1}^{\ell} P_i$ is a set of n colored points such that all points in P_i have the same color, and each cluster must have no

more than a fraction $1/t$ (for some given constant $t > 1$) of its points sharing the same color. Thus, for each cluster C and $i \in [\ell]$, $|C \cap P_i| \leq |C|/\ell$. We note that each point can have only one color. We note that the t-diversity clustering is a special case of (α, β)-fair clustering without the lower bound constraints involving parameter β, and $\alpha = 1/t$.

· In chromatic clustering, again $P = \cup_{i=1}^{\ell} P_i$ is a set of n colored points such that all points in P_i have the same color. The constraint is that each cluster must contain at most one point from each P_i. Naturally, this problem a;so falls into the category of color-constrained clustering problems.

For more discussion about various constraint problems and the formal definitions of fairness constraints and general color constraints, we refer to Chapter 7.

3.2 Low-Rank Approximation

Classical *principal component analysis* (PCA) is one of the most popular and successful techniques used for dimension reduction in data analysis and machine learning [167, 114, 80]. In PCA one seeks the best low-rank approximation of data matrix $\mathbf{A}$ by solving

$$\text{minimize } \|\mathbf{A} - \mathbf{L}\|_F^2 \tag{3.1}$$

$$\text{subject to } \operatorname{rank}(\mathbf{L}) \leq r. \tag{3.2}$$

By the Eckart-Young-Mirsky theorem [80, 160], PCA is efficiently solvable via Singular Value Decomposition (SVD). Recall that the singular value decomposition of a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ is given by $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{n \times n}$ and $\mathbf{V} \in \mathbb{R}^{d \times d}$ are real orthogonal matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{n \times d}$ is a rectangular diagonal matrix with non-negative real numbers on the diagonal, known as the singular values of $\mathbf{A}$. The Eckart-Young-Mirsky theorem states that truncating the SVD of $\mathbf{A}$ to its first r singular values yields the best possible r-rank approximation of $\mathbf{A}$. That is, the solution to (3.2) is the matrix $\mathbf{A}_r = \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^\top$, where $\mathbf{U}_r \in \mathbb{R}^{n \times r}$ are the first r columns of $\mathbf{U}$, $\mathbf{V}_r \in \mathbb{R}^{d \times r}$ are the first r columns of $\mathbf{V}$, and $\mathbf{\Sigma}_r \in \mathbb{R}^{r \times r}$ is the diagonal matrix containing the first r singular values. Since SVD of $\mathbf{A}$ can be found in polynomial time, the same holds for the matrix $\mathbf{A}_r$. PCA is used as a preprocessing step in a great variety of modern applications including face recognition, data classification, and analysis of social networks.

As mentioned above, contrary to k-Clustering, the basic version of the low-rank approximation problem is long known to be efficiently solvable. However, just as in the case of clustering, various extensions of the problem arise wherever there is a need to go beyond the most basic scenarios, and the algorithmic complexity of these extensions is not as simple anymore. In what follows, we first consider robust extensions of PCA, specifically the version with outliers. Second, we discuss

extending PCA from the Frobenius norm to operator norms, and from real-valued data to categorical data.

3.2.1 Outliers

The key interest of Part IV of this work is a variant of PCA with outliers, proposed in [190, 191, 55, 176]. Suppose that we have n points (observations) in d-dimensional space. We know that a part of the points are arbitrarily located (say, produced by corrupted observations) while the remaining points are close to an r-dimensional true subspace. We do not have any information about the true subspace and about the corrupted observations. Our task is to learn the true subspace and to identify the outliers. As a common practice, we collect the points into $n \times d$ matrix $\mathbf{A}$, thus each of the rows of $\mathbf{A}$ is a point and the columns of $\mathbf{A}$ are the coordinates. However, it is very likely that PCA of $\mathbf{A}$ will not reveal any reasonable information about non-corrupted observations — a well-documented drawback of PCA is its vulnerability to even a very small number of outliers, an example is shown in Figure 3.2.

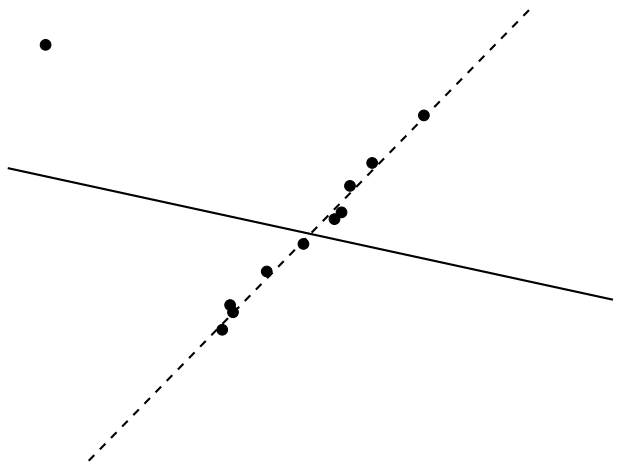


Figure 3.2: An illustration on how outliers impact PCA. The optimal approximation line (in dashed) of the given set of points without the evident outlier shows the linear structure of the dataset. However, when the outlier is present, the principal component (in solid) changes drastically.

Matrix formulation suggests the following interpretation: we seek a low-rank matrix $\mathbf{L}$ that, with an exception in few rows, approximates $\mathbf{A}$ best.

PCA WITH OUTLIERS

Input: Data matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, integer parameters r and k.
Task:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{A} - \mathbf{L} - \mathbf{N}\|_F^2 \\ \text{subject to} \quad & \mathbf{L}, \mathbf{N} \in \mathbb{R}^{n \times d}, \\ & \operatorname{rank}(\mathbf{L}) \leq r, \text{ and} \\ & \mathbf{N} \text{ has at most } k \text{ non-zero rows.} \end{aligned}$$

PCA WITH OUTLIERS has a natural geometric interpretation. Given n points in $\mathbb{R}^d$, represented by the rows of $\mathbf{A}$, we seek for a set of k outliers, represented by the non-zero rows of $\mathbf{N}$, whose removal leaves the remaining $n - k$ inliers as close as possible to an r-dimensional subspace. The matrix $\mathbf{L}$ contains then the orthogonal projections of the inliers to this subspace.

An important special case of PCA WITH OUTLIERS is the ROBUST SUBSPACE RECOVERY problem, where the task is to decide whether for an instance of PCA WITH OUTLIERS the value of the optimal solution is 0 or not. Equivalently, the input to ROBUST SUBSPACE RECOVERY is an instance $(\mathbf{A}, r, k)$ of PCA WITH OUTLIERS, and the task is to decide whether there exist matrices $\mathbf{L}$ and $\mathbf{N}$ such that $\mathbf{A} = \mathbf{L} + \mathbf{N}$, the rank of $\mathbf{L}$ is at most r, and $\mathbf{N}$ has at most k non-zero rows.

PCA WITH OUTLIERS belongs to the large class of extensively studied robust PCA problems, see e.g. [183, 191, 36]. In the robust PCA setting we observe a noisy version $\mathbf{A}$ of data matrix L whose principal components we have to discover. In the case when $\mathbf{A}$ is a "slightly" disturbed version of $\mathbf{L}$, PCA performed on M provides a reasonable approximation for $\mathbf{L}$. However, when $\mathbf{A}$ is very "noisy" version of $\mathbf{L}$, like being corrupted by a few outliers, even one corrupted outlier can arbitrarily alter the quality of the approximation.

One of the approaches to robust PCA, which is relevant to our work, is to model outliers as an additive sparse matrix. Thus we have a data $d \times n$ matrix $\mathbf{A}$, which is the superposition of a low-rank component $\mathbf{L}$ and a sparse component $\mathbf{N}$. That is, $\mathbf{A} = \mathbf{L} + \mathbf{N}$. This approach became popular after the works of Candès et al. [43], Wright et al. [190], and Chandrasekaran et al. [47]. The study of the following natural, and seemingly more difficult extension of (3.2) to the PCA with outliers, was initiated by Xu et al. [191].

$$\begin{aligned} \text{minimize} \quad & \operatorname{rank}(\mathbf{L}) + \lambda \|\mathbf{N}\|_0 \\ \text{subject to} \quad & \mathbf{A} = \mathbf{L} + \mathbf{N}. \end{aligned} \tag{3.3}$$

Here $\|\mathbf{N}\|_0$ denotes the number of non-zero columns in matrix $\mathbf{N}$ and λ is a regularizing parameter. Xu et al. [191] approached this problem by building its convex

surrogate and applying efficient convex optimization-based algorithm for the surrogate. Chen et al. [55] studied the variant of the problem with the partially observed data.

The problem strongly related to (3.2) was studied in computational complexity under the name Matrix Rigidity [105, 182]. Here, for a given matrix $\mathbf{A}$, and integers r and k, the task is to decide whether at most k entries of $\mathbf{A}$ can be changed so that the rank of the resulting matrix is at most r. Equivalently, this is the problem to decide whether a given matrix $\mathbf{A} = \mathbf{L} + \mathbf{N}$, where $\text{rank}(\mathbf{L}) \leq r$ and $\|\mathbf{N}\|_{0,0} \leq k$. Fomin et al. [91] gave an algorithm solving Matrix Rigidity in time $2^{\mathcal{O}(r\cdot k\cdot \log(r\cdot k))}\cdot (nd)^{\mathcal{O}(1)}$. On the other hand, they show that the problem is W[1]-hard parameterized by k. In particular, this implies that an algorithm of running time $f(k)\cdot (rnd)^{\mathcal{O}(1)}$ for this problem is highly unlikely for any function f of k only.

A natural extension of the robust PCA approach (3.3) is to consider the noisy version of robust PCA: Given $\mathbf{A} = \mathbf{L} + \mathbf{N} + \mathbf{\Delta}$, where $\mathbf{L}$, $\mathbf{N}$, and $\mathbf{\Delta}$ are unknown, but $\mathbf{L}$ is known to be low rank, $\mathbf{N}$ is known to have a few non-zero rows, and the noise matrix $\mathbf{\Delta}$ is of small Frobenius norm, recover $\mathbf{L}$. Wright et al. [190] studied the following model of noisy robust PCA:

$$\begin{aligned} \text{minimize} \quad & \text{rank}(\mathbf{L}) + \lambda\|\mathbf{N}\|_0 \\ \text{subject to} \quad & \|\mathbf{A} - \mathbf{L} - \mathbf{N}\|_F^2 \leq \varepsilon. \end{aligned} \tag{3.4}$$

Thus (3.4) models the situations when we want to learn the principal components of n points in d-dimensional space under the assumption that a small number of coordinates is corrupted.

Much less is known about algorithms with guaranteed performance for PCA with Outliers and Robust Subspace Recovery. Hardt and Moitra [111] used the Small Set Expansion conjecture (SSE) to show that for Robust Subspace Recovery, even if one allows to select $(1+\delta)k$ outliers, for some $\delta > 0$, it is still unlikely that a polynomial time algorithm can find a $c\cdot r$-dimensional subspace containing all remaining $n - (1+\delta)k$ points for any $c > 0$. [26] used the smallest edge r-subgraph conjecture to show that, there is exists a constant $c > 0$, such that no polynomial time algorithm can find a rn^c-dimensional subspace that results in a multiplicative approximation to the objective cost. By using the hardness of the rank reduction problem for matroids [89, Proposition 8.1], it is possible to show that Robust Subspace Recovery is W[1]-hard parameterized by k.

For PCA with Outliers, [26] provides two $(1+\varepsilon)$-approximation bicriteria algorithms. While the cost of their solution is preserved within $(1+\varepsilon)$ factor, the number of outliers k and the dimension of the subspace r in the solution are also approximated.

3.2.2 Beyond the Frobenius Norm

In fact, Eckart-Young-Mirsky theorem guarantees that PCA is not only efficiently solvable for the Frobenius norm, but also in the *spectral* norm, given by $\|\mathbf{A}\|_2 = \sup_{\mathbf{x}\neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$. The spectral norm is an "extremal" norm — it measures the worst-case stretch of the matrix. On the other hand, the Frobenius norm is "averaging" in this sense. Spectral norm is usually applied in situations when one is interested in actual columns for the subspaces they define and is of greater interest in scientific computing and numerical linear algebra. The Frobenius norm is widely used in statistics and machine learning, see the survey of Mahoney [152] for further discussions.

Recently there has been considerable interest in developing algorithms for low-rank matrix approximation problems for binary (categorical) data. Such variants of dimension reduction for high-dimensional data sets with binary attributes arise naturally in applications involving binary data sets, like latent semantic analysis [25], pattern discovery for gene expression [174], or web search models [133], see [72, 123, 107, 135, 166, 192] for other applications. In many such applications it is much more desirable to approximate a binary matrix $\mathbf{A}$ with a binary matrix $\mathbf{L}$ of small ($\mathsf{GF}(2)$ or Boolean) rank because it could provide a deeper insight into the semantics associated with the original matrix. There is a big body of work done on binary and Boolean low-rank matrix approximation, see [18, 21, 72, 149, 158, 159, 161, 181, 180] for further discussions.

Unfortunately, SVD is not applicable for the binary case which makes such problems computationally much more challenging. For a binary matrix, its squared Frobenius norm is equal to the number of its 1-entries, that is $\|\mathbf{A}\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{d} \mathbf{A}_{ij}$. Thus, the value $\|\mathbf{A}-\mathbf{L}\|_F^2$ measures the total Hamming distance from points (columns) of $\mathbf{A}$ to the subspace spanned by the columns of $\mathbf{L}$. For this variant of the low-rank binary matrix approximation, a number of approximation algorithms were developed, resulting in efficient polynomial time approximation schemes obtained in [17, 88]. However, the algorithmic complexity of the problem for any vector-induced norm, including the spectral norm, remained open.

For binary matrices, the natural "extremal" norm to consider is the $\|\cdot\|_1$ norm, also known as *column-sum norm*, operator 1-norm, or Hölder matrix 1-norm. That is, for a matrix $\mathbf{A}$,

$$\|\mathbf{A}\|_1 = \sup_{\|\mathbf{x}\|_1\neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_1}{\|\mathbf{x}\|_1} = \max_{1\leq j\leq d} \sum_{i=1}^{n} |\mathbf{A}_{ij}|.$$

In other words, the column-sum norm is the maximum number of 1-entries in a column in $\mathbf{A}$, whereas the Frobenius norm is the total number of 1-entries in $\mathbf{A}$. The column-sum norm is analogous to the spectral norm, only it is induced by the ℓ_1 vector norm, not the ℓ_2 vector norm.

We consider the problem, where for an $n \times d$ binary data matrix $\mathbf{A}$ and a positive

integer constant r, one seeks a binary matrix $\mathbf{L}$ optimizing

$$\begin{aligned} &\text{minimize } \|\mathbf{A}-\mathbf{L}\|_1 \\ &\text{subject to } \operatorname{rank}(\mathbf{L}) \leq r. \end{aligned}$$

Here, by the rank of the binary matrix $\mathbf{L}$ we mean its $\mathsf{GF}(2)$-rank. We refer to the problem defined above by the name L_1-Rank-r Approximation over $\mathsf{GF}(2)$. The value $\|\mathbf{A}-\mathbf{L}\|_1$ is the maximum Hamming distance from each of the columns of $\mathbf{A}$ to the subspace spanned by columns of $\mathbf{L}$ and thus, compared to approximation with the Frobenius norm, it could provide a more accurate dimension reduction.

It is easy to see by the reduction from the Closest String problem, that already for $r = 1$, L_1-Rank-r Approximation over $\mathsf{GF}(2)$ is NP-hard. In Chapter 11 we show that L_1-Rank-r Approximation over $\mathsf{GF}(2)$ admits a polynomial time approximation scheme (PTAS). Next we discuss other related work.

The variant of low-rank approximation with both matrices $\mathbf{A}$ and $\mathbf{L}$ binary, and the objective being the Frobenius norm, is known as Low GF(2)-Rank Approximation. Due to numerous applications, various heuristic algorithms for Low GF(2)-Rank Approximation can be found in the literature [122, 123, 95, 135, 174].

When it concerns rigorous algorithmic analysis of Low GF(2)-Rank Approximation, Gillis and Vavasis [100] and Dan et al. [72] have shown that Low GF(2)-Rank Approximation is NP-complete for every $r \geq 1$. Fomin et al. studied parameterized algorithms for Low GF(2)-Rank Approximation in [90]. The first approximation algorithm for Low GF(2)-Rank Approximation is due to Shen et al. [174], who gave a 2-approximation algorithm for the special case of $r = 1$. For rank $r > 1$, Dan et al. [72] have shown that a $(r/2+1+\frac{r}{2(2^r-1)})$-approximate solution can be formed from r columns of the input matrix $\mathbf{A}$. Recently, these algorithms were significantly improved in [17, 88], where efficient polynomial time approximation schemes were obtained.

Also note that for general (non-binary) matrices a significant amount of work is devoted to $L1$-PCA, where one seeks a low-rank matrix $\mathbf{L}$ approximating given matrix $\mathbf{A}$ in *entrywise* ℓ_1 norm, see e.g. [178].

While the main motivation to study L_1-Rank-r Approximation over $\mathsf{GF}(2)$ stems from low-rank matrix approximation problems, this problem also generalizes Closest String, very well-studied problem about strings. Given a set of binary strings $S = \{s_1, s_2, \ldots, s_n\}$, each of length m, the Closest String problem is to find the smallest d and a string s of length m which is within Hamming distance d to each $s_i \in S$. Let us note that Closest String can be seen as a special case of L_1-Rank-r Approximation over $\mathsf{GF}(2)$ for $r = 1$. Indeed, Closest String is exactly the variant of L_1-Rank-r Approximation over $\mathsf{GF}(2)$, where columns of $\mathbf{A}$ are strings of S and the approximation matrix $\mathbf{L}$ is required to have all columns equal. Note that in a binary matrix $\mathbf{L}$ of rank 1 all non-zero columns are equal. However, it is easy to construct an equivalent instance of Closest String by attaching to each

string of S a string 1^{m+1}, such that the solution to L_1-Rank-r Approximation over $\mathsf{GF}(2)$ for $r = 1$ does not have zero columns.

A long history of algorithmic improvements for Closest String was concluded by the PTAS of running time $n^{\mathcal{O}(\varepsilon^{-5})}$ by Li, Ma, and Wang [144], and this running time was later improved to $n^{\mathcal{O}(\varepsilon^{-2})}$ [150]. Cygan et al. [71] proved that the existence of an EPTAS for Closest String, that is $(1+\varepsilon)$-approximation in time $n^{\mathcal{O}(1)} \cdot f(\varepsilon)$, for any computable function f, is unlikely, as it would imply that $\mathsf{FPT} = \mathsf{W}[1]$, a highly unexpected collapse in the hierarchy of parameterized complexity classes. They also showed that the existence of a PTAS for Closest String with running time $f(\varepsilon)n^{o(1/\varepsilon)}$, for any computable function f, would contradict the Exponential Time Hypothesis. The result of Cygan et al. implies that L_1-Rank-r Approximation over $\mathsf{GF}(2)$ also does not admit EPTAS (unless $\mathsf{FPT} = \mathsf{W}[1]$) already for $r = 1$.

A generalization of Closest String, k-closest strings is also known to admit a PTAS [125, 99]. This problem corresponds to the variant of L_1-Rank-r Approximation over $\mathsf{GF}(2)$, where approximating matrix $\mathbf{L}$ is required to have at most k different columns. However, it is not clear how solution to this special case can be adopted to solve L_1-Rank-r Approximation over $\mathsf{GF}(2)$.

4

Toolbox

In this chapter we give a brief survey of the fundamental techniques that the main body of this thesis relies on. These include dimensionality reduction, coresets and computational algebraic geometry methods.

4.1 Dimensionality Reduction

Dimensionality reduction is a ubiquitous tool for preprocessing matrix data. The motivation behind it is that the input data lies often in a high-dimensional space, however for the purpose of computing approximately, say, a k-means clustering or a k-low rank approximation, working with the original vectors is not necessary. In fact, it is possible to project the original high-dimensional matrix into a much smaller space by multiplying it with a simple randomized *sketch* matrix, while preserving the objective value up to a factor of $(1+\varepsilon)$. The current state-of-the-art results allow for just $\mathcal{O}(k/\varepsilon)$ dimensions [60] for preserving approximately the cost of any orthogonal k-rank projection of the given matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, which in particular preserves the best k-low rank approximation, and just $\mathcal{O}(\log(k/\varepsilon)/\varepsilon^2)$ dimensions for preserving the cost of every k-means clustering [153]. Designing dimensionality reduction routines is a long line of work, where many variants of sketch matrices were studied, allowing for different trade-offs between the approximation guarantee, the target dimension, and the cost of computing the sketch. Moreover, dimensionality reduction was studied for various objectives and norms. The sheer volume of the field does not allow us to give an extensive overview in this work; we refer the reader to the surveys of Woodruff [188] and Mahoney [152] for an overview of this area. Next we give a few definitions and results that we use in our work. Specifically, we explore the applicability of dimensionality reduction methods in Chapter 7 for color-constrained k-Means, where the size of the coreset depends on dimension, and in Chapter 10 for PCA with Outliers, where the dimension enters the exponential part of the

running time.

Definition 4.1 (ε-embedding)**.** *Given a subset $\mathcal{W} \subset R^d$ and $\varepsilon \in (0,1)$, an* ε-embedding *is a matrix $\mathbf{S} \in R^{d\times s}$ for some $s \geq 0$ such that for all $\mathbf{x} \in \mathcal{W}$, we have*

$$\left\|\mathbf{x}^\top \mathbf{S}\right\|_2^2 = (1 \pm \varepsilon) \left\|\mathbf{x}\right\|_2^2 .$$

When $\mathcal{W}$ is a linear subspace we call $\mathbf{S}$ an ε-subspace embedding.

Essentially, an ε-embedding $\mathbf{S}$ is a linear transform, with small $s << d$, providing an approximate isometry over the embedded space $\mathcal{W} \subset R^d$. Observe that in this language, the classical Johnson–Lindenstrauss Lemma states that for any set W of cardinality n, a random rank-k projection achieves an ε-embedding of dimension $s = \Theta(\log n/\varepsilon^2)$. However, in this work we will require embeddings for linear subspaces, preserving approximately the distances between arbitrary points in the same low-dimensional subspace, where the embedding dimension depends only on the dimension of the preserved subspace (and the error parameter ε). To the best of our knowledge, the notion of subspace embedding in numerical linear algebra appeared for the first time due to Sarlos [172], and since then it has emerged as a powerful tool for accelerating various statistical learning procedure like ℓ_p-regression, low-rank approximation, and PCA. The theorem of Indyk and Motwani [120] below provides the bound on embedding dimension s of a Normal Transform for it to be a subspace embedding

Theorem 4.2 (**Normal Transform** [120])**.** *Let $0 < \varepsilon, \delta < 1$ and $\mathbf{S} = \frac{1}{\sqrt{s}}\mathbf{G} \in R^{d\times s}$ where the entries of matrix $\mathbf{G}$ are independent standard normal random variables. Then if $s = \Theta((r + \log(1/\delta))\varepsilon^{-2})$, then for any fixed r-dimensional linear subspace $\mathcal{U} \subset R^d$, with probability at least $1 - \delta$, $\mathbf{S}$ is an ε-subspace embedding.*

The following theorem, observed in [172], says that the solution to an embedded ℓ_2-regression problem provides a good approximate solution to the original regression problem. The proof is an immediate application of ℓ_2-subspace embedding to the ℓ_2-regression problem; for completeness, we give the proof here.

Theorem 4.3 ([172])**.** *Given a matrix $\mathbf{V} \in R^{r\times d}$ and $\mathbf{a} \in R^d$, we have*

$$(1-\varepsilon)\operatorname{dist}^2(\mathbf{a}^\top, \mathbf{V}) \leq \operatorname{dist}^2(\mathbf{a}^\top \mathbf{S}, \mathbf{V}\mathbf{S}) \leq (1+\varepsilon)\operatorname{dist}^2(\mathbf{a}^\top, \mathbf{V}).$$

where $\mathbf{S} \in R^{d\times s}$ is an ε-subspace embedding for subspace spanned by $\operatorname{row}(\mathbf{V})$ *and* $\mathbf{a}$.

Here $\operatorname{dist}^2(\mathbf{a}^\top, \mathbf{V})$ denotes $\min_{\mathbf{x}} \left\|\mathbf{a}^\top - \mathbf{x}^\top \mathbf{V}\right\|_2^2$, which is why we refer to this as an ℓ_2-regression problem. Observe that when $\mathbf{V}$ is an orthonormal basis of an r dimensional subspace of $\mathbb{R}^d$, $\operatorname{dist}^2(\mathbf{a}^\top, \mathbf{V})$ gives precisely the squared Euclidean distance for the point $\mathbf{a}$ to this subspace. This is why Theorem 4.3 is of key importance for the low-rank approximation problem, as we can for example guarantee that the

distance from a point to the optimal subspace is preserved under the embedding. Additionally, if the embedding is, like a normal transform matrix, *oblivious*, i.e. does not depend on the particular matrix $\mathbf{V}$, then for this kind of argument we do not need to know the actual optimal subspace.

Proof. Let (a) $\tilde{\mathbf{x}}_i = \arg\min_{\mathbf{x}} \left\|\mathbf{a}^\top \mathbf{S} - \mathbf{x}^\top \mathbf{V}\mathbf{S}\right\|_2^2$ and (b) $\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}} \left\|\mathbf{a}^\top - \mathbf{x}^\top \mathbf{V}\right\|_2^2$. Since $\mathbf{S}$ is a ε-embedding for $\mathrm{col}([\mathbf{V}^\mathbf{T}|\mathbf{a}])$, we have

$$\begin{aligned}
(1-\varepsilon)\operatorname{dist}^2(\mathbf{a}^\top, \mathbf{V}) &= (1-\varepsilon)\left\|\mathbf{a}^\top - \hat{\mathbf{x}}_i^\top \mathbf{V}\right\|_2^2 \\
&\leq (1-\varepsilon)\left\|\mathbf{a}^\top - \tilde{\mathbf{x}}_i^\top \mathbf{V}\right\|_2^2 \quad (\text{By } (b)) \\
&\leq \left\|\mathbf{a}^\top \mathbf{S} - \tilde{\mathbf{x}}_i^\top \mathbf{V}\mathbf{S}\right\|_2^2 \\
&= \operatorname{dist}^2(\mathbf{a}^\top \mathbf{S}, \mathbf{V}\mathbf{S}).
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
\operatorname{dist}^2(\mathbf{a}^\top \mathbf{S}, \mathbf{V}\mathbf{S}) &= \left\|\mathbf{a}^\top \mathbf{S} - \tilde{\mathbf{x}}_i^\top \mathbf{V}\mathbf{S}\right\|_2^2 \\
&\leq \left\|\mathbf{a}^\top \mathbf{S} - \hat{\mathbf{x}}_i^\top \mathbf{V}\mathbf{S}\right\|_2^2 \quad (\text{By } (a)) \\
&\leq (1+\varepsilon)\left\|\mathbf{a}^\top - \hat{\mathbf{x}}_i^\top \mathbf{V}\right\|_2^2 \\
&= (1+\varepsilon)\operatorname{dist}^2(\mathbf{a}^\top, \mathbf{V}).
\end{aligned}$$

which gives us

$$\begin{aligned}
(1-\varepsilon)\operatorname{dist}^2(\mathbf{a}^\top, \mathbf{V}) &\leq \operatorname{dist}^2(\mathbf{a}^\top \mathbf{S}, \mathbf{V}\mathbf{S}) \\
&\leq (1+\varepsilon)\operatorname{dist}^2(\mathbf{a}^\top, \mathbf{V}).
\end{aligned}$$

□

In fact, one can go further and achieve the same guarantee for all vectors $\mathbf{x} \in \mathbb{R}^r$, i.e. all potential solutions to the regression problem. Next we introduce the concept of Affine Embeddings from [59] that defines this idea formally, and also extends it to multiple regression problems simultaneously.

Definition 4.4 (Affine embedding)**.** *Let $\mathbf{U} \in \mathbb{R}^{r\times d}$ and $\mathbf{A} \in \mathbb{R}^{n\times d}$, then $\mathbf{S} \in \mathbb{R}^{d\times s}$ is an ε-affine embedding for $(\mathbf{U}, \mathbf{A})$ if for every $\mathbf{X} \in \mathbb{R}^{n\times r}$, we have*

$$\|(\mathbf{A} - \mathbf{X}\mathbf{U})\mathbf{S}\|_2^2 = (1 \pm \varepsilon)\|\mathbf{A} - \mathbf{X}\mathbf{U}\|_2^2.$$

Surprisingly, even for such a strong kind of a guarantee, it is possible to achieve the embedding dimension of $\Theta(r \log(1/\delta)\varepsilon^{-2})$. We defer the technical discussion of this construction to Chapter 10, where this notion will be used. The precise result on ε-affine embeddings is stated in Theorem 10.14.

In Chapter 7, we show how to apply dimensionality reduction tools to effectively replace the dimension d by $\mathcal{O}(k/\varepsilon)$ for color-constrained k-MEANS in order to achieve a reduction to an instance of size independent of d. For that we require slightly different properties than above, we do not need that the embedding is oblivious, therefore we can use an SVD-based dimensionality reduction instead of a normal transform embedding. To deal with clustering, it is helpful to employ the notion of a *projection-cost preserving sketch*, defined next. In what follows, we use the results and notation of Cohen et al. [60].

Definition 4.5 (Definition 2 in [60])**.** *For a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, $\mathbf{S} \in \mathbb{R}^{n\times s}$ is a rank k projection-cost preserving sketch with one-sided error $0 \leq \varepsilon < 1$ if for all rank k orthogonal projection matrices $\mathbf{P} \in \mathbb{R}^{n\times n}$*

$$||\mathbf{A} - \mathbf{PA}||_F^2 \leq ||(\mathbf{A} - \mathbf{PA})\mathbf{S}||_F^2 + c \leq (1+\varepsilon)||\mathbf{A} - \mathbf{PA}||_F^2,$$

for some fixed non-negative constant c that may depend on $\mathbf{A}$ and $\mathbf{S}$ but is independent of $\mathbf{P}$.

Before we move on to discussing constructions of projection-cost preserving sketches, let us clarify why this object is of great use for clustering problems. It is well-known that, for a data matrix $\mathbf{A}$, any k-means clustering of the rows of $\mathbf{A}$ may be represented by a particular orthogonal projection matrix $\mathbf{P}$, such that the cost of the clustering is equal to $||\mathbf{A} - \mathbf{PA}||_F^2$, see e.g. Section 2.3 in [60]. Specifically, for any clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ of the rows of $\mathbf{A} \in \mathbb{R}^{n\times d}$ that has rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$, consider the *cluster indicator matrix* $\mathbf{X}^{\mathcal{C}} \in \mathbb{R}^{n\times k}$ where

$$\mathbf{X}^{\mathcal{C}}_{ij} = \begin{cases} 1/\sqrt{|C_j|}, & \text{if the row } \mathbf{a}_i \text{ is assigned to the cluster } C_j, \\ 0, & \text{otherwise,} \end{cases}$$

for $i \in [n]$, $j \in [k]$. Now consider the projection matrix $\mathbf{P}^{\mathcal{C}} = \mathbf{X}^{\mathcal{C}}(\mathbf{X}^{\mathcal{C}})^\top$, by definition of $\mathbf{X}^{\mathcal{C}}$, $\mathbf{P}^{\mathcal{C}}\mathbf{A} = \mathbf{X}^{\mathcal{C}}(\mathbf{X}^{\mathcal{C}})^\top\mathbf{A}$ has its i-th row equal to the mean of all the rows belonging to the cluster that the i-th row of $\mathbf{A}$ belongs to. Thus,

$$||\mathbf{A} - \mathbf{P}^{\mathcal{C}}\mathbf{A}||_F^2 = \sum_{i=1}^{n} ||\mathbf{a}_i - \mathbf{c}_j||_2^2,$$

where $\mathbf{c}_j$ is the mean of the cluster C_j such that $\mathbf{a}_i \in C_j$, which is exactly the cost of the clustering $\mathcal{C}$ provided that the centers are selected optimally as the means of the respective clusters. To see that $\mathbf{P}^{\mathcal{C}}$ is a rank k orthogonal projection matrix, observe that the matrix $\mathbf{X}^{\mathcal{C}}$ consists of k columns that have disjoint support and norm one.

From the discussion above we get that, if $\mathbf{S}$ is a projection-cost preserving sketch for $\mathbf{A}$ with one-sided error ε, then for any clustering $C_1, \ldots, C_k$ of the rows of $\mathbf{A}$, and the corresponding clustering $\tilde{C}_1, \ldots, \tilde{C}_k$ of the rows of $\mathbf{AS}$, it holds that

$$\operatorname{cost}(C_1, \ldots, C_k) \leq \operatorname{cost}(\tilde{C}_1, \ldots, \tilde{C}_k) + c \leq (1+\varepsilon)\operatorname{cost}(C_1, \ldots, C_k).$$

In particular, this holds for any clustering that is subject to a given clustering constraint, for example fair clustering or capacitated clustering. Thus, in fact, projection-cost preserving sketches allow readily to reduce dimension for any constrained variant of Euclidean k-Means, which we will use in Chapter 7.

A variety of constructions provide a projection-cost preserving sketch, with different values of embedding dimension. In particular, it suffices to take a random projection matrix with the embedding dimension $s = \Theta(k/\varepsilon^2)$ (Theorem 12 in [60]). For a better dependence on ε, we will employ a dimensionality reduction scheme based on approximate singular value decomposition, the key result is stated next.

Proposition 4.6 (Theorem 8 in [60])**.** *Let $s = \lceil k/\varepsilon \rceil$. For any $\mathbf{A} \in \mathbb{R}^{n\times d}$ and any orthonormal matrix $\mathbf{S} \in \mathbb{R}^{d\times s}$ satisfying*

$$||\mathbf{A} - \mathbf{ASS}^\top||_F^2 \leq (1+\varepsilon')||\mathbf{A} - \mathbf{A}_s||_F^2,$$

the sketching matrix $\mathbf{S}$ satisfies the conditions of Definition 4.5 with error $(\varepsilon + \varepsilon')$. Here $\mathbf{A}_s$ is the matrix $\mathbf{A}$ projected onto its top s singular vectors.

There is a long line of work providing algorithms to compute this sort of relative approximation to the SVD, we use the algorithm of Boutsidis et al. [33], stating the version appearing in [35].

Proposition 4.7 (Lemma 4 in [35])**.** *Given $\mathbf{A} \in \mathbb{R}^{n\times d}$ of rank ρ, a target rank $2 \leq s < \rho$, and $0 < \varepsilon < 1$, there exists a randomized algorithm that computes an orthonormal matrix $\mathbf{S} \in \mathbb{R}^{d\times s}$ such that*

$$\mathrm{E}||\mathbf{A} - \mathbf{ASS}^\top||_F^2 \leq (1+\varepsilon)||\mathbf{A} - \mathbf{A}_s||_F^2.$$

The proposed algorithm runs in time $\mathcal{O}(nds/\varepsilon)$.

Together, Proposition 4.6 and Proposition 4.7 provide the following theorem, that we will use for dimensionality reduction in our study of constrained clustering problems.

Theorem 4.8. *There is a polynomial-time algorithm that, given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, an integer k and an error parameter $0 < \varepsilon < 1$, computes a matrix $\mathbf{S} \in \mathbb{R}^{d\times s}$ that is a rank k projection-cost preserving sketch of $\mathbf{A}$ with one-sided error ε, where $s = \lceil k/\varepsilon \rceil$.*

4.2 Coresets

Similarly to dimensionality reduction, constructing a coreset from the input data allows to reduce the size of the input while preserving approximately the desired objective. With respect to a clustering problem, informally, given a set of points, its coreset is a small weighted set of points in the same space that for any set of k centers has approximately the same cost of clustering as the original points. This idea can also be extended to low-rank approximation and other matrix problems, by requiring to replace the input matrix by a matrix with few weighted rows, that approximates the original matrix well. However, at least for the standard low-rank approximation problem in Frobenius norm, reducing the number of rows (i.e. constructing a coreset) is equivalent to reducing the number of columns (i.e. performing dimensionality reduction), as the objective is indifferent to transposing the input matrix. While for a clustering problem rows and columns are not interchangeable, thus a task of constructing a small-sized coreset cannot be directly reduced to dimensionality reduction, although we note that dimensionality reduction is in fact used in a number of coreset constructions as an intermediate step. For this section, we focus on coresets for clustering problems.

Now we proceed to the formal definition of a coreset. We are interested in coresets for both Euclidean space and general metric spaces, thus we use the notations of the general k-Clustering problem. Recall that we denote the space by $(\mathcal{X}, \mathrm{dist})$, where the set of the input points $P \subset \mathcal{X}$ is given, and the set of potential centers is denoted as $F \subset \mathcal{X}$. Additionally, consider a *weight function* $w : P \to \mathbb{R}_{\geq 0}$, then the weighted set of points $W \subset P \times \mathbb{R}_{\geq 0}$ given by the weight function w is the set

$$\{(p, w(p)) \mid p \in P \text{ and } w(p) > 0\}.$$

Now, for a set of centers $C \subset F$, recall that

$$\mathrm{cost}(P, C) = \sum_{p \in P} \mathrm{dist}(p, C),$$

and we use similar notation for weighted points,

$$\mathrm{wcost}(W, C) = \sum_{(p,w) \in W} w \cdot \mathrm{dist}(p, C).$$

Now we are ready to define a coreset.

Definition 4.9 (Coreset for k-Clustering)**.** *A coreset for a given instance of k-Clustering is a set of weighted points $W \subseteq P \times \mathbb{R}_{\geq 0}$ such that for every set of centers $C \subset F$ of size k,*

$$(1 - \varepsilon) \cdot \mathit{cost}(P, C) \leq \mathit{wcost}(W, C) \leq (1 + \varepsilon) \cdot \mathit{cost}(P, C).$$

We say that the size of a coreset is the cardinality of the set W.

Observe that if $w(p) = 1$ for all $p \in P$, then for the weighted set W given by w, $\text{wcost}(W, C) = \text{cost}(P, C)$ for all sets of centers C, in this way the original set of points is a trivial coreset for itself. Also note that a coreset for a general discrete metric space does not immediately give a coreset for the Euclidean variant of k-Clustering, as in the former the set of potential centers is finite and given in the input, while in the latter the set of potential centers is the whole space $\mathbb{R}^d$.

Apart from the advantage that any clustering algorithm can be applied to the coreset to efficiently retrieve a nearly optimal clustering, coresets are also vital in the streaming setting for the following reasons. First, they take less space compared to the original data, and most coreset constructions are simple and require only a few passes over the data. Second, an important property of the coresets is that they are *composable*: If the input P is partitioned into subsets $P_1, \ldots, P_t$, and for each subset P_i a coreset W_i is computed, then the union of the coresets $\bigcup_{i=1}^{t} W_i$ is a coreset for the whole input P. From Definition 4.9, this can be trivially seen to be true; although in the literature there are other definitions of coresets that not necessarily have this property. Composability allows to split a large body of data that cannot be handled at one place to several machines, such that each machine computes the coreset of its portion independently, and then just collecting the outputs from each machine gives an accurate summary of the whole input. Additionally, a coreset that is composable can be computed in the merge-and-reduce fashion, allowing for one-pass computation of the coreset in streaming. This concept goes back to [22], and to the best of our knowledge was first used for the design of geometrical algorithms in the streaming model in [2]. First, the coreset for a small portion of the input that fits into the small memory is computed. Then the coreset for the second small portion is computed. Then these two coresets are merged and reduced by computing a smaller coreset from their union. In this way, the algorithm proceeds along the corresponding tree, such that at each node only a small-sized coreset, resulting from the direct application of the coreset construction algorithm, is stored. The MapReduce programming model used for processing big data is strongly related to the above input partitioning methods, as well as its popular implementations, e.g. [186].

Over the years, researchers have paid increasing attention to the design of coreset construction algorithms to optimize the coreset size. Many different schemes have been proposed for coreset construction. In the earlier works considering the Euclidean space, exponential grids and similar structures have been used that led to coresets with size depending exponentially on the dimension d [110, 109, 92]. To give the intuition to [110], the work that coined the term "coreset", consider an instance of Euclidean k-Means and a set of centers C that achieves a constant approximation of the k-Means objective. Around each center c from C, construct the exponential grid: it partitions the space around the center C into cubes of side length that grows exponentially as the cubes become further from c, in such a way that the side length is approximately ε times the distance to c. Then, because of the triangle inequality, all points of P inside the particular cube can be condensed into a single point, and

the error resulting from that can be attributed to the error that each point pays in the approximate solution, i.e. its distance to c, times ε. By carefully cutting the grid at the close and far ranges of c, one can get an order of $\log n$ levels of the grid. In the end, this results in $\mathcal{O}(\varepsilon^{-d} \log n)$ cells around each center, and the total coreset size is $\mathcal{O}(k\varepsilon^{-d} \log n)$.

We note that the majority of known coreset constructions partitions in a way the space around the centers of an approximate solution, however more advanced coreset constructions use sampling inside the resulting regions to avoid exponential dependence on d. The first construction of this form was given by Chen [53], who showed a coreset of size polynomial in k, d, ε, $\log n$ in the Euclidean space, and also a coreset of size polynomial in k, ε and $\log n$ in general discrete metric spaces. Our coreset result in Chapter 7 stems, in fact, from this construction, and more details about it are given in that chapter. Subsequently, the size of the coreset has been further improved [139, 83]. Finally, the dependence on d was removed for both k-Means and k-Median in $\mathbb{R}^d$ in [84] and [177], respectively, resulting in coresets of size $(k/\varepsilon)^{O(1)}$. See also [20, 37, 116] for recent improvements. For general discrete metric spaces, the best-known upper bound on coreset size is $\mathcal{O}((k \log n)/\varepsilon^2)$ [83] and the lower bound is known to be $\Omega((k \log n)/\varepsilon)$ [16]. Thus, in this case the dependence on $\log n$ is unavoidable.

4.2.1 Coresets for Constrained Clustering

The above covers the situation with coresets for unconstrained clustering problems, however, when we move to constrained clustering, the story is different. First of all, none of the results for the unconstrained problem are immediately applicable as, per Definition 4.9, a coreset preserves the cost with respect to every possible set of centers, where cost is computed as the sum of distances from every point to its closest center. Since a constrained version of the clustering problem allows only certain partitions of the points into clusters, the cost in this version might be very different than in the unconstrained assignment to the closest centers. Second, even if we try to tweak Definition 4.9 to represent the cost only of the allowed clusterings, some issues arise with the definition itself. Specifically, given a family $\mathcal{C}$ of the allowed partitions of the points, it would be natural to replace $\mathrm{cost}(P, C)$ in Definition 4.9 with its constrained variant, $\mathrm{cost}_{\mathcal{C}}(P, C)$ that only allows assignments from P to C that are allowed by $\mathcal{C}$. However, doing the same thing with the definition of weighted cost $\mathrm{wcost}(W, C)$ would not be sound, as replacing points of P by a few weighted points of W might lose the information required to check that an assignment satisfies $\mathcal{C}$. Moreover, even if for a particular $\mathcal{C}$, a weighted subset W has enough information to check whether $\mathcal{C}$ is satisfied, an important property that the coresets are composable will not necessarily hold. Take for an example a special case of (α, β)-Fair Clustering, where the input points are colored in two colors, red and blue, and each cluster must have exactly the same number of red and blue points. Then it might be the case that even if the whole input has a very low cost of clustering with a given set C,

partitioning it into two parts might not yield any feasible clustering at all, if the parts do not contain the same number of red and blue points. Even if we demand that the parts must be balanced too, failing to keep the whole cluster in the same part might lead to the optimal cost of clustering in one part being very high for the need to balance the resulting cluster by merging parts of two different clusters in the optimal solution. And if the cost of clustering with respect to C is high in a part, the ε-factor error will also be too high compared to the cost of the optimal clustering as a whole. For the details of the last example, see [173].

Thus we arrive to a stronger definition of a coreset for constrained clustering problems, introduced in [173]. First, we make formal the intuition of a coloring constraint, introduced in Subsection 3.1.4. For a clustering problem with k centers and ℓ groups $P_1, \ldots, P_\ell$, identified among the whole set of points P, a *coloring constraint* is a $k \times \ell$ matrix $\mathbf{M}$ having non-negative integer entries. We say that a clustering $C_1, \ldots, C_k$ of P satisfies the coloring constraint $\mathbf{M}$ if for each $i \in [k], j \in [\ell]$, $|C_i \cap P_j| = \mathbf{M}_{ij}$, that is, exactly $\mathbf{M}_{ij}$ points from the group P_j are assigned into the cluster C_i. Next, we have the following observation, which was noted in [173, 115].

Proposition 4.10. *For an instance of* (α, β)-Fair Clustering *and a fixed set* C *of* k *centers, there exists a collection of coloring constraints* $\mathcal{M}$ *with the following property. A clustering* $C_1, \ldots, C_k$ *of* P *is* (α, β)*-fair if and only if there exists a coloring constraint* $\mathbf{M} \in \mathcal{M}$ *such that the clustering* $C_1, \ldots, C_k$ *satisfies* $\mathbf{M}$*.*

The proof is by simply observing that knowing how many points from each group ends up in each cluster, which is exactly represented by a coloring constraint, allows us to verify that the clustering is fair. Thus listing all corresponding $k \times \ell$ matrices $\mathbf{M}$ represents exactly the fairness conditions, in the sense of Proposition 4.10. In fact, capacitated clustering can also be reduced to a collection of coloring constraints in the same fashion. Given an upper bound U on cluster size, we simply set $\ell = 1$, $P_1 = P$, and the collection of those matrices $\mathbf{M} \in \mathbb{Z}_{\geq 0}^{k \times 1}$ where each entry is at most U. Similarly, all constraints listed under Subsection 3.1.4 can be expressed by coloring constraints.

Now, in the stronger definition, a coreset is required to preserve the optimal clustering cost w.r.t. all coloring constraints. Hence, it also preserves the optimal fair clustering cost, the optimal capacitated clustering cost, or any other color constraint. Next, we formally define the cost of a clustering w.r.t. a set of centers and a coloring constraint. Suppose we are given a weight function $w : P \to \mathbb{R}_{\geq 0}$. Let $W \subseteq P \times \mathbb{R}$ be the set of pairs $\{(p, w(p)) \mid p \in P \text{ and } w(p) > 0\}$. For a set of centers $C = \{c_1, \ldots, c_k\}$ and a coloring constraint $\mathbf{M}$, $\mathrm{wcost}(W, \mathbf{M}, C)$ is the minimum value $\sum_{p \in P, c_i \in C} \psi(p, c_i) \cdot \mathrm{dist}(p, c_i)$ over all assignments $\psi : P \times C \to \mathbb{R}_{\geq 0}$ such that

(i) For each $p \in P$, $\sum_{c_i \in C} \psi(p, c_i) = w(p)$.

(ii) For each $c_i \in C$ and group $1 \leq j \leq \ell$, $\sum_{p \in P_j} \psi(p, c_i) = \mathbf{M}_{ij}$.

For k-means, $\mathrm{wcost}(W, \mathbf{M}, C)$ is defined in the same way except it is the minimum value $\sum_{p \in P, c_i \in C} \psi(p, c_i) \cdot \mathrm{dist}(p, c_i)^2$. If there is no such assignment ψ, $\mathrm{wcost}(W, \mathbf{M}, C) = \infty$. When $w(p) = 1$ for all $p \in P$, we simply denote W by P and $\mathrm{wcost}(W, \mathbf{M}, C)$ by $\mathrm{cost}(P, \mathbf{M}, C)$. From the above definitions of cost and wcost, we derive a generalization of Definition 4.9. We call this object a universal coreset, as it is required to preserve optimal clustering cost w.r.t. all coloring constraints.

Definition 4.11 (Universal coreset)**.** *A universal coreset for a clustering objective is a set of weighted points $W \subseteq P \times \mathbb{R}$ such that for every set of centers C of size k and any coloring constraint $\mathbf{M}$,*

$$(1 - \varepsilon) \cdot \mathit{cost}(P, \mathbf{M}, C) \leq \mathit{wcost}(W, \mathbf{M}, C) \leq (1 + \varepsilon) \cdot \mathit{cost}(P, \mathbf{M}, C).$$

Schmidt et al. [173] and subsequently Huang et al. [115] designed deterministic algorithms in $\mathbb{R}^d$ that construct universal coresets whose sizes exponentially depend on d, employing the exponential grid construction by [110]. To remove this exponential dependency on d, Schmidt et al. [173] proposed an interesting open question whether it is possible to use random sampling for construction of fair coresets. Huang et al. [115] also suggested the same open question. Besides, Huang et al. asked whether it is possible to achieve a similar size bound as in the vanilla setting. In Chapter 7 we resolve this question affirmatively, and provide a novel analysis of the construction by Chen [53] that achieves a coreset for any color constrained clustering of size $\mathsf{poly}(\ell, k, \log n, \varepsilon)$, where ℓ is the number of groups/colors (in the case the groups are disjoint), in general discrete metric spaces. For Euclidean k-Median and k-Means, the size of the coreset also depends polynomially on the dimension d, although for k-Means we show how to employ dimensionality reduction to get rid of the dependence on d. Specifically for capacitated clustering, Cohen-Addad and Li [66] gave earlier similar results, our analysis for the general color constrained clustering problem follows their approach.

4.3 Algebraic Geometry

In Chapters 9 and 10, we study the PCA with Outliers problem using heavily the tools of computational algebraic geometry, and we outline the basics of the approach in this section. On the highest level, the idea is to bound the inherent geometric complexity of the problem, which is much smaller in small-dimensional spaces than the space-oblivious combinatorial complexity. To the best of our knowledge, the first similar usage of these tools for the design of geometrical algorithms is due to Inaba et al. [118], who presented a $n^{\mathcal{O}(dk)}$-time algorithm for k-Means in $\mathbb{R}^d$. The intuition of this approach is as follows, on the example of the k-Means algorithm of [118]. We parameterize the unknown object by real-valued variables, in this case, we take a variable for each coordinate of each unknown cluster center, dk variables in total. Then, we introduce a family of polynomial conditions such that their signs

characterize completely the desired object: for each input point $\mathbf{p}$ and each pair of the unknown centers $\mathbf{c}_i$ and $\mathbf{c}_j$, we write the condition that $||\mathbf{p}-\mathbf{c}_i||_2^2 < ||\mathbf{p}-\mathbf{c}_j||_2^2$, i.e. that $\mathbf{p}$ is closer to $\mathbf{c}_i$ than to $\mathbf{c}_j$. Knowing the signs of these conditions for all pairs of centers gives uniquely the index of the closest center to $\mathbf{p}$, and when we know this for all points, we know the actual clustering. Thus, for any given values of signs of these polynomials, we can construct the corresponding clustering and compute its cost. Finally, a generic algebraic geometry theorem bounds the number of different values of signs of the polynomial family, based on the number of variables and the number of polynomial conditions. This provides the bound of $n^{\mathcal{O}(dk)}$ possible clusterings to consider. In what follows, we give technical definitions and the statement of the above-mentioned theorem, that allow to formalize the intuition above. We use notations and results from the seminal book of Basu et al. [19].

We denote the ring of polynomials in variables $X_1, \ldots, X_d$ with coefficients in $\mathbb{R}$ by $\mathbb{R}[X_1, \ldots, X_d]$. By saying that an algebraic set V in $\mathbb{R}^d$ is defined by a polynomial $Q \in \mathbb{R}[X_1, \ldots, X_d]$, we mean that $V = \{\mathbf{x} \in \mathbb{R}^d \mid Q(\mathbf{x}[1], \ldots, \mathbf{x}[d]) = 0\}$. For a set of s polynomials $\mathcal{P} = \{P_1, \ldots, P_s\} \subset \mathbb{R}[X_1, \ldots, X_d]$, a sign condition is specified by a sign vector $\sigma \in \{-1, 0, +1\}^s$, and the sign condition is non-empty over V with respect to $\mathcal{P}$ if there is a point $\mathbf{x} \in V$ such that

$$\sigma = (\text{sign}(P_1(\mathbf{x})), \ldots, \text{sign}(P_s(\mathbf{x}))),$$

where $\text{sign}(z)$ is the sign function on real numbers defined as

$$\text{sign}(z) = \begin{cases} 1, & \text{if } z > 0, \\ 0, & \text{if } z = 0, \\ -1, & \text{if } z < 0 \end{cases}$$

for $z \in \mathbb{R}$.

The *realization space* of $\sigma \in \{-1, 0, +1\}^s$ over V is the set

$$R(\sigma) = \{\mathbf{x} \mid \mathbf{x} \in V, \sigma = (\text{sign}(P_1(\mathbf{x})), \ldots, \text{sign}(P_s(\mathbf{x})))\}.$$

If $R(\sigma)$ is not empty then each of its non-empty semi-algebraically connected (which is equivalent to just connected on semi-algebraic sets as proven in [19], Theorem 5.22) components is a *cell* of $\mathcal{P}$ over V.

For an algebraic set V its real dimension is the maximal integer d' such that there is a homeomorphism of $[0, 1]^{d'}$ in V. Naturally, if $V \subset \mathbb{R}^d$, then $d' \leq d$.

The following theorem from [19] gives an algorithm to compute a point in each cell of $\mathcal{P}$ over V.

Theorem 4.12 ([19], Theorem 13.22)**.** *Let V be an algebraic set in $\mathbb{R}^d$ of real dimension d' defined by $Q(X_1, \ldots, X_d) = 0$, where Q is a polynomial in $\mathbb{R}[X_1, \ldots, X_d]$ of degree at most b, and let $\mathcal{P} \subset \mathbb{R}[X_1, \ldots, X_d]$ be a finite set of s polynomials with each $P \in \mathcal{P}$ also of degree at most b. Let D be a ring generated by the coefficients*

of Q and the polynomials in $\mathcal{P}$. There is an algorithm which takes as input Q, d' and $\mathcal{P}$ and computes a set of points meeting every non-empty cell of V over $\mathcal{P}$. The algorithm uses at most $s^{d'}b^{\mathcal{O}(d)}$ arithmetic operations in D.

Going back to the k-Means algorithm, the algebraic set V is simply $\mathbb{R}^{k\times d}$, so Q is a zero polynomial. Consider variables $X_1, \ldots, X_{dk}$, where the first d variables denote the coordinates of the center $\mathbf{c}_1$, the next d variables are the coordinates of $\mathbf{c}_2$, and so on. Then the set of polynomials $\mathcal{P} \subset \mathbb{R}^{dk}$ consists of polynomials $P_p^{i,j}(X_1, \ldots, X_{dk})$ for each input point p and each pair $1 \leq i < j \leq k$, where

$$P_p^{i,j}(X_1, \ldots, X_{dk}) = \sum_{t=1}^{d} (\mathbf{p}[t] - X_{(i-1)d+t})^2 - \sum_{t=1}^{d} (\mathbf{p}[t] - X_{(j-1)d+t})^2,$$

corresponding to the value $\|\mathbf{p} - \mathbf{c}_i\|_2^2 - \|\mathbf{p} - \mathbf{c}_j\|_2^2$. The size of $\mathcal{P}$ is $n\binom{k}{2}$, and each polynomial has degree two. Thus, by Theorem 4.12, there is an algorithm running in time $(n\binom{k}{2})^{dk}2^{\mathcal{O}(d)} = n^{\mathcal{O}(dk)}$ that outputs a point from each non-empty cell of V over $\mathcal{P}$. Thus, evaluating $\mathcal{P}$ on each of these points provides all non-empty sign conditions of $\mathcal{P}$, from which we can construct all possible clusterings where each point goes to the closest center, resulting in an $n^{\mathcal{O}(dk)}$-time algorithm for k-Means. Our parameterization for PCA with Outliers presented in Chapter 9 follows a similar approach, although the polynomial system and the underlying algebraic set is somewhat more complicated, as we parameterize all possible r-dimensional subspaces.

On the practical side, we note that a number of routines from [19] is implemented in the SARAG library [44].

Part II

Clustering as an Editing Problem

5

Parameterized L_p-k-Clustering

In this chapter we investigate how the complexity of L_p-k-Clustering depends on the cost of clustering D. First we recall the problem's definition.

L_p-k-Clustering

Input: A multiset X of n vectors in $\mathbb{Z}^d$, a positive integer k, and a nonnegative number D.

Task: Decide whether there is a partition of X into k clusters $\{C_i\}_{i=1}^k$ and k vectors $\{\mathbf{c}_i\}_{i=1}^k$ in $\mathbb{R}^d$ such that

$$\sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \mathrm{dist}_p(\mathbf{x}, \mathbf{c}_i) \leq D.$$

Note that the vector set X (like the row set of a matrix) can contain many equal vectors. Here for $0 < p < \infty$, dist_p is the p-th power of the Minkowsky L_p-norm, i.e.

$$\mathrm{dist}_p(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_p^p = \sum_{i=1}^{d} |\mathbf{x}[i] - \mathbf{y}[i]|^p.$$

We also consider the natural extension of the above to the cases $p = 0$ and $p = \infty$,

$$\mathrm{dist}_0(\mathbf{x}, \mathbf{y}) = |\{i \in \{1, \ldots, d\} \mid \mathbf{x}[i] \neq \mathbf{y}[i]\}|,$$
$$\mathrm{dist}_\infty(\mathbf{x}, \mathbf{y}) = \max_{i \in \{1, \ldots, d\}} |\mathbf{x}[i] - \mathbf{y}[i]|.$$

Let us remark that we consider the situation where the given vectors X have integral coordinates, while cluster centers are not necessarily from X. Moreover, coordinates of the centers can be any real values. Thus, in the case $p = 2$, L_p-k-Clustering

is exactly the well-known k-Means problem, restricted to integer-valued instances. We believe restricting the input to integral values is the most natural model for studying complexity of L_p-k-Clustering with respect to the parameter D. Considering D as a parameter only makes sense if the input values are suitably discretized. Imagine input vectors could have arbitrary real-valued (or rational-valued) entries, then for a given instance it is always possible to scale the values down by the same factor such that the cost of an optimal clustering is arbitrarily small, but the structure of the instance is completely preserved. Thus the restriction to integer values in our study is a natural discretization of the problem. It allows the parameter D to bear deep structural significance, as our results demonstrate.

It might be noted that the regime where the cost of clustering D is small compared to the number of points n, is quite special. Indeed, if the cost of clustering is at most D, then there are but a few points that are not equal to the respective cluster centers. Thus, the problem we study has the spirit of an editing problem: check whether a given instance is close to a "structured" one, where in our case a "structured" instance has at most k distinct points, and closeness is measured via the sum of L_p-distances. Editing problems are extensively studied in the parameterized algorithms literature, ranging from the vast area of graph modification (see e.g. a recent survey by [68]) to studies very close to ours, like the Consensus Patterns algorithm by Marx [156], and the study of Binary r-Means by Fomin et al. [90] that is essentially a special case of our L_0-k-Clustering problem, where the input is binary. And still, even in this highly structured regime, our results show a very intricate picture: for instance, for L_p-k-Clustering parameterized just by D, we provide a non-trivial FPT algorithm in the case $0 < p \leq 1$. While on the other hand, conditionally, the same scheme could not lead to an analogous algorithm in the case $p = 2$, and there could not be any FPT algorithm at all in the cases $p = 0$ and $p = \infty$, under the widely believed assumption FPT $\neq$ W[1]. Finally we believe that studying L_p-k-Clustering with respect to the parameter D is an essential question provided the notorious hardness of the problem. Recall that for the combination of the two other natural parameters, the dimension d and the number of clusters k, only a $O(n^{dk+1})$ algorithm of Inaba et al. is known [118], and the hardness result by Cohen-Addad et al. in [63] serves as a strong indication that a better algorithm might not exist.

The main algorithmic result of this chapter is the following theorem.

Theorem 5.1. *L_p-k-Clustering is solvable in time $2^{\mathcal{O}(D \log D)}(nd)^{\mathcal{O}(1)}$ for every $p \in (0, 1]$.*

Thus L_p-k-Clustering when parameterized by D is fixed-parameter tractable (FPT) for Minkowski distance dist_p of order $0 < p \leq 1$. In the first step of our algorithm we use color coding to reduce solution of the problem to the L_p-Cluster Selection problem, which we find interesting on its own. In L_p-Cluster Selection we have t groups of weighted vectors and the task is to select exactly one vector from each group such that the weighted cost of the composite cluster is at most D. More formally, the problem is defined as follows.

L_p-Cluster Selection

Input: A set of m vectors $X \subset \mathbb{Z}^d$ given together with a partition $X = X_1 \cup \cdots \cup X_t$ into t disjoint sets, a weight function $w : X \to \mathbb{Z}_{>0}$, and a nonnegative number D.

Task: Decide whether it is possible to select exactly one vector $\mathbf{x}_i$ from each set X_i such that the total cost of the composite cluster formed by $\mathbf{x}_1, \ldots, \mathbf{x}_t$ is at most D:

$$\min_{\mathbf{c} \in \mathbb{R}^d} \sum_{i=1}^{t} w(\mathbf{x}_i) \cdot \mathrm{dist}_p(\mathbf{x}_i, \mathbf{c}) \leq D.$$

The L_p-Cluster Selection problem is closely related to variants of the well-known Consensus Pattern problem. Namely, for the Hamming distance, the definition of L_0-Cluster Selection nearly coincides with the Colored Consensus Strings with Outliers problem studied in [31], only in the latter the alphabet is assumed to be of constant size.

Informally (see Theorem 5.10 for the precise statement), our reduction shows that if the distance norm satisfies some specific properties (which dist_p satisfies for all p) and if L_p-Cluster Selection is FPT parameterized by D, then so is L_p-k-Clustering. Therefore, in order to prove Theorem 5.1, all we need is to show that L_p-Cluster Selection is FPT parameterized by D when $p \in (0, 1]$. This is the most difficult part of the proof. Here we invoke the theorem of Marx [156] on the number of subhypergraphs in hypergraphs of bounded fractional edge cover.

Superficially, the general idea of the proof of Theorem 5.1 is similar to the idea behind the algorithm for Binary r-Means by [90]. In both cases, the classical color coding technique of Alon et al. [10] is used as a preprocessing step. However, the further steps in [90] strongly exploit the fact that the data is binary. As we will see in Theorem 5.2, the existence of an FPT algorithm for L_0-k-Clustering is highly unlikely. Thus the reductions from [90] cannot be applied in our case, and we need a new approach.

More precisely, for clustering in L_0 we prove the following theorem.

Theorem 5.2. *L_0-k-Clustering parameterized by $d + D$ and L_0-Cluster Selection parameterized by $d + t + D$ are W[1]-hard.*

In particular, this means that up to a widely-believed assumption in complexity that FPT $\neq$ W[1], Theorem 5.2 rules out algorithms solving L_0-k-Clustering in time $f(d, D) \cdot n^{\mathcal{O}(1)}$ and algorithms solving L_0-Cluster Selection in time $g(t, d, D) \cdot n^{\mathcal{O}(1)}$ for any functions $f(d, D)$ and $g(t, d, D)$. A similar hardness result holds for the dist_∞ version.

dist_p	L_p-k-Clustering	L_p-Cluster Selection
$p = 0$	W[1]-hard param. $d + D$ [Thm 5.2] NP-c for $k = 2$ [82]	W[1]-hard param. $d + t + D$ [Thm 5.2]
$0 < p \leq 1$	$2^{\mathcal{O}(D \log D)}(nd)^{\mathcal{O}(1)}$ [Thm 5.1]	$2^{\mathcal{O}(D \log D)}(nd)^{\mathcal{O}(1)}$ [Thm 5.14]
$p = 1$	NP-c for $k = 2$ [82] NP-c for $d = 2$ [157]	W[1]-hard param. $t + d$ [Thm 5.19]
$1 < p < +\infty$		W[1]-hard param. $t + D$ [Thm 5.5]
$p = 2$	FPT param. $d + D$ [Thm 5.4] NP-c for $k = 2$ [9] NP-c for $d = 2$ [151]	FPT param. $d + D$ [Thm 5.4]
$p = \infty$	W[1]-hard param. D [Thm 5.3] NP-c for $k = 2$ [Thm 5.29]	W[1]-hard param. $t + D$ [Thm 5.3]

Table 5.1: Complexity of L_p-k-Clustering and L_p-Cluster Selection.

Theorem 5.3. *L_∞-k-Clustering parameterized by D and L_∞-Cluster Selection parameterized by $t + D$ are W[1]-hard.*

This naturally brings us to the question: What happens with L_p-k-Clustering for $p \in (1, \infty)$, especially for the Euclidean distance, that is, the case $p = 2$. Unfortunately, we are not able to answer this question when the parameter is D only. However, we can prove the following.

Theorem 5.4. *L_2-k-Clustering and L_2-Cluster Selection are FPT when parameterized by $d + D$.*

Thus in particular, Theorem 5.4 implies that L_2-k-Clustering is FPT parameterized by $d + D$. On the other hand, we prove that

Theorem 5.5. *L_p-Cluster Selection is W[1]-hard for every $p \in (1, \infty)$ when parameterized by $t + D$.*

In particular, Theorem 5.5 yields that the approach we used to establish the tractability (with parameter D) of L_p-k-Clustering for $p = 1$ will not work for $p > 1$.

We summarize our and previously known algorithmic and hardness results for L_p-k-Clustering and L_p-Cluster Selection with different values of p in Table 5.1. Observe that Theorem 5.10 works also in the setting where possible cluster centers are restricted to be from a set given in the input, and so do our algorithmic Theorems 5.1 and 5.4 since L_p-Cluster Selection is trivially solvable in polynomial time in this setting.

The remaining part of this chapter is organized as follows. In Section 5.1 we prove Theorem 5.10 which provides us with FPT Turing reduction from L_p-k-Clustering to L_p-Cluster Selection. Theorem 5.10 appears to be a handy tool to establish tractability of L_p-k-Clustering. In Section 5.2 we collect the results on clustering with the distance dist_p for $p \in (0, 1]$. In particular, in Subsection 5.2.1, we prove

Theorem 5.1, the main algorithmic result of this work, stating that when $p \in (0, 1]$, L_p-k-Clustering and L_p-Cluster Selection admit FPT algorithms with parameter D. In Subsection 5.2.2 we complement the algorithmic upper bounds with lower bounds by proving that L_1-Cluster Selection is W[1]-hard when parameterized by $t + d$ (Theorem 5.19). In Section 5.3, we consider the case $p = 0$ and prove Theorem 5.2 establishing W[1]-hardness of L_0-k-Clustering and L_0-Cluster Selection. Section 5.4 is devoted to the case $p = \infty$. Here we establish two hardness results about L_∞-k-Clustering: W[1]-hardness when parameterized by D and NP-hardness in the case $k = 2$. In Section 5.5, we look at the case $p \in (1, \infty)$, with the particular emphasis on the most classical case $p = 2$. We show that when $d + D$ is the parameter, then L_2-Cluster Selection and L_2-k-Clustering are FPT. We also show that L_p-Cluster Selection is W[1]-hard when parameterized by $t + D$ for all $p \in (1, \infty)$. We conclude with open problems in Section 5.6.

5.1 From L_p-k-Clustering to Cluster Selection

In this section we present a general scheme for obtaining an FPT algorithm parameterized by D, which is later applied to various L_p-distances. Note that the results of this section hold in fact for any spaces and distances. However, we state them primarily for vectors in $\mathbb{Z}^d$ under dist_p, as we aim to use the results specifically for this case in the following sections.

First, we formalize the following intuition: there is no reason to assign equal vectors to different clusters.

Definition 5.6 (Initial cluster and regular partition)**.** *For a multiset of vectors X, an inclusion-wise maximal multiset $I \subset X$ such that all vectors in I are equal is called* an initial cluster*.*

We say that a clustering $\{C_1, \ldots, C_k\}$ of X is regular *if for every initial cluster I there is a $i \in \{1, \ldots, k\}$ such that $I \subset C_i$.*

Now we prove that it suffices to look only for regular solutions.

Proposition 5.7. *Let (X, k, D) be a yes-instance to L_p-k-Clustering. Then there exists a solution for (X, k, D) which is a regular clustering.*

Proof. Let us assume that the instance (X, k, D) has a solution. There are k clusters $\{C_i\}_{i=1}^k$ and k vectors $\{\mathbf{c}_i\}_{i=1}^k$ in $\mathbb{R}^d$ such that $\sum_{i=1}^k \sum_{x \in C_i} \mathrm{dist}(\mathbf{x}, \mathbf{c}_i) \leq D$. Note that for every $\mathbf{x} \in C_j$, $\mathrm{dist}(\mathbf{x}, \mathbf{c}_j) \geq \min_{1 \leq i \leq k} \mathrm{dist}(\mathbf{x}, \mathbf{c}_i)$. So if we consider a new clustering $\{C'_1, \ldots, C'_k\}$ with the same centers, where C'_j are all vectors from X for which $\mathbf{c}_j$ is the closest center, the total distance does not increase. If we also break ties in favor of the lower index, then for any initial cluster I the same center $\mathbf{c}_i$ will be the closest, and all vectors from I will end up in C'_i, so $\{C'_1, \ldots, C'_k\}$ is a regular clustering. □

From now on, we consider only regular solutions.

Definition 5.8 (Simple and composite clusters)**.** *We say that a cluster C is* simple *if it is an initial cluster. Otherwise, the cluster is* composite*.*

Next we state a property of k-Clustering with a particular distance, which is required for the algorithm. Intuitively, each unique vector adds at least some constant to the cluster cost.

Definition 5.9 (α-property)**.** *We say that a distance has the* α-property *for some $\alpha > 0$ if for any s the cost of any composite cluster which consists of s initial clusters is at least $\alpha(s-1)$.*

In the subsequent sections we show that the α-property holds for all the distance measures for which we present algorithmic results. Namely, dist_p has the α-property with a certain constant α for each $p \in [0,1] \cup \{2, \infty\}$. Analogously to the case $p = 2$, one can show that it holds for all other values of p between 1 and ∞ as well, although we will not require this fact for the results of this chapter.

The L_p-Cluster Selection problem defined in the introduction is a key subroutine in our algorithm. In some cases the problem is solvable trivially, but it presents the main challenge for our main algorithmic result in the L_1 distance. The intuition to the weight function in the definition of L_p-Cluster Selection is that it represents sizes of initial clusters, that is, how many equal vectors are there in each if the initial clusters.

We also need a procedure to enumerate all values of the possible cluster costs that are at most D, with respect to an optimally selected cluster center. It may not be straightforward since not all distances in our consideration are integral. So for the purpose of stating Theorem 5.10 in the general case, we assume that the set of all possible optimal cluster costs which are less than D is also given in the input. For the L_p-distances we consider, in the respective algorithmic theorems we show how to provide this set without raising any additional assumptions or increasing the running time. Now we are ready to state the result formally.

Theorem 5.10. *Assume that the α-property holds, L_p-Cluster Selection is solvable in time $\Phi(m, d, t, D)$, where Φ is a non-decreasing function of its arguments, and we are given the set $\mathcal{D}$ of all possible optimal cluster costs which are at most D. Then L_p-k-Clustering is solvable in time*

$$2^{\mathcal{O}(D \log D)}(nd)^{\mathcal{O}(1)}|\mathcal{D}|\Phi(n, d, 2D/\alpha, D).$$

Proof. By the α-property, in any solution there are at most D/α composite clusters, since each contains at least two initial clusters. Moreover, there are at most $2D/\alpha$ initial clusters in all composite clusters.

Thus by Proposition 5.7, solving L_p-k-Clustering is equivalent to selecting at most $T := \lceil 2D/\alpha \rceil$ initial clusters and grouping them into composite clusters such that the total cost of these clusters is at most D. We design an algorithm which, taking as a subroutine an algorithm for L_p-Cluster Selection, solves L_p-k-Clustering. The algorithm is sketched in Figure 5.2, and an example is shown in Figure 5.1.

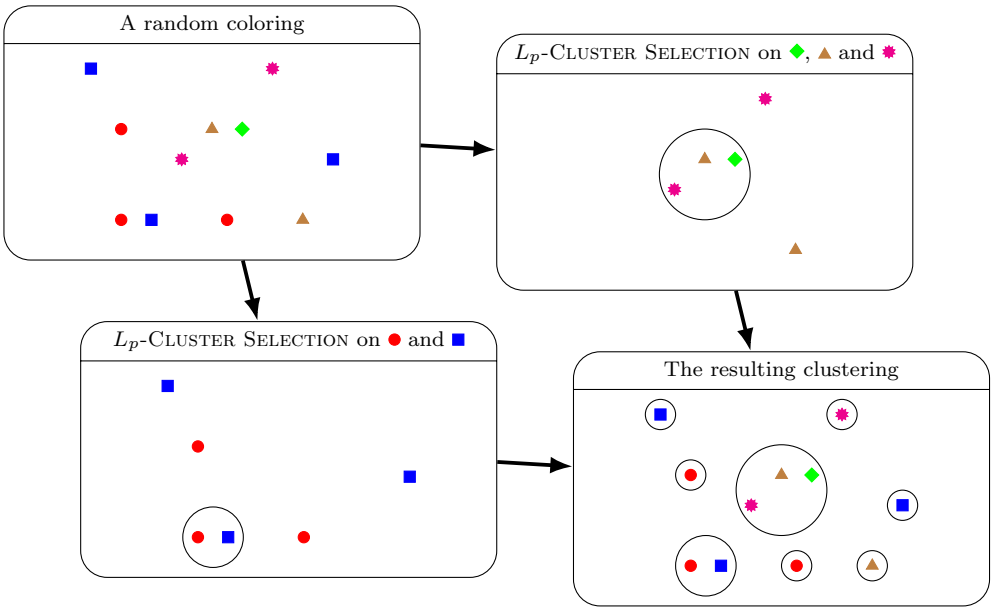


Figure 5.1: An illustration to the algorithm in Theorem 5.10. We start with a particular random coloring and a particular partition of colors $\mathcal{P} = \{P_1, P_2\}$, where $P_1 = \{●, ■\}$ and $P_2 = \{◆, ▲, ✸\}$. We make two calls to L_p-Cluster Selection with respect to P_1 and P_2 and construct the resulting clustering. In the example, all input vectors are distinct.

To perform the selection and grouping, our algorithm uses the color coding technique of Alon, Yuster, and Zwick from [10]. Consider the input as a family of initial clusters $\mathcal{I}$. We color initial clusters from $\mathcal{I}$ independently and uniformly at random by T colors 1, 2, ..., T. Consider any solution, and the particular set of at most T initial clusters which are included into composite clusters in this solution. These initial clusters are colored by distinct colors with probability at least $\frac{T!}{T^T} \geq e^{-T}$. Now we construct an algorithm for finding a colorful solution.

We consider all possible ways to split colors between clusters (some colors may be unused). Hence we consider all possible families $\mathcal{P} = \{P_1, \ldots, P_h\}$ of pairwise disjoint non-empty subsets of $\{c \in \{1, \ldots, T\} : \text{ there exists } J \in \mathcal{I} \text{ colored by } c\}$. Each family $\mathcal{P}$ corresponds to a partition of the set of colors $\{1, \ldots, T\}$ if we add one

L_p-k-Clustering $(X, k, D, \alpha, \mathcal{D})$
Input : A multiset $X \subset \mathbb{Z}^d$, a positive integer k, real nonnegative values D and α, a set $\mathcal{D}$, an algorithm $\mathcal{A}$ for L_p-Cluster Selection
Output: *Yes* or *No*

$T \leftarrow \lceil 2D/\alpha \rceil$
$\mathcal{I} \leftarrow$ initial clusters of X
for $\lceil e^T \rceil$ *iterations* **do**
  Fix a random coloring c of $\mathcal{I}$ with colors $\{1, \ldots, T\}$
  for *valid partitions* $\mathcal{P}$ *of* $\{1, \ldots, T\}$ **do**
    for $i = 1$ **to** $|\mathcal{P}|$ **do**
      $P_i = \{i_1, \ldots, i_t\}$
      for $j = 1$ **to** t **do**
        $X_j \leftarrow \emptyset$
        for $J \in \mathcal{I} : c(J) = i_j$ **do**
          $\mathbf{x} \leftarrow$ a point from J
          $X_j \leftarrow X_j \cup \{\mathbf{x}\}$
          $w(\mathbf{x}) \leftarrow |J|$
      $d_i \leftarrow D + 1$
      foreach $d \in \mathcal{D}$ **do**
        if $\mathcal{A}(X_1, \ldots, X_t, w, d)$ **then**
          $d_i \leftarrow d$
          BREAK
    if $\sum_{i=1}^{t} d_i \leq D$ **then**
      Yes, STOP
No, STOP

Figure 5.2: L_p-k-Clustering algorithm from Theorem 5.10

fictitious subset for colors which are not used in the composite clusters. The total number of partitions does not exceed $T^T = 2^{\mathcal{O}(D \log D)}$.

When partition $\mathcal{P}$ is fixed, we form clusters by solving instances of L_p-Cluster Selection: For each $i \in \{1, \ldots, h\}$, we take initial clusters colored by elements of P_i, bundle together those with the same color, and pass the resulting family to the L_p-Cluster Selection algorithm. First note that there cannot be $P \in \mathcal{P}$ of size at most one, since then L_p-Cluster Selection has to make a simple cluster while we assume that all clusters obtained from $\mathcal{P}$ are composite. Second, the total number of clusters has to be k, the number of clusters is $|\mathcal{I}| - \sum_{P \in \mathcal{P}} |P| + |\mathcal{P}|$. For each $\mathcal{P}$ we check that both conditions hold, and if not, we discard the choice of $\mathcal{P}$ and move to the next one, before calling the L_p-Cluster Selection subroutine.

Next, we formalize how we call the L_p-Cluster Selection algorithm. We fix the set of colors $P_i = \{c_1, \ldots, c_t\}$, then take the sets $I_j = \{J \in \mathcal{I} : J \text{ is colored by } c_j\}$

for $j \in \{1, \ldots, t\}$. We turn each set of initial clusters I_j into a set of weighted vectors X_j naturally: For each $J \in I_j$, we put one vector $\mathbf{x} \in J$ into X_j, and $w(\mathbf{x}) := |J|$. The family of sets of vectors $X_1, \ldots, X_t$ and the weight function w are the input for L_p-Cluster Selection. Then we search for the minimum cluster cost bound $d_i \leq D$ from $\mathcal{D}$, for which the instance $(X_1, \ldots, X_t, d_i)$ of L_p-Cluster Selection is a yes-instance, running each time the algorithm for L_p-Cluster Selection.

If for some i setting d_i to D leads to a no-instance, or if $\sum_{i=1}^{h} d_i > D$, then we discard the choice of the partition $\mathcal{P}$ and move to the next one. Otherwise, we report that L_p-k-Clustering has a solution and stop. Next, we prove that in this case the solution indeed exists.

We reconstruct the solution to L_p-k-Clustering as follows: For each $i \in \{1, \ldots, h\}$ the corresponding to $P_i = \{c_1, \ldots, c_t\}$ instance of L_p-Cluster Selection has a solution $\{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$. For each $j \in \{1, \ldots, t\}$, consider the corresponding initial cluster J_j consisting of $w(\mathbf{x}_j)$ vectors equal to $\mathbf{x}_j$. For each $i \in \{1, \ldots, h\}$ we obtain a composite cluster $\cup_{j=1}^{t} J_j$, all other clusters are simple. So the total cost is $\sum_{i=1}^{h} d_i$, which is at most D. Thus, if the algorithm finds a solution, then (X, d, D) is a yes-instance.

In the opposite direction. If there is a solution to L_p-k-Clustering, then there is a regular solution, and with probability at least e^{-T} initial clusters which are parts of composite clusters in this solution are colored by distinct colors. Then, there is a partition $\mathcal{P} = \{P_1, \ldots, P_h\}$ which corresponds to this solution. This partition is obtained as follows: put into P_1 colors from the first composite cluster, into P_2 from the second and so on. At some point our algorithm checks the partition $\mathcal{P}$, and as it finds the optimal cost value for each cluster, then it is at most the cost of the corresponding cluster of the solution from which we started.

To analyze the running time, we consider $2^{\mathcal{O}(D \log D)}$ partitions $\mathcal{P}$, for each $\mathcal{P}$ we $|\mathcal{P}| = \mathcal{O}(D)$ times search for optimal d_i. And for each of $|\mathcal{D}|$ possible values [1] of d_i we make one call to the L_p-Cluster Selection algorithm, which takes time at most $\Phi(n, d, T, D)$.

To amplify the error probability to be at most $1/e$, we do $N = \lceil e^T \rceil$ iterations of the algorithm, each time with a new random coloring. As each iteration succeeds with probability at least e^{-T}, the probability of not finding a colorful solution after N iterations is at most $(1 - e^{-T})^{e^T} \leq e^{-1} < 1$. So the total running time is $2^{\mathcal{O}(D \log D)} \cdot (nd)^{\mathcal{O}(1)} |\mathcal{D}| \Phi(n, d, 2D/\alpha, D)$.

The algorithm could be derandomized by the standard derandomization technique using perfect hash families [10, 162]. So L_p-k-Clustering is solvable in the same deterministic time. □

[1] We could also binary search for the optimal $d_i \in \mathcal{D}$ instead, thus replacing $|\mathcal{D}|$ by $\log |\mathcal{D}|$ in the running time. However, for all choices of $\mathcal{D}$ we consider this does not make a difference.

5.2 Algorithms and Lower Bounds for $p \in (0, 1]$

The main motivation for the results in this section is the study of k-Clustering with the L_1 distance, the natural variant of Euclidean k-Median with the rectilinear/Manhattan distance. However, our main algorithmic result also extends to distances of order $p \in (0, 1)$ since in some sense they behave similarly to the L_1 distance.

5.2.1 FPT Parameterized by D

In this subsection, we prove Theorem 5.1: when $p \in (0, 1]$, L_p-k-Clustering admits an FPT algorithm with parameter D. First we state basic geometrical observations for cases $p = 1$ and $p \in (0, 1)$, Then we propose a general algorithm for L_p-Cluster Selection which relies only on these properties. Finally, we show how Theorem 5.10 could be applied.

The next two claims deal with the structure of optimal cluster centers. We state and prove them in the case of weighted vectors where each vector has a positive integer weight given by a weight function w. The unweighted case is just a special case when the weight of each vector is one.

First, we show that coordinates of cluster centers could always be selected among the values present in the input, which helps greatly in enumerating cluster centers that may be optimal.

Claim 5.11. *Assume $p \in (0, 1]$, let $C = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ be a cluster and $w : \{\mathbf{x}_1, \cdots, \mathbf{x}_t\} \to \mathbb{Z}_{>0}$ be a weight function. There is an optimal (subject to the weighted distance $w(\mathbf{x}_i) \cdot \mathrm{dist}_p(\mathbf{x}_i, \mathbf{c})$) center $\mathbf{c}$ for the cluster C such that for each $i \in \{1, \ldots, d\}$, the i-th coordinate $\mathbf{c}[i]$ of the center $\mathbf{c}$ is from the values present in the input in this coordinate, that is $\mathbf{c}[i] \in \{\mathbf{x}_1[i], \ldots, \mathbf{x}_t[i]\}$. Moreover, for $p = 1$ we may assume that the optimal value is a weighted median of the values present in the i-th coordinate.*

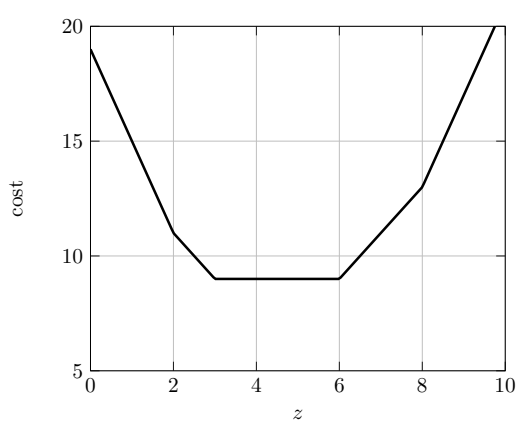
Proof. For cluster C, consider the corresponding multiset of unweighted vectors $C' = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$, where each vector $\mathbf{x} \in C$ is repeated $w(\mathbf{x})$ times. We define $y_j = \mathbf{x}_j[i]$ for $j \in \{1, \ldots, t\}$. Assume that $y_1 \le y_2 \le \cdots \le y_t$. Let us consider an optimal cluster center c for C and denote $z = \mathbf{c}[i]$. Figure 5.3 shows how the cluster cost behaves with respect to z on a concrete set of values $\{y_i\}$ for $p = 1$ and $p = 1/2$.

For the formal proof, we start with the case $p = 1$. The total cost of C contributed by the i-the coordinate is

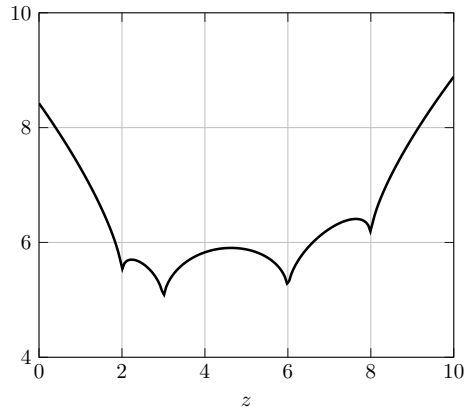
$$|y_1 - z| + |y_2 - z| + \cdots + |y_t - z|.$$

If $z \in (y_i, y_{i+1})$ for $i \in \{1, \ldots, t-1\}$, then the derivative with respect to z is

$$((z - y_1) + \cdots + (z - y_i) + (y_{i+1} - z) + \cdots + (y_t - z))' = i - (t - i).$$



(a) $\text{cost}(z) = |z-2|+|z-3|+|z-6|+|z-8|$



(b) $\text{cost}(z) = |z-2|^{1/2} + |z-3|^{1/2} + |z-6|^{1/2} + |z-8|^{1/2}$

Figure 5.3: Graphs of cluster cost over different values of z: dist_1 in the left plot, $\text{dist}_{1/2}$ in the right plot. The set of coordinate values is given as $y_1 = 2$, $y_2 = 3$, $y_3 = 6$, $y_4 = 8$.

Analogously, when $z = y_i$ for $i \in \{1, \ldots, t\}$, the derivative is $i - 1 - (t - i)$. When $z < y_1$ the derivative is $-t$, and when $z > y_t$ the derivative is t. So if t is odd, then the derivative is zero at $y_{\lceil t/2 \rceil}$, strictly negative before and strictly positive after, so $y_{\lceil t/2 \rceil}$, which is the only median, is the optimal value for z. If t is even, then the derivative is zero on $[y_{t/2}, y_{t/2+1}]$, strictly negative before and strictly positive after. So any value from $[y_{t/2}, y_{t/2+1}]$ is optimal, and we may assume that it is one of the two medians $y_{t/2}$, $y_{t/2+1}$.

Now to the case $p \in (0, 1)$, the contribution of the coordinate i is

$$|y_1 - z|^p + |y_2 - z|^p + \cdots + |y_t - z|^p.$$

When z is between y_i and y_{i+1}, then the derivative of the above with respect to z is equal to

$$p \cdot \left((z - y_1)^{p-1} + \cdots + (z - y_i)^{p-1} - (y_{i+1} - z)^{p-1} - \cdots - (y_t - z)^{p-1} \right).$$

It is monotone on (y_i, y_{i+1}): when z increases, the sum decreases, as terms of the form $(z - y_j)^{p-1}$ decrease and terms of the form $(y_j - z)^{p-1}$ increase, because $p - 1 < 0$. Thus, the optimal value on this interval is achieved at one of its ends. Doing the same for all intervals, we conclude that the optimal value for z must be in $\{y_1, \ldots, y_t\}$. □

In particular, by Claim 5.11 we may assume that the coordinates of optimal cluster centers are integers. Then, the α-property holds with $\alpha = 1$ since at most one of the initial clusters could have distance zero to the cluster center, and all others

have distance at least one since the cluster center is integral. Namely, let $\mathbf{x}$ be a vector in the cluster, and $\mathbf{c}$ be the cluster center, if $\mathbf{x} \neq \mathbf{c}$, then there is a coordinate j where $\mathbf{x}$ and $\mathbf{c}$ differ, and since they are both integral, $|\mathbf{x}[j] - \mathbf{c}[j]| \geq 1$, and

$$\mathrm{dist}_p(\mathbf{x}, \mathbf{c}) = \sum_{i=1}^{d} |\mathbf{x}[i] - \mathbf{c}[i]|^p \geq |\mathbf{x}[j] - \mathbf{c}[j]|^p \geq 1^p = 1.$$

In what follows, the expression *half of vectors by weight* means that the total weight of the corresponding set of vectors is at least half of the total weight of C.

Claim 5.12. *If at least half of the vectors by weight in the cluster C have the same value z in some coordinate i, then the optimal cluster center is also equal to z in this coordinate.*

Proof. Let S be the weight-respecting multiset of values which vectors from C have in the i-th coordinate: $S = \{\mathbf{x}[i] : \mathbf{x} \in C, w(\mathbf{x}) \text{ times}\}$. Consider the difference between selecting z and some other value z' as the i-th coordinate of the center:

$$\sum_{y \in S} |y - z|^p - \sum_{y \in S} |y - z'|^p \leq \sum_{y \in S, y \neq z} (|y - z|^p - |y - z'|^p - |z - z'|^p).$$

The inequality holds since at least half of the elements of S are equal to z, and so for any value $y \neq z$ there is a term $|z - z'|^p$ in $\sum_{y \in S} |y - z'|^p$ corresponding to one of the values from S equal to z. The last sum is non-positive because in every term

$$|y - z|^p \leq |y - z'|^p + |z - z'|^p,$$

as $p \in (0, 1]$. This concludes the proof. □

In order to apply Theorem 5.10, we need an FPT algorithm for L_p-Cluster Selection. Before obtaining it, we state some properties of hypergraphs, which we need for the algorithm. Intuitively, our algorithm reduces selecting a center in a L_p-Cluster Selection instance to finding a subhypergraph with certain properties.

A *hypergraph* G is a set of vertices $V(G)$ and a collection of hyperedges $E(G)$, each hyperedge is a subset of $V(G)$. If G and H are hypergraphs, we say that H *appears* at $V' \subset V(G)$ as a *subhypergraph* if there is a bijection $\pi : V(H) \to V'$ with a property that for any $E \in E(H)$ there is $E' \in E(G)$ such that $\pi(E) = E' \cap V'$. Here we consider that the action of π is extended to subsets of $V(H)$ in a natural way, $\pi(E) = \{\pi(v)\}_{v \in E}$ for $E \subset V(H)$.

A *fractional edge cover* of a hypergraph H is an assignment $\psi : E(H) \to [0, 1]$ such that for every $v \in V(H)$, $\sum_{E \in E(H): v \in E} \psi(E) \geq 1$. The *fractional cover number* $\rho^*(H)$ is the minimum of $\sum_{E \in E(H)} \psi(E)$ taken over all fractional edge covers ψ.

We need the following result of Marx [156] about finding occurrences of one hypergraph in another.

Theorem 5.13 ([156]). *Let H be a hypergraph with fractional cover number $\rho^*(H)$, and let G be a hypergraph where each hyperedge has size at most ℓ. There is an algorithm that enumerates in time $|V(H)|^{\mathcal{O}(|V(H)|)} \cdot \ell^{|V(H)|\rho^*(H)+1} \cdot |E(G)|^{\rho^*(H)+1} \cdot |V(G)|^2$ every subset $V' \subset V(G)$ where H appears in G as a subhypergraph.*

We are ready to proceed with the proof that L_p-Cluster Selection with $p \in (0, 1]$ is FPT when parameterized by D.

Theorem 5.14. *For every $p \in (0, 1]$, L_p-Cluster Selection is solvable in time $2^{\mathcal{O}(D \log D)}(md)^{\mathcal{O}(1)}$.*

Proof. First we check if any of the given vectors could be the center of the resulting composite cluster. When the center is fixed, we find the optimal solution in polynomial time by just selecting the cheapest vector with respect to this center from each set. If at some point we find a suitable center, then we return that the solution exists. If not, we may assume that the center is not equal to any of the given vectors. As a consequence, any vector $\mathbf{x}$ selected into the solution cluster contributes at least $w(\mathbf{x})$ to the total distance, since the center must be integral by Claim 5.11. So we may now consider only vectors of weight at most D and, moreover, the total weight of the resulting cluster is at most D.

Consider a resulting cluster C with the center $\mathbf{c}$. There is some $\mathbf{x}_1$ in C from X_1, and $\mathrm{dist}_p(\mathbf{x}_1, \mathbf{c}) \leq D$. So if we try all possible $\mathbf{x}_1$ from X_1 (there are at most m of them), any feasible center is at distance at most D from at least one of them. Since $\mathbf{x}_1$ and $\mathbf{c}$ are integral, they could be different in at most D coordinates, as $\mathrm{dist}_p(\mathbf{x}_1, \mathbf{c}) = \sum_{i=1}^{d} |\mathbf{x}_1[i] - \mathbf{c}[i]|^p \leq D$.

We try all possible $\mathbf{x}_1 \in X_1$. After $\mathbf{x}_1$ is fixed, we enumerate all subsets P of coordinates $\{1, \ldots, d\}$ where $\mathbf{x}_1$ and $\mathbf{c}$ could differ, we show how to do it efficiently afterwards. When the subset of coordinates P is fixed, we consider all possible centers, which are integral, equal to $\mathbf{x}_1$ in all coordinates except P, and differ from $\mathbf{x}_1$ by at most $D^{1/p}$ in each of coordinates from P. If $|\mathbf{x}_1[i^*] - \mathbf{c}[i^*]| > D^{1/p}$ for some coordinate i^*, then $\mathrm{dist}_p(\mathbf{x}_1, \mathbf{c}) = \sum_{i=1}^{d} |\mathbf{x}_1[i] - \mathbf{c}[i]|^p \geq |\mathbf{x}_1[i^*] - \mathbf{c}[i^*]|^p > D$, so $\mathbf{c}$ cannot be a center. With restrictions stated above, there are at most $2^{\mathcal{O}(D \log D)}$ possible centers.

It remains to show that we could enumerate all possible coordinate subsets efficiently. We reduce this task to the task of finding a specific subhypergraph and then apply Theorem 5.13.

Claim 5.15. *There are $2^{\mathcal{O}(D \log D)}$ coordinate subsets where $\mathbf{x}_1$ and an optimal cluster center c could differ. There exists an algorithm which enumerates all of them in time $2^{\mathcal{O}(D \log D)}(md)^{\mathcal{O}(1)}$.*

Proof. Let G be a hypergraph with $V(G) = \{1, \ldots, d\}$, one vertex for each coordinate, and for each vector $\mathbf{x}$ in $\cup_{j=1}^{t} X_j$ we take $w(\mathbf{x})$ multiple hyperedges $E_{\mathbf{x}}$ which contains exactly the coordinates where $\mathbf{x}$ and $\mathbf{x}_1$ differ. We add an edge only if there are at

$D = 2$

v	1	2	3	4	5
$\mathbf{x}_1$	0	2	1	3	2
$\mathbf{x}_2$	0	**1**	1	3	**1**
$\mathbf{x}_3$	**1**	2	1	3	**1**
$\mathbf{x}_4$	0	2	**2**	3	2
$\mathbf{x}_5$	0	2	**2**	3	**1**
$\mathbf{c}$	0	2	**2**	3	2

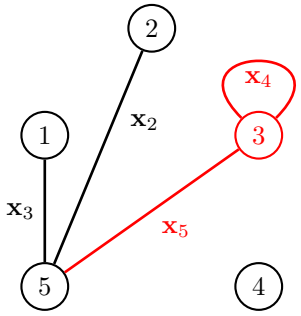


Figure 5.4: An illustration of the hypergraph construction in Claim 5.15. On the left, the vector $\mathbf{x}_1$ and other input vectors $\mathbf{x}_2, \ldots, \mathbf{x}_5$ are given. On the right, the corresponding hypergraph G. The solution is marked in red on both sides: on the left, the resulting cluster $\{\mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_5\}$ of cost 2; on the right, the corresponding to $\{\mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_5\}$ subhypergraph H. Note that in H the hyperedge $\mathbf{x}_5$ is restricted to the only vertex 3, so its size is one.

most D such coordinates, otherwise $\mathbf{x}$ can not be in the same cluster as $\mathbf{x}_1$. So hyperedges in G are of size at most D. Since we consider only vectors of weight at most D, $|E(G)| \leq Dm$.

For a solution, let $\mathbf{x}_j$ be the vector selected from the corresponding X_j, for $j \in \{1, \ldots, t\}$, $C = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ be the solution cluster and $\mathbf{c}$ be the center. All vectors in C are identical in all coordinates except at most D, since if there are different values in at least $D + 1$ coordinates, the cost is at least $D + 1$. Denote this subset of coordinates as Q, $\mathbf{c}$ could also differ from $\mathbf{x}_1$ only at Q. Denote the subset of coordinates where $\mathbf{c}$ differs from $\mathbf{x}_1$ as P, $P \subset Q$ and so $|P| \leq D$. The solution $(C, \mathbf{c})$ induces a subhypergraph H of G in the following way. Leave only hyperedges corresponding to the vectors in C, and restrict them to vertices in P. There are at most D vertices and at most D hyperedges in H, since the total weight is at most D. An example of the correspondence between input vectors and hypergraphs is given in Figure 5.4.

The next claim shows that the fractional cover number of H is bounded by a constant.

Claim 5.16. *Each vertex in H is covered by at least half of the hyperedges of H, and $\rho^*(H) \leq 2$.*

Proof. Consider a vertex $p \in P$, and assume that less than half of the hyperedges cover p. It means that in the p-th coordinate the center c differs from $\mathbf{x}_1$, but less than half of the vectors in C by weight differ from $\mathbf{x}_1$ in this coordinate. This contradicts Claim 5.12.

So each vertex is covered by at least half of the hyperedges, and setting $\psi \equiv \frac{2}{|E(H)|}$ leads to $\rho^*(H) \leq 2$. □

In order to enumerate all possible subsets of coordinates P, we try all hypergraphs H with at most D vertices and at most D hyperedges, and if each vertex is covered by at least half of the hyperedges, we find all places where H appears in G by Theorem 5.13. The last step is done in $2^{\mathcal{O}(D \log D)} \cdot (md)^{\mathcal{O}(1)}$ time. However, the number of possible H could be up to $2^{\Omega(D^2)}$. The following claim, which is analogous to Proposition 6.3 in [156], shows that we could consider only hypergraphs with a logarithmic number of hyperedges.

Claim 5.17. *If $D \geq 2$, it is possible to delete all except at most $160 \ln D$ hyperedges from H so that in the resulting hypergraph H^* each vertex is covered by at least $1/4$ of the hyperedges, and $\rho^*(H^*) \leq 4$.*

Proof. Denote $s = |E(H)|$, construct a new hypergraph H^* on the same vertex set $V(H)$ by independently selecting each hyperedge of H with probability $(120 \ln D)/s$. Applying Proposition 11.14 with $\beta = 1/3$, probability of selecting more than $160 \ln D$ hyperedges is at most $\exp((-120 \ln D)/27) < 1/D^2$. By Claim 5.16, each vertex v of H is covered by at least $s/2$ hyperedges, and the expected number of hyperedges covering v in H^* is at least $60 \ln D$. By Proposition 11.14 with $\beta = 1/3$, the probability that v is covered by less than $40 \ln D$ hyperedges in H^* is at most $\exp(-60 \ln D/18) \leq 1/D^3$. By the union bound, with probability at least $1 - 1/D^2 - D \cdot 1/D^3 > 0$ we select at most $160 \ln D$ hyperedges and each vertex is covered by at least $40 \ln D$ hyperedges. So the claim holds, and $\rho^*(H^*) \leq 4$ by setting $\psi \equiv \frac{4}{|E(H^*)|}$. □

Thus, if there is a subhypergraph H in G corresponding to a solution, then there is also a subhypergraph H^* in G appearing at the same subset of $V(G)$ with at most $160 \ln D$ hyperedges and where each vertex is covered by at least $1/4$ of the hyperedges. Since we only need to enumerate possible coordinate subsets, in our algorithm we try all hypergraphs of this form and apply Theorem 5.13 for each of them. Since there are at most $2^{\mathcal{O}(D \log D)}$ hypergraphs with at most $160 \ln D$ hyperedges and since the fractional cover number is still bounded by a constant, the total running time is $2^{\mathcal{O}(D \log D)} \cdot (md)^{\mathcal{O}(1)}$, as desired. □

With Claim 5.15 proven, the proof of the theorem is complete. The pseudocode given in Figure 5.5 summarizes the main steps of the algorithm. □

Combining Theorem 5.10 and Theorem 5.14, we obtain an FPT algorithm for L_p-k-Clustering. This proves Theorem 5.1, which we recall here.

Theorem 5.1. *L_p-k-Clustering is solvable in time $2^{\mathcal{O}(D \log D)}(nd)^{\mathcal{O}(1)}$ for every $p \in (0, 1]$.*

Proof. We have an algorithm for L_p-Cluster Selection whose running time is specified by Theorem 5.14. By Claim 5.11, the α-property holds. The only missing part is to describe the way of producing the set $\mathcal{D}$ of all possible cluster costs which are at most D.

L_p-Cluster Selection $(X_1, \ldots, X_t, w, D)$
Input : Sets of vectors $X_1, \ldots, X_t$, a weight function w, a nonnegative integer D
Output: *Yes* or *No*

for *vector* $\mathbf{c}$ *in the input* **do**
  if $\sum_{i=1}^{t} \min_{\mathbf{x}_i \in X_i} w(\mathbf{x}_i) \operatorname{dist}_p(\mathbf{x}_i, \mathbf{c}) \leq D$ **then**
    Yes, STOP
for $\mathbf{x}_1 \in X_1$ **do**
  $G \leftarrow$ hypergraph with $V(G) = \{1, \ldots, d\}, E(G) =$ {positions where $\mathbf{x}_1$ and $\mathbf{x}$ differ : $\mathbf{x} \in \cup_{j=1}^{t} X_j, w(\mathbf{x})$ times}
  for *hypergraph* H *with at most* D *vertices and at most* $160 \ln D$ *hyperedges* **do**
    if *each vertex of* H *is covered by at least 1/4 of its hyperedges* **then**
      for *place* P *where* H *appears in* G *as subhypergraph* **do**
        for *integer vector* $\mathbf{c}$ *which differs from* $\mathbf{x}_1$ *only at* P *by at most* $D^{1/p}$ **do**
          if $\sum_{i=1}^{t} \min_{\mathbf{x}_i \in X_i} w(\mathbf{x}_i) \operatorname{dist}_p(\mathbf{x}_i, \mathbf{c}) \leq D$ **then**
            Yes, STOP
No, STOP

Figure 5.5: L_p-Cluster Selection algorithm from Theorem 5.14

In the case $p = 1$ all distances are integral since optimal centers have integral coordinates by Claim 5.11, and we can take $\mathcal{D} = \{0, \ldots, D\}$.

For the general case, let $\mathcal{B} = \{b^p : b \in \{1, \ldots, \lceil D^{1/p} \rceil\}\}$. Consider a cluster $C = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ and the corresponding optimal cluster center $\mathbf{c}$. For any $\mathbf{x}_j \in C$, $\operatorname{dist}_p(\mathbf{x}_j, \mathbf{c}) = \sum_{i=1}^{d} |\mathbf{x}_j[i] - \mathbf{c}[i]|^p$ is a combination of elements of $\mathcal{B}$ with nonnegative integer coefficients. This is because $\mathbf{x}_j$ and $\mathbf{c}$ are integral and the cluster cost is at most D, hence $|\mathbf{x}_j[i] - \mathbf{c}[i]| \leq D^{1/p}$ for each $i \in \{1, \ldots, d\}$. Since weights are also integral, the whole cluster cost is a combination of distances between cluster vectors and the center with nonnegative integer coefficients, and so also a combination of elements of $\mathcal{B}$ with nonnegative integer coefficients. This means that we can take

$$\mathcal{D} = \left\{ \sum_{b \in \mathcal{B}} \mathbf{a}_b \cdot b : \mathbf{a}_b \in \mathbb{Z}, \mathbf{a}_b \geq 0, \sum_{b \in \mathcal{B}} \mathbf{a}_b \leq D \right\},$$

where $\mathbf{a}$ is a vector indexed by the set $\mathcal{B}$. The sum of coefficients $\mathbf{a}_b$ is at most D since all elements of $\mathcal{B}$ are at least 1. The size of $\mathcal{D}$ is at most $|\mathcal{B}|^D = 2^{\mathcal{O}(D \log D)}$. □

Another widely studied version of k-Clustering is where the centers $\mathbf{c}_i$ could be selected only among the set of given vectors, this version is usually referred to as

discrete k-Clustering. Naturally, Theorem 5.1 also holds in this setting since L_p-Cluster Selection is then trivially solvable in polynomial time. As was observed in the proof of Theorem 5.14, if the cluster center is fixed, we can pick the cheapest vector from each of the sets given to a L_p-Cluster Selection algorithm, and there are now only polynomially many candidates for the cluster center.

Note that Claim 5.11 and Claim 5.12 do not hold in the case $1 < p < \infty$, and our algorithm relies heavily on the structure provided by these claims. Therefore, it does not seem that the algorithm could be extended to the case $1 < p < \infty$. Moreover, in Theorem 5.5 we formally prove that L_p-Cluster Selection parameterized by D is W[1]-hard for $1 < p < \infty$.

In the cases $p = 0$ and $p = \infty$ there are different obstacles for the algorithm above. In L_0-Cluster Selection, even knowing the center that differs in at most k positions from an optimal one, is not enough, as any distinct value in the coordinate would incur the same cost of one. For $p = \infty$, it simply does not hold that the number of coordinates where points and the center can differ is small: any number of coordinates might differ as long as the absolute difference is at most D. To formalize this intuition we later prove Theorem 5.2 and Theorem 5.3, showing that L_p-k-Clustering parameterized by D is W[1]-hard for both $p = 0$ and $p = \infty$.

5.2.2 W[1]-hardness Parameterized by $t + d$ for L_1

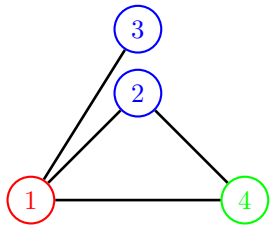
In this subsection, we restrict our attention to the $p = 1$ case. What happens when D is not bounded, but the dimension d and the number of clusters k are parameters? There is a trivial XP-algorithm in time $n^{\mathcal{O}(kd)}$, as by Claim 5.11 it suffices to try all possible combinations of the values present in coordinates as possible cluster centers. There are at most n distinct values in each coordinate, so at most n^d candidates for a cluster center. After the cluster centers are fixed, each vector goes to the cluster with the closest center. The next observation is the result of this discussion.

Observation 5.18. *L_1-k-Clustering is solvable in time $n^{\mathcal{O}(kd)}$.*

We do not know of a lower bound for L_1-k-Clustering complementing Observation 5.18. However, we are able to show the hardness of L_1-Cluster Selection with respect to the dimension.

Theorem 5.19. *L_1-Cluster Selection is W[1]-hard when parameterized by $t+d$.*

Proof. We construct a reduction from Multicolored Clique with the input G and k. We set d to k, for each pair of colors $1 \leq i < j \leq k$ and each $e = \{u, v\}$ between a vertex u of color i and a vertex v of color j we add a vector $\mathbf{x}_e$ to the set $X_{i,j}$, such that $\mathbf{x}_e[i] = u$, $\mathbf{x}_e[j] = v$ and all other coordinates are set to zero, and a vector $\mathbf{y}_e$ to the set $Y_{i,j}$ which is the same as $\mathbf{x}_e$, only coordinates other that i and j are set to $|V(G)| + 1$. We will refer to 0 and $|V(G)| + 1$ as boundary values. The sets $X_{i,j}$ and $Y_{i,j}$ are the input to L_1-Cluster Selection, so t is $2\binom{k}{2}$, and we set D to



$X_{1,2}$	1	2	0
	1	3	0
$X_{2,3}$	0	2	4
$X_{1,3}$	1	0	4
$Y_{1,2}$	1	2	5
	1	3	5
$Y_{2,3}$	5	2	4
$Y_{1,3}$	1	5	4

$\mathbf{x}_{12}$	1	2	0
$\mathbf{x}_{24}$	0	2	4
$\mathbf{x}_{14}$	1	0	4
$\mathbf{y}_{12}$	1	2	5
$\mathbf{y}_{24}$	5	2	4
$\mathbf{y}_{14}$	1	5	4
$\mathbf{c}$	1	2	4

Figure 5.6: An example illustrating the reduction in Theorem 5.19: an input graph G with vertices colored in three colors, the sets of vectors produced by the reduction, and the resulting optimal cluster, corresponding to the clique on $\{1, 2, 4\}$.

$k(|V(G)| + 1)\binom{k-1}{2}$. Intuitively, the set $X_{i,j}$ corresponds to the choice of the clique edge between i-th and j-th color, and $Y_{i,j}$ mirrors it. All vectors have weight one. An example is given in Figure 5.6.

Note that in any feasible cluster, each coordinate i has exactly $2(k-1)$ values in $[1, |V(G)|]$, one from each of the sets $X_{i,j}$ and $Y_{i,j}$ for $j \neq i$. Out of all $2(\binom{k}{2} - k + 1) = 2\binom{k-1}{2}$ other values, exactly half are zero and half are $|V(G)| + 1$. So the median is always in $[1, |V(G)|]$, and the boundary values in each column contribute exactly $(|V(G)| + 1)\binom{k-1}{2}$ to the total distance.

Assume there is a colorful k-clique in G, with vertices $v_1, v_2, \ldots, v_k$. We form the resulting cluster by choosing the vector corresponding to the clique's edge between its i-th and j-th vertices from $X_{i,j}$, and also from $Y_{i,j}$, for all $1 \leq i < j \leq k$. For this cluster, in the i-th coordinate we have all non-boundary values equal to v_i. So the median is also v_i, and the total distance is D, since non-boundary values do not contribute anything.

In the other direction, if we are able to select a cluster of cost exactly D, then all non-boundary values in each coordinate must be equal, denote this common value in the i-th coordinate as v_i. We claim that vertices $v_1, v_2, \ldots, v_k$ form a colorful clique in G. Indeed, since we have $2(k-1)$ times v_i in the i-th column, then we have $(k-1)$ of them from the sets $X_{i,j}$, one from each, and in the j-th column the only non-boundary value is v_j. So v_i must have an edge to each v_j for $j \neq i$. By construction, vertices in the i-th coordinate are of color i. □

5.3 The L_0 Distance

In this section, we consider the case of the dist_0 distance. It is a natural measure of difference to consider since observation parameters are often incomparable, and we very well may be interested in counting only the number of different entries. From another point of view, the L_0 distance gives the k-Clustering problem a more

combinatorial flavor, since the input vectors could be viewed as strings and we are interested about how close they are according to the Hamming distance. However, in comparison to a number of problems on strings, the size of the alphabet is unbounded.

First, note that there is a simple rule for finding the optimal cluster center for a given cluster.

Observation 5.20. *For a given cluster C, the coordinates of the optimal cluster center $\mathbf{c}$ could be set as*

$$\mathbf{c}[i] = \textit{the most frequent element of the multiset } \{\mathbf{x}[i]\}_{\mathbf{x}\in C},\ 1 \leq i \leq d,$$

breaking ties in favor of the lowest values.

By Observation 5.20, we may assume that optimal cluster centers could never have values not present in the input, and in particular that they are integral.

We prove W[1]-hardness of L_0-k-Clustering by showing a reduction from Clique. The reduction also shows hardness of L_0-Cluster Selection.

Note that when d is fixed, we could apply Theorem 5.10 to obtain an FPT algorithm: L_0-Cluster Selection can be solved trivially by trying every present value in each coordinate as a value for the center, there are only n^d variants. The α-property holds for L_0 distance with $\alpha = 1$ since at most one initial cluster could coincide with the cluster center, and all others have distance at least one. We state this formally in the next observation.

Observation 5.21. L_0-Cluster Selection *is solvable in time $n^{\mathcal{O}(d)}$, and L_0-k-*Clustering *is solvable in time $2^{\mathcal{O}(D \log D)} n^{\mathcal{O}(d)}$.*

Next we restate and prove Theorem 5.2. Note that Theorem 5.2 essentially complements the trivial algorithms given by Observation 5.21.

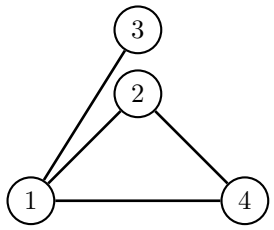
Theorem 5.2. L_0-k-Clustering *parameterized by $d + D$ and L_0-*Cluster Selection *parameterized by $d + t + D$ are* W[1]*-hard.*

Proof. First we show how to obtain an FPT reduction from Clique parameterized by the clique size to L_0-k-Clustering.

Given an instance (G, k) of Clique, for each pair of indices $\{i, j\}$, $1 \leq i < j \leq k$, we make $|E(G)|$ vectors in $\mathbb{Z}^k$, assume $k \geq 3$. For each $e = \{u, v\} \in E(G)$, we add a vector $\mathbf{x}_{i,j,e}$: two coordinates are set to vertex values, $\mathbf{x}_{i,j,e}[i] = u$, $\mathbf{x}_{i,j,e}[j] = v$, and in all other coordinates $\mathbf{x}_{i,j,e}$ is set to the special padding value $c_{i,j,e} = |V(G)| + (k \cdot i + j) \cdot |E(G)| + e$. In total, there are $n = \binom{k}{2}|E(G)|$ vectors and $|V(G)| + \binom{k}{2}|E(G)|$ different values, since there are $|V(G)|$ vertex values, all padding values are distinct from vertex values and from each other.

Finally, we set $k' = n - \binom{k}{2} + 1$ and $D = \binom{k}{2}(k-2)$. An example of the reduction is shown in Figure 5.7.

Now we prove that the original instance has a k-clique iff the transformed instance has a k'-clustering of cost at most D.



$\mathbf{x}_{1,2,\cdot}$	1	2	·
	1	3	·
	1	4	·
	2	4	·
$\mathbf{x}_{1,3,\cdot}$	1	·	2
	1	·	3
	1	·	4
	2	·	4
$\mathbf{x}_{2,3,\cdot}$	·	1	2
	·	1	3
	·	1	4
	·	2	4

$\mathbf{x}_{1,2,12}$	1	2	·
$\mathbf{x}_{1,3,14}$	1	·	4
$\mathbf{x}_{2,3,24}$	·	2	4
$\mathbf{c}$	1	2	4

Figure 5.7: An example illustrating the reduction in Theorem 5.2: an input graph G, the vectors produced by the reduction (for clarity, they are separated over corresponding pairs $\{i, j\}$, and padding values are replaced by dots), and the only composite cluster in the resulting optimal clustering of cost 3, corresponding to the clique on $\{1, 2, 4\}$.

If there is a k-clique, there is a clustering with cost D: we take one nontrivial cluster of size $\binom{k}{2}$ and all other clusters are of size 1. Let v_1,..., v_k be the vertices of the clique, for each $\{i, j\}$, $1 \leq i < j \leq k$ we take $\mathbf{x}_{i,j,\{v_i,v_j\}}$ into the cluster. The cluster center is $(v_1, ..., v_k)$, each vector in the cluster has distance to the center of exactly $(k - 2)$.

Now to the opposite direction. Assume that there is a clustering of cost at most D, and there are t composite clusters: C_1, ..., C_t. In each cluster and each coordinate, by Observation 5.20 we may assume that we select the most frequent vertex there as the value of the center, since all padding values are distinct. If there are no vertex values in this cluster in this coordinate, we may assume that we select any of the occurring padding values. For a cluster C, denote the number of vertex-containing coordinates as $\beta(C)$, and the total number of vertex-valued entries which do not match with the center value in the corresponding coordinate as $\gamma(C)$. We could write the total cost of the clustering as

$$\sum_{i=1}^{t} \left(|C_i|(k-2) - (k - \beta(C_i)) + \gamma(C_i) \right).$$

That holds since in each cluster C_i each of the $|C_i|(k - 2)$ padding values is not matched with the cluster center and increases the total distance by one, except for the $(k - \beta(C_i))$ vertex-free coordinates, where exactly one of the padding values is selected as the value of the center. Also each vertex-valued entry which is not matched with the center increases the total distance by one, there are $\gamma(C_i)$ of them.

There are $n - \binom{k}{2} + 1$ clusters in total, $n - \binom{k}{2} + 1 - t$ of them are simple. We may assume that in the optimal clustering there are no empty clusters, since we could

always move a vector from a composite cluster to an empty one without increasing the cost. So there are $n - (n - \binom{k}{2} + 1 - t) = t + \binom{k}{2} - 1$ vectors in the composite clusters, which is equal to $\sum_{i=1}^{t} |C_i|$. We could rewrite the total cost as

$$(t+\binom{k}{2}-1)(k-2)-tk+\sum_{i=1}^{t}(\beta(C_i)+\gamma(C_i)) = \binom{k}{2}(k-2)-(k-2)+\sum_{i=1}^{t}(\beta(C_i)-2+\gamma(C_i)).$$

Now we show that for any clustering the value $\sum_{i=1}^{t}(\beta(C_i)-2+\gamma(C_i))$ is at least $(k-2)$, and it is equal to $(k-2)$ only in the k-clique clustering. It suffices to prove the following lemma.

Lemma 5.22. *For any cluster C such that $2 \le |C| \le \binom{k}{2}$, $\frac{\beta(C)-2+\gamma(C)}{|C|-1} \ge \kappa$, where $\kappa = \frac{k-2}{\binom{k}{2}-1} = \frac{2}{k+1}$, and the equality holds only when C is a k-clique.*

The lemma implies

$$\sum_{i=1}^{t}(\beta(C_i) - 2 + \gamma(C_i)) = \sum_{i=1}^{t} \frac{\beta(C_i) - 2 + \gamma(C_i)}{|C_i| - 1}(|C_i| - 1)$$
$$\ge \kappa \sum_{i=1}^{t}(|C_i| - 1) = \kappa \left(\binom{k}{2} - 1 \right) = k - 2,$$

and also that the equality holds only when each term is equal to κ, so each C_i is a k-clique, but then $t = 1$ since $\sum_{i=1}^{t}(|C_i| - 1) = \binom{k}{2} - 1$. So G must contain a k-clique if there is a clustering of cost at most D, and the reduction is correct. Note that none of the C_i could have size larger than $\binom{k}{2}$ since there are $n - \binom{k}{2} + 1$ clusters in total.

Proof of Lemma 5.22. First, we consider the case $\gamma(C) = 0$, so in each coordinate all vertex values are equal.

Claim 5.23. *If C is a cluster of vectors obtained by applying the reduction described in the proof of Theorem 5.2 to any graph H, $\gamma(C) = 0$, and $\binom{\ell}{2} < |C|$, then $\beta(C) \ge \ell + 1$.*

Proof. The proof is by induction on ℓ. The base is $\ell = 1$, and each non-empty cluster contains at least one vector and so at least 2 coordinates with vertices, we assume $\binom{1}{2} = 0$.

For the general case, if there are at least ℓ occurrences of a vertex v in a coordinate i, then there are at least $(\ell + 1)$ coordinates with vertices. Each vector with v in the i-th coordinate has also some other vertex in some other coordinate. As in each coordinate all vertex values are equal, it could not be that two of the vectors with the value v in the i-th coordinate share the second vertex-valued coordinate, since then they would represent the same edge.

So each coordinate has at most $(\ell - 1)$ vertex occurrences, otherwise the claim holds. Select a coordinate j which contains some vertex value u and remove the j-th coordinate and all vectors which have the value u in the j-th coordinate. That corresponds to the natural restriction C' of the cluster C to a subgraph $H - u$. The size of C' is at least $\binom{\ell}{2} + 1 - (\ell - 1) = \binom{\ell-1}{2} + 1$, and by induction there are at least ℓ coordinates which contain vertex values, so the original cluster C has at least $\ell + 1$ such coordinates, since there is also the j-th coordinate with the vertex value u. □

Now consider a cluster C with $\gamma(C) = 0$. Let ℓ be the largest value with $\binom{\ell}{2} + 1 \leq |C|$, so $|C| \leq \binom{\ell+1}{2}$. Since $|C| \leq \binom{k}{2}$, $\ell + 1 \leq k$. By Claim 5.23, $\beta(C) \geq \ell + 1$, then

$$\frac{\beta(C) - 2}{|C| - 1} \geq \frac{\ell - 1}{\binom{\ell+1}{2} - 1} = \frac{2}{\ell + 2} \geq \frac{2}{k + 1} = \kappa,$$

and so if $\ell + 1 < k$, the inequality is strict. It is also strict if $\ell + 1 = k$ and $|C| < \binom{k}{2}$, as the denominator becomes larger in the first step. Thus the only possibility of getting exactly κ is when $|C| = \binom{k}{2}$.

But then we have exactly $k \cdot (k - 1)$ vertex values across k coordinates, and each coordinate has at most $(k - 1)$ vertex values by the argument in Claim 5.23, so each coordinate must have exactly $(k - 1)$ vertex values. Since $\gamma(C) = 0$, they must be all equal. Denote the common vertex value in the i-th coordinate as v_i. Since each occurrence of v_i in the i-th coordinate corresponds to an edge to a different v_j, vertices $v_1, \ldots, v_k$ form a clique in G.

In the case $\gamma(C) > 0$, consider a new cluster C' which is obtained from C by removing all vectors which have a vertex-valued entry not equal to the center value. Assume for now that $|C'| \geq 2$. By the proof above, $\frac{\beta(C')-2}{|C'|-1} \geq \kappa$, since $\gamma(C') = 0$. The value $\frac{\beta(C)-2+\gamma(C)}{|C|-1}$ could be obtained from $\frac{\beta(C')-2}{|C'|-1}$ by adding $\gamma(C) + (\beta(C) - \beta(C')$ to the numerator and $|C| - |C'|$ to the denominator. Removing vectors could not increase β, so $\beta(C) - \beta(C') \geq 0$, and $\gamma(C) \geq |C| - |C'|$ since each of the removed vectors has at least one vertex value not equal to the center value. If $\frac{\beta(C')-2}{|C'|-1} \geq 1$, then the new fraction is also at least 1 and so strictly greater than κ. If $|C'| \leq 1$, then $\frac{\beta(C)-2+\gamma(C)}{|C|-1} \geq 1$ since $\beta(C) \geq 2$ and $\gamma(C) \geq |C| - |C'|$. If $\frac{\beta(C')-2}{|C'|-1} < 1$, then the new fraction became strictly larger, and so strictly larger than κ. In all cases, the inequality is strict when $\gamma(C) > 0$. □

Now to L_0-Cluster Selection: the reduction is almost the same, only we start from Multicolored Clique, and for each pair of indices $\{i, j\}$, $1 \leq i < j \leq k$ we obtain the set of vectors $X_{i,j}$ from edges in G starting in color i and ending in color j. The vectors are constructed in the same way as in the previous reduction. All weights are set to one. The value of D is the same, $D = \binom{k}{2}(k - 2)$.

Since vectors are constructed in the same way, all statements about the cost of grouping them remain valid, in particular Lemma 5.22. Only now the statement of

L_0-Cluster Selection already guarantees that we select exactly one cluster and exactly one vector from each $X_{i,j}$, so exactly one edge between each pair of colors. And by Lemma 5.22 only the proper k-clique has the optimal cost. □

Note that L_0-Cluster Selection is very similar to the known problem Consensus String With Outliers, studied e.g. in [32]. The only difference of Cluster Selection [0] is that we have to select one point from each of the given sets, whereas in Consensus String With Outliers the goal is to select the arbitrary subset of size $(n-k)$. The construction from Theorem 5.2 also shows W[1]-hardness of Consensus String With Outliers with respect to $(d+D+n-k)$ in the case of unbounded alphabet.

5.4 The L_∞ Distance

In this section, we consider the case $p = \infty$. We prove two hardness results for L_∞-k-Clustering: W[1]-hardness when parameterized by D and NP-hardness in the case $k = 2$.

First, we prove some useful facts about the structure of optimal cluster centers. The one thing, in which the L_∞ distance is harder than all other distances in our consideration, is that even when the cluster is given, we cannot simply find the optimal cluster center by optimizing the value in each coordinate independently. So there seems to be no simple rule of finding the optimal cluster center of a given cluster. However, one could still do that in polynomial time by solving a linear program, as we show in the next claim.

Claim 5.24. *Given a multiset C of vectors in $\mathbb{Z}^d$, there is a polynomial time algorithm to find $\mathbf{c} \in \mathbb{R}^d$ minimizing*

$$\sum_{\mathbf{x} \in C} \mathrm{dist}_\infty(\mathbf{x}, \mathbf{c}).$$

Proof. We reduce to solving a linear program, which we define next. Denote $C = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, introduce variables $c_1, \ldots, c_d$ corresponding to coordinates of the unknown cluster center and variables $d_1, \ldots, d_n$, where d_i corresponds to the value $\mathrm{dist}_\infty(\mathbf{x}_i, \mathbf{c})$. Consider the following linear program.

$$\sum_{i=1}^{n} d_i \to \min$$
$$\mathbf{x}_i[j] - c_j \le d_i \quad \forall\, i, j : 1 \le i \le n, 1 \le j \le d$$
$$c_j - \mathbf{x}_i[j] \le d_i \quad \forall\, i, j : 1 \le i \le n, 1 \le j \le d$$

Clearly, solving this linear program provides an optimal cluster center by the values $c_1, \ldots, c_d$. □

The next claim shows that we could only consider half-integral cluster centers.

Claim 5.25. *For any multiset C of vectors in $\mathbb{Z}^d$, the vector $\mathbf{c} \in \mathbb{R}^d$ which minimizes*

$$\sum_{\mathbf{x} \in C} \mathrm{dist}_\infty(\mathbf{x}, \mathbf{c})$$

could always be chosen from $\frac{1}{2}\mathbb{Z}^d$ (coordinates are either integer or half-integer).

Proof. Assume that we have an optimal solution $\mathbf{c}$ that has at least one coordinate not of the form $z/2$, $z \in \mathbb{Z}$. For $a \in \mathbb{R}$ we denote $\mathrm{frac}(a) = a - \lfloor a \rfloor$, and

$$\mathrm{rem}(a) = \begin{cases} \mathrm{frac}(a), & \text{if } \mathrm{frac}(a) < 1/2 \\ 1 - \mathrm{frac}(a), & \text{otherwise} \end{cases},$$

calling this value the *remainder* of a.

We could partition all coordinates into equivalence classes by remainder of $\mathbf{c}$. One could also define a partition of all vectors by the remainder of the distance to $\mathbf{c}$. These two partitions are related in the following sense: if $\mathrm{dist}_\infty(\mathbf{x}, \mathbf{c})$ has remainder ξ then each coordinate j where $|\mathbf{x}[j] - \mathbf{c}[j]| = \mathrm{dist}_\infty(\mathbf{x}, \mathbf{c})$ also has remainder ξ, and vice versa. Now we take one particular remainder and show that we can shift it without losing optimality.

There are two kinds of vectors with the particular remainder ξ: call *bottom* those vectors x for which $\mathrm{frac}(\mathrm{dist}_\infty(\mathbf{x}, \mathbf{c})) = \xi$, and call *top* those vectors $\mathbf{x}$ for which $\mathrm{frac}(\mathrm{dist}_\infty(\mathbf{x}, \mathbf{c})) = 1 - \xi$. Similarly, there are also two kinds of coordinates of $\mathbf{c}$, which we also call bottom and top depending of the value of $\mathrm{frac}(\mathbf{c}[j])$.

Consider a bottom coordinate j. Increasing $\mathbf{c}[j]$ increases $|\mathbf{x}[j] - \mathbf{c}[j]|$ for all bottom vectors $\mathbf{x}$, and decreases $|\mathbf{x}[j] - \mathbf{c}[j]|$ for all top vectors x. Decreasing $\mathbf{c}[j]$ does the opposite, as well as increasing a top coordinate. So if we take some sufficiently small value β and simultaneously increase all bottom coordinates and decrease all top coordinates by β then for all bottom vectors their distance will become larger by β, and for all top vectors — smaller by β. An if we do the opposite, the bottom vectors will cost less and the top vectors will cost more. Then, we could just take the group which has more vectors (bottom or top) and choose that action which decreases the distance for these vectors. The larger group has at least as many vectors as the smaller group, so the total distance does not increase.

It remains to see which value of β we could take. We could safely shift until we either reach a value in $\frac{1}{2}\mathbb{Z}$ or another remainder. In any case, we reduce the number of distinct remainders by one, and so we conclude the proof by doing this inductively over the number of distinct remainders. □

By Claim 5.25, the α-property holds with $\alpha = 1/2$, since at most one vector could be equal to the cluster center, and all others have distance at least $1/2$ due to half-integrality. We can also see that when the problem is parameterized by $d + D$, it admits an FPT algorithm..

Claim 5.26. *L_∞-k-Clustering distance is* FPT *when parameterized by $d + D$.*

Proof. We use Theorem 5.10. We have the α-property, and for the set $\mathcal{D}$ of all possible cluster costs not exceeding D we could take all half-integral values not exceeding D by Claim 5.25. All that remains is to solve L_∞-Cluster Selection in FPT time.

For that, we try all possible $\mathbf{x}_1 \in X_1$, and then try each possible resulting cluster center $\mathbf{c}$. Since $\text{dist}_\infty(\mathbf{x}_1, \mathbf{c}) \leq D$ and $\mathbf{c}$ is half-integral by Claim 5.25, we can try only vectors $\mathbf{c}$ of this form, and that is done in time $(2D + 1)^d$. □

5.4.1 W[1]-hardness Parameterized by D

Knowing that L_∞-k-Clustering is in FPT when parameterized by $d + D$, the next natural question is, is the problem FPT or W[1]-hard when parameterized only by D? We show that W[1]-hardness is the case, proving Theorem 5.3, which we recall here for convenience.

Theorem 5.3. *L_∞-k-Clustering parameterized by D and L_∞-Cluster Selection parameterized by $t + D$ are* W[1]*-hard.*

Proof. First, we show a reduction from Clique to L_∞-k-Clustering. Given a graph G and a clique size k, we construct the following instance of the clustering problem.

We set the dimension to $|V(G)| + \binom{|V(G)|}{2} - |E(G)|$. We take $|V(G)|$ vectors $\{\mathbf{x}_i\}_{i=1}^{|V(G)|}$ corresponding to vertices. For the vertex v, first $|V(G)|$ coordinates are set to zero, except v-th coordinate, which is set to 2.

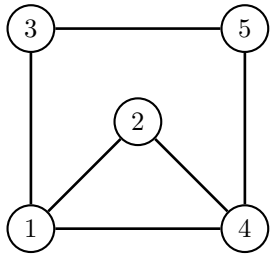
The last $\binom{|V(G)|}{2} - |E(G)|$ coordinates correspond to non-edges, vertex pairs which are not connected by an edge. For each vertex pair $\{u, v\} \notin E(G)$ in the coordinate $\{u, v\}$ we set $\mathbf{x}_u$ to 2, $\mathbf{x}_v$ to -2, the order on u, v is chosen arbitrarily, and all other vectors to zero.

Finally, we set the number of clusters to $|V(G)| - k + 1$ and the total distance to k. We show an example on how the reduction works in Figure 5.8.

If there is a clique of size k in G, then we have a solution of cost k: take k vectors corresponding to the clique vertices in one cluster, and make all other clusters trivial. For the only nontrivial cluster C, we can always choose $\mathbf{c}$ so that $|\mathbf{x}[j] - \mathbf{c}[j]| \leq 1$ for any $\mathbf{x} \in C$ and for any coordinate j. Each vertex coordinate has only 0 and 2, so setting $\mathbf{c}$ to 1 there suffices. As in C we have an edge between any two vertices, in any non-edge coordinate j there are either all zeros, or zeros and 2, or zeros and -2. In each of the cases there is a suitable value for $\mathbf{c}[j]$: 0, 1 or -1 correspondingly.

Next, we prove that any solution has cost at least k, and any solution which is not a k-clique has strictly larger cost. For that, we prove the following claim.

Claim 5.27. *In the instance above, the cost of any cluster C containing at least two vectors is at least $|C|$. If there is at least one non-edge in C, then the cost is at least $|C| + 1$.*



	1	2	3	4	5	23	34	15	25
$\mathbf{x}_1$	2	0	0	0	0	0	0	2	0
$\mathbf{x}_2$	0	2	0	0	0	2	0	0	2
$\mathbf{x}_3$	0	0	2	0	0	-2	2	0	0
$\mathbf{x}_4$	0	0	0	2	0	0	-2	0	0
$\mathbf{x}_5$	0	0	0	0	2	0	0	-2	-2

$\mathbf{x}_1$	2	0	0	0	0	0	0	2	0
$\mathbf{x}_2$	0	2	0	0	0	2	0	0	2
$\mathbf{x}_4$	0	0	0	2	0	0	-2	0	0
$\mathbf{c}$	1	1	0	1	0	1	-1	1	1

Figure 5.8: An example illustrating the reduction in Theorem 5.3: an input graph G, the vectors produced by the reduction (for clarity, the coordinates corresponding to vertices and to non-edges are separated), and the only composite cluster in the resulting optimal clustering of cost 3, corresponding to the clique on $\{1, 2, 4\}$. Note that $\mathrm{dist}_\infty(\mathbf{x}_1, \mathbf{c}) = \mathrm{dist}_\infty(\mathbf{x}_2, \mathbf{c}) = \mathrm{dist}_\infty(\mathbf{x}_4, \mathbf{c}) = 1$.

Proof. Denote the cluster center as $\mathbf{c}$. If each vector $\mathbf{x}$ in C has $\mathrm{dist}_\infty(\mathbf{x}, \mathbf{c}) \geq 1$, the first statement is trivial. So assume that there is a vector $\mathbf{x}^*$ in C such that $\mathrm{dist}_\infty(\mathbf{x}^*, \mathbf{c}) = \xi < 1$. Consider the coordinate j^* which corresponds to the same vertex as the vector $\mathbf{x}^*$, $\mathbf{x}^*[j^*] = 2$, and all other vectors are zero in the coordinate j^*. As $\mathrm{dist}_\infty(\mathbf{x}^*, \mathbf{c}) = \xi$, $\mathbf{c}[j^*] \geq 2 - \xi$. Then, for any other $\mathbf{x} \in C$, $\mathrm{dist}_\infty(\mathbf{x}, \mathbf{c}) \geq 2 - \xi > 1$. The total cost of the cluster is at least $\xi + (|C| - 1)(2 - \xi) = 2 + (|C| - 2)(2 - \xi) \geq |C|$, as $2 - \xi > 1$.

Now to the second part of the claim. Assume there are only two vectors in C and they do not have an edge, there is a coordinate j^* where one is 2 and the other is -2. No matter what we choose for $\mathbf{c}[j^*]$, the cost is at least $|2 - \mathbf{c}[j^*]| + |-2 - \mathbf{c}[j^*]| \geq 4$, and the statement follows. So assume that $|C| \geq 3$ and there is a coordinate j^* corresponding to a non-edge in C. One vector from C has 2 in the coordinate j^*, another -2, and all others have 0. Then there is a vector in C with distance to $\mathbf{c}$ of at least 2, as either $\mathbf{c}[j^*] \geq 0$ and $|-2 - \mathbf{c}[j^*]| \geq 2$ or $\mathbf{c}[j^*] < 0$ and $|2 - \mathbf{c}[j^*]| > 2$. Let us just forget about this vector and consider all other vectors in C. There are $|C| - 1 \geq 2$ of them, and by the reasoning in the proof of the first statement, their cost is at least $|C| - 1$. In this proof we considered only vertex coordinates, so the vector we forgot and the j^*-th coordinate (which is a non-edge coordinate) does not affect it. So, the total cost is at least $|C| - 1 + 2 = |C| + 1$. □

Assume that we have $l \geq 1$ nontrivial clusters of sizes $\{t_i\}_{i=1}^{l}$, nontrivial means that the size is at least two, $t_i \geq 2$ for $i \in \{1, \ldots, l\}$. By Claim 5.27, the total cost is

at least

$$\sum_{i=1}^{l} t_i = k + l - 1 \geq k,$$

as there are $|V(G)| - k + 1$ clusters in total, $|V(G)| - k + 1 - l$ trivial clusters, and the total number of vectors is $|V(G)| = \sum_{i=1}^{l} t_i + |V(G)| - k + 1 - l$, from which it follows that $\sum_{i=1}^{l} t_i = k + l - 1$. So no solution has cost less than k.

Also, if there are at least two nontrivial clusters, then $k + l - 1 \geq k + 1$. So if a solution has cost k, it must have only one nontrivial cluster, and its size must be k.

Finally, assume that the solution indeed has only one nontrivial cluster, but there is a non-edge in it. Then, as the size is k, by Claim 5.27 its cost is at least $k + 1$. So only a k-clique has cost k, which proves the correctness of the reduction.

Now, to L_∞-Cluster Selection. We consider essentially the same reduction, only we start from Multicolored Clique. We obtain sets of vectors $X_1, \ldots, X_k$ in the same way as X in the reduction above, only vectors obtained from vertices of color j are put into X_j. The total distance parameter is also set to k. So parameters t and D of the obtained instance have the same value as the starting parameter k.

Since vectors are constructed in the same way, Claim 5.27 still works. And now the statement of L_∞-Cluster Selection enforces that exactly one cluster of k vectors is selected. By Claim 5.27 it could be done with the cost k if and only if there is a colorful k-clique in the original graph. □

5.4.2 NP-hardness for $k = 2$

In this subsection we prove NP-hardness of L_∞-k-Clustering when $k = 2$. Intuitively, if we consider the previous reduction, partitioning the vectors optimally into two clusters loosely corresponds to partitioning the vertices into two sets such that there are as many as possible vertices having no edges inside their set. Which, in turn, is Odd Cycle Transversal: the problem of removing the smallest number of vertices so that the remaining graph is bipartite. However, to make everything really work, we need to consider a modified version of Odd Cycle Transversal which we call Half-Integral Odd Cycle Transversal.

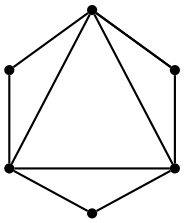
Half-Integral Odd Cycle Transversal

Input:	An undirected graph G, an integer t.
Task:	Is there an assignment $\delta : V(G) \to \{0, 1, 2\}$, such that $\sum_{v \in V(G)} \delta(v) \leq t$ and $G - S$ is bipartite, where $S = \{\{u, v\} \in E(G) : \delta(u) + \delta(v) \geq 2\}$?

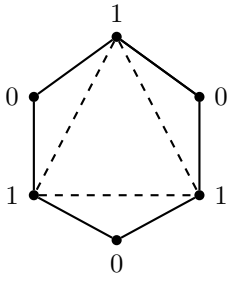
The definition of Half-Integral Odd Cycle Transversal is illustrated in Figure 5.9.

First we show that Half-Integral Odd Cycle Transversal is also NP-hard by constructing a reduction from 3-SAT.

(a) A graph G.



(b) An optimal assignment δ of cost 3. The set S, i.e. "deleted" edges, is in dashed.



(c) A different choice of δ. This assignment models the optimal solution for Odd Cycle Transversal on the same graph, and has the cost of 4.

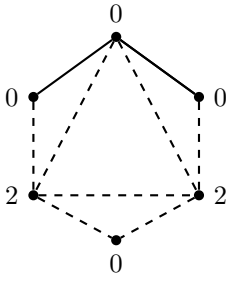


Figure 5.9: An illustration to Half-Integral Odd Cycle Transversal.

Lemma 5.28. *There is a polynomial time reduction from* 3-SAT *to* Half-Integral Odd Cycle Transversal.

Proof. Given an instance of 3-SAT with n variables and m clauses, make a graph G as follows. The example of the reduction is given in Figure 5.10. For each variable x_i, introduce two vertices x_i and x'_i, connect them with an edge. Also introduce $2n+1$ vertices $y_{i,j}$ connect them to both x_i and x'_i.

For each clause C_j introduce four vertices $C_{j,1},\ldots,C_{j,4}$. Consider following seven vertices: $C_{j,1}, \ldots, C_{j,4}$, and three variable vertices which are present in C_j: if $x_i \in C_j$ then we consider the vertex x_i, and if $\neg x_i \in C_j$ then we consider the vertex x'_i. Connect all these seven vertices in a cycle such that each variable vertex is adjacent to two clause vertices. Finally, set t to $2n$.

First, assume there is a satisfying assignment. Consider the following $\delta : V(G) \to \{0,1,2\}$: if x_i is true, $\delta(x_i) = 2$, otherwise $\delta(x'_i) = 2$, on all other vertices $\delta \equiv 0$. Clearly, $\sum_{v \in V(G)} \delta(v) = 2n$.

Since δ does not take value 1, deleting edges $\{u,v\}$ with $\delta(u) + \delta(v) \geq 2$ is equivalent to deleting vertices on which δ is 2. From each vertex gadget we deleted either x_i or x'_i, so the remaining part is a star with leaves $y_{i,j}$ and center x_i or x'_i. Since the assignment we started from is satisfying, from each clause cycle we deleted at least one vertex. So each cycle present in G lost at least one vertex, and what remains is bipartite.

Now assume there is a solution δ to the Half-Integral Odd Cycle Transversal instance. We claim that $\delta(x_i) + \delta(x'_i) \geq 2$ for each variable x_i. Consider a 2-coloring of $G - S$: either x_i and x'_i have the same color or not. In the former case, $\delta(x_i) + \delta(x'_i) \geq 2$ since the edge $\{x_i, x'_i\}$ must be removed.

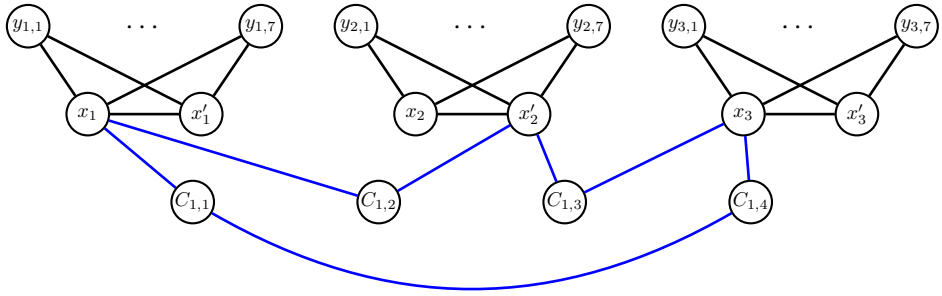


Figure 5.10: A graph obtained from the 3CNF-formula $(x_1 \vee \neg x_2 \vee x_3)$ by the reduction from Lemma 5.28. A 7-cycle corresponding to the only clause of the formula is highlighted in blue.

If x_i and x'_i have different colors, assume that $\delta(x_i) \leq 1$ and $\delta(x'_i) \leq 1$. Then, each of the $2n+1$ vertices $y_{i,j}$ takes one of the two colors, and so has an incident edge to x_i or x'_i which needs to be deleted. But then, $\delta(y_{i,j}) \geq 1$ for each j, and the total cost on these vertices is already $2n+1$. Then either $\delta(x_i) = 2$ or $\delta(x'_i) = 2$.

So we have n variables and δ is at least 2 on each pair of variable vertices, and in total δ is at most $2n$. Then δ has to be exactly 2 on each variable pair, and zero on all other vertices. Now we claim that on each clause cycle there is a variable vertex v with $\delta(v) = 2$. If not, then none of the cycle edges gets deleted, as δ is equal to zero on clause vertices. But then the remaining graph could not be bipartite, since it contains an odd cycle.

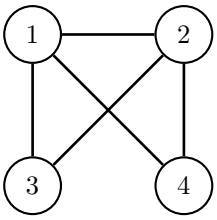
To get a satisfying assignment, set x_i to true if $\delta(x_i) = 2$, or to false otherwise. In particular, if $\delta(x'_i) = 2$, x_i is set to false, since $\delta(x_1) + \delta(x'_1) = 2$. Each clause is satisfied since each clause cycle contains a variable vertex on which δ is equal to 2. □

Now we prove NP-hardness of L_∞-k-Clustering when $k = 2$ by constructing a reduction from Half-Integral Odd Cycle Transversal.

Theorem 5.29. *L_∞-k-Clustering is NP-hard when $k = 2$.*

Proof. Consider an instance (G, t) of Half-Integral Odd Cycle Transversal, if $t \geq |V(G)|$, we have a yes-instance since $\delta \equiv 1$ deletes all edges from the graph, so we may assume $t < |V(G)|$. Remove all isolated vertices in G and add $t+5$ isolated edges to G, it clearly does not change the type of the instance. The number of clusters k is 2, set the dimension d to $|E(G)|$, each coordinate corresponds to an edge. For each vertex $v \in V(G)$ add a vector $\mathbf{x}_v$ to X with all coordinates set to zero. Then, for each edge $\{u, v\} \in E(G)$ set $\mathbf{x}_u[u, v]$ to 2 and $\mathbf{x}_v[u, v]$ to -2, the order on u, v is chosen arbitrarily. Finally, set D to $|V(G)| + t$. An example is given in Figure 5.11, additional isolated edges are dropped out for clarity.

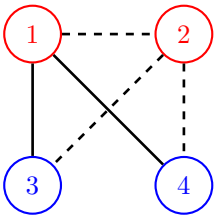
(a) A starting graph G, $t = 2$.



(b) The obtained instance: set of vectors $X = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$, $D = 6$.

edges:	12	13	14	23	24
$\mathbf{x}_1 = ($	2	2	2	0	0 $)$
$\mathbf{x}_2 = ($	-2	0	0	2	2 $)$
$\mathbf{x}_3 = ($	0	-2	0	-2	0 $)$
$\mathbf{x}_4 = ($	0	0	-2	0	-2 $)$

(c) A possible solution: $\delta(1) = \delta(3) = \delta(4) = 0$, $\delta(2) = 2$. Edges from S are dashed, a 2-coloring of $G - S$ is in red and blue.



(d) The corresponding clustering of cost 6, $C_1 = \{\mathbf{x}_1, \mathbf{x}_2\}$, $C_2 = \{\mathbf{x}_3, \mathbf{x}_4\}$, and optimal centers $\mathbf{c}_1$, $\mathbf{c}_2$.

$$\mathbf{c}_1 = (\ 1,\ 1,\ 1,\ 1,\ 1)$$
$$\mathbf{x}_1 = (\ 2,\ 2,\ 2,\ 0,\ 0), \quad \mathrm{dist}_\infty(\mathbf{x}_1, \mathbf{c}_1) = 1$$
$$\mathbf{x}_2 = (-2,\ 0,\ 0,\ 2,\ 2), \quad \mathrm{dist}_\infty(\mathbf{x}_2, \mathbf{c}_1) = 3$$

$$\mathbf{c}_2 = (\ 0, -1, -1, -1, -1)$$
$$\mathbf{x}_3 = (\ 0, -2,\ 0, -2,\ 0), \quad \mathrm{dist}_\infty(\mathbf{x}_3, \mathbf{c}_1) = 1$$
$$\mathbf{x}_4 = (\ 0,\ 0, -2,\ 0, -2), \quad \mathrm{dist}_\infty(\mathbf{x}_4, \mathbf{c}_1) = 1$$

Figure 5.11: An illustration of the reduction from Theorem 5.29.

If (G, t) is a yes-instance of Half-Integral Odd Cycle Transversal, consider the solution δ. Split vectors into clusters according to any proper 2-coloring of $G - S$. Now we show the way to select cluster centers so that each vertex v has distance at most $1 + \delta(v)$ to the corresponding center. We consider separately each of two clusters and each coordinate, indexed by an edge $\{u, v\} \in E(G)$. For a cluster C, there are three cases on how u and v are present in the cluster, for each of them we assign a particular value to the cluster center c in the coordinate $\{u, v\}$.

- If u and v are both not in C, for vectors in C all entries in the coordinate $\{u, v\}$ are zero, and we set $\mathbf{c}[u, v]$ also to zero. Each vector is at distance zero to the center in this coordinate.

- If only one of u and v are in C, for vectors in C all entries in the corresponding coordinate are zero, except one entry corresponding to the edge's endpoint belonging to C, which is either 2 or -2. Set $\mathbf{c}[u, v]$ to 1 or -1, correspondingly, then each vector is at distance 1 in this coordinate.

- If both u and v are in C, w.l.o.g. $\mathbf{x}_u[u, v]$ is 2 and $\mathbf{x}_v[u, v]$ is -2, and all other points are zero. It must hold that $\delta(u) + \delta(v) \geq 2$, either $\delta(u) = \delta(v) = 1$ or w.l.o.g $\delta(u) = 2$ and $\delta(v) = 0$. In the former case, set $\mathbf{c}[u, v]$ to zero, then all vectors have distance zero, $\mathbf{x}_u$ and $\mathbf{x}_v$ have distance 2 in this coordinate. In

the latter case, set $\mathbf{c}[u, v]$ to -1, then u is at distance 3, and all other vectors, including v, are at distance 1.

For any $v \in V(G)$, since it holds for all coordinates that distance from $\mathbf{x}_v$ to the corresponding cluster center is at most $1+\delta(v)$, then the L_∞ distance is also at most $1 + \delta(v)$, and the total cost of the clustering defined above is at most

$$\sum_{v \in V(G)} 1 + \delta(v) = |V(G)| + t.$$

In the other direction, assume there is a clustering C_1, C_2 with centers $\mathbf{c}_1$, $\mathbf{c}_2$ such that the total cost is at most D. By Claim 5.25 we may assume that centers are integral, and for any vector the distance to the nearest center is also an integer. We also may assume that centers are between -2 and 2 in each coordinate since all the input vectors have entries in this range, and so we could move the centers to the same range without increasing distances.

So, each vector has distance in $\{0, 1, 2, 3, 4\}$ to the closest center. We claim that it could not be that a vector $\mathbf{x}_v$ has distance zero: in this case w.l.o.g $\mathbf{x}_v = \mathbf{c}_1$, and so $\mathbf{c}_1$ is equal to 2 or -2 in some coordinate, since each vertex has at least one incident edge. But then each vector in C_1 has distance at least 2 to $\mathbf{c}_1$. And since at most two vectors could be equal to the centers, each of the remaining $|V(G)| - 2$ vectors has distance at least 1. Consider $t + 5$ isolated edges, at least $t + 3$ of them do not have any endpoint equal to one of $\mathbf{c}_1$ and $\mathbf{c}_2$. For these edges, the total distance of their endpoints is at least 3: either their endpoints are in different clusters, and so the endpoint in C_1 costs at least 2, or both endpoints are in the same cluster, and in total they cost 4 since there are simultaneously values 2 and -2 in the coordinate corresponding to this edge. So each of the $t+3$ edges increases the cost by additional 1, and the total cost is at least $|V(G)| - 2 + t + 3 > |V(G)| + t$.

Since each vector has distance at least 1, we may assume that the centers are in $\{-1, 0, 1\}^d$. If we have 2 (or -2) we could change it to 1 (or -1), all vectors which could become farther from the centers have 2 in this coordinate. But then the distance for these vectors is still at most 1. We also may assume that distances are in $\{1, 2, 3\}$, since distance 4 could be only from 2 to -2.

We claim that if we set $\delta(v) := \min_{i=1}^{2} \mathrm{dist}_\infty(\mathbf{x}_v, \mathbf{c}_i)$, δ is a solution to Half-Integral Odd Cycle Transversal. Remove all edges $\{u, v\}$ with $\delta(u)+\delta(v) \geq 2$, and consider 2-coloring of G induced by the partition $\{C_1, C_2\}$. Assume that we have an edge $\{u, v\}$ such that $\delta(u) + \delta(v) \leq 1$ and u and v are in the same cluster (w.l.o.g C_1). Then we have a coordinate $\{u, v\}$ such that w.l.o.g $\mathbf{x}_u[u, v] = 2$ and $\mathbf{x}_v[u, v] = -2$, but $\mathrm{dist}_\infty(\mathbf{x}_u, \mathbf{c}_1) + \mathrm{dist}_\infty(\mathbf{x}_v, \mathbf{c}_1) \leq 3$ due to $\delta(u) + \delta(v) \leq 1$ and so $|\mathbf{x}_u[u, v] - \mathbf{c}_1[u, v]| + |\mathbf{x}_v[u, v] - \mathbf{c}_1[u, v]| \leq 3$, which is a contradiction. So (G, t) is also a yes-instance. □

Note that the reduction from the proof of Theorem 5.29 also implements k-Coloring, if we set k to the number of colors and D to $|V(G)|$, since with such

a small budget we can not allow any same-colored neighbors in the optimal clustering. However, k-Coloring is only known to be NP-hard for 3 or colors. Thus in Theorem 5.29 we show a reduction from Half-Integral Odd Cycle Transversal to show the hardness of L_∞-k-Clustering even for two clusters.

5.5 The Case $p \in (1, \infty)$

In this section we consider the case $p \in (1, \infty)$, with the particular emphasis on the most commonly used case $p = 2$. With the L_2 distance, the k-Clustering problem is widely studied under the name k-Means.

5.5.1 FPT Parameterized by $d + D$ for L_2

When we consider both d and D as the parameters, L_2-Cluster Selection becomes FPT, and so L_2-k-Clustering is also FPT by Theorem 5.10.

Note that in any composite cluster, each vector except at most one is at distance at least $1/4$ from the center, so the α-property holds with $\alpha = 1/4$. Consider two different vectors, they have different values in some coordinate, and in this coordinate at least one of them is at distance at least $(1/2)^2 = 1/4$ from the center.

Now we prove Theorem 5.4, which we restate here.

Theorem 5.4. *L_2-k-Clustering and L_2-Cluster Selection are FPT when parameterized by $d + D$.*

Proof. We start with the proof that L_2-Cluster Selection is FPT. The distance dist_2 satisfies the α-property. Hence if $t > 4D + 1$ then any composite cluster costs more than D and the instance is clearly a no-instance. So we may assume that $t \leq 4D + 1$.

We claim that there are at most $4mtD$ possible total weights of the resulting composite cluster. First, in the resulting cluster there could be at most one vector with weight strictly larger than $4D$. Otherwise, let us consider two such vectors and the coordinate in which they differ. No matter which value the center has there, it is at distance of at least $1/2$ from at least one of the vectors, so the total cost is larger than $4D(1/2)^2 \geq D$. So there are at most m possibilities for the largest weight, and all of the other $(t - 1)$ weights are at most $4D$.

We fix the total resulting cluster weight W, the vector in the resulting cluster with the largest weight $\mathbf{x}_{j^*} \in X_{j^*}$, and the coordinate i. Since the center $\mathbf{c}$ is the mean of the vectors in the resulting cluster, $\mathbf{c}[i]$ is of form $\frac{y}{W}$, where $y \in \mathbb{Z}$. We claim that the distance from y to $W \cdot \mathbf{x}_{j^*}[i]$ is bounded by a function of D, and so each possible y could be enumerated in FPT time. Moreover, all possible centers could also be enumerated in FPT time since d is a parameter.

Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$ be the resulting cluster, $\mathbf{x}_j \in X_j$ for all $j \in \{1, \ldots, t\}$. The difference between $\mathbf{c}[i]$ and $\mathbf{x}_{j^*}[i]$ could be written as

$$\mathbf{x}_{j^*}[i] - \mathbf{c}[i] = \mathbf{x}_{j^*}[i] - \sum_{j=1}^{t} \frac{w(\mathbf{x}_j)\mathbf{x}_j[i]}{W} = \frac{\sum_{j=1}^{t} w(\mathbf{x}_j)(\mathbf{x}_{j^*}[i] - \mathbf{x}_j[i])}{W}.$$

The absolute value of the numerator is $\mathcal{O}(D^3)$ since $t = \mathcal{O}(D)$, $w(\mathbf{x}_{j^*})$ gets multiplied by zero, and all other weights are at most $4D$. Also, for any $j \in \{1, \ldots, t\}$, $|\mathbf{x}_{j^*}[i] - \mathbf{x}_j[i]| \leq 4D$, since

$$4D \geq 4\left((\mathbf{x}_{j^*}[i] - \mathbf{c}[i])^2 + (\mathbf{x}_j[i] - \mathbf{c}[i])^2\right) \geq (\mathbf{x}_{j^*}[i] - \mathbf{x}_j[i])^2 \geq |\mathbf{x}_{j^*}[i] - \mathbf{x}_j[i]|.$$

The total running time is at most

$$4mtd \cdot m \cdot \mathcal{O}(D^3)^d \cdot m,$$

since we try all possible cluster weights, all possible $\mathbf{x}_{j^*}$ out of the input vectors, then all possible centers which differ from $\mathbf{x}_{j^*}$ by $\mathcal{O}(D^3)$ in each coordinate. And then for each center we check whether the optimal cluster for it has cost at most D by selecting the best $\mathbf{x}_j \in X_j$ for each $j \in \{1, \ldots, t\}$. This concludes the proof that L_2-Cluster Selection is FPT when parameterized by $d + D$.

Now we proceed with showing that L_2-k-Clustering is FPT parameterized by $d + D$. For that we employ Theorem 5.10. We already have the α-property and an FPT algorithm for L_2-Cluster Selection. Hence the only thing left is to enumerate the set $\mathcal{D}$ of all possible optimal cluster costs not exceeding D.

Since there are n vectors in total, each cluster contains from 1 to n vectors. For each possible cluster size s the center is of the form $\frac{y}{s}$, where $y \in \mathbb{Z}$. Since input vectors have integer coordinates, the cost of any cluster of size s is of form $\frac{z}{s^2}$, where $z \in \mathbb{Z}$. And since the cost is at most D, $z \in \{0, \ldots, Ds^2\}$. We enumerate all possible cluster sizes in $\{1, \ldots, n\}$, and for each cluster size s all possible cluster costs in $\{0/s^2, \ldots, Ds^2/s^2\}$. In this way we obtain $\mathcal{D}$, and $|\mathcal{D}| = \mathcal{O}(Dn^3)$. □

5.5.2 W[1]-hardness Parameterized by $t + D$

In our setting, L_2-k-Clustering seems to be harder than L_1-k-Clustering, since we do not have the nice property that if many vectors have the same value in some coordinate then the center must also have this value. On the contrary, even if only one vector diverges from the rest, the optimal center also diverges. So the approach with enumerating nontrivial coordinate sets, which we successfully used in the $p \in (0, 1]$ case, is not likely to work.

We are able to prove that L_p-Cluster Selection for $p \in (1, \infty)$ is W[1]-hard parameterized by $t + D$. It remains open whether L_p-k-Clustering for $p \in (1, \infty)$ or specifically for $p = 2$ is W[1]-hard or not, but our result shows that at least the

approach we used to obtain an FPT algorithm in the $p \in (0, 1]$ case would not yield an FPT algorithm for $p \in (1, \infty)$.

First we state and prove two technical claims about the geometrical properties of clustering zero-one valued vectors in the $p \in (1, \infty)$ case.

Claim 5.30. *If we have a cluster of size $a+b$ where a vectors have zero and b vectors have one in the coordinate i, then the optimal center value in this coordinate is equal to*

$$\frac{b^{\frac{1}{p-1}}}{a^{\frac{1}{p-1}} + b^{\frac{1}{p-1}}},$$

and the coordinate i contributes

$$\frac{ab}{\left(a^{\frac{1}{p-1}} + b^{\frac{1}{p-1}}\right)^{p-1}},$$

to the total cost.

Proof. Assume that the center value in the coordinate i is equal to c, then the cost is

$$ac^p + b(1-c)^p.$$

It is easy to see that $c < 0$ is worse than $c = 0$, and similarly $c > 1$ is worse than $c = 1$, so we could restrict c to $[0, 1]$. The derivative with respect to c is

$$p(ac^{p-1} - b(1-c)^{p-1}),$$

as $p > 1$, the derivative is zero if and only if

$$\begin{aligned} ac^{p-1} &= b(1-c)^{p-1} \\ \left(\frac{c}{1-c}\right)^{p-1} &= \frac{b}{a} \\ \frac{c}{1-c} &= \left(\frac{b}{a}\right)^{\frac{1}{p-1}} \\ c = \frac{1}{1 + \left(\frac{a}{b}\right)^{\frac{1}{p-1}}} &= \frac{b^{\frac{1}{p-1}}}{a^{\frac{1}{p-1}} + b^{\frac{1}{p-1}}}. \end{aligned}$$

The derivative increases monotonically: when we increase c, c^{p-1} increases and $(1-c)^{p-1}$ decreases as $p - 1 > 0$. So the optimal value must be at its unique root defined by the expression above. Thus, the optimal cost is equal to

$$a\frac{b^{\frac{p}{p-1}}}{\left(a^{\frac{1}{p-1}} + b^{\frac{1}{p-1}}\right)^p} + b\frac{a^{\frac{p}{p-1}}}{\left(a^{\frac{1}{p-1}} + b^{\frac{1}{p-1}}\right)^p} = \frac{ab}{\left(a^{\frac{1}{p-1}} + b^{\frac{1}{p-1}}\right)^{p-1}},$$

finishing the proof. □

Now we prove that it is optimal to have as many ones in the same coordinate as possible. For that, we calculate how much each one adds to the total cost depending on how many ones are there in a coordinate.

Claim 5.31. *Consider a cluster of s zero-one valued vectors, denote as $f(b)$ the contribution of a coordinate in which there are b ones and $s - b$ zeros. The function $f(b)/b$ is strictly decreasing for $0 < b < s$.*

Proof. Denote the number of zeros in the coordinate as $a := s - b$. By Claim 5.30, the contribution of the coordinate per each one is

$$\frac{f(b)}{b} = \frac{ab}{\left(a^{\frac{1}{p-1}} + b^{\frac{1}{p-1}}\right)^{p-1}} \cdot \frac{1}{b} = \frac{a/s}{\left((a/s)^{\frac{1}{p-1}} + (1-a/s)^{\frac{1}{p-1}}\right)^{p-1}}.$$

Let us denote $x = a/s$, $0 < x < 1$, the derivative of the above with respect to x is equal to

$$\frac{d}{dx}\left(\frac{x}{\left(x^{\frac{1}{p-1}} + (1-x)^{\frac{1}{p-1}}\right)^{p-1}}\right) = \left(x^{\frac{1}{p-1}} + (1-x)^{\frac{1}{p-1}}\right)^{-(p-2)} \cdot \left((1-x)^{\frac{1}{p-1}} + x(1-x)^{\frac{1}{p-1}-1}\right),$$

which is strictly positive for $0 < x < 1$, hence proving the claim. □

Now we are ready to prove the hardness result, which was stated in the introduction as Theorem 5.5. We recall the statement here.

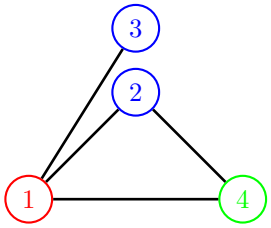
Theorem 5.5. *L_p-Cluster Selection is $\mathsf{W}[1]$-hard for every $p \in (1, \infty)$ when parameterized by $t + D$.*

Proof. We construct a reduction from Multicolored Clique. Given a graph G and a clique size k, we construct the following instance of L_p-Cluster Selection.

We set t to $\binom{k}{2}$, each input set of vectors represents a choice of an edge of the clique between two particular colors, so we number them by unordered pairs of indices from 1 to k. We set the dimension d to $|V(G)|$, coordinates are numbered by vertices.

The set $X_{i,j}$ consists of the following vectors: for each edge $\{u, v\} \in E(G)$ between a vertex u of color i and vertex v of color j, we add a vector with 1 in the coordinate u and 1 in the coordinate v, all other coordinates are set to zero. All vectors have weight one. Finally, we set

$$D = k \cdot \frac{(k-1)\binom{k-1}{2}}{\left((k-1)^{\frac{1}{p-1}} + \binom{k-1}{2}^{\frac{1}{p-1}}\right)^{p-1}}.$$



	1	2	3	4
$X_{1,2}$	1	1	0	0
	1	0	1	0
$X_{2,3}$	0	1	0	1
$X_{1,3}$	1	0	0	1

$\{1,2\}$	1	1	0	0
$\{2,4\}$	0	1	0	1
$\{1,4\}$	1	0	0	1
$\mathbf{c}$	$\frac{2}{3}$	$\frac{2}{3}$	0	$\frac{2}{3}$

Figure 5.12: An example illustrating the reduction in Theorem 5.5: an input graph G colored in three colors, the vector sets produced by the reduction, and the resulting optimal cluster of cost 2, corresponding to the clique on $\{1, 2, 4\}$. Note that in the resulting cluster, each non-zero coordinate has the maximal number of ones, $(k-1)$.

In Figure 5.12, we show the intuition behind the reduction by considering a simple example.

If there is a colorful k-clique in G then we construct a solution to our instance of L_p-Cluster Selection. Assume the clique is formed by vertices v_1, v_2, …, v_k, for each $i \in \{1, \cdots, l\}$ vertex v_i is of color i. From each $X_{i,j}$ choose the vector corresponding to the edge $\{v_i, v_j\} \in E(G)$. Among the chosen vectors, in every coordinate of the form v_i there are $(k-1)$ ones from edges to v_i and $\binom{k}{2} - (k-1) = \binom{k-1}{2}$ zeros. All other coordinates are zeros in the chosen vectors, so they do not contribute anything to the total distance. By Claim 5.30, the total distance is

$$k \cdot \frac{(k-1)\binom{k-1}{2}}{\left((k-1)^{\frac{1}{p-1}} + \binom{k-1}{2}^{\frac{1}{p-1}}\right)^{p-1}} = D.$$

In the other direction, we prove that only the solution described above could have the cost D, all others have strictly larger cost. First notice that in any resulting cluster there are at most $(k-1)$ ones in each coordinate, since for any vertex $v \in V(G)$, if we denote its color by i, only vectors from $(k-1)$ sets of the form $X_{i,j}$ ($j \in \{1, \ldots, k\} \setminus \{i\}$) have ones in the coordinate v, and we take one vector from each set by the definition of L_p-Cluster Selection.

Each vector has exactly two ones, so in any resulting cluster there are $2 \cdot \binom{k}{2}$ ones in total. By Claim 5.31, any resulting cluster which does not have $(k-1)$ ones in k coordinates has strictly larger cost, since only coordinates with exactly $(k-1)$ ones have the optimal cost per each one.

So, if the resulting cluster has the cost D, then there are k coordinates such that in each of them exactly $(k-1)$ of the chosen vectors have one. We show that in this case the original instance of Clique has a k-clique. For any color $i \in \{1, \ldots, k\}$ there are at most $(k-1)$ ones in all coordinates indexed by vertices of color i in the resulting cluster. So all of these ones are in the same coordinate v_i for some v_i.

We claim that the vertices $v_1, \ldots, v_k$ form a clique. Consider vertices v_i and v_j, we have taken some vector from $X_{i,j}$, and this vector must have added a one to the coordinates v_i and v_j, then by construction the edge $\{v_i, v_j\}$ is in $E(G)$. □

5.6 Conclusion and Open Problems

In this chapter, we presented an FPT algorithm for L_p-k-Clustering with $p \in (0, 1]$ parameterized by D. However, for the case $p \in (1, \infty)$ we were only able to show the W[1]-hardness of L_p-Cluster Selection. While intractability of L_p-Cluster Selection does not exclude that L_p-k-Clustering could be FPT with $p \in (1, \infty)$, it indicates that the proof of this (if it is true at all) would require an approach completely different from ours. Thus an interesting open question concerns the parameterized complexity of L_p-k-Clustering with $p \in (1, \infty)$ and parameter D.

Another open question is about the fine-grained complexity of L_p-k-Clustering when parameterized by $k + d$. For several distances, we know XP-algorithms: an $\mathcal{O}(n^{dk+1})$ algorithm by Inaba et. al. [118] for $p = 2$, as well as trivial algorithms for $p \in [0, 1]$. For the case when the possible cluster centers are given in the input, the matching lower bound is shown in [63]. However, we are not aware of a lower bound complementing the algorithmic results in the case when any point in Euclidean space can serve as a center.

Finally, let us note that our W[1]-hardness reductions could be easily adapted to obtain ETH-hardness results. Our reductions are from Clique and, assuming ETH, there is no $n^{o(k)}$ algorithm for Clique. In most of our results, the ETH lower bounds derived from our reductions, can be complemented by matching upper bounds through a trivial algorithm for L_p-Cluster Selection in time $n^{\mathcal{O}(d)}$ or $n^{\mathcal{O}(t)}$ and, consequently, an algorithm for L_p-k-Clustering obtained by Theorem 5.10. However, the reduction in Theorem 5.5 excludes only an $(nd)^{o(t^{1/2}+D^{1/2})}$ algorithm for L_p-Cluster Selection with $p \in (1, \infty)$ under ETH. Both the trivial algorithm in time $n^{\mathcal{O}(t)}$ and the algorithm from Theorem 5.4 in time $D^{\mathcal{O}(d)}$ (which could also be turned into a $d^{\mathcal{O}(D)}$-time algorithm) fail to match this lower bound. So, another open question is, whether there exists a better reduction or a subexponential algorithm could be obtained in this case.

6

Robust Categorical Clustering

In this chapter, we are interested in studying clustering problems on *categorical* data. Naturally, the most common similarity (or dissimilarity) measure for categorical data objects is the Hamming distance, which is basically the number of mismatched attributes of the objects. As Hamming distance is a metric, the existing constant approximations for clustering problems in general metrics can be easily adapted for categorical data. However, these algorithms need the whole metric space (points, potential centers and all-pair distances) as input, whose size depends exponentially on the dimension of the Hamming space. By increasing the cost by a constant factor, one can assume that the potential centers are the input points. This leads to polynomial time approximations, albeit with large constant approximation factors. Consequently, researchers have focused on obtaining clustering algorithms with better performance guarantees tailored for Hamming spaces. While a large number of such algorithms have been proposed in recent years, the development of provably good algorithms remains an intriguing challenge [11]. Among these results, the most relevant to our work is the one in [90], which shows that the k-median clustering problem on binary data is FPT parameterized by the cost of clustering. While the previous chapter covers the generalization of this work to integer-valued vectors under various L_p-norms, here we consider a different extension. As discussed in Chapter 3, our main interest for categorical clustering lies in the study of robust variants of the problem. Specifically, we are interested in Categorical k-Clustering with Row Outliers, where it is allowed to remove a given number of points from the dataset, and Categorical k-Clustering with Column Outliers, where instead we remove a predefined number of features. In both variants, the removed rows/columns do not contribute to the cost of clustering. Thus the task in both problems can be thought of as to remove the noisy part of data so that the remaining part fits the Categorical k-Clustering objective the best. Again, we are primary interest in the exact parameterized complexity of these problems with the respect to the parameter D, representing the cost of the target clustering. We refer to Chapter 5 for

the discussion on the choice of the parameter; here we note that in the robust setting parameterizing by D becomes more powerful, as the removed rows/columns do not contribute to the cost. We recall the formal definitions of the two problems next. Let Σ be a finite set of non-negative integers. We refer to Σ as the alphabet and we denote the d-dimensional space over Σ by Σ^d.

Categorical k-Clustering with Row Outliers

Input: An alphabet Σ, an $n \times d$ matrix $\boldsymbol{A}$ with rows $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ such that $\mathbf{a}_i \in \Sigma^d$ for all $1 \leq i \leq n$, a positive integer k, non-negative integers D and ℓ.

Task: Decide whether there is a subset $O \subset \{1, 2, \ldots, n\}$ of size at most ℓ, a partition of $\{1, 2, \ldots, n\} \setminus O$ into k sets $\{I_1, I_2, \ldots, I_k\}$ called clusters, and vectors $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k \in \Sigma^m$ such that the cost of clustering is at most D, that is,

$$\sum_{t=1}^{k} \sum_{i \in I_i} d_H(\mathbf{a}_i, \boldsymbol{c}_t) \leq D.$$

Now we recall the feature selection variant. For a set of indices $S \subset \{1, 2, \ldots, d\}$ and an $n \times d$ matrix $\boldsymbol{A}$, $\boldsymbol{A}^{-S}$ is the matrix obtained from $\boldsymbol{A}$ by removing the columns with indices in S.

Categorical k-Clustering with Column Outliers

Input: An alphabet Σ, an $n \times d$ matrix $\boldsymbol{A}$ with rows $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ such that $\mathbf{a}_j \in \Sigma^d$ for all $1 \leq i \leq n$, a positive integer k, non-negative integers D and ℓ.

Task: Decide whether there is a subset $O \subset \{1, 2, \ldots, d\}$ of size at most ℓ, a partition $\{I_1, I_2, \ldots, I_k\}$ of $\{1, 2, \ldots, n\}$, and vectors $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k \in \Sigma^{d-|O|}$ such that

$$\sum_{t=1}^{k} \sum_{i \in I_i} d_H(\boldsymbol{a}_i^{-O}, \boldsymbol{c}_t) \leq D.$$

For a set of indices $S \subset \{1, 2, \ldots, m\}$ and an $m \times n$ matrix $\boldsymbol{A}$, let $\boldsymbol{A}^{-S}$ be the matrix obtained from $\boldsymbol{A}$ by removing the rows with indices in S. We denote the rows of $\boldsymbol{A}^{-S}$ by $\boldsymbol{a}_j^{-S}$ for $1 \leq j \leq n$.

While Categorical k-Clustering with Column Outliers looks very similar to the one with row outliers, it appears that the former is computationally much more challenging. Later, we will discuss more about this. Our first result is the following theorem.

Theorem 6.1. Categorical k-Clustering with Column Outliers *is solvable in time* $f(k, D, |\Sigma|) \cdot d^{g(k,|\Sigma|)} \cdot n^2$*, where* f *and* g *are computable functions.*

In particular, this implies that for fixed k and $|\Sigma|$, the problem is FPT parameterized by D. Note that the running time of the algorithm does not depend on the number of irrelevant features ℓ. In particular, for fixed k, D, and $|\Sigma|$, it runs in polynomial time even when $\ell = \Omega(d)$. Also, the theorem could be used to identify the minimum number of irrelevant features ℓ such that the cost of k-clustering on the remaining features does not exceed D. Note that our time complexity also exponentially depends on the number of clusters k. In this regard, one can compare our result with the result in [90] that shows that the binary version of the problem without the row outliers is FPT parameterized by only D. However, in the presence of the row outliers the dependence on k is unavoidable as we state in our next theorem.

Theorem 6.2. Categorical k-Clustering with Column Outliers *is* W[1]*-hard parameterized by*

- *either* $k + (d - \ell)$
- *or* ℓ

even when $D = 0$ *and* $\Sigma = \{0, 1\}$*. Moreover, assuming the Exponential Time Hypothesis (ETH), the problem cannot be solved in time* $f(k) \cdot d^{o(k)} \cdot n^{\mathcal{O}(1)}$ *for any function* f*, even when* $D = 0$ *and the alphabet* Σ *is binary.*

Note that when $D = 0$ and $\Sigma = \{0, 1\}$, from Theorem 6.1 it follows that Categorical k-Clustering with Column Outliers can be solved in time $f(k) \cdot d^{g(k)} \cdot n^2$. Theorem 6.2 shows that the dependence on such a function g on k is inevitable, unless W[1] = FPT, and $g(k)$ is unlikely to be sublinear up to ETH. Theorem 6.1 follows from a more general algorithm for Constrained Clustering with Outliers. In this problem, one seeks a clustering with centers of clusters satisfying the property imposed by a set of relations. Constrained clustering was introduced in [88] as the tool in the design of approximation algorithms for binary low-rank approximation problems. The Constrained Clustering with Outliers problem is basically the robust variant of this problem. This problem encompasses a number of well-studied problems related to robust clustering and dimensionality reduction, and our algorithm for constrained clustering implies fixed-parameter tractability for all these problems. Now we move to define Constrained Clustering with Outliers formally.

A *p-ary relation* on Σ is a collection of p-tuples whose elements are in Σ. For example, $R = \{(0, 1, 0), (1, 1, 0)\}$ is a 3-ary relation on $\{0, 1\}$.

Definition 6.3 (**Vectors satisfying** $\mathcal{R}$)**.** *An ordered set* $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_p\}$ *of* d*-dimensional vectors in* Σ^d *is said to satisfy a set* $\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$ *of* p*-ary relations on* Σ *if for all* $1 \leq i \leq d$*, the* p*-tuple formed by the* i*-th coordinates of vectors from* C*, that is* $(\boldsymbol{c}_1[i], \boldsymbol{c}_2[i], \ldots, \boldsymbol{c}_p[i])$*, belongs to* R_i*.*

For example, consider $\Sigma = \{0, 1\}$, $d = 2$, $p = 3$, and relation $\mathcal{R} = \{R_1, R_2\}$, where $R_1 = \{(0, 1, 0), (1, 1, 0)\}$, $R_2 = \{(1, 0, 0), (1, 1, 1)\}$. Then the set with

$$\boldsymbol{c}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \boldsymbol{c}_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \boldsymbol{c}_3 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

satisfies $\mathcal{R}$, as $(\boldsymbol{c}_1[1], \boldsymbol{c}_2[1], \boldsymbol{c}_3[1]) = (0, 1, 0) \in R_1$ and $(\boldsymbol{c}_1[2], \boldsymbol{c}_2[2], \boldsymbol{c}_3[2]) = (1, 0, 0) \in R_2$. On the other hand, the set with

$$\boldsymbol{c}_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \boldsymbol{c}_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{c}_3 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

does not satisfy $\mathcal{R}$, because $(\boldsymbol{c}_1[1], \boldsymbol{c}_2[1], \boldsymbol{c}_3[1]) = (0, 0, 0) \notin R_1$.

We define the following constrained variant of robust categorical clustering.

Constrained Clustering with Outliers

Input: An alphabet Σ, an $n \times d$ matrix $\boldsymbol{A}$ with rows $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ such that $\mathbf{a}_j \in \Sigma^d$ for all $1 \leq j \leq n$, a positive integer k, non-negative integers D and ℓ, a set $\mathcal{R} = \{R_1, R_2, \ldots, R_d\}$ of k-ary relations on Σ.

Task: Decide whether there is a subset $O \subset \{1, 2, \ldots, n\}$ of size at most ℓ, a partition $\mathcal{I} = \{I_1, I_2, \ldots, I_k\}$ of $\{1, 2, \ldots, n\} \setminus O$, and a set $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$ of d-dimensional vectors in Σ^d such that C satisfies $\mathcal{R}$ and

$$\sum_{i=1}^{k} \sum_{j \in I_i} d_H(\mathbf{a}_j, \boldsymbol{c}_i) \leq D.$$

Thus in Constrained Clustering with Outliers we want to identify a set of outliers $\mathbf{a}_i, i \in O$, such that the remaining $n - \ell$ vectors could be partitioned into k *clusters* $\{I_1, I_2, \ldots, I_k\}$. Each cluster I_j could be identified by its *center* $\boldsymbol{c}_j \in \Sigma^d$ as the set of vectors that are closer to $\boldsymbol{c}_j \in \Sigma^d$ than to any other center (ties are broken arbitrarily). Then the cost of each cluster I_j is the sum of the Hamming distances between its vectors and the corresponding center $\boldsymbol{c}_j \in \Sigma^d$. However, there is an additional condition that the set of cluster centers $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$ must satisfy the set of k-ary relations $\mathcal{R}$. And, the total sum of costs of all clusters must not exceed D.

We prove the following theorem.

Theorem 6.4. Constrained Clustering with Outliers *is solvable in time* $(kD)^{O(kD)}|\Sigma|^{kD} \cdot n^{O(k)} \cdot d^2$.

Note that Categorical k-Clustering with Row Outliers is also a restricted version of Constrained Clustering with Outliers corresponding to the case when every relation $R_i \in \mathcal{R}$ contains all possible k-tuples over Σ, that is, there are no constraints on the centers. Hence, by Theorem 6.4, we readily obtain the same result for this problem. However, in this special case we show that it is possible to obtain an improved result that matches the result without outliers.

Theorem 6.5. Categorical k-Clustering with Column Outliers *is solvable in time* $2^{O(D \log D)}|\Sigma|^D \cdot (nd)^{O(1)}$.

In particular, the theorem implies that the problem is FPT parameterized by D and $|\Sigma|$. We note that the running time of Theorem 6.5 matches the running time in [90] obtained for the (vanilla) k-median clustering problem without outliers on binary data. The interesting feature of the theorem is that the running time of the algorithm does not depend on the number of outliers ℓ, matching the bound of the problem without outliers. Most of the clustering procedures in robust statistics, data mining and machine learning perform well only for small number of outliers. Our theorem implies that if all of the inlier points could be naturally partitioned into k distinct clusters with small cost, then such a clustering could be efficiently recovered even after arbitrarily many outliers are added. Indeed, the time complexity in all of our results are independent of the number of outliers ℓ, which essentially portray the same kind of behavior. One of our main contributions is to design algorithms that return exact solutions and are insensitive to the amount of noise and irrelevant features as well.

Beyond the proof of Theorem 6.1, Theorem 6.4 has a number of interesting applications on low-rank approximation type problems. In particular, it implies a number of FPT algorithms for Robust L_0-Low Rank Approximation, Robust Low Boolean-Rank Approximation and Robust Projective Clustering. We define these problems and explain the connections with Theorem 6.4 later in this chapter.

Both of our algorithmic results, Theorems 6.4 and 6.5, have at their core the sub-hypergraph enumeration technique introduced by Marx [156]. This is fairly natural, since our algorithms solve generalized versions of the vanilla binary clustering problem, and the only known FPT algorithm [90] for the latter problem parameterized by D relies on the hypergraph enumeration as well. In fact, our algorithm for Categorical k-Clustering with Row Outliers closely follows this established approach of applying the hypergraph construction to clustering problems, presented in the previous chapter for L_p-k-Clustering. However, for the Constrained Clustering with Outliers problem the existing techniques do not work immediately. To deal with this, we generalize the previously used hypergraph construction.

In what remains of this chapter, we present the algorithms and lower bounds discussed above. We begin with the simpler case of Categorical k-Clustering with Row Outliers in Section 6.1. In Section 6.2 we discuss implications of the

Constrained Clustering with Outliers algorithm for other problems, including Categorical k-Clustering with Column Outliers and several low-rank approximation problems. Then in Section 6.3 we move on to the main result, the FPT algorithm for Constrained Clustering with Outliers. We present the hardness result for Categorical k-Clustering with Column Outliers in Section 6.4. Finally, in Section 6.5, we conclude this chapter with some open problems.

6.1 Categorical k-Clustering with Row Outliers

In this section we give a proof of Theorem 6.5. Let us recall that the theorem states that Categorical k-Clustering with Row Outliers *is solvable in time* $2^{O(D \log D)}|\Sigma|^D(nd)^{O(1)}$.

First, we briefly discuss the intuition of the algorithm. Given an instance $(\boldsymbol{A}, k, D, \ell)$ of Categorical k-Clustering with Row Outliers, we note that at most $2D$ distinct rows can belong to "nontrivial" clusters (with at least 2 distinct rows), exactly like in the case without the outliers. So we employ a color-coding scheme to partition the rows in a way so that every row belonging to a nontrivial cluster of a fixed feasible solution is colored with its own color. Thus we reduce to multiple instances of the problem we call Cluster Selection. In Cluster Selection, we are given sets of rows $U_1, U_2, \ldots, U_p$ and a parameter D. The goal is to select p rows $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_p$ and a cluster center $\boldsymbol{s}$ such that $\boldsymbol{b}_t \in U_t$ for $1 \leq t \leq p$ and $\sum_{t=1}^{p} d_H(\boldsymbol{b}_t, \boldsymbol{s}) \leq D$. Up to the change to categorical data, this is the same problem as defined in [90] under the same name, and as L_0-Cluster Selection defined in the previous chapter. The hypergraph-based algorithm for it is essentially the same too, although we present it for completeness in this section.

A specific issue we need to deal with in the Categorical k-Clustering with Row Outliers problem is the following. If we encounter a number of identical rows, we need to treat them as an indivisible set (*initial cluster*), as only a cluster with sufficiently many different rows can have large cost. However, a potential flaw of this approach is that in an optimal solution, identical rows can belong to different clusters and to the set of outliers. To handle this, we prove that the optimal solution can always be "normalized". Namely, we can rearrange clusters of the optimal solution in a way that at most one initial cluster is split between a solution cluster and the set of outliers, and every other initial cluster is completely contained in either one of the resulting clusters or the set of outliers. Then, guessing the one initial cluster being split solves the issue. After the color-coding and solving Restricted Clustering, each remaining initial cluster is either a trivial cluster in the solution, or belongs to the outliers. This can be decided greedily.

Now we move to the formal proof of Theorem 6.5. Let $(\boldsymbol{A}, k, D, \ell)$ be the input to Categorical k-Clustering with Row Outliers, where $\boldsymbol{A}$ is an $n \times d$ matrix, k is the number of clusters, D is the cost of the solution and ℓ is the number of outliers.

Let β be the number of pairwise distinct rows in $\boldsymbol{A}$. We start with partitioning

the rows of $\boldsymbol{A}$ into sets of identical rows $\mathcal{J} = \{J_1, J_2, \ldots, J_\beta\}$. That is, all rows in each J_i are equal. We refer to such a set J_i as an *initial cluster*.

We show that for any yes-instance, there is a feasible solution such that the rows of at most one initial cluster are "split" by the solution. More formally, consider a clustering $\mathcal{C} = \{I_1, I_2, \ldots, I_k\}$ of $\{1, 2, \ldots, n\} \setminus O$. Then each of the initial clusters J_i is exactly one of the following types *with respect to* $\mathcal{C}$.

(i) $\{J_i\}$ such that there is t with $J_i \subseteq I_t$,

(ii) $\{J_i\}$ such that J_i is not of type (i), and $J_i \subseteq \cup_{j=1}^{k} I_j$,

(iii) $\{J_i\}$ such that $J_i \subseteq O$,

(iv) $\{J_i\}$ such that $J_i \not\subseteq \cup_{j=1}^{k} I_j$, $J_i \not\subseteq O$, and there is t with $J_i \subseteq I_t \cup O$, and

(v) $\{J_i\}$ such that $J_i \not\subseteq \cup_{j=1}^{k} I_j$, $J_i \not\subseteq O$, and there is no t with $J_i \subseteq I_t \cup O$.

Lemma 6.6. *Let $(\boldsymbol{A}, k, D, \ell)$ be a yes-instance of* Categorical k-Clustering with Row Outliers. *Then there is a set $O \subseteq \{1, 2, \ldots, n\}$ of size ℓ and k-clustering $\mathcal{C}$ of $\{1, 2, \ldots, n\} \setminus O$ of cost at most D such that every initial cluster of $\boldsymbol{A}$ with respect to $\mathcal{C}$ is of type (i), (iii) or (iv). Moreover, there is at most one initial cluster of type (iv).*

Proof. Because $(\boldsymbol{A}, k, D, \ell)$ is a yes-instance, there is a set of outliers O and a k-clustering of the remaining vectors of cost at most D. Among all such clusterings, we select a clustering $\mathcal{C} = \{I_1, I_2, \ldots, I_k\}$ of $\{1, 2, \ldots, n\} \setminus O$ of cost at most D such that with respect to $\mathcal{C}$

$$\text{the number of initial clusters of type (ii) is minimum,} \tag{6.1}$$

subject to (6.1)

$$\text{the number of initial clusters of type (v) is minimum,} \tag{6.2}$$

and subject to (6.1) and (6.2),

$$\text{the number of initial clusters of type (iv) is minimum.} \tag{6.3}$$

We claim that $\mathcal{C}$ is the required clustering. Targeting towards a contradiction, assume first that there is an initial cluster of $\mathcal{J}$ of type (ii) with respect to $\mathcal{C}$. Without loss of generality, let this cluster be J_1. Then J_1 is not fully contained in any cluster of $\mathcal{C}$, but it is fully contained in the union of all clusters. We convert J_1 into a type (i) initial cluster by the following modification of $\mathcal{C}$. Let $\boldsymbol{c}_i$ be the center of cluster I_i for $i \in \{1, \ldots, k\}$. Also, let $\mathbf{a}$ be the unique row corresponding to J_i (the whole initial cluster consists of rows equal to $\mathbf{a}$). Let $\boldsymbol{c}_q$ be a center that minimizes the cost $d_H(\mathbf{a}, \boldsymbol{c}_i)$ over all centers $\boldsymbol{c}_i$. We construct a new clustering $\mathcal{C}'$ by reassigning the

rows of J_1 to the cluster I_q. Thus in the new clustering J_1 becomes of type (i). The assignment of all other initial clusters remains the same. Since $d_H(\mathbf{a}, \boldsymbol{c}_q) \leq d_H(\mathbf{a}, \boldsymbol{c}_1)$, we have that the cost of the new clustering $\mathcal{C}'$ is at most D, thus contradicting assumption (6.1). Hence, there are no initial clusters of $\mathcal{J}$ of type (ii) with respect to $\mathcal{C}$.

By applying the same arguments to the non-outlier part of a type (v) initial cluster with respect to $\mathcal{C}$. The only difference that now type (v) initial cluster becomes type (iv) thus contradicting assumption (6.2).

Finally, we already know that there are no types (ii) and (v) initial clusters with respect to $\mathcal{C}$. Assume that there are at least two, say J_1 and J_2, initial clusters of type (iv) with respect to $\mathcal{C}$. Then, there are clusters C_{i_1} and C_{i_2} in $\mathcal{C}$ such that $J_1 \subseteq C_{i_1} \cup O$ and $J_2 \subseteq C_{i_2} \cup O$. (We do not exclude the possibility of $i_1 = i_2$ here.) Let $\mathbf{c}_{i_1}$ and $\mathbf{c}_{i_2}$ be the centers of C_{i_1} and C_{i_2} respectively. Let also $\mathbf{a}_1$ and $\mathbf{a_2}$ be the unique rows corresponding to J_1 and J_2. Without loss of generality, we assume that $d_H(\mathbf{a}_1, \mathbf{c}_{i_1})) \leq d_H(\mathbf{a}_2, \mathbf{c}_{i_2}))$. There are two cases: (1) $|J_2 \cap C_{i_2}| \leq |J_1 \cap O|$, and (2) $|J_2 \cap C_{i_2}| > |J_1 \cap O|$. In the first case, we can assign additional $|J_2 \cap C_{i_2}|$ rows of J_i to C_{i_1}, and move $|J_t \cap C_{i_2}|$ rows of J_2 from C_{i_2} to O. The cost increase is $|J_2 \cap C_{i_2}| \cdot (d_H(\mathbf{a}_1, \mathbf{c}_{i_1}) - d_H(\mathbf{a}_2, \mathbf{c}_{i_2})) \leq 0$. Also, type of J_2 is now changed to (iii) with respect to the new clustering. Thus, the number of type (iv) initial clusters has decreased by at least 1 and no type (ii) or (iv) clusters were created. For the second case, assign the $|J_1 \cap O|$ rows of J_1 from O to C_{i_1}, and move $|J_1 \cap O|$ more rows of J_2 from C_{i_2} to O. The cost increase is $|J_1 \cap O| \cdot (d_H(\mathbf{a}_1, \mathbf{c}_{i_1}) - d_H(\mathbf{a}_2, \mathbf{c}_{i_2})) \leq 0$. Also, type of J_1 is now changed to (i) with respect to the new clustering. Thus, in this case also, the number of type (iv) initial clusters has decreased by 1. In both cases we constructed clusterings contradicting (6.3). Hence there is at most one initial cluster of type (iv) with respect to $\mathcal{C}$. □

Now, we describe our algorithm. The algorithm tries to find a feasible clustering (if any) that satisfies the property of Lemma 6.6. As there is at most one initial cluster that can get split, we can guess it in advance. Suppose J_t be this initial cluster. We can also guess the number of rows ℓ' of J_t that would be part of the outliers set. We construct a new instance of the problem which contains all the rows in J_i for $1 \leq i \leq \beta$ and $i \neq t$, and exactly $|J_t| - \ell'$ rows of J_t. Let $\boldsymbol{A}'$ be the matrix $\boldsymbol{A}$ containing these rows. Then, our new instance is $(\boldsymbol{A}', k, D, \ell - \ell')$. The advantage of working with this instance is that in the desired clustering no initial cluster gets split. Next, we show how to find such a clustering for this instance.

For simplicity of notations, let us denote the input instance by $(\boldsymbol{A}, k, D, \ell)$. As no initial cluster in the desired clustering gets split, an initial cluster can either form its own cluster, or becomes part of the outliers, or gets merged with other initial clusters to form a new cluster. We refer to the last type of clusters as composite clusters. Now, we have the following simple observation.

Observation 6.7. *In a feasible clustering, the total number of initial clusters that*

can be part of the composite clusters is at most $2D$. The total number of composite clusters is at most D.

Next, we use the color coding technique of [10] to randomly color the initial clusters by one of the colors in $\{1, 2, \ldots, 2D\}$. This is done analogously to Section 5.1, where we reduced L_p-k-Clustering to L_p-Cluster Selection in the case of clustering integer-valued points with the distance dist_p. The idea is that with good probability the initial clusters that form the composite clusters get colored by pairwise distinct colors. Thus, assuming this event occurs, it is sufficient to select only one initial cluster from each color class to find out such initial clusters. However, we do not know how to combine those initial clusters. Nevertheless, we can guess such combination by trying all possible partitions of $\{1, 2, \ldots, 2D\}$ that contains at most D parts. For each part, we can also guess the cost of the corresponding composite cluster. By Observation 6.7, the selected initial clusters corresponding to each part form a composite cluster. Each initial cluster which is not selected would either form its own cluster or become part of the outliers. However, an initial cluster that forms its own cluster does not incur any cost. Hence, any subsets of the appropriate number of non-selected initial clusters can form their own clusters without incurring any extra cost. However, we need to ensure that the number of outliers is at most ℓ. To ensure this, we assign the non-selected initial clusters to the outliers set O in non-decreasing order of their cardinalities. This makes sure that we can pack maximum number of initial clusters into O.

In the light of the above discussion, the problem boils down to the Cluster Selection problem with exactly one cluster that we recall here. Let S_j be the set of indices of the initial clusters colored by the color j for all $1 \leq j \leq 2D$. Also, denote the unique row in J_j by $\boldsymbol{b}_j$ for $1 \leq j \leq \beta$, and let the weight of $\boldsymbol{b}_j$, $w_j = |J_j|$. Fix a partition $T_1, T_2, \ldots, T_\tau$ of $\{1, 2, \ldots, 2D\}$. Also, guess τ positive integers $D_1, D_2, \ldots, D_\tau$ such that $\sum_{i=1}^{\tau} D_i = D$. D_i is the guess for the cost of the i^{th} composite cluster.

Fix a part i of the partition, where $1 \leq i \leq \tau$. Let the set of color indices $T_i = \{i_1, i_2, \ldots, i_p\}$. For each $1 \leq t \leq p$, let $U_t^i = \cup_{j \in S_{i_t}} \boldsymbol{b}_j$, i.e the set of unique rows of the initial clusters which are colored by color i_t. The Cluster Selection problem that we need to solve is the following. We are given the sets of rows $U_1^i, U_2^i, \ldots, U_p^i$ and a parameter D_i. The goal is to select p rows $\boldsymbol{b}_{j_1}, \boldsymbol{b}_{j_2}, \ldots, \boldsymbol{b}_{j_p}$ and a cluster center $\boldsymbol{s}^i$ such that $\boldsymbol{b}_{j_t} \in U_t^i$ for $1 \leq t \leq p$ and $\sum_{t=1}^{p} w_{j_t} \cdot d_H(\boldsymbol{b}_{j_t}, \boldsymbol{s}^i) \leq D_i$.

Solving Cluster Selection for categorical data equipped with the Hamming distance is practically identical to the case of Theorem 5.14. The only difference is that when we obtain a candidate center and coordinates where its value needs to be changed, we do a bruteforce enumeration of all possible values in time $|\Sigma|^{D_i}$. Thus, we obtain an algorithm for Cluster Selection with running time $2^{O(D_i \log D_i)} |\Sigma|^{D_i} \cdot (nd)^{O(1)}$. For completeness, we present the detailed proof next.

Solving Cluster Selection for categorical data

For simplicity, we drop the suffix i from all the notations. Thus, now we are given the sets of rows $U_1, U_2, \ldots, U_p$ and a parameter D. Let $\boldsymbol{c}$ be a center corresponding to an optimum feasible clustering $\mathcal{I}$ with the set $B = \{\boldsymbol{b}_{j_1}, \boldsymbol{b}_{j_2}, \ldots, \boldsymbol{b}_{j_p}\}$ of chosen rows. Also, let $U = \cup_{t=1}^{p} U_t$.

Suppose $\boldsymbol{x}$ is a vector such that there are at most D positions h with $\boldsymbol{x}[h] \neq \boldsymbol{c}[h]$. Consider the hypergraph H defined in the following way with respect to $\boldsymbol{x}$. The labels of the vertices of H are in $\{1, 2, \ldots, d\}$. For each $\boldsymbol{b}_{j_t} \in B$, we add w_{j_t} copies of the edge $S \subseteq \{1, 2, \ldots, d\}$ such that $h \in S$ if $\boldsymbol{x}[h] \neq \boldsymbol{y}[h]$.

Lemma 6.8. *Suppose the input is a yes-instance. Consider a vector $\boldsymbol{x}$ such that there is at most D positions h where $\boldsymbol{x}[h] \neq \boldsymbol{c}[h]$. Also, consider the hypergraph H defined in the above with respect to $\boldsymbol{x}$. There exists a subhypergraph H_0^* of H with the following properties.*

(i) $|V(H_0^*)| \leq D$.

(ii) $|E(H_0^*)| \leq 200 \ln D$.

(iii) The indices in $V(H_0^*)$ are the exact positions h such that $\boldsymbol{x}[h] \neq \boldsymbol{c}[h]$.

(iv) The fractional cover number of H_0^* is at most 4.

To prove the above lemma, first, we show the existence of a subhypergraph that satisfies all the properties except the second one. Let P be the set of positions h such that $\boldsymbol{x}[h] \neq \boldsymbol{c}[h]$. Let H_0 be the subhypergraph of H induced by P. We show that H_0 satisfies the first, third and fourth properties mentioned in Lemma 6.8.

By definition of $\boldsymbol{x}$, P contains at most D indices. The first property follows immediately. Next, we show that the third property also follows for H_0. As $V(H_0) \subseteq P$, in every position $h \in V(H_0)$, $\boldsymbol{x}[h] \neq \boldsymbol{c}[h]$. It is sufficient to show that for any position h with $\boldsymbol{x}[h] \neq \boldsymbol{c}[h]$, $h \in V(H_0)$. Consider any such position h. Then, if there is at least one $y \in B$ such that $\boldsymbol{x}[h] \neq \boldsymbol{y}[h]$, there is an edge e in H that contains h. Now, $h \in P$ by definition. Thus, the edge $e' = e \cap P \in H_0$ also contains h. Hence, $h \in V(H_0)$. In the other case, for all $y \in B$, $\boldsymbol{x}[h] = \boldsymbol{y}[h]$. However, this case does not occur. Otherwise, we can replace the value $\boldsymbol{c}[h]$ by $\boldsymbol{x}[h]$. The new center $\boldsymbol{c}$ would incur lesser cost, as now $\boldsymbol{c}[h] = \boldsymbol{y}[h]$ for all $\boldsymbol{y} \in B$ and previously $\boldsymbol{c}[h] \neq \boldsymbol{y}[h]$ for all $\boldsymbol{y} \in B$. Next, we show that the fourth property holds for H_0.

Lemma 6.9. *The fractional cover number of H_0 is at most* 2.

Proof. Note that the total number of edges of H_0 is $\tau = \sum_{t=1}^{p} w_{j_t}$. We claim that each vertex of H_0 is contained in at least $\tau/2$ edges.

Consider any vertex h of H_0. Let h be contained in $\delta\tau$ edges. Thus, by definition, $(1-\delta)\tau$ vectors of B (with multiplicity equal to their weights) do not differ from $\boldsymbol{x}$ at location h. Consider replacing $\boldsymbol{c}[h]$ by $\boldsymbol{x}[h]$ at position h of $\boldsymbol{c}$. Next, we analyze

the change in cost of the clustering $\mathcal{I}$. Note that the cost corresponding to positions other than h remains same. As $\boldsymbol{c}[h] \neq \boldsymbol{x}[h]$, previously each of the $(1-\delta)\tau$ rows were paying a cost of at least 1 corresponding to position h. Now, as $\boldsymbol{c}[h]$ is replaced by $\boldsymbol{x}[h]$ and consequently $\boldsymbol{c}[h] = \boldsymbol{x}[h]$, for these rows, the cost decrease is at least $(1-\delta)\tau$. Now, the cost increase for the remaining $\delta\tau$ rows is at most $\delta\tau$.

As $\mathcal{I}$ is an optimum feasible clustering, the cost decrease must be at most the cost increase. It follows that $\delta \geq 1/2$. □

So far we have proved the existence of a subhypergraph that satisfies all the properties except the second. Next, we prove the existence of a subhypergraph that satisfies all the properties. The proof of the following lemma is very similar to the proof of Lemma 6.20.

Lemma 6.10. *Let $D \geq 2$. Consider the subhypergraph H_0 that satisfies all the properties of Lemma 6.8 except (2). It is possible to select at most $200 \ln D$ edges from H_0 such that the subhypergraph H_0^* obtained by removing all the other edges from H_0 satisfies all the properties of Lemma 6.8.*

The Algorithm

We consider all possible rows $\boldsymbol{x} \in U$ and for each of them construct a hypergraph G whose vertices are in $\{1, 2, \ldots, m\}$. For each row $\boldsymbol{b}_j \in U$, we add w_j copies of the edge $S \subseteq \{1, 2, \ldots, m\}$ such that $h \in S$ if $\boldsymbol{x}[h] \neq \boldsymbol{b}_j[h]$. For all hypergraphs H_0 having at most D vertices and at most $200 \ln D$ edges, we check if each vertex of H_0 is contained in at least $1/4$ fraction of the edges. If that is the case, we use the algorithm of Theorem 5.13 to find every place P' where H_0 appears in G as subhypergraph. For each such set P', we perform all possible $D' \leq D$ edits in $\boldsymbol{x}$ at the locations in P'. After each such edit, we retrieve the edited vector $\boldsymbol{x}$ and construct a clustering considering $\boldsymbol{x}$ as its center in the following way. For each $1 \leq t \leq p$, we select a vector $\boldsymbol{b}_j$ from U_t such that $d_H(\boldsymbol{x}, \boldsymbol{b}_j)$ is the minimum over all $\boldsymbol{b}_j \in U_t$. If the cost of this clustering is at most D, we output $\boldsymbol{x}$ as a candidate center. If no such vector is output as a candidate center, we return "NO".

Next, we analyze the correctness and time complexity of this algorithm.

The Analysis.

Lemma 6.11. *The above algorithm successfully outputs $\boldsymbol{c}$ as a candidate center.*

Proof. Consider any vector $\boldsymbol{x} \in B$. Note that there is at most D positions h where $\boldsymbol{x}[h] \neq \boldsymbol{c}[h]$. Consider the hypergraph G constructed by the algorithm corresponding to $\boldsymbol{x}$. Note that the hypergraph H defined in Lemma 6.8 is a partial subgraph of G. Thus, the subhypergraph H_0^* of H is also a subhypergraph of G. As we enumerate all hypergraphs having at most D vertices, at most $200 \ln D$ edges and at most 4 fractional covering number, H_0^* must be considered by the algorithm. Let P' be the

place in G where H_0^* appears. By the third property of Lemma 6.8, the locations in P' are the exact positions h such that $\boldsymbol{x}[h] \neq \boldsymbol{c}[h]$. It follows that an edit corresponding to P' generates the vector $\boldsymbol{c}$, as $d_H(\boldsymbol{x}, \boldsymbol{c}) \leq D$. Hence, $\boldsymbol{c}$ must be an output of the algorithm. □

Time Complexity. Next, we discuss the time complexity of our algorithm. For each choice of $\boldsymbol{x}$, the hypergraph G can be constructed in $n^{O(1)}$ time. The number of distinct hypergraphs H_0 with at most D vertices and at most $200 \ln D$ edges is $2^{O(D \log D)}$, since there are 2^D possibilities for each edge. Now we analyze the time needed for locating a particular H_0 in G. For any vector $\boldsymbol{y} \in B$, $d_H(\boldsymbol{y}, \boldsymbol{c}) \leq D$. By triangle inequality, $d_H(\boldsymbol{x}, \boldsymbol{y}) \leq 2D$. Thus, the size of any edge in H is at most $2D$, and we can remove any edge of G of size more than $2D$. From Theorem 5.13, it follows that every occurrence of H_0 in G can be found in $D^{O(D)} \cdot (2D)^{4D+1} \cdot n^5 \cdot m^2 = D^{O(D)} n^{O(1)} m^2$ time. If H_0 appears at some place in G, it would take $O((D|\Sigma|)^D)$ time to edit $\boldsymbol{x}$. Hence, in total the algorithm runs in $2^{O(D \log D)} |\Sigma|^D (nm)^{O(1)}$ time.

Finally, Theorem 6.5 is proven. We restate it here for convenience.

Theorem 6.5. Categorical k-Clustering with Column Outliers *is solvable in time* $2^{O(D \log D)} |\Sigma|^D \cdot (nd)^{O(1)}$.

6.2 Constrained Clustering Applications

Before showing the algorithm for Constrained Clustering with Outliers, we discuss the implications of solving this general problem. This section is dedicated to two applications of Constrained Clustering with Outliers, the Categorical k-Clustering with Column Outliers problem, and several binary low-rank approximation problems.

6.2.1 Categorical k-Clustering with Column Outliers

Here we show that Categorical k-Clustering with Column Outliers is a special case of Constrained Clustering with Outliers. Observe that for Categorical k-Clustering with Column Outliers, the approach of Section 6.1 for Categorical k-Clustering with Row Outliers is not applicable, for several reasons. Most crucially, it does not seem that one can partition the problem into k independent instances of a simpler single-center selection problem, in a way that we reduce Categorical k-Clustering with Row Outliers to k instances of Cluster Selection (for a fixed coloring). Intuitively, the possibility to remove outlier rows does not allow such a partition as all the clusters depend simultaneously on the choice of the outlier rows. Moreover, our hardness result shows that for Categorical k-Clustering with Column Outliers the running time cannot match with

Categorical k-Clustering with Row Outliers, as no $n^{o(k)}$ time algorithm is possible for constant D, assuming ETH.

Intuitively, the hardness of Categorical k-Clustering with Column Outliers lies in the fact that we have to select the outlier rows and to group columns into clusters at the same time. Thus, to make the problem more accessible, it is natural to give a reduction to Constrained Clustering with Outliers, where both partitioning into clusters and removing the outliers are with respect to the columns, at the cost of having arbitrary row-wise constraints on the cluster centers.

The general idea of the reduction from Categorical k-Clustering with Column Outliers to Constrained Clustering with Outliers is as follows. For any matrix, a set of row outliers can be treated as column outliers of the transposed matrix. Similarly, a clustering of the columns of the transposed matrix induces a clustering of the rows of the original matrix. However, we want to find a clustering of the columns of the original matrix. To do this, one can add extra constraints on the cluster centers so that a clustering of the columns of the transposed matrix implicitly induces also a clustering of its rows. Essentially, grouping rows into k clusters corresponds to grouping columns into $|\Sigma|^k$ clusters such that the centers of these column clusters can be "read off" from the k row centers. Simply by transposing, the clustering of the columns of the transposed matrix induces a clustering of the rows of the original matrix, which is our desired clustering. This is the crux of our reduction. The next lemma shows this reduction formally.

Lemma 6.12. *For any instance* $(\boldsymbol{A}, k, D, \ell)$ *of* Categorical k-Clustering with Column Outliers*, one can construct in time* $\mathcal{O}(nd+k\cdot|\Sigma|^k)$ *an equivalent instance* $(\mathbf{B}, k', D', \ell', \mathcal{R})$ *of* Constrained Clustering with Outliers *such that* $\mathbf{B}$ *is the transpose of* $\boldsymbol{A}$*,* $k' = |\Sigma|^k$*,* $D' = D$ *and* $\ell' = \ell$*.*

Proof. Given an instance $I = (\boldsymbol{A}, k, D, \ell)$ of Categorical k-Clustering with Column Outliers, we construct an instance $I' = (\mathbf{B}, k', D', \ell', \mathcal{R})$ of Constrained Clustering with Outliers such that $\mathbf{B}$ is the transpose of $\boldsymbol{A}$, $k' = |\Sigma|^k$, $D' = D$ and $\ell' = \ell$. The set of constraints $\mathcal{R}$ is defined as follows. Let $S = \{s_1, s_2, \ldots, s_{|\Sigma|^k}\}$ be the lexicographic ordered collection of all $|\Sigma|^k$ strings of length k on alphabet Σ. Also, let Q be the $|\Sigma|^k \times k$ matrix such that the i-th row of Q is the i-th string of S. Thus, each element of Q is in Σ. For each column $\boldsymbol{q}_j$ of Q with $1 \leq j \leq k$, we construct a tuple $z_j = (\boldsymbol{q}_j[1], \boldsymbol{q}_j[2], \ldots, \boldsymbol{q}_j[|\Sigma|^k])$. Let R be the set of these tuples. To construct $\mathcal{R}$ we set $R_i = R$ for all i. This finishes our construction.

Next, we prove that I is a yes-instance of Categorical k-Clustering with Column Outliers if and only if I' is a yes-instance of Constrained Clustering with Outliers. The lemma follows immediately.

First, suppose I' is a yes-instance of Constrained Clustering with Outliers. Let $X = \{X_1, X_2, \ldots, X_{|\Sigma|^k}\}$ be a feasible clustering of the rows of $\mathbf{B} \setminus O$, the matrix $\mathbf{B}$ after removing the rows in O, where O is the set of outliers. Also, let $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_{|\Sigma|^k}$ be the corresponding centers. Note that as this clustering satisfies

the constraints in $\mathcal{R}$, each tuple $(\boldsymbol{c}_1[j], \boldsymbol{c}_2[j], \ldots, \boldsymbol{c}_{|\Sigma|^k}[j])$ is one of the k tuples of R. Let $Y = \{Y_1, Y_2, \ldots, Y_k\}$ be the partition of the columns of $\mathbf{B} \setminus O$, such that Y_t contains all the indices j such that $(\boldsymbol{c}_1[j], \boldsymbol{c}_2[j], \ldots, \boldsymbol{c}_{|\Sigma|^k}[j]) = z_t$ for $1 \leq t \leq k$. We reorder the columns of matrix $\mathbf{B} \setminus O$ based on this partition. In particular, we modify the column numbers in a way so that the columns in Y_1 become the first $|Y_1|$ columns, the columns in Y_2 become the next $|Y_2|$ columns, and so on (see Figure 6.1). Now, let us go back to the clustering $\{X_1, X_2, \ldots, X_{|\Sigma|^k}\}$. Note that this is still a clustering of the new matrix $\mathbf{B} \setminus O$ with cost at most D with columns of the centers reordered in the same way. Indeed, by construction, the reordering of the centers ensures that the center of X_i is the row vector $(z_1[i], \ldots |Y_1| \text{ times }, z_2[i], \ldots |Y_2| \text{ times } , \ldots, z_k[i], \ldots |Y_k| \text{ times})$ (see Figure 6.1). Now, note that $\{Y_1, Y_2, \ldots, Y_k\}$ is a clustering of the rows in $\boldsymbol{A}^{-O}$. We claim that its cost is at most D. As $|O| \leq l$, it follows that I is a yes-instance of Categorical k-Clustering with Column Outliers. Note that some clusters might be empty. But, as they do not incur any cost, we keep them for simplicity. To analyze the cost, set the center of Y_j to the row vector $(z_j[1], \ldots |X_1| \text{ times }, z_j[2], \ldots |X_2| \text{ times } , \ldots, z_j[|\Sigma|^k], \ldots |X_{|\Sigma|^k}| \text{ times})$ (see Figure 6.1). Now, because of the symmetry between the two clusterings X and Y, their costs are the same. Hence, the claim follows.

Now, suppose I is a yes-instance of Categorical k-Clustering with Column Outliers and consider a feasible clustering $Y = \{Y_1, Y_2, \ldots, Y_k\}$. Let O be the set of indices of the outliers. Also, let $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k$ be the corresponding centers. Note that each string of the form $\boldsymbol{c}_1[j]\boldsymbol{c}_2[j] \ldots \boldsymbol{c}_k[j] \in S$. For a string $s \in S$, we say column j is of type s if $\boldsymbol{c}_1[j]\boldsymbol{c}_2[j] \ldots \boldsymbol{c}_k[j] = s$. Let $\{X_1, X_2, \ldots, X_{|\Sigma|^k}\}$ be the partition of the columns of $\boldsymbol{A}^{-O}$ based on their types. In particular, for the i-th string $s \in S$, X_i contains all columns of type s. Note that $\{X_1, X_2, \ldots, X_{|\Sigma|^k}\}$ is also a clustering of the rows of $\mathbf{B} \setminus O$. We show that the cost of this clustering is at most D. To analyze the cost, choose the row vector $(s_i[1], \ldots |Y_1| \text{ times } , s_i[2], \ldots |Y_2| \text{ times } , \ldots, s_i[k], \ldots |Y_k| \text{ times})$ as the center vector of the cluster X_i of the columns of $\mathbf{B} \setminus O$. Note that the centers chosen in this way satisfy the constraints in $\mathcal{R}$, as the center tuple $(s_1[j], s_2[j], \ldots, s_{|\Sigma|^k}[j]) = (\boldsymbol{q}_j[1], \boldsymbol{q}_j[2], \ldots, \boldsymbol{q}_j[|\Sigma|^k])$ corresponding to index j is in R. It follows that the cost of this clustering is at most D, as it is induced by the clustering Y and the centers $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k$. □

Lemma 6.12 together with Theorem 6.4 immediately give us Theorem 6.1 with $f(k, D, |\Sigma|) = (k'D)^{O(k'D)}|\Sigma|^{k'D}$ and $g(k, |\Sigma|) = O(k')$, where $k' = |\Sigma|^k$. The theorem is restated here.

Theorem 6.1. Categorical k-Clustering with Column Outliers *is solvable in time* $f(k, D, |\Sigma|) \cdot d^{g(k,|\Sigma|)} \cdot n^2$, *where* f *and* g *are computable functions.*

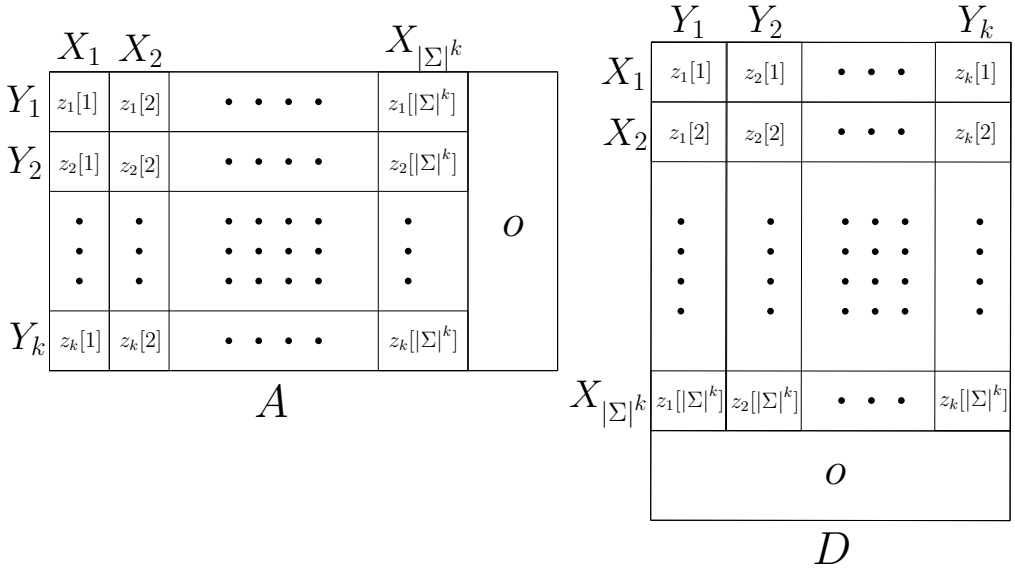


Figure 6.1: Figure showing the clustering X of $\boldsymbol{D} \setminus O$ and Y of $\boldsymbol{A}^{-O}$. The common center coordinate values are shown inside submatrices.

6.2.2 Low Rank Approximation

Now we show that Constrained Clustering with Outliers encapsulates not only Categorical k-Clustering with Column Outliers, but also robust variants of binary/finite field low-rank approximations problems, and projective clustering. We give the formal definitions next.

Robust L_0-Low Rank Approximation. The vanilla case of the L_0-Low Rank Approximation problem is the following. Given an $n \times d$ matrix $\mathbf{A}$ over GF(p) (a finite field of order p), the task is to find an $n \times d$ matrix $\mathbf{B}$ over GF(p) of GF(p)-rank at most r which is closest to $\mathbf{A}$ in the L_0-*entrywise norm*. Thus, the goal is to minimize $\|\mathbf{A} - \mathbf{B}\|_{0,0}$, the number of different entries in $\mathbf{A}$ and $\mathbf{B}$. In the robust version of this problem some of the rows of $\mathbf{A}$ could be outliers, which brings us to the following problem.

Robust L_0-Low Rank Approximation

Input: An $n \times d$ matrix $\mathbf{A}$ over GF(p), a positive integer r, non-negative integers D and ℓ.

Task: Decide whether there is a matrix $\mathbf{B}$ of GF(p)-rank at most r, and a matrix $\mathbf{C}$ over GF(p) with at most ℓ non-zero rows such that

$$\|\mathbf{A} - \mathbf{B} - \mathbf{C}\|_{0,0} \leq D$$

Note that in this definition the non-zero rows of $\mathbf{C}$ can take any values. However, it is easy to see that the problem would be equivalent if the rows of $\mathbf{C}$ were constrained to be either zero rows or the respective rows of $\mathbf{A}$. This holds since if $\mathbf{C}$ contains a non-zero row, it could be replaced by the respective row of $\mathbf{A}$, and the respective row of $\mathbf{B}$ can be replaced by a zero row. This does not increase the cost nor the rank of $\mathbf{B}$. Thus any of the two formulations allows to restore the row outliers in $\mathbf{A}$ from $\mathbf{C}$.

The following lemma gives a reduction from Constrained Clustering with Outliers.

Lemma 6.13. *For any instance* $(\mathbf{A}, r, D, \ell)$ *of* Robust L_0-Low Rank Approximation*, one can construct in time* $\mathcal{O}(n+d+p^{\mathcal{O}(r)})$ *an equivalent instance* $(\boldsymbol{X}, k', D', \ell, \mathcal{R})$ *of* Constrained Clustering with Outliers*, where* $\boldsymbol{X} = \mathbf{A}$*,* $k' = p^r$*,* $D' = D$ *and* $\mathcal{R}$ *is a set of constraints.*

(By equivalent instances we mean that $(\mathbf{A}, r, D, \ell)$ is a yes-instance if and only if $(\boldsymbol{X}, k', D', \ell, \mathcal{R})$ is a yes-instance and solution corresponding to a yes-instance $(\boldsymbol{X}, k', D', \ell, \mathcal{R})$ can be used to construct a solution for the yes-instance $(\mathbf{A}, r, D, \ell)$.) The proof of Lemma 6.13 is almost identical to the proof of [88, Lemma 1] and we omit it here. By Lemma 6.13 and Theorem 6.4, we derive the following corollary.

Theorem 6.14. Robust L_0-Low Rank Approximation *is* FPT *parameterized by* D *when* p *and* r *are constants.*

Robust Low Boolean-Rank Approximation is a variant of the robust low-rank approximation problem when the approximation matrix $\mathbf{B}$ is of low Boolean rank. More precisely, let $\mathbf{A}$ be a binary $n \times d$ matrix. Now we consider the elements of $\mathbf{A}$ to be *Boolean* variables. The *Boolean rank* of $\mathbf{A}$ is the minimum r such that $\mathbf{A} = \mathbf{U} \wedge \mathbf{V}$ for a Boolean $n \times r$ matrix $\mathbf{U}$ and a Boolean $r \times d$ matrix $\mathbf{V}$, where the product is Boolean, that is, the logical $\wedge$ plays the role of multiplication and $\vee$ the role of sum. Here $0 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 1 = 1$, $0 \vee 0 = 0$, $0 \vee 1 = 1$, and $1 \vee 1 = 1$. Thus the matrix product is over the Boolean semi-ring $(0, 1, \wedge, \vee)$. This can be equivalently expressed as the normal matrix product with addition defined as $1 + 1 = 1$. Binary matrices equipped with such algebra are called *Boolean matrices*. The variant of the low Boolean-rank matrix approximation is the following problem.

Robust Low Boolean-Rank Approximation

Input: A binary $n \times d$ matrix $\mathbf{A}$ a positive integer r, non-negative integers D and ℓ.

Task: Decide whether there is a binary matrix $\mathbf{B}$ of Boolean-rank at most r, and a binary matrix $\mathbf{C}$ with at most ℓ non-zero rows such that

$$\|\mathbf{A} - \mathbf{B} - \mathbf{C}\|_0 \leq D$$

By Theorem 6.4 and reduction from constrained clustering to the Boolean-rank matrix approximation identical to [88, Lemma 2], we derive the following corollary.

Theorem 6.15. Robust Low Boolean-Rank Approximation *is* FPT *parameterized by D when r is a constant.*

Robust Projective Clustering generalizes Categorical k-Clustering with Row Outliers and Robust Low GF(p)-Rank Approximation. Here the centers of clusters are linear subspaces of bounded dimension r. More precisely, in Robust Projective Clustering we are given a set $X \subseteq \Sigma^d$ of n d-dimensional vectors over GF(p) and positive integers k, r, D and ℓ. The task is to decide whether there is a family of r-dimensional linear subspaces $\mathcal{C} = \{C_1, \ldots, C_k\}$ over GF(p) and a set of outliers $O \subseteq X$ of size at most ℓ, such that

$$\sum_{\mathbf{x} \in X \setminus O} \min_{1 \leq i \leq k} d_H(\mathbf{x}, C_i) \leq D.$$

The proof of the following lemma is similar to the proof of [88, Lemma 3].

Lemma 6.16. *For any instance (X, k, r, D, ℓ) of* Robust Projective Clustering*, one can construct in time $\mathcal{O}(n+d+p^{k \cdot r})$ an equivalent instance $(\boldsymbol{X}, k', D', \ell, \mathcal{R})$ with $\boldsymbol{X} = X$, $k' = p^{k \cdot r}$, $D' = D$, and $\ell' = \ell$ of* Constrained Clustering with Outliers.

Then by Theorem 6.4, we obtain the following.

Theorem 6.17. Robust Projective Clustering *is* FPT *parameterized by D when k, r and p are constants.*

Related work. To give more context on the problems introduced above, here we survey briefly relevant literature. For the general introduction to robust variants of the low-rank approximation problem, we refer to Chapter 3 of this work; here we focus on the results concerning finite fields/Boolean rank.

For robust variants of categorical low-rank matrix approximation, we are not aware of any approximation or parameterized algorithms. For the vanilla variant, L_0-Low Rank Approximation, a number of parameterized and approximation algorithms were developed. Ban et al. [17] gave a PTAS for L_0-Low Rank Approximation when the rank r of matrix $\mathbf{B}$ is a constant. For the case when the field is GF(2), faster EPTAS was obtained in [17] and [88], see also [138]. When the rank r is not bounded, Ban et al. [17] proved that assuming P≠NP, there is no constant factor approximation in polynomial time for the problem. Moreover, assuming ETH, for every $\alpha > 1$ there is $\delta > 0$ such that no α-approximate solution could be found in time $2^{r^\delta}(mn)^{\mathcal{O}(1)}$.

L_0-Low Rank Approximation is strongly related to the problem of finding the rigidity of a matrix. (For a target rank r, the *rigidity* of a matrix $\mathbf{A}$ over field GF(p) is the minimum ℓ_0-distance between $\mathbf{A}$ and a matrix of rank at most r.) Rigidity is a classical concept in Computational Complexity Theory studied due to its connections with lower bounds for arithmetic circuits [105, 182, 170]. We refer to [148] for an extensive survey on this topic. Most of the known results about the parameterized complexity of L_0-Low Rank Approximation follows from the results for Matrix Rigidity. Fomin et al. have proved in [91] that for every finite field, L_0-Low Rank Approximation is W[1]-hard being parameterized by D. However, when parameterized by D and r, the problem becomes fixed-parameter tractable. The result of Fomin et al. [91] also implies that L_0-Low Rank Approximation admits a polynomial kernel of size $\mathcal{O}(r^2D^2)$. For the special case of the binary field GF(2), algorithm of running time $2^{\mathcal{O}(r\sqrt{D\log(rD)})}(nd)^{\mathcal{O}(1)}$ was given in [90].

Low Boolean-Rank Approximation has attracted much attention, especially in the data mining and knowledge discovery communities. In data mining, matrix decompositions are often used to produce concise representations of data. Since much of the real data is binary or even Boolean in nature, Boolean low-rank approximation could provide a deeper insight into the semantics associated with the original matrix. There is a big body of work done on Low Boolean-Rank Approximation, see e.g. [18, 21, 72, 149, 158, 159, 180]. While computing GF(2)-rank (or rank over any other field) of a matrix can be performed in polynomial time, deciding whether the Boolean rank of a given matrix is at most r is already an NP-complete problem. Thus, Low Boolean-Rank Approximation is NP-complete already for $D = 0$. This follows from the well-known relation between the Boolean rank and covering edges of a bipartite graph by bicliques [104]. The latter Biclique Cover problem is known to be NP-complete [164]. Biclique Cover is solvable in $2^{2^{\mathcal{O}(r)}} \cdot (nm)^{\mathcal{O}(1)}$ time [103] and unless Exponential Time Hypothesis (ETH) fails, it cannot be solved in $2^{2^{o(r)}} \cdot (nm)^{\mathcal{O}(1)}$ time [46]. For the special case $r = 1$ and $D \leq \|A\|_0/240$, an exact algorithm of running time $2^{D/\sqrt{\|A\|_0}} \cdot (nm)^{\mathcal{O}(1)}$ for Low Boolean-Rank Approximation was given in [38]. EPTAS for this problem were obtained in [17] and [88]. Finally, parameterized algorithm for Low Boolean-Rank Approximation of running time

$2^{\mathcal{O}(r2^r\sqrt{D\log D})}(nm)^{\mathcal{O}(1)}$ is known [90].

6.3 Solving Constrained Clustering with Outliers

In this section we prove Theorem 6.4 by giving an algorithm for Constrained Clustering with Outliers that runs in $(kD)^{O(kD)}n^{O(k)}d^2|\Sigma|^{kD}$ time.

Clearly, the approach used for Categorical k-Clustering with Row Outliers fails for Constrained Clustering with Outliers, as the constraints on the centers do not allow to form clusters independently. Instead, we generalize the hypergraph construction used for Cluster Selection to handle the choice of all k centers simultaneously, as opposed to just one center at a time. This is the most technical part of this chapter, and we will now sketch the generalized construction briefly. The main idea is to base the hypergraph on k-tuples of rows instead of just single rows.

Fix an optimal solution: a subset of outlier rows $O \subset \{1, 2, \ldots, n\}$, a partition $\mathcal{I} = \{I_1, I_2, \ldots, I_k\}$ of $\{1, 2, \ldots, n\} \setminus O$, and a k-tuple $C = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k)$ of d-dimensional vectors representing the centers. First, pick a k-tuple $x = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ such that it is at distance at most D to the optimal k-tuple of centers C. This can be done by guessing a row $\boldsymbol{x}_j$ from each of the optimal clusters I_j, in total time $n^{O(k)}$. To obtain the desired set C, it is sufficient to know in which rows x is different from C, that is, positions $h \in \{1, \ldots, d\}$ such that $(\boldsymbol{x}_1[h], \ldots, \boldsymbol{x}_k[h]) \neq (\boldsymbol{c}_1[h], \ldots, \boldsymbol{c}_k[h])$. Denote this set of positions by P, which is a subset of $\{1, \ldots, d\}$. There can be at most D such rows, since x is at distance at most D from C, so $|P| \leq D$. As there are $|\Sigma|^k$ choices for a row, we can try all the $|\Sigma|^{kD}$ possible choices for C once the set of row positions P is identified.

If x does not satisfy the constraints at a position $h \in \{1, \ldots, d\}$, meaning that $x[h] = (\boldsymbol{x}_1[h], \ldots, \boldsymbol{x}_k[h]) \notin R_h$, then definitely $x[h] \neq C[h]$. For each such h, replace the row $x[h]$ with any tuple from R_h, this does not increase the number of rows where C and x are different. From now on we assume that all rows of x satisfy the constraints.

To identify the deviating rows, construct a hypergraph H in the following way. Vertices of H are labeled by $\{1, \ldots, d\}$ representing row indices, like in the Cluster Selection construction. However, the edges correspond to k-tuples of rows now. For each k-tuple $y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)$ of rows of $\boldsymbol{A}$, we add an edge $S \subseteq \{1, 2, \ldots, d\}$ containing the rows where y and x are not equal. That is, $h \in \{1, 2, \ldots, d\}$ belongs to S if $(\boldsymbol{x}_1[h], \ldots, \boldsymbol{x}_k[h]) \neq (\boldsymbol{y}_1[h], \ldots, \boldsymbol{y}_k[h])$. Now observe that the set of positions P and the optimal clustering $\mathcal{I}$ correspond to the following subhypergraph H_0 of H. The vertex set $V(H_0)$ is exactly P. As for the edges, we have to carefully select the subset of $E(H)$. Recall that to apply Theorem 5.13 we crucially need that the fractional edge cover number of H_0 is bounded by a constant. It turns out that restricting all edges of $E(H)$ to $V(H_0)$ does not provide us with such a bound. Instead, let T be the set of all k-tuples $a = (\mathbf{a}_{i_1}, \ldots, \mathbf{a}_{i_k})$ such that $i_j \in I_j$ for all

$1 \leq j \leq k$. The hyperedge set $E(H_0)$ is the subset of $E(H)$ corresponding to the tuples in T, and restricted to the vertices in $V(H_0)$. Intuitively, the tuples in T are those that are relevant for the optimal clustering, and as we show next this gives us the required fractional edge cover bound.

We claim that the fractional edge cover number of H_0 is at most 2. Surprisingly, the choice of T allows us to essentially lift the single-row argument to k-tuples of rows as well. It suffices to show that for every vertex h in $V(H_0)$, at least half of edges in $E(H_0)$ cover h. Or equivalently, at least for half of tuples y in T it holds that $y[h] \neq x[h]$. Fix h, assume there is a cluster I_j such that there are at least $\lceil |I_j|/2 \rceil$ elements a in I_j with $a[h] \neq \boldsymbol{x}_j[h]$. Then at least half of tuples in T do not agree with x in the row h, since T is the Cartesian product of all optimal clusters I_j. Otherwise, for every cluster I_j there are strictly less than $\lceil |I_j|/2 \rceil$ elements a in I_j such that $a[h] \neq \boldsymbol{x}_j[h]$. This contradicts the optimality of C. Replace $C[h]$ by $x[h]$, since the tuple x can be assumed to satisfy the constraints, the clustering is still valid. If $\boldsymbol{c}_j[h] = \boldsymbol{x}_j[h]$ for $j \in \{1, \ldots, r\}$, the cost for the j-th cluster does not change. If $\boldsymbol{c}_j[h] \neq \boldsymbol{x}_j[h]$, then the cost decreases as $\boldsymbol{x}_j[h]$ matches more than half of the elements of I_j, while $\boldsymbol{c}_j[h]$ could only match at most the remaining portion of the elements.

By Theorem 5.13, we can enumerate all possible locations where H_0 appears in H in FPT time, as the fractional edge cover number of H_0 is bounded. Since we can also enumerate all possible hypergraphs on k vertices in FPT time (and even in $k^{O(k)}$ time by repeating the proof of Lemma 6.20), we can enumerate all possible sets of row positions where x differs from C, and Theorem 6.4 follows.

Now we move to the detailed proof. First, we show a structural lemma that would be useful for analysis of the algorithm.

6.3.1 Structural Lemma

In this section, we prove a structural lemma that shows the existence of a "good" subhypergraph. Consider a feasible clustering $\mathcal{I} = \{I_1, I_2, \ldots, I_k\}$ having the minimum cost. Let $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$ be a fixed set of centers corresponding to $\mathcal{I}$. Also, let T be the set of all tuples of the form $(\mathbf{a}_{i_1}, \mathbf{a}_{i_2}, \ldots, \mathbf{a}_{i_k})$ such that $i_j \in I_j$ for all j. Note that we do not actually need to know the set T — we just introduce the notation for the sake of analysis. For a k-tuple $x = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$, we denote the tuple $(\boldsymbol{x}_1[j], \boldsymbol{x}_2[j], \ldots, \boldsymbol{x}_k[j])$ by $x[j]$. Two k-tuples x and y are said to differ from each other at location j if $x[j] \neq y[j]$.

Let C be the k-tuple such that $C = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k)$. Suppose $x = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ is such that there are at most D positions h where $x[h] \neq C[h]$, and for each $1 \leq j \leq d$, $x[j] \in R_j$. Consider the hypergraph H defined in the following way with respect to x. The labels of the vertices of H are in $\{1, 2, \ldots, d\}$. For each k-tuple $y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)$ of T, we add an edge $S \subseteq \{1, 2, \ldots, d\}$ such that $h \in S$ if $x[h] \neq y[h]$.

In the following lemma, we show that the hypergraph H has a "good" subhypergraph.

Lemma 6.18. *(Structural Lemma) Suppose the input is a yes-instance. Consider a k-tuple $x = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ such that there are at most D positions h where $x[h] \neq C[h]$ and for each $1 \leq j \leq d$, $x[j] \in R_j$. Also, consider the hypergraph H defined in the above with respect to x. There exists a subhypergraph H^* of H with the following properties.*

(i) $|V(H^*)| \leq D$.

(ii) $|E(H^*)| \leq 200 \ln D$.

(iii) The indices in $V(H^*)$ are the exact positions h such that $x[h] \neq C[h]$.

(iv) The fractional cover number of H^* is at most 4.

To prove the above lemma, first, we show the existence of a subhypergraph that satisfies all the properties except the second one. Let P be the set of positions h such that $x[h] \neq C[h]$. Let H_0 be the subhypergraph of H induced by P. We show that H_0 satisfies the first, third and fourth properties mentioned in Lemma 6.18.

By definition of x, P contains at most D indices. The first property follows immediately. Next, we show that the third property also follows for H_0. As $V(H_0) \subseteq P$, in every position $h \in V(H_0)$, $x[h] \neq C[h]$. It is sufficient to show that for any position h with $x[h] \neq C[h]$, $h \in V(H_0)$. Consider any such position h. Then, if there is at least one tuple $y \in T$ such that $x[h] \neq y[h]$, there is an edge e in H that contains h. Now, $h \in P$ by definition. Thus, the edge $e' = e \cap P \in H_0$ also contains h. Hence, $h \in V(H_0)$. In the other case, for all tuples y in T, $x[h] = y[h]$. We show that this case does not occur. For the sake of contradiction, suppose this case does occur. This implies, for $1 \leq j \leq k$, all rows in I_j contains the value $\boldsymbol{x}_j[h]$ at location h. Now, by definition of x, there is a k-tuple z in R_h such that $z = x[h]$. Replace $C[h]$ by z at position h of C. Let us compute the change in cost of the clustering $\mathcal{I}$. Note that the cost corresponding to positions other than h remains same. Previously $C[h] \neq x[h]$. Thus, there is at least one $1 \leq j \leq k$ such that $\boldsymbol{c}_j[h] \neq \boldsymbol{x}_j[h]$. As all rows in I_j contains the value $\boldsymbol{x}_j[h]$ at location h, the previous cost corresponding to location h is at least 1. However, after replacement the cost corresponding to location h is exactly 0. Thus, the cost decrease is at least 1, which contradicts the optimality of the previously chosen centers. Thus, the current case cannot occur, and hence the third property follows.

Next, we show that the fourth property holds for H_0. Indeed, we show a stronger bound of 2.

Lemma 6.19. *The fractional cover number of H_0 is at most 2.*

Proof. Note that the total number of edges of H_0 is $\tau = |T|$. We claim that each vertex of H_0 is contained in at least $\tau/2$ edges.

Consider any vertex h of H_0. Suppose there is a $1 \leq j \leq k$, such that for at least $\lceil |I_j|/2 \rceil$ rows in I_j the value at location h is not $\boldsymbol{x}_j[h]$. Note that each such

row contributes to $\Pi_{t^1=1}^{j-1}|I_{t^1}| \cdot \Pi_{t^2=j+1}^{k}|I_{t^2}| = \tau/|I_j|$ tuples $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)$ of T such that $\boldsymbol{y}_j[h] \neq \boldsymbol{x}_j[h]$. Thus, the edge corresponding to each such tuple contains h. It follows that, at least $\lceil |I_j|/2 \rceil \cdot (\tau/|I_j|) \geq \tau/2$ edges in $E(H_0)$ contain h.

In the other case, for all $1 \leq j \leq k$ and less than $\lceil |I_j|/2 \rceil$ rows in I_j, the value at location h is not $\boldsymbol{x}_j[h]$. We prove that this case does not occur. Note that there is a k-tuple z in R_h such that $z = x[h]$. Consider replacing $C[h]$ by z at position h of C. Next, we analyze the change in cost of the clustering $\mathcal{I}$. Note that the cost corresponding to positions other than h remains same. For a $1 \leq j \leq k$, if previously $\boldsymbol{c}_j[h] = \boldsymbol{x}_j[h]$, the cost remains same. Otherwise, $\boldsymbol{c}_j[h] \neq \boldsymbol{x}_j[h]$. Note that for more than $\lceil |I_j|/2 \rceil$ rows in I_j, the value at location h is $\boldsymbol{x}_j[h]$. Thus, by replacing $\boldsymbol{c}_j[h]$ by $\boldsymbol{x}_j[h]$, the cost decrement corresponding to the index j and location h is at least 1. As $x[h] \neq C[h]$, there is an index j such that $\boldsymbol{c}_j[h] \neq \boldsymbol{x}_j[h]$. It follows that the overall cost decrement is at least 1, which contradicts the optimality of the previously chosen centers. Hence, this case cannot occur. This completes the proof of the lemma. □

So far we have proved the existence of a subhypergraph that satisfies all the properties except the second. Next, we show the existence of a subhypergraph that satisfies all the properties. The following claim completes the proof of Lemma 6.18. Its proof follows a standard sampling argument, and is nearly identical to the proof of the corresponding Claim 5.17 for L_p-Cluster Selection.

Claim 6.20. *Let $D \geq 2$. Consider the subhypergraph H_0 that satisfies all the properties of Lemma 6.18 except (2). It is possible to select at most $200 \ln D$ edges from H_0 such that the subhypergraph H_0^* obtained by removing all the other edges from H_0 satisfies all the properties of Lemma 6.18.*

Proof. Recall that τ is the number of edges in H_0. Construct a hypergraph H_0' by selecting each edge of H_0 independently at random with probability $\frac{150 \ln D}{\tau}$. The expected number of edges selected is $150 \ln D$. Also, by Proposition 11.14 with $\beta = 1/3$, the probability that at least $200 \ln D$ edges are selected is at most $exp(-150 \ln D/27) < 1/D^2$. Each vertex of H_0 is contained in at least $\tau/2$ edges. Thus, the expected number of edges that cover a vertex of H_0' is at least $75 \ln D$. Moreover, by Proposition 11.14 with $\beta = 1/3$, the probability that a given vertex of H_0^* is covered by at most $50 \ln D$ edges is at most $exp(-75 \ln D/18) < 1/D^3$. Therefore, with probability at least $1 - 1/D^2 - D \cdot 1/D^3$, H_0' contains at most $200 \ln D$ edges and each vertex of H_0' is covered by at least $50 \ln D$ edges. Thus, the fractional cover number of H_0' is at most 4. Hence, the second and the fourth property are satisfied. As the vertex set does not get changed, the first and the third property also hold. It follows that there exists a subhypergraph H_0^* that contains at most $200 \ln D$ edges from H_0 and satisfies all the properties of Lemma 6.18. □

6.3.2 The Algorithm

In this section, we first describe our algorithm, and then provide the analysis of its correctness and running time. Note that the goal of our algorithm is to retrieve a feasible clustering if there is one.

A k-tuple x of rows of $\boldsymbol{A}$ is said to differ from $\mathcal{R}$ at a position j for $1 \leq j \leq d$ if $x[j] \notin R_j$.

First, we consider all k-tuples $x = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ such that $\boldsymbol{x}_j$ is a row of $\boldsymbol{A}$ for $1 \leq j \leq k$, and apply the following refinement process on each of them.

- Let $P \subseteq \{1, 2, \ldots, d\}$ be the set of positions where x differs from $\mathcal{R}$.
- If $|P| > D$, probe the next k-tuple x.
- For each position $h \in P$, replace $x[h]$ by any element of R_h.

Next, for each refined k-tuple $x = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$, we construct a hypergraph G whose vertices are in $\{1, 2, \ldots, d\}$. For each k-tuple $y = (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k)$ of rows of $\boldsymbol{A}$, we add an edge $S \subseteq \{1, 2, \ldots, d\}$ such that $h \in S$ if $x[h] \neq y[h]$. For all hypergraphs H_0^* having at most D vertices and at most $200 \ln D$ edges, we check if each vertex of H_0^* is contained in at least $1/4$ fraction of the edges. If that is the case, we use the algorithm of Theorem 5.13 to find every place P' where H_0^* appears in G as subhypergraph. For each such set P', we perform all possible $D' \leq D \cdot k$ edits of the tuple x at the locations in P'. In particular, for each D', the editing is done in the following way. For each possible D' entries $(a_1, \ldots, a_{D'})$ in $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ at the locations in P' and each set of D' symbols $(s_1, \ldots, s_{D'})$ from Σ, we put s_j at the entry a_j for all j. After each such edit, we retrieve the edited tuple $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ and perform a sanity check on this tuple to ensure that it is a valid k-tuple center. In particular, for each index $1 \leq h \leq d$, if there is a $z \in R_h$ such that $z = x[h]$, we tag x as a valid tuple. Lastly, we output all such valid k-tuples as candidate centers if the corresponding cost of clustering is at most D. If no k-tuple is output as a candidate center, we return "NO".

Note that, given a k-tuple candidate center $z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k)$, one can compute a minimum cost clustering in the following greedy way, which implies that we can correctly compute the cost of clustering in the above. At each step i, we assign a new row of $\boldsymbol{A}$ to a center. In particular, we add a row $\mathbf{a}_j$ of $\boldsymbol{A}$ to a cluster I'_t such that $\mathbf{a}_j$ incurs the minimum cost over all unassigned rows if it is assigned to a center, i.e, it minimizes the quantity $\min_{t'=1}^{k} d(\mathbf{a}_j, \boldsymbol{z}_{t'})$, and $\boldsymbol{z}_t$ is a corresponding center nearest to $\mathbf{a}_j$. As we are allowed to exclude ℓ outliers, we assign $n - \ell$ rows. $\{I'_1, \ldots, I'_k\}$ is the desired clustering. This finishes the description of our algorithm.

Analysis Again consider the feasible clustering with partition $\mathcal{I} = \{I_1, I_2, \ldots, I_k\}$ and the corresponding center tuple $C = (\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k)$ having the minimum cost. First, we have the following observation.

Observation 6.21. *Suppose for a k-tuple x, x differs from C at D_1 positions. Then, after refinement, there is at most D_1 positions h such that $x[h] \neq C[h]$. Moreover, after refinement, $d_H(x, C) \leq D_1 \cdot k$.*

The first part is true for the following reason. If x was different from $\mathcal{R}$ at a position h, then $x[h] \neq C[h]$ as well. Thus, refinement is applied for a position h where $x[h]$ already differs from $C[h]$. Hence, refinement does not affect a position h where $x[h] = C[h]$. The moreover part follows trivially from the first part as x is a k-tuple. Now, it is sufficient to prove the following lemma.

Lemma 6.22. *Suppose there is a feasible clustering with partition $\mathcal{I} = \{I_1, I_2, \ldots, I_k\}$ as defined above. The above algorithm successfully outputs the k-tuple $(\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k)$.*

Proof. Consider a k-tuple $x = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ such that the row $\boldsymbol{x}_j$ is in cluster I_j. As the algorithm considers all possible k-tuples of rows in $\boldsymbol{A}$, it must consider x. Note that there can be at most D positions h such that $x[h] \neq C[h]$. By Observation 6.21, after refinement, there are at most D positions h where $x[h] \neq C[h]$. Also, for each $1 \leq j \leq d$, $x[j] \in R_j$. Let G be the hypergraph constructed by the algorithm corresponding to this refined x. Note that the hypergraph H defined in Lemma 6.18 is a partial subhypergraph of G. Thus, the subhypergraph H^* of H is also a subhypergraph of G. As we enumerate all hypergraphs having at most D vertices, at most $200 \ln D$ edges and at most 4 fractional covering number, H^* must be considered by the algorithm. Let P' be the place in G where H^* appears. By the third property of Lemma 6.18, the locations in P' are the exact positions h such that $x[h] \neq C[h]$. It follows that an edit corresponding to P' generates the tuple $C = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k)$, as $d_H(x, C) \leq D \cdot k$. It is easy to see that C also passes the sanity check. Hence, C must be an output of the algorithm. □

Given the tuple center $C = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k)$, we use the greedy assignment scheme (described in the algorithm) to find the underlying clustering. Note that given any k-tuple candidate center $z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k)$, this greedy scheme computes a minimum cost clustering with $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k$ being the cluster centers. Thus, the cost of the clustering computed by the algorithm is at most D. Hence, the algorithm successfully outputs C as a candidate center. We summarize our findings in the following lemma.

Lemma 6.23. *Suppose the input instance is a no-instance, then the algorithm successfully outputs "NO". If the input instance is a yes-instance, the algorithm correctly computes a feasible clustering.*

Time Complexity Next, we discuss the time complexity of our algorithm. The number of choices of x is $n^{O(k)}$. For each choice of x, the hypergraph G can be constructed in $n^{O(k)}$ time. The number of distinct hypergraphs H_0^* with at most D vertices and at most $200 \ln D$ edges is $2^{D \cdot 200 \ln D} = D^{O(D)}$, since there are 2^D possibilities for each edge. Now we analyze the time needed for locating a particular

H_0^* in G. For any tuple $y \in T$, $d_H(y, C) \leq D$. By triangle inequality, $d_H(x, y) \leq 2D$. Thus, the size of any edge in H is at most $2D$, and we can remove any edge of G of size more than $2D$. From Theorem 5.13, it follows that every occurrence of H_0^* in G can be found in $D^{O(D)} \cdot (2D)^{4D+1} \cdot n^{4k+k} \cdot d^2 = D^{O(D)} \cdot n^{O(k)} d^2$ time. If H_0^* appears at some place in G, it would take $O((kD|\Sigma|)^{kD})$ time to edit x. Hence, in total the algorithm takes $(kD)^{O(kD)}|\Sigma|^D \cdot n^{O(k)} d^2$ time. By the above discussion, we have Theorem 6.4.

Theorem 6.4. Constrained Clustering with Outliers *is solvable in time* $(kD)^{O(kD)}|\Sigma|^{kD} \cdot n^{O(k)} \cdot d^2$.

6.4 Hardness Results

In this section we present the lower bound results. First we recall a few complexity notions essential to our hardness proofs.

Our lower bounds hold for the Categorical k-Clustering with Column Outliers problem, and even for its special case where $D = 0$. This case may be viewed upon as the problem of partitioning rows of the input matrix into clusters where rows in each cluster are identical, except for the outlier columns. We show that this exact version of Categorical k-Clustering with Column Outliers is W[1]-hard when parameterized by the number of clusters k and the number of inliers $(n - \ell)$, and also W[1]-hard when parameterized by the number of outliers ℓ.

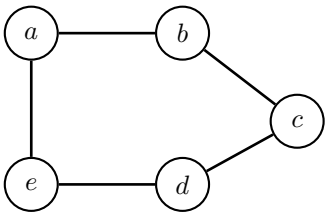
The first lower bound shows that the result in Theorem 6.1 is tight in the sense that k must be a constant. This is in contrast to the Categorical k-Clustering with Row Outliers problem, for which we show the FPT algorithm for the stronger parameterization (Theorem 6.5). Thus, Categorical k-Clustering with Column Outliers is fundamentally more difficult than Categorical k-Clustering with Row Outliers. Also, note that Categorical k-Clustering with Row Outliers is trivially solvable in polynomial time when the cost D is zero.

First, we describe briefly the idea of the proof. We give a reduction from Independent Set. Consider an instance (G, t) of the latter problem. Take as $\boldsymbol{A}$ the incidence matrix of G, such that columns correspond to the vertices, and rows to the edges. Now selecting a set of size t in G corresponds to selecting t inlier columns in $\boldsymbol{A}$. For an independent set of size t, we aim to select t columns of $\boldsymbol{A}$ that have a zero-cost clustering with $t+1$ clusters. Equivalently, we are to select t columns of $\boldsymbol{A}$ such that there are at most $t+1$ distinct rows in the resulting submatrix. Observe that whenever we select a subset $S \subset V(G)$, each edge inside S corresponds to a unique row with two ones. An edge from S to $V(G) \setminus S$ corresponds to a row with a single one in the column of the vertex of S, and an edge inside $V(G) \setminus S$ corresponds to a zero row. By a certain preprocessing of the input graph G, we can assume that for every t-sized subset S the zero row and all the t distinct rows with a single one necessarily appear in the corresponding submatrix of t columns. This is already

$t+1$ distinct rows, and every edge inside S must be a new row. Thus no edges can be inside S whenever there is a valid clustering. In the next lemma, we make the intuition above formal.

Lemma 6.24. Categorical k-Clustering with Column Outliers *is W[1]-hard when parameterized by* $k+(d-\ell)$ *when* $D=0$ *and* $\Sigma=\{0,1\}$. *Assuming ETH, there is no algorithm solving the problem with* $D=0$ *and the binary alphabet in time* $d^{o(k)}\cdot n^{\mathcal{O}(1)}$.

(a) The input graph G, $t=2$.



(b) The corresponding 5×5 matrix of the clustering problem. The number of columns to remove is 3, the number of clusters is 3.

	a	b	c	d	e
ab	1	1	0	0	0
bc	0	1	1	0	0
cd	0	0	1	1	0
de	0	0	0	1	1
ae	1	0	0	0	1

(c) A selection of two inlier columns achieving a zero-cost clustering (left), and a nonzero cost (right).

b	d
1	0
1	0
0	1
0	1
0	0

d	e
0	0
0	0
1	0
1	1
0	1

Figure 6.2: An illustration of the reduction in Theorem 6.2 showing a possible input graph, the obtained matrix, and two possible selections of the inliers.

Proof. We show a reduction from the Independent Set problem. Given a graph G and a parameter t, the task in Independent Set is to determine whether there is an independent set of size at least t in G. Consider an instance (G,t) of Independent Set. Assume that for any t-sized subset S of $V(G)$, each of s in S is adjacent to a vertex in $V(G)\setminus S$, and there is an edge inside $V(G)\setminus S$. Any graph G can be tweaked to meet this assumption by adding a $(t+2)$-sized clique C to the graph and connecting each vertex of G to each vertex of C. Note that no vertex of C can be in any independent set of size at least two, and any independent set of the original graph is present in the newly constructed graph. Thus, the new instance of Independent Set is equivalent to the original one, and it satisfies the assumption stated.

From (G,t), we construct an instance of Categorical k-Clustering with Column Outliers over the alphabet $\Sigma=\{0,1\}$. The input matrix $\boldsymbol{A}$ is the incidence matrix of G where columns are indexed by vertices and rows by edges. Set $k=t+1$, $\ell=|V(G)|-t$, $D=0$. See Figure 6.2 for an example.

Now it remains to show the correctness of the reduction. Assume there is an independent set I of size t in G. Let the inliers be the columns corresponding to I, and O be the set of the remaining outlier columns. Restricted to the inlier columns, every row of $\boldsymbol{A}$ is either a zero row, or has a single one, since no edge of G has both ends in I. Thus, there are at most $t+1$ distinct rows in $\boldsymbol{A}^{-O}$, so there is a zero-cost

$(t+1)$-clustering. The centers in this clustering are the following t-rows: the zero row and all the t possible rows with a single one.

For the other direction of the correctness proof, assume there is a solution to the constructed Categorical k-Clustering with Column Outliers instance. Consider the set I of the vertices corresponding to the inlier columns, and the set O of the outlier columns. By the assumption on G, every vertex in I is adjacent to a vertex in $V(G) \setminus I$, and there is an edge inside $V(G) \setminus I$. Thus, $\boldsymbol{A}^{-O}$ has a zero row and all t distinct rows with a single one. If there is an edge e inside I, then there are at least $t+2$ distinct rows in the matrix, since even after removing the columns of O, the row corresponding to e has two ones. In this case no zero-cost clustering with $t+1$ clusters is possible. Thus, I must be an independent set.

Thus, the reduction is valid. It is well-known that Independent Set is W[1]-hard, and also not solvable in time $f(t) \cdot |V(G)|^{o(t)}$ for any computable function f unless ETH fails. Observing that $k = t+1$ and $d - \ell = t$ concludes the proof of the lemma. □

Since Lemma 6.12 gives a parameterized reduction from Categorical k-Clustering with Column Outliers to Constrained Clustering with Outliers, we immediately get the following corollary.

Corollary 6.25. Constrained Clustering with Outliers *is W[1]-hard when parameterized by* $k + (n - \ell)$ *when* $D = 0$ *and* $\Sigma = \{0, 1\}$.

We do not get the analogous ETH bound however, as the number of clusters in the constructed instance of Constrained Clustering with Outliers is exponential.

The second lower bound shows that bounding both the number of outliers and the cost D is insufficient for an FPT algorithm as well. The reduction is essentially the same, only we start from the Partial Vertex Cover problem. The input matrix is again the incidence matrix of G, and now we argue that choosing the set of t outlier columns minimizing the number of distinct rows corresponds to the best partial vertex cover of G of size t. The analysis is sufficiently more technical than in the previous case.

Lemma 6.26. Categorical k-Clustering with Column Outliers *is W[1]-hard when parameterized by* ℓ *when* $D = 0$ *and* $\Sigma = \{0, 1\}$. *Assuming ETH, there is no algorithm solving the problem with* $D = 0$ *and the binary alphabet in time* $d^{o(\ell)} \cdot n^{\mathcal{O}(1)}$.

Proof. We show a reduction from the Partial Vertex Cover problem. The input to Partial Vertex Cover is a graph G and numbers t, q. The problem is to decide whether there is a subset $C \subset V(G)$ of size at most t such that at least q edges of G are covered by C. The Partial Vertex Cover problem is well-known to be as hard as Independent Set ([70], Theorem 13.6). That is, when Partial Vertex Cover is parameterized by t, there is a parameter-preserving reduction from Independent Set.

Consider an instance (G, t, q) of Partial Vertex Cover. First, we modify G to obtain a new graph G'. Let P be a set of two new vertices, and D be a set of d new vertices, where $d = 5+t+|E(G)|-q$. Set $V(G')$ to $V(G) \cup P \cup D$. Keep all edges of G on $V(G)$, and connect each vertex of P to all other vertices of G', including the other vertex of P. That is,

$$E(G') = E(G) \cup \{pv \mid p \in P, v \in V(G') \setminus \{p\}\}.$$

The total number of edges in G' is $|E(G)| + 2 \cdot |V(G')| - 3$.

The graph G' behaves in the same way as G with respect to Partial Vertex Cover, as stated in the following claim. Set $t' = t + 2$ and $q' = q + 2 \cdot |V(G')| - 3$.

Claim 6.27. *The instance (G, t, q) is a yes-instance of* Partial Vertex Cover *if and only if (G', t', q') is a yes-instance of* Partial Vertex Cover.

Proof. First, assume there is a subset $C \subset V(G)$ of size at most t covering at least q edges of G. Set $C' = C \cup P$, $|C'| \leq t + 2 = t'$, and vertices of P cover the additional $2 \cdot |V(G')| - 3$ edges which are not present in G.

To the other direction, assume there is a subset $C' \subset V(G')$ of size at most t' covering at least q' edges of G'. We may assume that $P \subset C'$, otherwise we may replace any vertex of C by a vertex of P, and the number of covered edges does not decrease since vertices in P are adjacent to all vertices. Now, $C = C' \cap V(G)$ is a solution for (G, t, q): $|C| \leq |C' \setminus P| \leq t$, and since $|E(G') \setminus E(G)| = q' - q$, C must cover at least q edges in G. □

Now we construct the input instance $(\boldsymbol{A}, k, D, \ell)$ of Categorical k-Clustering with Column Outliers over the alphabet $\Sigma = \{0, 1\}$. The input matrix $\boldsymbol{A}$ is the incidence matrix of G' where columns are indexed by vertices and rows by edges. Set $k = 1 + |V(G')| - t' + |E(G')| - q'$, $\ell = t'$, $D = 0$.

To show the correctness of the reduction, first assume there is a set $C \subset V'$ of size t' covering at least q' edges in G'. Let O be the set of columns in $\boldsymbol{A}$ corresponding to C. We claim that there are at most $1 + (|V(G')| - t') + (|E(G')| - q')$ distinct rows in $\boldsymbol{A}^{-O}$. That is, at most one zero row, at most $|V(G')| - t'$ rows with a single one corresponding to the vertices of $V(G') \setminus C$ adjacent to C, and at most $|E(G')| - q'$ rows with two ones corresponding to the edges not covered by C. Since there are at most k distinct rows, the cost of k-clustering is zero if we remove the columns of O.

In the other direction, assume there is a solution to the constructed Categorical k-Clustering with Column Outliers instance, that is, a set O of at most t' columns such that $\boldsymbol{A}^{-O}$ has at most k distinct rows. Consider the set C of vertices corresponding to the outlier columns O. First, we show that C must contain P. If this does not hold, either $C \cap P$ is empty, or $|C \cap P| = 1$. Assume $C \cap P$ is empty, in this case out of $2 \cdot |V(G')| - 3$ edges incident to P at most $2t'$ are covered by vertices of C. These edges correspond to at least $2 \cdot |V(G')| - 3 - 2t'$ rows of $\boldsymbol{A}$ that have

still two ones each in $\boldsymbol{A}^{-O}$, and thus they are all distinct. However, the number of clusters is smaller than the number of such rows, since

$$2\cdot|V(G')|-3-2t' = |V(G')|-2t'-1+|V(G)|+d > 1+|V(G')|-t'+|E(G')|-q' = k,$$

as $d = 3 + t' + |E(G)| - q$ and $|E(G)| - q = |E(G')| - q'$. This is a contradiction to O being a solution.

In the other case $|C \cap P| = 1$, denote by p the vertex of P that is in C, and by p' the other vertex of P. At most t' vertices in $V(G') \setminus \{p\}$ are in C, thus there are at least $|V(G')| - 1 - t'$ distinct rows in $\boldsymbol{A}^{-O}$ that correspond to edges from p to $V(G') \setminus C$, they each have a single one in the column corresponding to the second endpoint. There are also at least $|V(G')|-2-t'$ distinct rows in $\boldsymbol{A}^{-O}$ that correspond to edges from p' to $V(G') \setminus C$, they each have two ones in the columns corresponding to p' and the second endpoint. Thus, there are at least $2 \cdot |V(G')| - 3 - 2t'$ distinct rows in $\boldsymbol{A}^{-O}$, and we have already shown that this quantity is strictly larger than k, leading to the contradiction.

Now that we have shown $P \subset C$, we argue that all possible rows with less than two ones are present in $\boldsymbol{A}^{-O}$, giving an exact bound on the number of distinct rows with two ones. Formally, since $P \subset C$, there is a zero row in $\boldsymbol{A}^{-O}$ corresponding to the edge inside P. Take any $p \in P$, since $p \in C$ and any vertex v in $|V(G')| \setminus C$ is adjacent to p, there is a row in $\boldsymbol{A}^{-O}$ corresponding to the edge vp that has a single one in the column corresponding to v. This gives $|V(G')| - t'$ distinct rows with single ones, one for each vertex of $V(G') \setminus C$. Thus, there must be at most $k - 1 - (|V(G')| - t') = |E(G')| - q'$ distinct rows with two ones in $\boldsymbol{A}^{-O}$. These rows correspond exactly to the uncovered by C edges of G'. So C covers at least q' edges, and (G', t', q') is a yes-instance. This finishes the correctness proof, and from the hardness of Partial Vertex Cover the lemma follows. □

Clearly, Theorem 6.2, restated next for convenience, follows from Lemma 6.24 together with Lemma 6.26.

Theorem 6.2. Categorical k-Clustering with Column Outliers *is* W[1]*-hard parameterized by*

- *either* $k + (d - \ell)$
- *or* ℓ

even when $D = 0$ *and* $\Sigma = \{0, 1\}$. *Moreover, assuming the Exponential Time Hypothesis (ETH), the problem cannot be solved in time* $f(k) \cdot d^{o(k)} \cdot n^{\mathcal{O}(1)}$ *for any function* f*, even when* $D = 0$ *and the alphabet* Σ *is binary.*

6.5 Conclusion

The results presented in this chapter initiate the systematic study of parameterized complexity of robust categorical data clustering problems. In particular, for Cat-

egorical k-Clustering with Row Outliers, we proved that the problem can be solved in $2^{\mathcal{O}(D \log D)}|\Sigma|^D \cdot (nd)^{\mathcal{O}(1)}$ time. Further, we considered the case of row outliers and proved that Categorical k-Clustering with Column Outliers is solvable in time $f(k, D, |\Sigma|) \cdot d^{g(k,|\Sigma|)}n^2$. We also proved that we cannot avoid the dependence on k in the degree of the polynomial of the input size in the running time unless $\mathsf{W}[1] = \mathsf{FPT}$, and the problem cannot be solved in $d^{o(k)} \cdot n^{\mathcal{O}(1)}$ time, unless ETH is false. To deal with row outliers, we introduced the Constrained Clustering with Outliers problem and obtained the algorithm with running time $(kD)^{\mathcal{O}(kD)}|\Sigma|^{kD} \cdot d^2 n^{\mathcal{O}(k)}$. This problem is very general, and the algorithm for it not only allowed us to get the result for Categorical k-Clustering with Row Outliers, but also led to algorithms for robust low rank approximation problems. In particular, we obtained that Robust L_0-Low Rank Approximation is FPT if k and p are constants when the problem is parameterized by D. However, even if the low rank approximation problems are closely related to the matrix clustering problems, they are quite different. Just to give an illustrative example, the parameterized complexity of the column and row outliers variants of the clustering problem considered in our paper are different with respect to k, but clearly the low rank approximation problems for matrices over fields are symmetric with respect to rows and columns. This leads to the question whether Robust L_0-Low Rank Approximation, Robust Low Boolean-Rank Approximation and Robust Projective Clustering could be solved by better algorithms specially tailored for these problems. It is unlikely that the possible improvements would considerably change the general qualitative picture. For example, Robust L_0-Low Rank Approximation for $p = 2$ and $\ell = 0$ is NP-complete if $k = 2$ [72, 100] and $\mathsf{W}[1]$-hard when parameterized by D [91]. It is also easy to observe that Robust L_0-Low Rank Approximation for $p = 2$, $D = 0$ and $k = n - \ell - 1$ is equivalent to asking whether the input matrix $\mathbf{A}$ has $n - \ell$ linearly dependent rows. This immediately implies that Robust L_0-Low Rank Approximation for $p = 2$ and $D = 0$ is $\mathsf{W}[1]$-hard when parameterized by k or $n - \ell$ by the recent results about the Even Set problem [28]. The most interesting open question, by our opinion, is whether the exponential dependence on k in the degree of the polynomial of the input size in the running time produced by the reduction of Robust L_0-Low Rank Approximation to Constrained Clustering with Outliers from Lemma 6.13 could be avoided, even if p is a constant. Can the dependence of k be made polynomial (or even linear)?

Part III

FPT Approximation Schemes for Clustering

7

Coreset Approach: Color-constrained Clustering

This chapter is dedicated to clustering problems with color constraints. Recall that these are the problems where the input is partitioned into ℓ groups, and the constraints are defined in terms of how many points from each group are assigned to each cluster. The primary tool we use in this chapter are coresets, small-sized summaries of the input. As we discussed in Chapter 4, coresets are of independent interest for a variety of uses in the big data setting, and they also help in designing FPT-time approximation algorithms. We will discuss results of both kinds in this chapter. Our main example is the fair clustering problem introduced by Chierichetti et al. [56]. We employ the most general version of this problem due to [23], we recall the formal definition next.

Definition 7.1 (Definition 1, [23]). *In the fair version of a clustering problem, one is additionally given ℓ many (not necessarily disjoint)* groups *of P, namely P_1, P_2, ..., P_ℓ. One is also given two* fairness vectors $\alpha, \beta \in [0,1]^\ell$, $\alpha = (\alpha_1, \ldots, \alpha_\ell)$, $\beta = (\beta_1, \ldots, \beta_\ell)$. *The objective is to select a set of at most k centers $C \subset F$ and an assignment $\varphi : P \to S$ such that φ satisfies the following* fairness constraints*:*

$$|\{x \in P_i : \varphi(x) = c\}| \leq \alpha_i \cdot |\{x \in P : \varphi(x) = c\}|, \quad \forall c \in C, \forall i \in [\ell],$$
$$|\{x \in P_i : \varphi(x) = c\}| \geq \beta_i \cdot |\{x \in P : \varphi(x) = c\}|, \quad \forall c \in C, \forall i \in [\ell],$$

and $\mathrm{cost}(\varphi)$ *is minimized among all such assignments.*

In the (α, β)-Fair k-median problem, $\mathrm{cost}(\varphi) := \sum_{x \in P} \mathrm{dist}(x, \varphi(x))$, and in the (α, β)-Fair k-means problem, $\mathrm{cost}(\varphi) := \sum_{x \in P} \mathrm{dist}(x, \varphi(x))^2$. To refer to these two problems together, we will use the term (α, β)-Fair Clustering. We call φ that satisfies the fairness constraints *a fair assignment*. We denote the minimum cost of a fair assignment of a set of points P to a set of k centers C by $\mathrm{faircost}(P, C)$, and

faircost(P) denotes the minimum of faircost(P, C') over all possible sets of k centers C'.

Our first main result is the following theorem.

Theorem 7.2 (Informal). *For general discrete metric spaces, there is an $\mathcal{O}(n(k+\ell))$ time randomized algorithm that w.p. at least $1 - 1/n$, computes a coreset of size $\mathcal{O}(\Gamma(k \log n)^2/\varepsilon^3)$ for (α, β)-fair k-Median and $\mathcal{O}(\Gamma(k \log n)^7/\varepsilon^5)$ for (α, β)-fair k-Means, where Γ is the number of distinct collections of groups to which a point may belong. If the groups are disjoint, the algorithm runs in $\mathcal{O}(nk)$ time. Moreover, in $\mathbb{R}^d$, the coreset sizes are*

$$O\left(\frac{\Gamma}{\varepsilon^3} \cdot k^2 \log n(\log n + d \log(1/\varepsilon))\right)$$

for (α, β)-fair k-Median and

$$O\left(\frac{\Gamma}{\varepsilon^5} \cdot k^7 (\log n)^6 (\log n + d \log(1/\varepsilon))\right)$$

for (α, β)-fair k-Means.

Theorem 7.2 provides the first coreset construction for fair clustering problem in general metric spaces. Our result is comparable to the best-known bound of $O_\varepsilon(k \log n)$ [83] in the vanilla case. In particular, if the number of groups in our case is just 1, we obtain coresets of size $O_\varepsilon(\text{poly}(k \log n))$, which matches with the best-known bound in the vanilla case, up to a small degree polynomial factor. We note, that this is the first sampling based coreset construction scheme for fair clustering, and in $\mathbb{R}^d$, the first coreset construction scheme where the size of the coreset does not depend exponentially on the dimension d. In fact, the dependency on d is only linear. Additionally, for k-means objective this dependency can be avoided (replaced by k/ε) by using standard dimension reduction techniques [60, 84] (this was also noted in [173]). Hence, our result solves the open question proposed in [173] and partly solves the open question proposed in [115]. As we already mentioned, in all of the previous results [173, 115], coreset sizes depended exponentially on d (see Table 7.1). We note that the formal statement of Theorem 7.2 appears in Theorems 7.26, 7.30 and 7.37.

In fact, our coreset construction scheme is much more general in the following sense. The coreset can preserve not only the cost of optimal fair clustering, but also the cost of any optimal clustering with group-cardinality constraints. In particular, for any set C of k centers and any constraint matrix $\mathbf{M} \in \mathbb{Z}^{k \times \ell}$, our coreset approximately preserves the cost of an optimal clustering that satisfies $\mathbf{M}$,

$$(1 - \varepsilon) \cdot \text{cost}(P, \mathbf{M}, C) \leq \text{wcost}(W, \mathbf{M}, C) \leq (1 + \varepsilon) \cdot \text{cost}(P, \mathbf{M}, C).$$

	k-Median	k-Means
[173]		$\mathcal{O}(\Gamma k\varepsilon^{-d-2}\log n)$
[115]	$\mathcal{O}(\Gamma k^2\varepsilon^{-d})$	$\mathcal{O}(\Gamma k^3\varepsilon^{-d-1})$
Thm. 7.30 and 7.37	$\mathcal{O}(\frac{\Gamma}{\varepsilon^3}\cdot k^2\log n(\log n+ d\log(1/\varepsilon)))$	$\mathcal{O}(\frac{\Gamma}{\varepsilon^5}\cdot k^7(\log n)^6(\log n+ d\log(1/\varepsilon)))$

Table 7.1: Previous and current coreset sizes in $\mathbb{R}^d$.

Therefore, for any clustering problem with constraints where the constraints can be represented by a set of matrices, we obtain a small size coreset. This gives rise to coresets for a wide range of clustering problems including lower-bounded clustering [179, 5, 23]. Notably, in the case of lower-bounded clustering, the input consists of only one group of points, and thus $\mathbf{M}$ is a column matrix.

We further exploit the new coreset construction to design clustering algorithms in various settings. In general metrics, we obtain the first fixed-parameter tractable (FPT) constant-factor approximation for (α,β)-fair clustering with parameters k and Γ. That is, the running time of our algorithm is exponential only in the values k and Γ while polynomial in the size of the input. All previous constant-approximation algorithms were bicriteria and violated the fairness constraints by some additive factors. Hence, the study of FPT approximation is well-motivated. Our approximation factors are reasonably small and improve the best-known approximation factors of the existing bicriteria algorithms (see Table 7.2). Moreover, our coreset leads to improved constant FPT approximations for many other clustering problems. For example, we obtain an improved ≈ 3-approximation algorithm for lower-bounded k-median [179, 5, 23] that is FPT parameterized by k. Previously, the best-known factor for FPT approximation for this problem was 3.736 [23].

Based on our coreset, we also obtain the first FPT $(1+\varepsilon)$-approximation for (α,β)-fair clustering in $\mathbb{R}^d$ with parameters k and Γ. Furthermore, the running time has a near-linear dependency on n and does not depend exponentially on d. A comparison with the running time of the previous $(1+\varepsilon)$-approximation algorithms can be found in Table 7.3. We also obtain FPT $(1+\varepsilon)$-approximation algorithms with parameter k for the Euclidean version of several other problems including capacitated clustering [66, 62] and lower-bounded clustering. We note that these are the first $(1+\varepsilon)$-approximations for these problems with near-linear dependency on n. For Euclidean capacitated clustering, quadratic time FPT algorithms follow due to [75, 27] (see Table 7.4). Also, the $(1+\varepsilon)$-approximation for Euclidean capacitated clustering in [66] and [62] have running time $(k\varepsilon^{-1})^{k\varepsilon^{-O(1)}}n^{\mathcal{O}(1)}$ and at least $n^{\varepsilon^{-O(1)}}$ (see Table 7.4).

Our coreset also leads to small space $(1+\varepsilon)$-approximation in streaming setting for (α,β)-fair clustering in $\mathbb{R}^d$ when the groups are disjoint. We show how to maintain

an $\mathcal{O}(d^2\ell \cdot \text{poly}(k \log n)/\varepsilon^4)$ size coreset in each step. One can apply our $(1+\varepsilon)$-approximation algorithm on the coreset to compute a near-optimal clustering. In the previous streaming algorithms [173], the space complexity depended exponentially on either d or k.

Our technical contributions are summarized in Section 7.1.

	multi	k-Median approx.	k-Means approx.	running time
[24]		$(4.675, 1)$	$(62.856, 1)$	$\text{poly}(n)$
[23]	✓	$(O(1), 4\Lambda + 3)$	$(O(1), 4\Lambda + 3)$	$\text{poly}(n)$
Thm. 7.50		≈ 3	≈ 9	$(k\ell)^{\mathcal{O}(k\ell)} n \log n$
Thm. 7.50	✓	≈ 3	≈ 9	$(k\Gamma)^{\mathcal{O}(k\Gamma)} n \log n$

Table 7.2: Approximation results for (α, β)-fair clustering in general metrics. "multi" denotes if the algorithm can handle overlapping groups. In "approx." columns, the first (resp. second) value in a tuple is the approximation factor (resp. violation). [23] does not explicitly compute the $\mathcal{O}(1)$ factor, but it is $> 3+\varepsilon$ (resp. $> 9+\varepsilon$) for k-median (resp. k-means), where ε is a sufficiently large constant.

	running time	version
[173]	$n^{\mathcal{O}(k/\varepsilon)}$	2-color, $(1, k)$-fair clustering
[115]	$(k^2\varepsilon^{-d})^{\mathcal{O}(k/\varepsilon)} + O(k\varepsilon^{-d+1}n)$	2-color, $(1, k)$-fair clustering
[30]	$n^{\text{poly}(k/\varepsilon)}$	ℓ-color, $(1, k)$-fair clustering
Thm. 7.45	$2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(k\Gamma)^{\mathcal{O}(k\Gamma)} nd \log n$	(α, β)-fair clustering

Table 7.3: The running time of the $(1+\varepsilon)$-approximations for fair clustering in $\mathbb{R}^d$.

Schmidt et al. [173] defined the concept of fair coresets and gave coreset of size $\mathcal{O}(\ell k \varepsilon^{-d-2} \log n)$ for the disjoint group case of Euclidean (α, β)-fair k-means. This

	running time
[75]	$2^{\text{poly}(k/\varepsilon)} n^2 (\log n)^{k+2} d$
[27]	$2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})} \cdot n^2 (\log n)^2 d$
[66]	$(k\varepsilon^{-1})^{k\varepsilon^{-O(1)}} n^{\mathcal{O}(1)}$
[62]	$n^{\varepsilon^{-O(1)}} (d = 2)$ $n^{(\log n/\varepsilon)^{\mathcal{O}(d)}} (d \geq 3)$
Thm. 7.58	$2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})} nd^{\mathcal{O}(1)} + nk^2 \varepsilon^{-O(1)} \log n$

Table 7.4: The running time of the $(1+\varepsilon)$-approximations for capacitated clustering in $\mathbb{R}^d$.

can be extended to the overlapping case by replacing ℓ with Γ in the size bound. Using a sophisticated dimension reduction technique [60], they showed how to stream coreset whose size does not depend exponentially on d. Unfortunately, this coreset size depends exponentially on k. Schmidt et al. also gave an $n^{\mathcal{O}(k/\varepsilon)}$ time $(1+\varepsilon)$-approximation for the two-color version of the problem. Note that this chapter improves over all these results (see Tables 7.1 and 7.3). Using the framework in [109], Huang et al. [115] improved the coreset size bound of [173] by a factor of $\Theta\left(\frac{\log n}{\varepsilon k^2}\right)$ and gave the first coreset for Euclidean (α, β)-fair k-median of size $\mathcal{O}(\Gamma k^2 \varepsilon^{-d})$. Both the coreset construction schemes in [173] and [115] use deterministic algorithms, and thus they proposed whether random sampling can be employed to remove the curse of dimensionality. Note that our result based on random sampling improves the bound (for k-median) in [115] by a factor of $\Theta\left(\frac{\varepsilon^{-d+3}}{\log n(\log n+d)}\right)$ (see Table 7.1). By applying the $(1+\varepsilon)$-approximation of [173] on their coreset, Huang et al. [115] obtained an algorithm with improved running time. However, the algorithm of [173] is only for two colors. Moreover, due to the inherent exponential dependency on d of the coreset size, the running time of the algorithm in [115] still depends exponentially on d (see Table 7.3). Böhm et al. [30] considered $(1, k)$-fair clustering with multiple colors. They designed near-linear time constant-approximation algorithms in this restricted setting. They also obtained an $n^{\text{poly}(k/\varepsilon)}$ time $(1+\varepsilon)$-approximation for the Euclidean version in the same setting. An FPT $(1+\varepsilon)$-approximation follows from our results for this version (see Table 7.3).

Chierichetti et al. [56] gave a polynomial time $\Theta(t)$-approximation for (t, k)-fair k-median with two groups (or colors). We improve their result by giving an FPT constant-approximation algorithm with parameters k and ℓ for (t, k)-fair clustering with arbitrary number of colors. Based on the framework implicitly mentioned in [45], Bera et al. [23] obtained polynomial time $\mathcal{O}(1)$-approximation for (α, β)-fair clustering that violates the fairness constraints by at most an additive factor of

$4\Lambda + 3$. This framework first computes k centers using a ρ-approximation algorithm for vanilla clustering, and then finds an assignment of the points to these centers that satisfies the fairness constraints. They showed, e.g, for k-median, there is always such an assignment whose cost is at most $\rho + 2$ times the optimal cost of fair clustering. However, computing such an assignment is not an easy task. Indeed, this is a big hurdle one faces while studying fair clustering, which makes this problem substantially harder compared to other clustering problems like capacitated clustering. Based on the algorithm due to Király et al. [131], Bera et al. [23] showed that an optimal assignment can be computed by violating any fairness constraint by the mentioned factor. For the disjoint group case, their violation factor is only 3. Independently, Bercea et al. [24] obtained algorithms with the same approximation guarantees as in [23] for the disjoint version, but with at most 1 additive factor violation. We show that the above mentioned assignment problem for (α, β)-fair clustering can be solved exactly in FPT time parameterized by k and Γ. Plugging this in with our coreset, we obtain algorithms with better constant approximation factors compared to [23] and [24] that do not violate any constraint (see Table 7.2).

Ding and Xu [75] gave an unified framework with running time $2^{\text{poly}(k/\varepsilon)}(\log n)^{k+1}nd$ that generates a collection of candidate sets of centers for clustering problems with constraints in $\mathbb{R}^d$. Subsequently, Bhattacharya et al. [27] and Feng et al. [86] designed similar frameworks having improved time complexity. None of these works study fair clustering. The results of this chapter can be viewed as an extension of these works to general metrics in the sense that we obtain constant-approximations for a range of constrained clustering problems. Furthermore, by applying the framework of [27] on our coreset, we obtain $(1+\varepsilon)$-approximation algorithms with improved time complexity bounds for several clustering problems in $\mathbb{R}^d$.

Organization. Section 7.1 summarizes the main technical ideas used to obtain the new results. The "stronger coreset" construction algorithm appers in Section 7.2. In the rest of the chapter, we describe the applications of our coresets. In Section 7.3, we describe an algorithm for solving an assignment problem, which we will need to design our algorithms for (α, β)-fair clustering. In Section 7.4 and 7.5, we describe our approximation algorithms for the Euclidean and metric case of (α, β)-fair clustering, respectively. In Section 7.6, we apply our coreset to design improved algorithms for other constrained clustering problems. In Section 7.7, we show how to maintain our coreset in the streaming setting. Finally, in Section 7.8, we conclude with some open questions.

7.1 Our Techniques

In this section, we summarize the techniques and key ideas used to obtain the new results of this chapter. The detailed version of our results and formal proofs appear

in the following sections. For simplicity, we limit our discussion here to k-median clustering. We start with the coreset results.

7.1.1 Universal Coreset Construction

Our coreset construction algorithms are based on random sampling and we will prove that our algorithms produce universal coresets with high probability (w.h.p.). At first glance, it is not easy to see how to sample points in the overlapping group case, as the decision has an effect on multiple groups. To give intuition to the reader, at first we discuss the disjoint group case.

The Disjoint Group Case

Our coreset construction algorithm is built upon the coreset construction algorithm for vanilla clustering due to Chen [53]. In our case, we have points from ℓ disjoint color classes. So, we apply Chen's algorithm for each color class independently. Note that Chen's algorithm was used to show that for any given set of centers C, the constructed coreset approximately preserves the optimal clustering cost. However, we would like to show that for any given set of centers C, the constructed coreset approximately preserves the optimal clustering cost corresponding to any given constraint $\mathbf{M}$. At this stage, it is not clear why Chen's algorithm should work in such a generic setting. Our main technical contribution is to show that sampling based approaches like Chen's algorithm can be used even for such a stronger notion of universal coreset. We will try to give some intuition after describing our algorithm. Our algorithm is as follows.

Given the set of points P, first we apply the algorithm of Indyk [119] for computing a vanilla k-median clustering of P. This is a bicriteria approximation algorithm that uses $\mathcal{O}(k)$ centers and runs in $\mathcal{O}(nk)$ time. Let C^* be the set of computed centers, ν be the constant approximation factor and Π be the cost of the clustering. Also, let $\mu = \Pi/(\nu n)$ be a lower bound on the average cost of the points in any optimal k-median clustering. Note that for any point p, $\text{dist}(p, C^*) \leq \Pi = \nu n \cdot \mu$.

For each center $c_i^* \in C^*$, let $P_i^* \subseteq P$ be the corresponding cluster of points assigned to c_i^*. We consider the ball $B_{i,j}$ centered at c_i^* and having radius $2^j\mu$ for $0 \leq j \leq N$, where $N = \lceil \log(\nu n) \rceil$. We note that any point at a distance $2^N\mu \geq \nu n \cdot \mu$ from c_i^* is in $B_{i,N}$, and thus all the points in P_i^* are also in $B_{i,N}$. Let $B'_{i,0} = B_{i,0}$ and $B'_{i,j} = B_{i,j} \setminus B_{i,j-1}$ for $1 \leq j \leq N$. We refer to each such $B'_{i,j}$ as a ring for $1 \leq i \leq k, 0 \leq j \leq N$. For each $0 \leq j \leq N$ and color $1 \leq t \leq \ell$, let $P'_{i,j,t}$ be the set of points in $B'_{i,j}$ of color t. Let $s = \Theta(k \log n/\varepsilon^3)$ for a sufficiently large constant hidden in $\Theta(.)$.

For each center $c_i^* \in C^*$, we perform the following steps.

Random Sampling. For each color $1 \leq t \leq \ell$ and ring index $0 \leq j \leq N$, do the following. If $|P'_{i,j,t}| \leq s$, add all the points of $P'_{i,j,t}$ to $W_{i,j}$ and set the weight of each

such point to 1. Otherwise, select s points from $P'_{i,j,t}$ independently and randomly (without replacement) and add them to $W_{i,j}$. Set the weight of each such point to $|P'_{i,j,t}|/s$.

The set $W = \cup_{i,j} W_{i,j}$ is the desired universal coreset. As the number of rings is $\mathcal{O}(k \log n)$, the size of W is $\mathcal{O}(\ell(k \log n)^2/\varepsilon^3)$. From [53], it follows that for each color, the coreset points can be computed in time linear in the number of points of that color times $\mathcal{O}(k)$. Thus, our coreset construction algorithm runs in $\mathcal{O}(nk)$ time.

An Intuitive Discussion about Correctness. Note that we need to show that for any set of centers C, the optimal clustering cost is approximately preserved w.r.t. all possible combination of cluster sizes as defined by the constraint matrices. In Chen's analysis, it was sufficient to argue that for any set of centers C, the optimal clustering cost needs to be preserved. This seems much easier compared to our case. (Obviously, the details are much more complicated even in the vanilla case.) For example, suppose $p \in P$ be a point that is assigned to a center $c \in C$ in an optimal clustering. Note that c must be a closest center to p. For simplicity, suppose p has a unique closest center. Now, if p is chosen in the coreset, then the total weight of p must also be assigned to c in any optimal assignment w.r.t. C. Thus, the assignment function for original and coreset points remains same in the vanilla case. This fact is in the heart of their analysis. Let h be this assignment function: $h(p) = \text{dist}(p, C)$ and for any set S, $h(S) = \sum_{p \in S} h(p)$. Consider any point set V and an uniformly drawn random subset $U \subseteq V$. Also, assume that $h(p)$ lies in an interval of size T. Then, using a result due to Haussler [112], one can show that if $|U|$ is sufficiently large, then w.h.p, $\left|\frac{h(V)}{|V|} - \frac{h(U)}{|U|}\right| \leq \varepsilon T$. Now, we can apply this observation to each ring separately. Note that for any ring $B'_{i,j}$ with points $P_{i,j}$, and for all $p \in P_{i,j}$, $h(p)$ is in an interval I of length at most the diameter of the ball $B_{i,j}$, i.e, $2(2^j\mu)$. It follows that,

$$\left|\sum_{p \in P_{i,j}} \text{dist}(p, C) - \sum_{p \in W_{i,j}} w(p) \cdot \text{dist}(p, C)\right| \leq |P_{i,j}| \cdot \left|\frac{h(P_{i,j})}{|P_{i,j}|} - \frac{|P_{i,j}|}{|W_{i,j}|} \cdot \frac{h(W_{i,j})}{|P_{i,j}|}\right|$$
$$\leq |P_{i,j}| \cdot \varepsilon 2^{j+1}\mu$$

The first inequality follows, as the weight of each point in $W_{i,j}$ was set to $\frac{|P_{i,j}|}{|W_{i,j}|}$. The second inequality follows from the observation mentioned above. Summing over all rings, we get,

$$\left|\sum_{p \in P} \text{dist}(p, C) - \sum_{p \in W} w(p) \cdot \text{dist}(p, C)\right| \leq \sum_{(i,j)} |P_{i,j}| \cdot \varepsilon 2^{j+1}\mu$$

Now, as we show later, one can upper-bound this by $\mathcal{O}(\varepsilon \cdot \text{OPT}_v)$, where OPT_v is the optimal cost of vanilla clustering. This is shown by charging the error bound

for each point with its cost in the bicriteria solution. Now, note that in the case of vanilla k-median, cost of a weighted set S of points in an optimal clustering with centers in C, $\text{wcost}(S, C) = \sum_{p \in S} w(p) \cdot \text{dist}(p, C)$ (similarly define $\text{cost}(P, C)$). By scaling ε appropriately and taking union bound over all rings, we obtain that w.h.p,

$$\big|\text{cost}(P, C) - \text{wcost}(W, C)\big| \leq \varepsilon \cdot \text{cost}(P, C).$$

This is how Chen obtained the bound for k-median. Note that the observation that a coreset point has the same optimal assignment as the one w.r.t. the original point set is not-necessarily true in our case. We cannot just use the nearest neighbor assignment scheme, as in our case cluster sizes are predefined through $\mathbf{M}$. Indeed, in our case we might very well need to assign the weight of a coreset point to multiple centers to satisfy $\mathbf{M}$. In general, this is the main hurdle one faces while analyzing a sampling based approach for fair coreset construction.

For analyzing our algorithm, we follow an approach similar to the one by Cohen-Addad and Li in [66]. They considered the capacitated clustering problem, where for each center c a capacity value U_c is given, and if the center c is chosen, at most U_c points can be assigned to c. They analyzed Chen's algorithm and showed that for any center C, the coreset approximately preserves the optimal capacitated clustering cost. In the following we describe their approach.

Fix a set C of centers. Again consider a single ring $B'_{i,j}$ and assume that we sample points from only this ring. Thus the coreset consists of sampled points from this ring and original points from the other rings. We would like to obtain an error bound for the points $P_{i,j}$ in $B'_{i,j}$ similar to the one in the vanilla case. For simplicity, let $P' = P_{i,j}$, $m = |P'|$ and $\mu' = 2^j \mu$. Also, let S be the samples chosen from P'. Recall that $|S| = s$. Let W' be the coreset, i.e, $W' = S \cup (P \setminus P')$. Instead of directly analyzing the sampling scheme of Chen, they consider a different sampling scheme. The two sampling schemes are same up to repetition as they argue. This is one of the most important ideas that they use in the analysis.

An Alternative Way of Sampling. For each $p \in P'$, select p w.p. s/m independently and set its weight to m/s. Otherwise, set its weight to 0. Let $X \in \mathbb{R}^m_{\geq 0}$ be the corresponding random vector such that $X[p] = m/s$ if p is selected, otherwise $X[p] = 0$.

We note two things here. First, for each p, $\mathrm{E}[X[p]] = 1$. Thus, $\mathrm{E}[X] = \mathbb{1}$, where $\mathbb{1}$ is the vector of length m whose entries are all 1. Intuitively, this shows that in expectation the chosen set of samples behave like the original points. They heavily use this connection in their analysis. Second, this sampling is different from the original sampling scheme in the sense that here we might end up selecting more (or less) than s samples. However, one can show that with sufficient probability, this sampling scheme selects exactly s points, as the expected number is $m \cdot (s/m) = s$. It follows that X contains exactly s non-zero entries with the same probability. Conditioned on this event, X accurately represents the outcome of the original sampling process.

Thus, both the sampling processes are same up to repetition. Henceforth, we assume that X contains exactly s non-zero entries.

The next crucial idea is to represent assignments through network flow. Suppose we are given a fixed set of centers and weighted input points and we would like to compute a minimum cost assignment of the points to the centers such that the capacities are not violated. This problem can be modeled as a minimum cost network flow problem. In particular, given any vector Y that represents weights of the points, one can compute a network G_Y. A minimum cost flow in this network corresponds to a minimum cost assignment. For any $Y \in \mathbb{R}^m_{\geq 0}$, we denote by $f(Y)$ the minimum cost of any feasible flow in G_Y. Note that as the weight of the points in $P \setminus P'$ are fixed, it is sufficient to consider an m-dimensional vector to represent the weights of the points in P'.

Now, note that $f(X)$ and $\text{wcost}(W', C)$ (for capacitated clustering) are identically distributed, as X contains exactly s non-zero entries. Also, as $\mathrm{E}[X] = \mathbb{1}$, $f(\mathrm{E}[X]) = f(\mathbb{1}) = \text{cost}(P, C)$. Thus it is sufficient to prove that w.h.p, $|f(X) - f(\mathrm{E}[X])| \leq \varepsilon m\mu'$. They show this in two steps. First, w.h.p, $|f(X) - \mathrm{E}[f(X)]| \leq \varepsilon m\mu'/2$, which can be proved using a variant of Chernoff bound. Then, they show that $|\mathrm{E}[f(X)] - f(\mathrm{E}[X])| \leq \varepsilon m\mu'/2$.

The proof in the second step is much more involved. First, they show that $f(\mathrm{E}[X]) \leq \mathrm{E}[f(X)]$. This follows from the fact that the value of $f(\mathbb{1})$ is not more than the average value of $f(X)$, as one can find an assignment of cost at most $\mathrm{E}[f(X)]$ where 1 weight is assigned for each point, by summing up the costs of all assignments weighted by their probabilities. The proof completes by showing $\mathrm{E}[f(X)] \leq f(\mathrm{E}[X]) + \varepsilon m\mu'/2$. It is not hard to prove that (i) $f(X) \leq f(\mathrm{E}[X]) + nm\mu'$. They show that (ii) w.p. at least $1 - 1/n^{10}$, $f(X) \leq f(\mathrm{E}[X]) + 0.49\varepsilon m\mu'$. From these above two claims, we obtain $\mathrm{E}[f(X)] \leq f(\mathrm{E}[X]) + \varepsilon m\mu'/2$. The proof that $f(X) \leq f(\mathrm{E}[X]) + 0.49\varepsilon m\mu'$ holds w.p. at least $1 - 1/n^{10}$ is the most crucial part of their analysis. To prove this, they start with an assignment corresponding to the cost $f(\mathbb{1})$, i.e, an original assignment where all points are assigned to the centers. They compute a feasible assignment corresponding to the vector X, by modifying this assignment whose cost is at most $f(\mathbb{1}) + 0.49\varepsilon m\mu'$ w.p. at least $1 - 1/n^{10}$. The details are much more involved. But, the crucial part is that the given assignment can be represented as a flow, and can be modified to obtain a new feasible flow in G_X whose cost is not much larger than $f(\mathbb{1})$.

Now, let us come back to fair clustering. The first hurdle to adapt the approach in [66] is that it is not possible to represent the assignment problem for fair clustering as a simple flow computation problem. It can be modeled as an ILP. But, then we loose the "nice" structure of the function f that is needed for analysis. For example, they show that f is a Lipschitz function and that helps them obtain good concentration bound. Thus it is not clear how to directly use their approach for fair clustering. However, we show that for a fixed constraint $\mathbf{M}$, the assignment problem can be modeled in the desired way. Thus, we can get high probability bound w.r.t. a fixed

constraint $\mathbf{M}$. However, to obtain a coreset for fair clustering we need to show this w.r.t. all such constraints (and this leads us towards a universal coreset). The number of such constraints can be as large as $n^{\Omega(k\ell)}$. Hence, to obtain the h.p. bound over all $\mathbf{M}$, we need to show that for a fixed $\mathbf{M}$ the error probability is at most $1/n^{\Omega(k\ell)}$. However, it is not clear how to show such a bound ($1/n^{\Omega(k)}$ bound can be shown). Nevertheless, we show that it is not necessary to consider all those choices of the constraints together – one can focus on a single color and the constraints w.r.t. that color only. Indeed, this is the reason that we apply Chen's algorithm to different color classes independently. Unfortunately, we pay a heavy toll for this: the coreset size is proportional to ℓ, unlike the vanilla coreset size. However, it is not clear how to avoid this dependency. Nevertheless, this solves our problem, as now we have only $n^{\Omega(k)}$ constraints.

The Overlapping Group Case

Recall that we are given ℓ groups of points $P_1, \ldots, P_\ell$ such that a point can potentially belong to multiple groups. In this section we design a sampling based algorithm for construction of universal coreset in this case. Note that the algorithm in the disjoint case clearly does not work. This is because we sample points from each group separately and independently, and thus it is not clear how to assign the weight of a point that belongs to multiple groups. One might think of the following trivial modification of the algorithm in the disjoint case. Assign each point to a single group to which it belongs. Based on this assignment, now we have disjoint groups, and we can apply our previous algorithm. However, this algorithm can have a very large error bound. For example, suppose a point p belongs to two groups i and j, and it is assigned to group i. Also, suppose p was not chosen in the sampling process. Note that the weight of p is represented by some other chosen point p', which was also assigned to group i. However, now we have lost the information that this weight of p was also contributing towards fairness of group j. Thus, the constructed coreset might not preserve any optimal fair clustering with a small error. In the overlapping case, it is not clear how to obtain a coreset whose size depends linearly in ℓ. Nevertheless, we design a new coreset construction algorithm that have very small error bound and its size depends linearly on Γ. As we noted before, in practice Γ is reasonably small, a polynomial in ℓ.

The main idea of our algorithm is to divide the points into equivalence classes based on their group membership and sample points from each equivalence class. Let $P = \cup_{i=1}^{\ell} P_i$. For each point $p \in P$, let $J_p \subseteq [\ell]$ be the set of indexes of the groups to which p belongs. Let I be the distinct collection of these sets $\{J_p \mid p \in P\}$ and $|I| = \Gamma$. In particular, let $I_1, \ldots, I_\Gamma$ be the distinct sets in I. Now, we partition the points in P based on these sets. For $1 \leq i \leq \Gamma$, let $P^i = \{p \in P \mid I_i = J_p\}$. Thus, $\{P^i \mid 1 \leq i \leq \Gamma\}$ defines equivalence classes for P such that two points $p, p' \in P$ belong to the same equivalence class if they are in exactly the same set of groups. Now we apply our algorithm in the disjoint case on the disjoint sets of points $P^1, \ldots, P^\Gamma$.

Let W be the constructed coreset.

Note that here we have Γ disjoint classes, and thus the coreset size is $\mathcal{O}(\Gamma(k \log n)^2/\varepsilon^3)$. As our coreset size is at least Γ, we assume that $\Gamma < n$. Note that the equivalence classes can be computed in $\mathcal{O}(n\ell)$ time, and thus the algorithm runs in time $\mathcal{O}(n\ell) + \mathcal{O}(nk) = \mathcal{O}(n(k+\ell))$. Next, we argue that W is indeed a universal coreset w.h.p.

An Intuitive Discussion of Correctness. Again, the idea here is to reduce the analysis to the one class case. However, this is not as straightforward as in the disjoint case. Note that although the classes $P^1, \ldots, P^\Gamma$ are disjoint, two classes can contain points from the same group. Moreover, the constraints are defined w.r.t. the groups. Thus, two classes need to interact to satisfy the constraints.

Fix a set of centers C. Let W_τ be the chosen samples from class τ. For any ring $B'_{i,j}$, let $P'_{i,j,\tau}$ be the points from class τ in the ring.

Consider any class $1 \leq t \leq \Gamma$. We can show that if our coreset contains samples from one specific class and original points from the other classes, then the error comes from only that class. In particular, we will show that for all matrices $\mathbf{M}$, w.h.p, $|\operatorname{cost}(P, \mathbf{M}, C) - \operatorname{wcost}(W_t \cup (P \setminus P^t), \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$.

Now, one can safely take union bound over all $\Gamma < n$ classes, to obtain the bound similar to the one in the disjoint case.

Next, we prove the above claim. Denote the size of the set I_t of indexes corresponding to points in P^t by Λ and WLOG, assume that $I_t = \{1, 2, \ldots, \Lambda\}$. To prove the above claim, we show that it is sufficient to prove that w.h.p, for all $k \times \Lambda$ matrix $\mathbf{M}'$ such that $\mathbf{M}'$ has Λ identical columns and the sum of the entries in each column is exactly $|P^t|$, $|\operatorname{cost}(P^t, \mathbf{M}', C) - \operatorname{wcost}(W_t, \mathbf{M}', C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$. Now, as $\mathbf{M}'$ contains all identical columns, points of P^t belong to the same set of groups, and we select samples from P^t separately and independently, this claim boils down to a case similar to the disjoint-group-one-color case.

One might find our approach in parallel with the one in [115], as they also reduce the problem with overlapping groups to a single class. However, in contrast to ours, their coreset construction algorithm is deterministic.

The Euclidean Case

The algorithm in the Euclidean case is the same as for general metrics, except we set s to $\Theta(k \log(nb)/\varepsilon^3)$ instead of $\Theta(k \log n/\varepsilon^3)$, where $b = \Theta(k \log(n/\varepsilon)/\varepsilon^d)$. The analysis for general metrics holds in this case, except the assumption that the number of distinct sets of centers is at most n^k is no longer true. Here any point in $\mathbb{R}^d$ is a potential center. This is the main challenge in the Euclidean case, as now it is not possible to take union bound over all possible sets of k centers. Nevertheless, we show that for every set $C \subseteq \mathbb{R}^d$ of k centers and constraint $\mathbf{M}$, the optimal cost is preserved approximately w.h.p. The idea is to use a discretization technique to

obtain a finite set of centers so that if instead we draw centers from this set, the cost of any clustering is preserved approximately.

First, we construct a set of points F that we will use as the center set. Recall that C^* is the set of centers computed by the bicriteria approximation algorithm. ν is the constant approximation factor and Π is the cost of this clustering. Also, $\mu = \Pi/(\nu n)$.

For each center $c_i^* \in C^*$, we consider the d-dimensional axis-parallel hypercubes $R_{i,j}$ having sidelength $2^j\mu$, and centered at c_i^* for $0 \leq j \leq N$, where N is sufficiently large. Let $R'_{i,0} = R_{i,0}$ and $R'_{i,j} = R_{i,j} \setminus R_{i,j-1}$ for $1 \leq j \leq N$. For each $0 \leq j \leq N$, we divide $R'_{i,j}$ into gridcells of sidelength $\varepsilon 2^j\mu$. Let Q_i be the exponential grid for $R'_{i,0}, \ldots, R'_{i,N}$, i.e., Q_i is the amalgamation of the gridcells in $R'_{i,0}, \ldots, R'_{i,N}$. For each gridcell in the exponential grid Q_i, we select any arbitrary point and add it to F_i.

We repeat the above process for all $c_i^* \in C^*$. Let $F = \cup_i F_i$. One can show that the size of F is $\mathcal{O}(k \log(n/\varepsilon)/\varepsilon^d)$.

Now we show that if the centers can only be chosen from F, then the analysis for general metrics holds in this case as well with the modified value of s mentioned above. We need to extend this argument for any set $C \subseteq \mathbb{R}^d$ of k centers. To do this, we consider two cases. In the first case, C contains a center $\hat{c}$ such that $\hat{c}$ is not in $\cup_i Q_i$. Thus, $\hat{c}$ is very far away from the centers of bicriteria solution. In this case we show that the cost of this clustering is at least $1/\varepsilon$ times the cost of the bicriteria solution. We also showed that the cost difference of any clustering w.r.t. P and W is at most the cost of the bicriteria solution. Together it follows that the above cost difference is at most ε times the cost of the bicriteria solution and we obtain the desired bound w.p. 1.

In the second case, all centers in C are in $\cup_i Q_i$. In this case we can approximate C with C' by choosing centers from F: for each center c, select the point c' in F chosen from the gridcell that contains c. Intuitively, the distance between c and c' is relatively small. Note that we showed before that W is a coreset w.r.t. points in F w.h.p, and so is w.r.t. C'. As C' approximates C, it follows that W is also a coreset w.r.t. C.

7.1.2 Approximation Algorithms Based on Universal Coresets

All the approximation algorithms that we show boil down to one general strategy: first, compute a suitable universal coreset, then, enumerate a small family of sets of possible k centers, such that at least one of them is guaranteed to provide a good approximation, and finally pick the best set of centers by finding the optimal fair assignment from the coreset to each of the center sets. Apart from the coreset construction, the notable challenge in the case of (α, β)-Fair Clustering is solving the assignment problem. We devise a general FPT time algorithm for the assignment problem. The approach for obtaining approximations for other problems are very

similar. Thus, in this summary part, we limit our discussion to fair clustering.

Solving the Assignment Problem

The fair assignment problem is the following: given an instance of (α, β)-Fair Clustering and a set of k centers C, compute a minimum-cost fair assignment to the centers of C. The fair assignment problem is one of the features that makes fair clustering harder than other constrained clustering problems. While often the optimal assignment can be found with the help of a network flow, like in the case of capacitated clustering or lower-bounded clustering, there was no previously known algorithms to compute an optimal or approximate fair assignment without violating the constraints. Moreover, it was observed by Bera et al. [23] that the assignment problem for (α, β)-Fair Clustering is NP-hard, so there is no hope to have a polynomial time assignment algorithm.

We show an assignment algorithm with running time $(k\Gamma)^{\mathcal{O}(k\Gamma)}n^{\mathcal{O}(1)}$, the formal statement and the proof is given in Theorem 7.39. The general idea is to reduce to a linear programming instance. The unknown optimal assignment can be naturally expressed in terms of linear inequalities by introducing a variable f_{ij} for the i-th point and the j-th center, denoting what fraction of the point is assigned to each center, and constraints $f_{ij} \geq 0$ for all i, j, and $\sum_{j=1}^{k} f_{ij} = 1$. Clearly this generalizes a discrete assignment, which corresponds to exactly one of $\{f_{ij}\}_{j=1}^{k}$ being equal to 1, for each $i \in [n]$. Observe that the condition that the assignment is fair can also be expressed as linear constraints: for each $j \in [k]$, summing all f_{ij} from the points belonging to a particular group provides the number of the points from this group assigned to the j-th center. And the fairness conditions just bound the ratio of points from a particular group to the size of the cluster.

However, the issue is that in general the optimal fractional solution to this linear programming problem is not integral, and the integrality gap could be arbitrarily large. Thus, an optimal fractional solution does not yield the desired assignment, and this is not surprising since the fair assignment problem is NP-hard. One possible solution would be to restrict the variables to be integral, solving an integer linear program (ILP) instead. But we cannot afford to make all variables integral, as the number of variables can be sufficiently large. Even if we aim to solve the assignment problem on the coreset, the number of points is polylogarithmic in n, and solving the ILP would take at least $(\log n)^{\Omega(\log n)}$ time, which is not FPT. Instead, we introduce the integral variable g_{tj} denoting how many points from the t-th point equivalence class gets to the j-th center, while leaving the $\{f_{ij}\}$ variables to be fractional. Thus, we obtain an instance of mixed-integer linear programming (MILP) with $k\Gamma$ integer variables and nk fractional variables. By using the celebrated result of Lenstra [142] with subsequent improvements by Kannan [127], and Frank and Tardos [93], we obtain an optimal solution to the MILP instance in time $(k\Gamma)^{\mathcal{O}(k\Gamma)}n^{\mathcal{O}(1)}$.

Now we explain that after constraining the $\{g_{tj}\}$ variables to be integral, we can assume that all the other variables $\{f_{ij}\}$ are integral too, thus we actually obtain an

optimal discrete assignment of the same cost. Consider a particular point equivalence class P^t, and the integral values $\{g_{tj}\}_{j=1}^{k}$ from the optimal solution to the MILP. When these values are fixed, the problem boils down to finding an assignment from P^t to C such that exactly g_{tj} points are assigned to the j-th center. This problem can be solved by a minimum-cost maximum flow in the network where each point has supply one, the j-th center has demand of g_{tj}, and the costs are the distances between the respective points. Moreover, the values $\{f_{ij}\}$ from the MILP correspond exactly to the flow values on the respective edges. Since there is an optimal integral flow in this network, this flow is also an optimal integral solution for $\{f_{ij}\}$.

The downside of Theorem 7.39 is that the dependency on n is a high degree polynomial, roughly n^5, and we cannot use it directly to obtain a near-linear time algorithm. So we also show how to obtain a fair assignment that has the cost of at most $(1+\varepsilon)$ times the optimal fair assignment cost in near-linear time with the help of the coreset. For this, we compute a universal coreset from the input points, and then compute the optimal fair assignment from the coreset to the centers C. Since the coreset preserves the cost of an optimal assignment w.r.t. any constraint matrix M, and fair assignments are precisely those that satisfy a certain set of constraint matrices, we obtain immediately that the cost of the optimal fair assignment on the coreset is within a factor of $(1+\varepsilon)$ from the optimal cost of the original instance. However, this does not yet give us a fair assignment of the original points to the centers. To construct this assignment, we take the values $\{g_{tj}\}$ computed by the assignment algorithm on the coreset, and then, for each point equivalence class P^t, we solve the simple assignment problem from P^t to C that assigns exactly g_{tj} points to the j-th center. As mentioned above, this can be done by a network flow algorithm. Since the network is bipartite and one of the parts is small, only of size k, this problem can be solved in near-linear time by a specialized flow algorithm given by [6]. Finally, the resulting assignment on the original points has cost at most $(1+\varepsilon)$ times the cost of the optimal assignment on the coreset. This holds since the coreset construction preserves the cost with respect to the set of centers C and any constraint matrix M, in particular the one that is constructed from the values $\{g_{tj}\}$. This argument is presented in full detail in Lemma 7.42. Combining the above steps, we obtain a near-linear time algorithm via coreset for the assignment problem on P given a set of centers.

$(1+\varepsilon)$-Approximation in $\mathbb{R}^d$

Apart from our coreset construction and our assignment algorithm, the key ingredient to obtain a $(1+\varepsilon)$-approximation algorithm is the general constrained clustering algorithm of Bhattacharya et al. [27]. Their algorithm outputs a list of $2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}$ candidate sets of k centers, such that for any clustering of the points there exists a set of centers C in this list that is only slightly worse than the optimal set of centers for this clustering. Naturally, this holds for any fair clustering too, thus there exists a set of centers C in the list such that $\operatorname{faircost}(P,C) \leq (1+\varepsilon)\operatorname{faircost}(P)$. Together

with our exact assignment algorithm this provides a $(1+\varepsilon)$-approximation algorithm with the running time of $2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(k\Gamma)^{\mathcal{O}(k\Gamma)}n^{\mathcal{O}(1)}d$: compute the list of candidate sets of centers, then find an optimal assignment to each set, and return the one with the smallest cost. Replacing the exact assignment algorithm with the approximate one that employs coreset, we obtain a $2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(k\Gamma)^{\mathcal{O}(k\Gamma)}nd(\log n)^2$-time algorithm. Finally, if for each candidate set of centers we solve the assignment problem on the coreset, then choose the best set of centers, and then solve (approximately) the assignment problem on the original points and this particular set of centers, we reduce the running time to $2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(k\Gamma)^{\mathcal{O}(k\Gamma)}nd\log n$.

$(3+\varepsilon)$-Approximation in General Metric

With the help of our universal coreset, the strategy to obtain $(3+\varepsilon)$-approximation for (α,β)-Fair k-median is essentially identical to that used in [64] and [66]: from each of the clusters in an optimal solution on the coreset we guess the closest point to the center, called a *leader* of that cluster. We also guess a suitably discretized distance from each leader to the center of the corresponding cluster. Finally, selecting any center that has roughly the guessed distance to the leader provides us with a $(3+\varepsilon)$-approximation. That holds since if we assign each point to the guessed center of its leader, the distance that this point contributes will be at most its distance in the optimal solution, plus the distance from the leader to the optimal center, plus the distance from the leader to the guessed center. Since the leader is the closest point in the cluster to the optimal center, this is at most $(3+\varepsilon)$ times the distance that the point contributes in the optimal solution. Note that this assignment is fair since the composition of the clusters is exactly the same as in the optimal solution.

We cannot directly find this assignment, but we can compute the lowest-cost fair assignment to this set of centers that can only be better. Thus, we solve the assignment problem on the coreset for each guess of the centers, choose the best set of centers, and then compute an approximately optimal fair assignment from the original points to these centers. By the property of the universal coreset, going to the coreset and back changes the cost of the optimal solution only slightly, so with the appropriate selection of error parameters the obtained assignment is a $(3+\varepsilon)$-approximate solution. There are $|W|^k$ possible choices for leaders and $(\log n/\varepsilon)^{\mathcal{O}(k)}$ for the respective distances, and we solve the assignment problem on our coreset for each such guess. Thus, we need a running time of $(k\Gamma)^{\mathcal{O}(k\Gamma)}/\varepsilon^{\mathcal{O}(k)}\cdot n\log n$ to compute the best set of centers and retrieve a corresponding assignment of the original points.

One technical difficulty is that for the distance guessing step we require that the *aspect ratio* of the instance, that is the ratio of the maximum distance between the points in the instance to the minimum, is polynomially bounded. Only in this case we can consider just $(\log n/\varepsilon)^{\mathcal{O}(k)}$ choices for the distances. The technique to reduce the aspect ratio of the instance is fairly standard, it was also employed in [66] for the case of capacitated clustering. It requires a bound on the cost of an optimal solution, and

one notable difference is that for (α, β)-Fair Clustering there were no previously known true approximation algorithm. Thus we also devise a simple linear-time $\mathcal{O}(n)$-approximation, based on the classical min-max algorithm for k-center.

7.2 Coreset Construction

In this section, we present in full detail the coreset construction. We start with k-Median, and first we deal with the simpler disjoint group case. Then we show how to extend the result from disjoint groups to overlapping groups. Finally, we extend We start with k-median in the disjoint group case in Section 7.2.1. Then we extend this result to the overlapping group case in Section 7.2.2. Section 7.2.3 describes the coreset construction for k-median in $\mathbb{R}^d$. Finally, the coreset constructions for k-means appear in Section 7.2.4.

7.2.1 Disjoint Group Case

In this section, we prove the following theorem.

Theorem 7.3. *Given a set P of n points in a metric space along with a color function $c : P \to \{1, \ldots, \ell\}$, there is an $\mathcal{O}(nk)$ time randomized algorithm that w.p. at least $1 - 1/n$, computes a universal coreset for k-median clustering of size $\mathcal{O}(\ell(k \log n)^2/\varepsilon^3)$.*

To prove this theorem, we analyze the coreset construction algorithm in the disjoint group case described in Section 7.1. For convenience of the reader, we again state our algorithm here.

Given the set of points P, first we apply the algorithm of Indyk [119] for computing a vanilla k-median clustering of P. This is a bicriteria approximation algorithm that uses $\mathcal{O}(k)$ centers and runs in $\mathcal{O}(nk)$ time. Let C^* be the set of computed centers, ν be the constant approximation factor and Π be the cost of the clustering. Also, let $\mu = \Pi/(\nu n)$ be a lower bound on the average cost of the points in any optimal k-median clustering. Note that for any point p, $\text{dist}(p, C^*) \leq \Pi = \nu n \cdot \mu$.

For each center $c_i^* \in C^*$, let $P_i^* \subseteq P$ be the corresponding cluster of points assigned to c_i^*. We consider the ball $B_{i,j}$ centered at c_i^* and having radius $2^j\mu$ for $0 \leq j \leq N$, where $N = \lceil \log(\nu n) \rceil$. We note that any point at a distance $2^N\mu \geq \nu n \cdot \mu$ from c_i^* is in $B_{i,N}$, and thus all the points in P_i^* are also in $B_{i,N}$. Let $B'_{i,0} = B_{i,0}$ and $B'_{i,j} = B_{i,j} \setminus B_{i,j-1}$ for $1 \leq j \leq N$. We refer to each such $B'_{i,j}$ as a ring for $1 \leq i \leq k, 0 \leq j \leq N$. For each $0 \leq j \leq N$ and color $1 \leq t \leq \ell$, let $P'_{i,j,t}$ be the set of points in $B'_{i,j}$ of color t. Let $s = \Theta(k \log n/\varepsilon^3)$ for a sufficiently large constant hidden in $\Theta(.)$.

For each center $c_i^* \in C^*$, we perform the following steps.

Random Sampling. For each color $1 \le t \le \ell$ and ring index $0 \le j \le N$, do the following. If $|P'_{i,j,t}| \le s$, add all the points of $P'_{i,j,t}$ to $W_{i,j}$ and set the weight of each such point to 1. Otherwise, select s points from $P'_{i,j,t}$ independently and randomly (without replacement) and add them to $W_{i,j}$. Set the weight of each such point to $|P'_{i,j,t}|/s$.

The set $W = \cup_{i,j} W_{i,j}$ is the desired universal coreset.

The Analysis

One way to prove that W is a universal coreset is to show that w.h.p. for any fixed set of centers C of size k and any coloring constraint $\mathbf{M}$,

$$(1-\varepsilon)\cdot \text{cost}(P,\mathbf{M},C) \le \text{wcost}(W,\mathbf{M},C) \le (1+\varepsilon)\cdot \text{cost}(P,\mathbf{M},C).$$

Then, by taking union bound over all C and $\mathbf{M}$, we obtain the desired bound. However, as we potentially have $n^{\Omega(k)}$ choices for C and $n^{\Omega(\ell k)}$ choices for $\mathbf{M}$, we need this bound for fixed C and $\mathbf{M}$ w.p. $1 - 1/n^{\Omega(\ell k)}$. It is not clear how to prove such a bound, as we pick only $\mathcal{O}(k \log n/\varepsilon^3)$ size sample from each ring corresponding to each color. Instead, we prove that for any fixed C, and for all $\mathbf{M}$, w.p. $1 - 1/n^{\Omega(k)}$, the above bound holds. In particular, we will show that for each ring $B'_{i,j}$ with points $P_{i,j}$ the error is bounded by $\varepsilon|P_{i,j}| \cdot 2^j\mu$.

Lemma 7.4. *For any fixed set C of k centers and for all $k \times \ell$ matrices $\mathbf{M}$, w.p. at least $1 - 1/n^{k+2}$, $|\text{cost}(P,\mathbf{M},C) - \text{wcost}(W,\mathbf{M},C)| \le \sum_{(i,j)} \varepsilon|P_{i,j}| \cdot 2^j\mu$.*

Now, consider all the rings $B'_{i,j}$ with $j = 0$. Then,

$$\sum_{(i,j):j=0} \varepsilon|P_{i,j}| \cdot 2^j\mu \le \varepsilon n \cdot \mu \le \varepsilon \cdot \text{OPT}_v \le \varepsilon \cdot \text{cost}(P,\mathbf{M},C).$$

Here, OPT_v is the optimal cost of vanilla k-median clustering. The last inequality follows, as the optimal cost of vanilla clustering is at most the cost of any constrained clustering. Now, for any ring $B'_{i,j}$ with $j \ge 1$ and any point p in the ring, $\text{dist}(p, c^*_i) \ge 2^{j-1}\mu$. Thus,

$$\sum_{(i,j):j\ge 1} \varepsilon|P_{i,j}| \cdot 2^j\mu \le \varepsilon \sum_{p\in P} 2 \cdot \text{dist}(p, C^*) \le 2\varepsilon \cdot \text{OPT}_v \le 2\varepsilon \cdot \text{cost}(P,\mathbf{M},C).$$

By taking union bound over all C and scaling ε down by a factor of 3, we obtain the desired result.

Lemma 7.5. *For every set C of k centers and every $k \times \ell$ matrix $\mathbf{M}$, w.p. at least $1 - 1/n$, $|\text{cost}(P,\mathbf{M},C) - \text{wcost}(W,\mathbf{M},C)| \le \varepsilon \cdot \text{cost}(P,\mathbf{M},C)$.*

This completes the proof of Theorem 7.3. Now, we are left with the proof of Lemma 7.4.

Proof of Lemma 7.4 Let P_τ be the points in P of color τ. Also, let W_τ be the chosen samples of color τ. For $1 \leq t \leq \ell - 1$, let $W^t = (\sum_{\tau=1}^{t} W_\tau) \cup (\cup_{\tau=t+1}^{\ell} P_\tau)$. Also, let $W^\ell = \sum_{\tau=1}^{\ell} W_\tau$ be the coreset points of all colors. Recall that for any ring $B'_{i,j}$, $P'_{i,j,\tau}$ is the points of color τ in the ring. Also, $P_{i,j} = \cup_{\tau=1}^{\ell} P'_{i,j,\tau}$.

Note that in the above, W^t contains the sampled points for color 1 to t and original points of color $t+1$ to ℓ. We will prove the following lemma that gives a bound when the coreset contains sampled points of a fixed color t and original points of the other colors.

Lemma 7.6. *Consider any color $1 \leq t \leq \ell$. For any fixed set C of k centers and for all $k \times \ell$ matrices $\mathbf{M}$, w.p. at least $1 - 1/n^{k+4}$, $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W_t \cup (P \setminus P_t), \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$.*

Note that for a particular color class if we select all original points in the coreset, then there is no error corresponding to those coreset points. This is true, as one can use the corresponding optimal assignment for these points. Assuming that the above lemma holds, now, we prove Lemma 7.4. Consider the coreset W^1. From the above lemma we readily obtain the following.

Corollary 7.7. *For any fixed set C of k centers and for all $k \times \ell$ matrices $\mathbf{M}$, w.p. at least $1 - 1/n^{k+4}$, $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W^1, \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,1}| \cdot 2^j \mu$.*

Now, in W^1 consider replacing the points of P_2 by the samples in W_2. We obtain the coreset W^2. Note that the samples in W_1 and W_2 are chosen independent of each other. Thus, by taking union bound over color 1 and 2, from Lemma 7.6 we obtain, for all $k \times \ell$ matrices $\mathbf{M}$, w.p. at least $1 - 2/n^{k+4}$, $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W^2, \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon (|P'_{i,j,1}| + |P'_{i,j,2}|) \cdot 2^j \mu$. Similarly, by taking union bound over all $\ell \leq n$ colors and noting that $W^\ell = W$, Lemma 7.4 follows.

Next, we prove Lemma 7.6, which is the key part of the analysis.

Proving Core Lemma

Recall that P_t is the set of points of color t, and W_t is the coreset points of color t. C is the given set of centers. For any matrix $\mathbf{M}$, let $\mathbf{M}^t$ be the t^{th} column of $\mathbf{M}$. We have the following observation that implies that it is sufficient to consider the points only in P_t to give the error bound.

Observation 7.8. *Suppose w.p. at least $1 - 1/n^{k+4}$, for all column matrix $\mathbf{M}'$, $|\text{cost}(P_t, \mathbf{M}', C) - \text{wcost}(W_t, \mathbf{M}', C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$. Then, with the same probability, for all $k \times \ell$ matrix $\mathbf{M}$, $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W_t \cup (P \setminus P_t), \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$.*

Proof. Consider any $k \times \ell$ matrix $\mathbf{M}$. Then,

$$\text{cost}(P, \mathbf{M}, C) = \sum_{\tau=1}^{\ell} \text{cost}(P_\tau, \mathbf{M}^\tau, C)$$

Also,

$$\text{wcost}(W_t \cup (P \setminus P_t), \mathbf{M}, C) = \text{wcost}(W_t, \mathbf{M}^t, C) + \sum_{\tau \in [\ell] \setminus \{t\}} \text{cost}(P_\tau, \mathbf{M}^\tau, C)$$

It follows that,

$$|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W_t \cup (P \setminus P_t), \mathbf{M}, C)| = |\text{cost}(P_t, \mathbf{M}^t, C) - \text{wcost}(W_t, \mathbf{M}^t, C)|$$

Now, by our assumption, it follows that the probability of the event: for all $\mathbf{M}$, $|\text{cost}(P_t, \mathbf{M}^t, C) - \text{wcost}(W_t, \mathbf{M}^t, C)|$ exceeds $\sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$ is at most $1/n^{k+4}$. Hence, the observation follows. □

By the above observation, it is sufficient to prove that w.p. at least $1 - 1/n^{k+4}$, for all column matrix $\mathbf{M}$, $|\text{cost}(P_t, \mathbf{M}, C) - \text{wcost}(W_t, \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$. The proof of this claim is similar to the analysis in [66]. In the rest of this section we prove this claim. For simplicity, we first do the analysis for a single ring. Later we will show how this idea in single ring case can be extended to obtain the h.p. bound for the multiple ring case.

Single Ring Case

We fix a ring $B'_{i,j}$ and rename the color t to γ. Note that we have points of only one color γ. For simplicity of notation, we rename P_γ to P. We do the analysis assuming that we sample points only from the ring $B'_{i,j}$. For simplicity, we denote this ring by B'. Let $P' = P'_{i,j,\gamma}$, $m = |P'|$, $\mu' = 2^j \mu$ and $c' = c^*_i$ for $1 \leq i \leq k$ and $0 \leq j \leq N$. Also, let S be the random sample chosen from P'. Thus in this case, our coreset W' consists of the points S, which have weight m/s and all the points in $P \setminus P'$, which have weight 1, i.e, $W' = S \cup (P \setminus P')$. We will show that the cost difference between P and W' is at most $\varepsilon m \mu'$ w.h.p. Intuitively, for each point in P', we allow at most $\varepsilon \mu'$ error on average.

For the rest of the proof we fix a column matrix $\mathbf{M}$ such that $\text{cost}(P, \mathbf{M}, C) < \infty$. We will prove the following theorem.

Theorem 7.9. *W.p. at least* $1 - 1/n^{2k+10}$*, it holds that* $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W', \mathbf{M}, C)| \leq \varepsilon m \mu'$.

By taking union bound over all (at most n^k) column matrices, we obtain the desired bound w.h.p. Towards this end, assume that $s < m$, otherwise $W' = P$ and the above theorem is trivially true.

An Alternative Way of Sampling. Consider the points of P' and the following alternative way of sampling points from P'. For each $p \in P'$, select p w.p. s/m independently and set its weight to m/s. Otherwise, set its weight to 0. Let $X \in \mathbb{R}^m_{\geq 0}$ be the corresponding random vector such that $X[p] = m/s$ if p is selected, otherwise $X[p] = 0$.

We note two things here. First, for each p, $\mathrm{E}[X[p]] = 1$. Thus, $\mathrm{E}[X] = \mathbb{1}$, where $\mathbb{1}$ is the vector of length m whose entries are all 1. Intuitively, this shows that in expectation the chosen set of samples behave like the original points. We will heavily use this connection in our analysis. Second, this sampling is different from our sampling scheme in the sense that here we might end up selecting more (or less) than s samples. However, one can show that with sufficient probability, this sampling scheme selects exactly s points, as the expected number is $m \cdot (s/m) = s$.

Claim 7.10. *[66] Let n be a positive integer, and $p \in (0, 1)$ such that np is an integer. The probability that Bernoulli$(n, p) = np$ is at least $\sqrt{p}$.*

Using the above claim with $n = m$ and $p = s/m$, it follows that X contains exactly s non-zero entries w.p. $\Omega(1/\sqrt{n})$. Conditioned on this event, X accurately represents the outcome of our sampling process. Thus, both the sampling processes are same up to repetition. Henceforth, we assume that X contains exactly s non-zero entries.

Representing Assignment By Network Flow. Given a vector $Y \in \mathbb{R}^m_{\geq 0}$ indexed by the points of P' we construct the following flow network G_Y. G_Y has two designated vertices s and t, which are called the source and the sink, respectively. For each point $p_j \in P$, there is a vertex u_j. For each center $c_i \in C$, there is a vertex v_i. There is also an auxiliary vertex w in G_Y corresponding to the center c' of the bicriteria solution. For each u_j, there is an edge between s and u_j, and also between w and u_j. s is also connected to w via an edge. w is connected to each v_i via an edge. Also, each v_i is connected to t via an edge. For each point p_j and center c_i, there is an edge between u_j and v_i. Formally, the vertex set V_Y of G_Y is defined as, $V_Y = \{s\} \cup \{t\} \cup \{w\} \cup \{u_j \mid 1 \leq j \leq n\} \cup \{v_i \mid 1 \leq i \leq k\}$. The set of edges $E_Y = \{(s, u_j) \mid 1 \leq j \leq n\} \cup \{(u_j, w) \mid 1 \leq j \leq n\} \cup \{(v_i, t) \mid 1 \leq i \leq k\} \cup \{(w, v_i) \mid 1 \leq i \leq k)\} \cup \{(u_j, v_i) \mid 1 \leq j \leq n, 1 \leq i \leq k\}$. For each $p_j \in P \setminus P'$, (s, u_j) has a demand of 1. For each $p_j \in P'$, (s, u_j) has a demand of $Y[p_j]$. The demand of (s, w) is exactly $m - \sum_{p \in P'} Y[p]$, which can be negative. The capacity of each edge (v_i, t) is exactly $\mathbf{M}[i]$, the i^{th} entry of $\mathbf{M}$. Lastly, the cost of all the edges is 0 except the edges of $\{(u_j, v_i)\}$, $\{(u_j, w)\}$ and $\{(w, v_i)\}$. The cost of (u_j, v_i) is $\mathrm{dist}(p_j, c_i)$ and the cost of (u_j, w) is $\mathrm{dist}(p_j, c')$. The cost of (w, v_i) is $\mathrm{dist}(c', c_i)$.

We note that the assignment of points in P to the centers in C corresponding to an optimal clustering (with $\mathrm{cost}(P, \mathbf{M}, C) < \infty$) induces a flow for G_Y with $Y = \mathbb{1}$ that satisfies all the demands, which sum to $|P|$. Hence, for any $Y \in \mathbb{R}^m_{\geq 0}$, G_Y always

has a feasible flow, as the sum of demands is exactly $|P \setminus P'| + \sum_{p \in P'} Y[p] + (m - \sum_{p \in P'} Y[p]) = |P|$.

For any $Y \in \mathbb{R}^m_{\geq 0}$, we denote by $f(Y)$ the cost of the minimum cost feasible flow in G_Y. Consider the random vector X defined before. We have the following important observation.

Observation 7.11. *$f(X)$ and wcost$(W', \mathbf{M}, C)$ are identically distributed. Moreover, $f(\mathrm{E}[X]) =$ cost$(P, \mathbf{M}, C)$.*

Proof. Note that the total demand in G_X is $|P|$, as argued before. This demand must be routed to t through the edges $\{(v_i, t)\}$. Now, the capacity of (v_i, t) is $\mathbf{M}[i]$. If $\mathbf{M}$ is a valid partition matrix, then $\sum_{i=1}^{k} \mathbf{M}[i]$ must be $|P|$. Thus, any feasible flow in G_X, which satisfies all the demands, must saturate all the edges $\{(v_i, t)\}$. It follows that from this flow we can retrieve an assignment of the points in W' to the centers in C, such that exactly $\mathbf{M}[i]$ weight is assigned to each center $c_i \in C$. Finally, as X contains exactly s non-zero entries, the cost of the minimum cost feasible flow in G_X and wcost$(W', \mathbf{M}, C)$ must be identically distributed.

The moreover part follows by noting that $\mathrm{E}[X] = \mathbb{1}$. □

From the above observation it follows that to prove Theorem 7.9, it is sufficient to prove that w.p. $1 - 1/n^{\Omega(k)}$, $|f(X) - f(\mathrm{E}[X])| \leq \varepsilon m \mu'$. Now, we have another observation which will be useful later.

Observation 7.12. *The function f is μ'-Lipschitz w.r.t. the ℓ_1 distance in $\mathbb{R}^m_{\geq 0}$.*

Proof. Consider two vectors $Y, Y' \in \mathbb{R}^m_{\geq 0}$ such that $Y' = Y + \delta \mathbb{1}_p$, where $\mathbb{1}_p$ is the m-dimensional vector which has a single non-zero entry 1 corresponding to $p \in P'$. Suppose we are given a minimum cost flow in G_Y. We can route δ additional flow from the vertex of p to w, which incurs $\delta\mu'$ cost. The modified flow is a feasible flow in $G_{Y'}$. Thus, $f(Y') \leq f(Y) + \delta\mu'$.

Similarly, suppose we are given a minimum cost flow in $G_{Y'}$. We can route δ additional flow from w to the vertex of p, which incurs $\delta\mu'$ cost. The modified flow is a feasible flow in G_Y. Thus, $f(Y) \leq f(Y') + \delta\mu'$. Together these show that f is μ'-Lipschitz. □

Towards this end, we state the following concentration bound, which will be useful in the analysis.

Lemma 7.13. *W.p. at least $1 - 1/n^{2k+20}$, $|f(X) - \mathrm{E}[f(X)]| \leq \varepsilon m \mu'/2$.*

The proof of this lemma is very similar to the proof of Lemma 15 in [66], which essentially follows from the fact that f is μ'-Lipschitz and from the following Chernoff type bound.

Theorem 7.14. *[66] Let $x_1, \ldots, x_n$ be independent random variables taking value b w.p. p and value 0 w.p. $1-p$, and let $g : [0,1]^n \to \mathbb{R}$ be an L-Lipschitz function in ℓ_1 norm. Define $X := (x_1, \ldots, x_n)$ and $\mu := \mathrm{E}[g(X)]$. Then, for $0 \leq \varepsilon \leq 1$:*

$$Pr[|g(X) - \mathrm{E}[g(X)]| \geq \varepsilon pnbL] \leq 2\exp(-\varepsilon^2 pn/3).$$

We apply the above theorem with $p = s/m$, $n = m$, $b = m/s$, $g = f$ and $L = \mu'$. Then,

$$\begin{aligned}
&\Pr[|f(X) - \mathrm{E}[f(X)]| \geq \varepsilon m\mu'/2] \\
&= \Pr[|f(X) - \mathrm{E}[f(X)]| \geq (\varepsilon/2)(s/m) \cdot m \cdot (m/s) \cdot \mu'] \\
&= \Pr[|f(X) - \mathrm{E}[f(X)]| \geq (\varepsilon/2) \cdot pnbL] \\
&\leq 2\exp(-(\varepsilon/2)^2 pn/3) \\
&= 2\exp(-(\varepsilon/2)^2 s/3) \\
&= 2\exp(-(\varepsilon^2/12)\Theta(k \log n/\varepsilon^3)) \\
&\leq 1/n^{2k+20}
\end{aligned}$$

The last inequality follows due to the sufficiently large constant hidden in the Θ notation. Now, we proceed towards the proof of Theorem 7.9. We will show the desired bound in two steps. Here we take a slightly different way than [66] for our convenience. First, we show that w.p. at least $1 - 1/n^{2k+20}$, $f(\mathrm{E}[X]) \leq f(X) + \varepsilon m\mu'$. Then, we show that w.p. at least $1 - 1/n^{2k+20}$, $f(X) \leq f(\mathrm{E}[X]) + \varepsilon m\mu'$.

The First Step. From Lemma 7.13 it follows that it is sufficient to prove $f(\mathrm{E}[X]) \leq \mathrm{E}[f(X)]$. Now, $\mathrm{E}[X] = \mathbb{1}$. Let Y be any outcome of X and $X = Y$ w.p. $p(Y)$. Let y_i be the value in Y corresponding to $p_i \in P'$. Then, there is a feasible flow in G_Y, where for each p_i, at least y_i demand is satisfied. Now, consider the flow ϕ obtained by summing, for each Y, the minimum cost feasible flow in G_Y scaled by $p(Y)$. Note that the cost of ϕ is $\sum_Y p(Y)f(Y) = \mathrm{E}[f(X)]$. Also, this flow does not violate any capacity, as the sum of the probabilities is 1. Now, in each flow corresponding to Y scaled by $p(Y)$, for each p_i, $p(Y) \cdot y_i$ demand is satisfied. Hence, in ϕ, for each p_i, at least $\sum_Y p(Y) \cdot y_i = 1$ demand is satisfied, as the expected value of y_i is 1. It follows that, $f(\mathbb{1}) = f(\mathrm{E}[X])$ is at most the cost of ϕ and we obtain the desired bound.

The Second Step. Here we will show that w.p. at least $1 - 1/n^{2k+20}$, $f(X) \leq f(\mathrm{E}[X]) + \varepsilon m\mu'$. First, we prove that it is sufficient to show that w.p. at least $1 - 1/n^3$, $f(X) \leq f(\mathrm{E}[X]) + \varepsilon m\mu'/3$.

Lemma 7.15. *If $f(X) \leq f(\mathrm{E}[X]) + \varepsilon m\mu'/3$ holds w.p. at least $1 - 1/n^3$, then w.p. $1 - 1/n^{2k+20}$, $f(X) \leq f(\mathrm{E}[X]) + \varepsilon m\mu'$.*

Proof. Here we will prove that $\mathrm{E}[f(X)] \leq f(\mathrm{E}[X]) + \varepsilon m\mu'/2$. Then, by Lemma 7.13, it follows that w.p. $1 - 1/n^{2k+20}$, $f(X) \leq \mathrm{E}[f(X)]| + \varepsilon m\mu'/2 \leq f(\mathrm{E}[X]) + \varepsilon m\mu'$.

First, note that $X \in [0, m/s]^m$. As the function f is μ'-Lipschitz by Observation 7.12, the values of $f(X)$ must lie in an interval of length at most $m/s \cdot m\mu' \leq m^2\mu'$. Similarly, $\mathrm{E}[X] = \mathbb{1} \in [0, m/s]^m$, and thus $f(\mathrm{E}[X])$ is also contained in that interval. Hence, $f(X) \leq f(\mathrm{E}[X]) + m^2\mu'$. Now,

$$\begin{aligned}\mathrm{E}[f(X)] &\leq (1 - 1/n^3) \cdot (f(\mathrm{E}[X]) + \varepsilon m\mu'/3) + (1/n^3) \cdot (f(\mathrm{E}[X]) + m^2\mu') \\ &\leq f(\mathrm{E}[X]) + \varepsilon m\mu'/3 + (1/n^2) \cdot m\mu' \\ &\leq f(\mathrm{E}[X]) + \varepsilon m\mu'/2\end{aligned}$$

□

The following lemma completes the proof of Theorem 7.9. Its proof is very similar to the proof of Lemma 20 in [66]. For completeness, we present the proof here.

Lemma 7.16. *W.p. at least* $1 - 1/n^3$, $f(X) \leq f(\mathbb{1}) + \varepsilon m\mu'/3$.

Proof. We consider a minimum cost feasible flow ϕ in G_Y for $Y = \mathbb{1}$. We can assume that this flow is integral, as all the demands and capacities are integral. We compute a feasible flow ϕ' in G_X modifying the flow ϕ whose cost is at most $f(\mathbb{1}) + \varepsilon m\mu'/3$ w.p. at least $1 - 1/n^3$.

The construction of the modified flow is as follows. For each point $p \in P \setminus P'$, we route the demand of p in ϕ' in the same way as in ϕ. Now consider the points in P'. Let P'_i be the subset of points of P' that are assigned to the center $c_i \in C$. Also, let Q'_i be the subset of points of P'_i that are sampled, and hence are contained in W'.

For each center $c_i \in C$, we have two cases. The first case is $|Q'_i| \leq |P'_i| \cdot s/m$. In this case, we route m/s amount of flow from each vertex u_j corresponding to the point p_j of Q'_i to the vertex v_i corresponding to c_i. We also route $|P'_i| - |Q'_i| \cdot m/s$ amount of flow from w to v_i. Note that the total amount of flow routed to v_i in these above two steps is exactly $|P'_i|$ and does not depend on $|Q'_i|$ as long as $|Q'_i| \leq |P'_i| \cdot s/m$. In the second case, $|Q'_i| > |P'_i| \cdot s/m$. In this case, first we select a random sample Q''_i from Q'_i of size $\lfloor |P'_i| \cdot s/m \rfloor$ and apply the same steps in the first case with Q''_i instead of Q'_i. Finally, route m/s amount of flow from the vertex corresponding to each point in $Q'_i \setminus Q''_i$ to w.

We note that the computed flow ϕ' in the above satisfies all the demands. Also, none of the capacities are violated, as the flow in and out for each vertex v_i remain the same as in ϕ. In the following we give a bound on the cost of ϕ'. To unify the analysis, in the first case, we set $Q''_i = Q'_i$. We consider two cases depending on the value of $\mathrm{E}[|Q'_i|]$ for every $c_i \in C$.

Case 1. $\mathrm{E}[|Q'_i|] \geq \varepsilon s/(100k)$**.** As Q'_i is distributed as Bernoulli$(|P'_i|, s/m)$, $\mathrm{E}[|Q'_i|] = |P'_i| \cdot s/m$. Thus, $|P'_i| \cdot s/m \geq \varepsilon s/(100k)$, or $|P'_i| \geq \varepsilon m/(100k)$. We have the following observation.

Observation 7.17. *W.p. at least* $1 - 1/n^{10}$, $||Q'_i| - |P'_i| \cdot s/m| \leq \varepsilon |P'_i| \cdot s/(50m)$.

Proof. Using the Chernoff bound, $\Pr[||Q'_i| - |P'_i| \cdot s/m| > \varepsilon |P'_i| \cdot s/(50m)] \leq exp(-\Theta(\varepsilon^2 \cdot |P'_i| \cdot s/m)) \leq exp(-\Theta(\varepsilon^2 \cdot \varepsilon(m/k) \cdot (s/m))) \leq exp(-\Theta(\log n)) \leq 1/n^{10}$, for sufficiently large constant hidden in $\Theta(.)$ in the definition of s. □

From the above observation and considering the fact that $|Q''_i| \geq |P'_i| \cdot s/m - 1$, we have the following bound.

Observation 7.18. *W.p. at least* $1 - 1/n^{10}$, $|Q'_i| - |Q''_i| \leq \varepsilon |P'_i| \cdot s/(40m)$.

From the above two observations, we have the following observation.

Observation 7.19. *W.p. at least* $1 - 1/n^{9}$, $|P'_i| \cdot s/m - |Q''_i| \leq \varepsilon |P'_i| \cdot s/(20m)$.

Now, we give bound on the cost of the computed flow. Note that we route m/s flow for each point in Q''_i to c_i whose total cost is $\sum_{p \in Q''_i} (m/s) \cdot d(p, c_i)$. To give bound on this cost we need the following lemma from [53].

Lemma 7.20. *(Lemma 3.2. of [53]) Let* $T \geq 0$ *and* η *be fixed constants, and let* $h(.)$ *be a function defined on a set* V *such that* $\eta \leq h(p) \leq \eta + T$ *for all* $p \in V$*. Let* $U = \{p_1, \ldots, p_r\}$ *be a set of* r *samples drawn independently and uniformly from* V*, and let* $\delta > 0$ *be a parameter. If* $r \geq (T^2/2\delta^2) \ln(2/\lambda)$*, then* $\Pr[|\frac{h(V)}{|V|} - \frac{h(U)}{|U|}| \geq \delta] \leq \lambda$*, where* $h(U) = \sum_{u \in U} h(u)$ *and* $h(V) = \sum_{v \in V} h(v)$.

Fix any integer $r \in [1 - \varepsilon/20, 1] \cdot |P'_i| \cdot s/m$ and consider the event that $|Q''_i| = r$. Conditioned on this event Q''_i is a set of r samples drawn independently and uniformly from P'_i. We apply Lemma 7.20 setting $T = 2\mu'$, $V = P'_i$, $U = Q''_i$, $h(p) = d(p, c_i)$, $\delta = \varepsilon\mu'/20$ and $\lambda = 1/n^{10}$. Note that,

$$r \geq (1 - \varepsilon/20) \cdot |P'_i| \cdot s/m \geq (1 - \varepsilon/20) \cdot \frac{\varepsilon m}{100k} \cdot \frac{s}{m} \geq \Theta(\log n/\varepsilon^2) \geq (T^2/2\delta^2) \ln(2/\lambda)$$

The last inequality follows assuming a sufficiently large constant is hidden in $\Theta(.)$ in the definition of s.

We obtain, w.p. at least $1 - 1/n^{10}$,

$$\left|\frac{h(P_i')}{|P_i'|} - \frac{h(Q_i'')}{|Q_i''|}\right| \leq \delta$$
$$\text{Or, } \left|\frac{h(P_i') \cdot |Q_i''|}{|P_i'|} - h(Q_i'')\right| \leq \delta |Q_i''|$$
$$\text{Or, } h(Q_i'') \leq \frac{h(P_i') \cdot |Q_i''|}{|P_i'|} + \delta \cdot (|P_i'| \cdot s/m)$$
$$\text{Or, } h(Q_i'') \cdot (m/s) \leq \frac{h(P_i') \cdot |Q_i''|}{|P_i'|} \cdot (m/s) + \varepsilon |P_i'| \cdot \mu'/20$$

Note that $h(Q_i'') \cdot (m/s)$ is exactly the cost of flow routing for the points in Q_i'' to c_i. Taking union bound over all possible values of $r = |Q_i''|$, we obtain the above bound w.p. at least $1 - 1/n^9$.

Now, we give a bound on the cost of flow routing from w to c_i. The cost is $(|P_i'| - |Q_i''| \cdot m/s) \cdot d(c', c_i)$. Now, for any $p \in P_i'$, $d(c', c_i) \leq d(p, c_i) + \mu'$. Averaging gives, $d(c', c_i) \leq h(P_i')/|P_i'| + \mu'$. Thus, the cost is at most,

$$\begin{aligned}
&(|P_i'| - |Q_i''| \cdot m/s) \cdot (h(P_i')/|P_i'| + \mu') \\
&\leq h(P_i') - \frac{h(P_i') \cdot |Q_i''|}{|P_i'|} \cdot (m/s) + (|P_i'| - |Q_i''| \cdot m/s) \cdot \mu' \\
&\leq h(P_i') - \frac{h(P_i') \cdot |Q_i''|}{|P_i'|} \cdot (m/s) + (\varepsilon |P_i'|/20)\mu' \\
&= h(P_i') - \frac{h(P_i') \cdot |Q_i''|}{|P_i'|} \cdot (m/s) + \varepsilon |P_i'| \mu'/20
\end{aligned}$$

The second inequality follows from Observation 7.19. Next, we bound the third and the last type of cost, which corresponds to flow routing from points in $Q_i' \setminus Q_i''$ to w. This cost is at most,

$$\begin{aligned}
&\sum_{p \in (Q_i' \setminus Q_i'')} (m/s) \cdot d(p, c') \\
&\leq (|Q_i'| - |Q_i''|) \cdot (m/s) \cdot \mu' \\
&\leq (\varepsilon |P_i'| \cdot s/(40m)) \cdot (m/s) \cdot \mu' \\
&\leq \varepsilon |P_i'| \mu'/40
\end{aligned}$$

The second inequality follows from Observation 7.18. Thus, in this case, the total cost is bounded by,

$$h(P_i') + \varepsilon|P_i'|\mu'/20 + \varepsilon|P_i'|\mu'/20 + \varepsilon|P_i'|\mu'/40$$
$$\leq h(P_i') + \varepsilon|P_i'|\mu'/8.$$

Case 2. $\mathrm{E}[|Q_i'|] < \varepsilon s/(100k)$**.** Note that in this case, $|P_i'| < \varepsilon m/(100k)$. First, we have the following observation.

Observation 7.21. *W.p. at least* $1 - 1/n^{10}$, $|Q_i'| \leq \varepsilon s/(50k)$.

Proof. We use the Chernoff bound:

$$\Pr[|Q_i'| \geq \varepsilon s/(50k)] \leq exp(-\Theta(\varepsilon s/k)) \leq exp(-\Theta(\log n)) \leq 1/n^{10}.$$

□

The cost of flow routing from points in Q_i'' to c_i is,

$$\sum_{p \in Q_i''} (m/s) \cdot d(p, c_i) \leq \sum_{p \in Q_i''} (d(p, c') + d(c', c_i)) \cdot (m/s)$$
$$\leq |Q_i''| \cdot \mu' \cdot (m/s) + |Q_i''| \cdot (m/s) \cdot d(c', c_i)$$

The cost of flow routing from w to c_i is,

$$(|P_i'| - |Q_i''| \cdot m/s) \cdot d(c', c_i) \leq \sum_{p \in P_i'} d(c', c_i) - |Q_i''| \cdot (m/s) \cdot d(c', c_i)$$
$$\leq \sum_{p \in P_i'} (d(c', p) + d(p, c_i)) - |Q_i''| \cdot (m/s) \cdot d(c', c_i)$$
$$\leq |P_i'| \cdot \mu' + \sum_{p \in P_i'} d(p, c_i) - |Q_i''| \cdot (m/s) \cdot d(c', c_i)$$

The cost of flow routing from points in $Q_i' \setminus Q_i''$ to w is,

$$\sum_{p \in (Q_i' \setminus Q_i'')} (m/s) \cdot d(p, c')$$
$$\leq |Q_i'| \cdot (m/s) \cdot \mu'$$

The total cost in this case is at most,

$$|Q_i''| \cdot \mu' \cdot (m/s) + |Q_i''| \cdot (m/s) \cdot d(c', c_i) + |P_i'| \cdot \mu' + \sum_{p \in P_i'} d(p, c_i) - |Q_i''| \cdot (m/s) \cdot d(c', c_i)$$
$$+ |Q_i'| \cdot (m/s) \cdot \mu'$$
$$\leq (\varepsilon s/(50k)) \cdot \mu' \cdot (m/s) + (\varepsilon m/(100k)) \cdot \mu' + \sum_{p \in P_i'} d(p, c_i) + (\varepsilon s/(50k)) \cdot \mu' \cdot (m/s)$$
$$\leq (\varepsilon m/(20k)) \cdot \mu' + \sum_{p \in P_i'} d(p, c_i)$$

The first inequality follows from Observation 7.21 and by noting that $|Q_i''| \leq |Q_i'|$.

General Upper Bound on the Cost. By merging the cost in both cases, we obtain the common upper bound, $\sum_{p \in P_i'} d(p, c_i) + (\varepsilon |P_i'| \cdot \mu'/8) + (\varepsilon m/(20k)) \cdot \mu'$. Summing over all the centers in C, we obtain,

$$f(X) \leq f(\mathbb{1}) + \varepsilon |P'| \cdot \mu'/8 + (\varepsilon m/20) \cdot \mu' \leq f(\mathbb{1}) + \varepsilon |P'| \cdot \mu'/3.$$

It is not hard to see that this bound holds w.p. at least $1 - 1/n^3$. This completes the proof of Lemma 7.16. □

Multiple Ring Case

In the previous section, we have shown how to bound the error for a fixed ring. Here we extend the ideas to the multiple ring case. Intuitively, we use a union bound over all rings to obtain the desired high probability bound. However, we need to consider the samples from all the rings corresponding to the color γ together. Let W' be the corresponding coreset.

We consider any arbitrary ordering of all the rings, and for any two rings $B_{i,j}'$ and $B_{i',j'}'$, we say $(i, j) < (i', j')$ if $B_{i,j}'$ precedes $B_{i',j'}'$ in this ordering. Consider any ring $B_{i,j}'$. We define a function $f_{i,j}$ corresponding to this ring similar to the function f. Let $P_{i,j}' = P_{i,j,\gamma}'$. Also, let $W_{i,j}'$ be the samples chosen from $P_{i,j}'$. The input to the function $f_{i,j}$ is a vector $Y \in \mathbb{R}_{\geq 0}^{|P_{i,j}'|}$ that is indexed by the points of the ring. We construct a network G_Y as before. But, as we consider samples from all the rings, the demands of the points are defined in a different way than before. For each point in $p \in P_{i,j}'$, its demand is $Y[p]$. Set the demand of w to $|P_{i,j}'| - \sum_{p \in P_{i,j}'} Y[p]$. For each ring $B_{i',j'}' \neq B_{i,j}'$, and for each point $p \in W_{i',j'}'$, set its demand to $|P_{i',j'}'|/s$. Note that the total demand corresponding to $B_{i',j'}'$ is $s \cdot |P_{i',j'}'|/s = |P_{i',j'}'|$. Thus, in G_Y we fix the samples of all the rings except $B_{i,j}'$. $f_{i,j}(Y)$ is the cost of the minimum cost flow in G_Y.

Let $\mathrm{E}_{W'_{i,j}:(i',j')>(i,j)}[f_{i,j}(Y)|W'_{i_1,j_1} : (i_1,j_1) < (i,j)]$ ($\mathrm{E}_{>(i,j)}[f_{i,j}(Y)]$ in short) be the expectation of $f_{i,j}(Y)$ over all samples $W'_{i',j'}$ for $(i',j') > (i,j)$ given fixed samples W'_{i_1,j_1} for all $(i_1,j_1) < (i,j)$. Similarly, define $\mathrm{E}_{W'_{i,j}:(i',j')\geq(i,j)}[f_{i,j}(Y)|W'_{i_1,j_1} : (i_1,j_1) < (i,j)]$ or $\mathrm{E}_{\geq(i,j)}[f_{i,j}(Y)]$ in short. Recall that in the single ring case we showed that w.p. at least $1 - 1/n^{2k+10}$, $|f(X) - f(\mathrm{E}[X])| \leq \varepsilon m\mu'$. Similarly, here we obtain the following lemma.

Lemma 7.22. *W.p. at least $1 - 1/n^{2k+10}$, for any ring $B'_{i,j}$,*

$$|\mathrm{E}_{>(i,j)}[f_{i,j}(Y)] - \mathrm{E}_{\geq(i,j)}[f_{i,j}(Y)]| \leq \varepsilon|P'_{i,j}| \cdot 2^j\mu.$$

Note that we would like to show the bound in terms of multiple rings together instead of just one ring $B'_{i,j}$. In particular, we would like to give a bound w.r.t. $\mathrm{wcost}(W', \mathbf{M}, C)$, where $\mathbf{M}$ is a column matrix. Correspondingly, we define $\mathrm{E}_{>(i,j)}[\mathrm{wcost}(W', \mathbf{M}, C)]$ and $\mathrm{E}_{\geq(i,j)}[\mathrm{wcost}(W', \mathbf{M}, C)]$. From Lemma 7.22, we readily obtain the following lemma.

Lemma 7.23. *W.p. at least $1 - 1/n^{2k+8}$,*

$$|\mathrm{E}_{>(i,j)}[\mathrm{wcost}(W', \mathbf{M}, C)] - \mathrm{E}_{\geq(i,j)}[\mathrm{wcost}(W', \mathbf{M}, C)]| \leq \varepsilon|P'_{i,j}| \cdot 2^j\mu.$$

Now consider going over all the rings in the ordering and applying the above lemma. Let (i_1,j_1) and (i',j') be the indexes of the first and last ring, respectively. Then the total difference between the values $\mathrm{E}_{\geq(i_1,j_1)}[\mathrm{wcost}(W', \mathbf{M}, C)]$ and $\mathrm{E}_{>(i',j')}[\mathrm{wcost}(W', \mathbf{M}, C)]$ is at most $\sum_{(i,j)} \varepsilon|P'_{i,j}|\cdot 2^j\mu$. But, $\mathrm{E}_{\geq(i_1,j_1)}[\mathrm{wcost}(W', \mathbf{M}, C)] = \mathrm{cost}(P, \mathbf{M}, C)$ and $\mathrm{E}_{>(i',j')}[\mathrm{wcost}(W', \mathbf{M}, C)] = \mathrm{wcost}(W', \mathbf{M}, C)$, and hence by taking union bound over all $\mathcal{O}(k \log n)$ rings, we obtain the following lemma.

Lemma 7.24. *For any fixed set C of k centers and any fixed column matrix $\mathbf{M}$, w.p. at least $1 - 1/n^{2k+5}$, $|\mathrm{cost}(P, \mathbf{M}, C) - \mathrm{wcost}(W', \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon|P'_{i,j}| \cdot 2^j\mu$.*

By taking union bound over all column matrices $\mathbf{M}$, we obtain the desired bound.

Lemma 7.25. *For any fixed set C of k centers and for all column matrices $\mathbf{M}$, w.p. at least $1 - 1/n^{k+4}$, $|\mathrm{cost}(P, \mathbf{M}, C) - \mathrm{wcost}(W', \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon|P'_{i,j}| \cdot 2^j\mu$.*

7.2.2 Overlapping Group Case

In this section, we prove the following theorem.

Theorem 7.26. *Given a collection of ℓ possibly overlapping groups consisting of n points in total in a metric space, there is an $\mathcal{O}(n(k+\ell))$ time randomized algorithm that w.p. at least $1 - 1/n$, computes a universal coreset for k-median clustering of size $\mathcal{O}(\Gamma(k \log n)^2/\varepsilon^3)$.*

Let $P = \cup_{i=1}^{\ell} P_i$. For each point $p \in P$, let $J_p \subseteq [\ell]$ be the set of indexes of the groups to which p belongs. Let I be the distinct collection of these sets $\{J_p \mid p \in P\}$ and $|I| = \Gamma$. In particular, let $I_1, \ldots, I_\Gamma$ be the distinct sets in I. Now, we partition the points in P based on these sets. For $1 \leq i \leq \Gamma$, let $P^i = \{p \in P \mid I_i = J_p\}$. Thus, $\{P^i \mid 1 \leq i \leq \Gamma\}$ defines equivalence classes for P such that two points $p, p' \in P$ belong to the same equivalence class if they are in exactly the same set of groups.

In the overlapping case, we will work with an even stronger definition of coresets. This is for the ease of computation of an optimal cost assignment of the points in the coreset. Here instead of $k \times \ell$ matrices, coloring constraints are defined by $k \times \Gamma$ matrices. The rows still correspond to k centers, but the columns now correspond to the Γ equivalence classes. Thus, for such a matrix $\mathbf{M}$, $\mathbf{M}_{ij}$ denotes the number of points from P^j that are in cluster i. Thus, the entries of $\mathbf{M}$ define a partition of the points in P. We note that Proposition 4.10 continues to hold, as any fair assignment of the points in P defines such a matrix $\mathbf{M}$. Now, the definition of universal coreset remains same, except here $\text{wcost}(W, \mathbf{M}, C)$ is defined in the following natural way.

Suppose we are given a weight function $w : P \to \mathbb{R}_{\geq 0}$. Let $W \subseteq P \times \mathbb{R}$ be the set of pairs $\{(p, w(p)) \mid p \in P \text{ and } w(p) > 0\}$. For a set of centers $C = \{c_1, \ldots, c_k\}$ and a coloring constraint $\mathbf{M}$, $\text{wcost}(W, \mathbf{M}, C)$ is the minimum value $\sum_{p \in P, c_i \in C} \psi(p, c_i) \cdot \text{dist}(p, c_i)$ over all assignments $\psi : P \times C \to \mathbb{R}_{\geq 0}$ such that

(i) For each $p \in P$, $\sum_{c_i \in C} \psi(p, c_i) = w(p)$.

(ii) For each $c_i \in C$ and class $1 \leq j \leq \Gamma$, $\sum_{p \in P^j} \psi(p, c_i) = \mathbf{M}_{ij}$.

If there is no such assignment ψ, $\text{wcost}(W, \mathbf{M}, C) = \infty$. When $w(p) = 1$ for all $p \in P$, we simply denote W by P and $\text{wcost}(W, \mathbf{M}, C)$ by $\text{cost}(P, \mathbf{M}, C)$. Note that for a fixed matrix $\mathbf{M}$, an optimal assignment ψ must be integral due to integrality of flow. This was not-necessarily true with our previous definition in the overlapping case. We will compute a coreset that satisfies this even stronger definition.

With the above definitions, our algorithm in the overlapping case is a natural extension of the one in the disjoint case. The main idea of our algorithm is to divide the points into disjoint equivalence classes based on their group membership and sample points from each equivalence class. We compute the disjoint classes $\{P^i \mid 1 \leq i \leq \Gamma\}$ defined above. Then, apply our algorithm in the disjoint case on these disjoint sets of points $P^1, \ldots, P^\Gamma$. Let W be the constructed coreset.

The Analysis

Recall that $P_{i,j}$ is the total number of points in each ring $B'_{i,j}$. We will prove the following lemma.

Lemma 7.27. *For any fixed set C of k centers and for all $k \times \Gamma$ matrix $\mathbf{M}$, w.p. at least $1 - 1/n^{k+2}$, $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W, \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon |P_{i,j}| \cdot 2^j \mu$.*

Like before, by taking union bound over all C, we obtain the desired result. This completes the proof of Theorem 7.26. Next, we prove Lemma 7.27.

Proof of Lemma 7.27 Note that P^τ is the points in P from class τ for $1 \leq \tau \leq \Gamma$. Let W_τ be the chosen samples from class τ. For any ring $B'_{i,j}$, let $P'_{i,j,\tau}$ be the points from class τ in the ring. Also, let $P_{i,j} = \cup_{\tau=1}^{\Gamma} P'_{i,j,\tau}$.

Like in the disjoint case, here also we will prove the following lemma that gives a bound when the coreset contains sampled points from a fixed class t and original points from the other classes.

Lemma 7.28. *Consider any class $1 \leq t \leq \Gamma$. For any fixed set C of k centers and for all $k \times \Gamma$ matrix $\mathbf{M}$, w.p. at least $1 - 1/n^{k+4}$, $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W_t \cup (P \setminus P^t), \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$.*

By using expectation argument similar to the one in the disjoint-group-multiple-ring case and taking union bound over all $\Gamma < n$ classes, Lemma 7.27 follows. Next, we prove Lemma 7.28.

Proof of Lemma 7.28 We have the following lemma that implies that it is sufficient to consider the points only in P^t to give the error bound.

Lemma 7.29. *Suppose w.p. at least $1-1/n^{k+4}$, for all $k \times 1$ matrix $\mathbf{M}'$ such that the sum of the entries in each column is exactly $|P^t|$, $|\text{cost}(P^t, \mathbf{M}', C) - \text{wcost}(W_t, \mathbf{M}', C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$. Then, with the same probability, for all $k \times \Gamma$ matrix $\mathbf{M}$, $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W_t \cup (P \setminus P^t), \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$.*

Proof. Consider any $k \times \Gamma$ matrix $\mathbf{M}$. Also consider a clustering $C_1, \ldots, C_k$ of P that has cost $\text{cost}(P, \mathbf{M}, C)$. We construct two $k \times \Gamma$ matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ from $\mathbf{M}$. For $j \neq t$, and for $1 \leq i \leq k$, $\mathbf{M}_1[i][j] = 0$ and $\mathbf{M}_2[i][j] = \mathbf{M}[i][j]$. For $1 \leq i \leq k$, $\mathbf{M}_1[i][t] = |C_i \cap P^t|$ and $\mathbf{M}_2[i][t] = \mathbf{M}[i][t] - |C_i \cap P^t|$.

$$cost(P, \mathbf{M}, C) = cost(P^t, \mathbf{M}_1, C) + cost(P \setminus P^t, \mathbf{M}_2, C)$$

Also, as $W_t \subseteq P^t$ and the sum of the weights of the points in W_t is $|P^t|$,

$$wcost(W_t \cup (P \setminus P^t), \mathbf{M}, C) = wcost(W_t, \mathbf{M}_1, C) + cost(P \setminus P^t, \mathbf{M}_2, C)$$

It follows that,

$$|cost(P, \mathbf{M}, C) - wcost(W_t \cup (P \setminus P^t), \mathbf{M}, C)| = |cost(P^t, \mathbf{M}_1, C) - wcost(W_t, \mathbf{M}_1, C)|$$

Let $\mathbf{M}'_1$ be the t^{th} column of $\mathbf{M}_1$. Now, considering the fact that P^t does not contain any points from any other classes, $cost(P^t, \mathbf{M}_1, C) - wcost(W_t, \mathbf{M}_1, C) = cost(P^t, \mathbf{M}'_1, C) - wcost(W_t, \mathbf{M}'_1, C)$. Also, by the definition of $\mathbf{M}_1$, the sum of the entries in $\mathbf{M}'_1$ is $\sum_{i=1}^{k} |C_i \cap P^t| = |P^t|$.

Now, by our assumption, it follows that the probability of the event: for all $\mathbf{M}$, $|cost(P^t, \mathbf{M}'_1, C) - wcost(W_t, \mathbf{M}'_1, C)|$ exceeds $\sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$ is at most $1/n^{k+4}$. Hence, the lemma follows. □

By the above observation, it is sufficient to prove that w.p. at least $1 - 1/n^{k+4}$, for all $k \times 1$ matrix $\mathbf{M}$ such that the sum of the entries in each column is exactly $|P^t|$, $|cost(P^t, \mathbf{M}, C) - wcost(W_t, \mathbf{M}, C)| \leq \sum_{(i,j)} \varepsilon |P'_{i,j,t}| \cdot 2^j \mu$. Now, as we select samples from P^t separately and independently, this claim boils down to the corresponding claim in the disjoint case. Recall that we proved this claim for a single ring first, and then extended to multiple rings. The proof of our claim here is very similar, and thus we omit it.

7.2.3 Euclidean Space

In this section, we prove the following theorem.

Theorem 7.30. *Given a collection of ℓ possibly overlapping groups consisting of n points in total in $\mathbb{R}^d$, there is an $\mathcal{O}(nd(k+\ell))$ time randomized algorithm that w.p. at least $1 - 1/n$, computes a universal coreset for Euclidean k-median clustering of size $O\left(\frac{\Gamma}{\varepsilon^3} \cdot k^2 \log n(\log n + d\log(1/\varepsilon))\right)$.*

The algorithm in the Euclidean case is the same as for general metrics, except we set s to $\Theta(k\log(nb)/\varepsilon^3)$ instead of $\Theta(k\log n/\varepsilon^3)$, where $b = \Theta(k\log(n/\varepsilon)/\varepsilon^d)$. The analysis for general metrics holds in this case, but the assumption that the number of distinct sets of centers is at most n^k is no longer true. Here any point in $\mathbb{R}^d$ is a potential center. Nevertheless, we show that for every set $C \subseteq \mathbb{R}^d$ of k centers and constraint $\mathbf{M}$, the optimal cost is preserved approximately w.h.p. The idea is to use a discretization technique to obtain a finite set of centers so that if instead we draw centers from this set, the cost of any clustering is preserved approximately.

In the following, we analyze the coreset construction algorithm in the overlapping case. First, we construct a set of points F that we will use as the center set. Recall that C^* is the set of centers computed by the bicriteria approximation algorithm. ν is the constant approximation factor and Π is the cost of clustering. Also, $\mu = \Pi/(\nu n)$. Note that for any point p, $\text{dist}(p, C^*) \leq \Pi = \nu n \cdot \mu$.

For each center $c_i^* \in C^*$, we consider the d-dimensional axis-parallel hypercubes $R_{i,j}$ having sidelength $2^j\mu$ and centered at c_i^* for $0 \leq j \leq N$, where $N = \lceil \log(14\nu n/\varepsilon) \rceil$. We note that any point at a distance $(2^N\mu)/2 \geq 7\nu n \cdot \mu/\varepsilon$ from c_i^* is in $R_{i,N}$. Let $R'_{i,0} = R_{i,0}$ and $R'_{i,j} = R_{i,j} \setminus R_{i,j-1}$ for $1 \leq j \leq N$. For each $0 \leq j \leq N$, we divide $R'_{i,j}$ into gridcells of sidelength $(\varepsilon 2^j\mu)/(10\nu)$. Let Q_i be the exponential grid for $R'_{i,0}, \ldots, R'_{i,N}$, i.e., Q_i is the amalgamation of the gridcells in $R'_{i,0}, \ldots, R'_{i,N}$. For each gridcell in the exponential grid Q_i, we select any arbitrary point and add it to F_i.

We repeat the above process for all $c_i^* \in C^*$. Let $F = \cup_i F_i$. Note that the total number of gridcells of Q_i is at most $\mathcal{O}(\log(n/\varepsilon)/\varepsilon^d)$. Now, from each such gridcell, we pick at most 1 point. As C^* contains $\mathcal{O}(k)$ centers, the size of F is $\mathcal{O}(k\log(n/\varepsilon)/\varepsilon^d)$.

Note that if the centers can only be chosen from F, then by the analysis for general metrics, we obtain the following lemma.

Lemma 7.31. *For any fixed set $C \subseteq F$ of k centers and for all $k \times \Gamma$ matrices $\mathbf{M}$, w.p. at least $1 - 1/(bn)^{k+2}$, $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W, \mathbf{M}, C)| \leq \varepsilon \cdot \text{cost}(P, \mathbf{M}, C)$.*

This lemma is similar to Lemma 7.27. The error probability is now $1/(bn)^{k+2}$ as s is set to the larger value $\Theta(k \log(nb)/\varepsilon^3)$ instead of $\Theta(k \log n/\varepsilon^3)$. Now the number of distinct sets of k centers from F is at most $|F|^k \leq b^k$. Thus, by taking union bound over all such sets, we obtain the bound in the above lemma for every $C \subseteq F$ w.h.p.

Lemma 7.32. *For every set $C \subseteq F$ of k centers and for all $k \times \Gamma$ matrices $\mathbf{M}$, w.p. at least $1 - 1/n^2$, $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W, \mathbf{M}, C)| \leq \varepsilon \cdot \text{cost}(P, \mathbf{M}, C)$.*

Next, we show that if in a clustering a center c is chosen that is not in any of the exponential grids considered before, then W preserves the cost of such clustering.

Lemma 7.33. *Consider any set $C \subseteq \mathbb{R}^d$ of k centers containing a center $\hat{c}$ such that a point $\hat{p} \in P$ is assigned to $\hat{c}$ in a clustering that satisfies a constraint $\mathbf{M}$. Moreover, suppose $\hat{c}$ is not in $\cup_i Q_i$. Then, $|\text{cost}(P, \mathbf{M}, C) - \text{wcost}(W, \mathbf{M}, C)| \leq \varepsilon \cdot \text{cost}(P, \mathbf{M}, C)$.*

Proof. Consider any class P^t and a ring $B'_{i,j}$. Let $P'_{i,j,t}$ be the points in $B'_{i,j}$ from P^t and $W_{i,j,t}$ be the points of $P'_{i,j,t}$ that are in W. Then, there is an assignment $\phi : P'_{i,j,t} \to W_{i,j,t}$ such that exactly $|P'_{i,j,t}|/|W_{i,j,t}|$ points are assigned to each point $q \in W_{i,j,t}$. Note that $\text{dist}(p, \phi(p)) \leq \text{dist}(p, c^*_i) + \text{dist}(c^*_i, \phi(p)) \leq 2^j\mu + 2^j\mu = 2^{j+1}\mu$. Now, consider an optimal assignment ψ corresponding to *cost*$(P, \mathbf{M}, C)$. We compute the following assignment for each $1 \leq t \leq \Gamma$ and ring $B'_{i,j}$. Assign 1 weight of each point $\phi(p) \in W_{i,j,t}$ to the center of C where p is assigned in ψ. (WLOG, one can assume that the weights of our coreset points are integral.) Note that for each point in $W_{i,j,t}$ exactly $|P'_{i,j,t}|/|W_{i,j,t}|$ amount of weight has been assigned. The new assignment for coreset points induces a valid clustering and satisfies $\mathbf{M}$. By triangle inequality it follows that,

$$
\begin{aligned}
|\mathrm{cost}(P,\mathbf{M},C)-\mathrm{wcost}(W,\mathbf{M},C)| &\le \sum_{t=1}^{\Gamma}\sum_{(i,j)}\sum_{p\in P'_{i,j,t}}\mathrm{dist}(p,\phi(p))\\
&\le \sum_{t=1}^{\Gamma}\sum_{(i,j)}\sum_{p\in P'_{i,j,t}}2^{j+1}\mu\\
&\le \sum_{(i,j)}\sum_{p\in P_{i,j}}2^{j+1}\mu\\
&= \sum_{i=1}^{k}\sum_{p\in P_{i,0}}2\mu+\sum_{p\in P_{i,j}|j\ge 1}2^{j+1}\mu\\
&\le 2n\mu+4\sum_{i=1}^{k}\sum_{p\in P_i^*}\mathrm{dist}(p,c_i^*)\\
&\le 2\cdot\mathrm{OPT}_v+4\cdot\Pi\le 6\cdot\Pi
\end{aligned}
$$

Here $P_i^* \subseteq P$ is the set of points assigned to c_i^*. The second last inequality follows, as for each point $p \in P_{i,j}$ with $j \geq 1$, $\mathrm{dist}(p, c_i^*) \geq 2^{j-1}\mu$. Now there is a point $\hat{p}$ that is assigned to $\hat{c} \in C$ such that $\hat{c}$ is not in $\cup_i Q_i$. Let $\hat{p} \in P_i^*$. It follows that,

$$\mathrm{cost}(P,\mathbf{M},C) \geq \mathrm{dist}(\hat{c},\hat{p}) \geq \mathrm{dist}(\hat{c},c_i^*) - \mathrm{dist}(c_i^*,\hat{p}) \geq 7\nu n\cdot\mu/\varepsilon - \nu n\mu \geq 6\nu n\cdot\mu/\varepsilon = 6\cdot\Pi/\varepsilon$$

The third inequality follows, as $\mathrm{dist}(\hat{c}, c_i^*) > (2^N\mu)/2 \geq \nu n \cdot \mu/\varepsilon$ and $\mathrm{dist}(c_i^*, \hat{p}) \leq \Pi = \nu n\mu$. Thus, $\Pi \leq \varepsilon \cdot \mathrm{cost}(P,\mathbf{M},C)/6$. Hence,

$$|\mathrm{cost}(P,\mathbf{M},C) - \mathrm{wcost}(W,\mathbf{M},C)| \leq 6\cdot\Pi \leq \varepsilon\cdot\mathrm{cost}(P,\mathbf{M},C).$$

□

Next, we consider the case when all points in C are in $\cup_i Q_i$. Let $C' \subseteq F$ be the set of centers constructed by replacing each point c in C, by the representative of the gridcell that contains c. Then, we have the following observation.

Observation 7.34. *$|cost(P,\mathbf{M},C) - cost(P,\mathbf{M},C')| \leq \varepsilon\cdot\mathrm{OPT}_v$ and $|cost(W,\mathbf{M},C) - cost(W,\mathbf{M},C')| \leq \varepsilon\cdot\mathrm{OPT}_v$.*

Lemma 7.35. *For every set $C \subseteq \mathbb{R}^d$ of k centers such that all centers are contained in $\cup_i Q_i$ and for all constraint $\mathbf{M}$, w.p. at least $1 - 1/n^2$, $|\mathrm{cost}(P,\mathbf{M},C) - \mathrm{wcost}(W,\mathbf{M},C)| \leq \varepsilon\cdot\mathrm{cost}(P,\mathbf{M},C)$.*

Proof. Define the set C' from C as above. It follows that,

$$\begin{aligned}
&|\mathrm{cost}(P,\mathbf{M},C) - \mathrm{wcost}(W,\mathbf{M},C)| \\
&\leq |\mathrm{cost}(P,\mathbf{M},C) - \mathrm{cost}(P,\mathbf{M},C') + \mathrm{cost}(P,\mathbf{M},C') - \mathrm{wcost}(W,\mathbf{M},C')+ \\
&\qquad\qquad \mathrm{wcost}(W,\mathbf{M},C') - \mathrm{wcost}(W,\mathbf{M},C)| \\
&\leq |\mathrm{cost}(P,\mathbf{M},C) - \mathrm{cost}(P,\mathbf{M},C')| + |\mathrm{cost}(P,\mathbf{M},C') - \mathrm{wcost}(W,\mathbf{M},C')|+ \\
&\qquad\qquad |\mathrm{wcost}(W,\mathbf{M},C') - \mathrm{wcost}(W,\mathbf{M},C)| \\
&\leq 2\varepsilon \cdot OPT_v + |\mathrm{cost}(P,\mathbf{M},C') - \mathrm{wcost}(W,\mathbf{M},C')|
\end{aligned}$$

The last inequality follows from Observation 7.34. By Lemma 7.32, we obtain for every C and all $\mathbf{M}$, w.p. at least $1 - 1/n^2$,

$$\begin{aligned}
|\mathrm{cost}(P,\mathbf{M},C) - \mathrm{wcost}(W,\mathbf{M},C)| &\leq 2\varepsilon \cdot OPT_v + \varepsilon \cdot \mathrm{cost}(P,\mathbf{M},C') \\
&\leq \varepsilon \cdot \mathrm{cost}(P,\mathbf{M},C) + 3\varepsilon \cdot OPT_v \leq 4\varepsilon \cdot \mathrm{cost}(P,\mathbf{M},C)
\end{aligned}$$

By scaling ε by a factor of 4, the lemma follows. □

Now, $\Theta(\log(nb)) = \Theta(\log n + \log k + \log\log(n/\varepsilon) + d\log(1/\varepsilon)) = \Theta(\log n + d\log(1/\varepsilon))$. Thus $s = \Theta\left(\frac{1}{\varepsilon^3} \cdot k(\log n + d\log(1/\varepsilon))\right)$. By Lemmas 7.35 and 7.33, Theorem 7.30 follows.

7.2.4 k-Means Clustering

Here we describe the changes needed to extend the coreset construction scheme for k-median to k-means. In the end of the section, we also show how to apply well-known dimensionality reduction techniques to obtain a coreset with the size independent of d in the Euclidean case. First, we consider the disjoint group case. The coreset construction algorithm is identical except here from each ring and for each color, we select a sample of size $\mathcal{O}(k\log n/\varepsilon^5)$. The analysis remains almost the same except in places we obtain worse bounds due to squaring of the distances.

In the single ring-single color case, instead of Theorem 7.9, we have the following modified theorem.

Theorem 7.36. *W.p. at least $1 - 1/n^{2k+10}$, it holds that*

$$|\mathrm{cost}(P,\mathbf{M},C) - \mathrm{wcost}(W',\mathbf{M},C)| \leq \varepsilon m {\mu'}^2 + O(\varepsilon) \cdot \mathrm{cost}(P,\mathbf{M},C).$$

The network G_Y is defined in a different way in this case to deal with the square of distances. In particular, we adapt a bipartite matching framework. The points (sources) have positive demands and are placed on the left side, and centers (sinks) have negative demands and are placed on the right. If the demand $m - \sum_{p\in P'} Y[p]$ corresponding to the bicriteria center c' is non-negative, it is placed on the left as a source. Otherwise, it is placed on the right as a sink. The costs of the edges are now set to square of the corresponding distances.

Lemma 7.13 continues to hold even in this case. Thus for the same reason we readily obtain, w.p. $1 - 1/n^{2k+20}$, $f(\mathrm{E}[X]) \leq f(X) + \varepsilon m\mu'$. To prove, w.p. $1 - 1/n^{2k+20}$, $f(X) \leq f(\mathrm{E}[X]) + \varepsilon m\mu'$, we need to show, w.p. at least $1 - 1/n^3$, $f(X) \leq f(\mathrm{E}[X]) + \varepsilon m\mu'/3$. Here we need significant amount of changes in the analysis. Again we have two cases based on the expectation of $|Q'_i|$. Here we need a slightly different bound on the expectation $\varepsilon^3 s/(100k)$ instead of $\varepsilon s/(100k)$.

Case 1. $\mathrm{E}[|Q'_i|] \geq \varepsilon^3 s/(100k)$**.** In this case, $|P'_i| \cdot s/m \geq \varepsilon^3 s/(100k)$, or $|P'_i| \geq \varepsilon^3 m/(100k)$. Note that Observation 7.17 continues to hold, as s is set to $\Theta(k \log n/\varepsilon^5)$, and thus Observations 7.18 and 7.19 as well.

Now, we give bound on the cost of the computed flow. Note that we route m/s flow for each point in Q''_i to c_i whose total cost is $\sum_{p \in Q''_i} (m/s) \cdot \mathrm{dist}(p, c_i)$.

For points $p \in P'_i$, the distances $\mathrm{dist}(p, c_i)$ lie in an interval of length at most $2\mu'$. Thus the average of these distances must also lie in this interval. It follows that,

$$\begin{aligned}
\mathrm{dist}(p, c_i)^2 &\leq \Big(\frac{1}{|P'_i|} \cdot \sum_{p \in P'_i} \mathrm{dist}(p, c_i) + 2\mu'\Big)^2 \\
&\leq 2\Big(\frac{1}{|P'_i|} \cdot \sum_{p \in P'_i} \mathrm{dist}(p, c_i)\Big)^2 + 8{\mu'}^2 \\
&\leq \frac{2}{(|P'_i|)^2} \cdot |P'_i| \cdot \sum_{p \in P'_i} \mathrm{dist}(p, c_i)^2 + 8{\mu'}^2 \\
&\leq \frac{2}{|P'_i|} \cdot \sum_{p \in P'_i} \mathrm{dist}(p, c_i)^2 + 8{\mu'}^2
\end{aligned}$$

The second last inequality follows from Cauchy-Schwarz's inequality. Now, we can apply Lemma 7.20 setting $T = 8{\mu'}^2$, $V = P'_i$, $U = Q''_i$, $h(p) = \mathrm{dist}(p, c_i)$, $\delta = \varepsilon{\mu'}^2/20$ and $\lambda = 1/n^{10}$. Note that,

$$r \geq (1 - \varepsilon/20) \cdot |P'_i| \cdot s/m \geq (1 - \varepsilon/20) \cdot \frac{\varepsilon^3 m}{100k} \cdot \frac{s}{m} \geq \Theta(\log n/\varepsilon^2) \geq (T^2/2\delta^2) \ln(2/\lambda)$$

The last inequality follows assuming a sufficiently large constant is hidden in $\Theta(.)$ in the definition of s.

We obtain, w.p. at least $1 - 1/n^{10}$,

$$\begin{aligned}
h(Q''_i) &\leq \frac{h(P'_i) \cdot |Q''_i|}{|P'_i|} + \delta \cdot (|P'_i| \cdot s/m) \\
\text{Or, } h(Q''_i) \cdot (m/s) &\leq \frac{h(P'_i) \cdot |Q''_i|}{|P'_i|} \cdot (m/s) + \varepsilon |P'_i| \cdot {\mu'}^2/20 \\
\text{Or, } h(Q''_i) \cdot (m/s) &\leq (1 + \frac{\varepsilon}{50}) \cdot h(P'_i) + \varepsilon |P'_i| \cdot {\mu'}^2/20
\end{aligned}$$

The last inequality follows from Observation 7.17 considering both cases in the flow construction. Next we compute the additional costs. We have two cases. In the first case, $|Q'_i| \leq |P'_i| \cdot s/m$ and we need to route $(|P'_i| - |Q'_i| \cdot m/s)$ amount of flow from c' to c_i. The cost is at most,

$$\begin{aligned}
&(|P'_i| - |Q'_i| \cdot m/s) \cdot \mathrm{dist}(c', c_i) \\
&\leq \varepsilon \cdot \frac{|P'_i|}{50} \cdot (\frac{2}{|P'_i|} \cdot \sum_{p \in P'_i} \mathrm{dist}(p, c_i)^2 + 8{\mu'}^2) \\
&\leq \frac{\varepsilon}{25} \cdot \sum_{p \in P'_i} \mathrm{dist}(p, c_i)^2 + \frac{4\varepsilon \cdot |P'_i|}{25} {\mu'}^2
\end{aligned}$$

The first inequality follows from Observation 7.17 and from the fact that c' is the ring center.

In the second case, $|Q'_i| > |P'_i| \cdot s/m$. Note that in this case we need to route at least $|P'_i| - |Q''_i| \cdot m/s$ flow from one point p to c_i, as $|Q''_i| = \lfloor |P'_i| \cdot s/m \rfloor$, and m/s flow for each point in $Q'_i \setminus Q''_i$ to w. The first cost is at most,

$$\begin{aligned}
&(|P'_i| - |Q''_i| \cdot m/s) \cdot \mathrm{dist}(p, c_i) \\
&\leq \varepsilon \cdot \frac{|P'_i|}{20} \cdot (\frac{2}{|P'_i|} \cdot \sum_{p \in P'_i} \mathrm{dist}(p, c_i)^2 + 8{\mu'}^2) \\
&\leq \frac{\varepsilon}{10} \cdot \sum_{p \in P'_i} \mathrm{dist}(p, c_i)^2 + \frac{2\varepsilon \cdot |P'_i|}{5} {\mu'}^2
\end{aligned}$$

The first inequality follows from Observation 7.19. The second cost can be bounded by,

$$\begin{aligned}
&\sum_{p \in (Q'_i \setminus Q''_i)} (m/s) \cdot \mathrm{dist}(p, c')^2 \\
&\leq (|Q'_i| - |Q''_i|) \cdot (m/s) \cdot {\mu'}^2 \\
&\leq (\varepsilon |P'_i| \cdot s/(40m)) \cdot (m/s) \cdot {\mu'}^2 \\
&\leq \varepsilon |P'_i| {\mu'}^2/40
\end{aligned}$$

The second inequality follows from Observation 7.18. Thus, in this case, the total cost is bounded by,

$$(1 + \frac{3\varepsilon}{25}) \cdot \sum_{p \in P'_i} \mathrm{dist}(p, c_i)^2 + \frac{19\varepsilon \cdot |P'_i|}{40} {\mu'}^2.$$

Case 2. $\mathrm{E}[|Q'_i|] < \varepsilon^3 s/(100k)$**.** We give separate bounds for the two cases. In the first case, $|Q'_i| \leq |P'_i| \cdot s/m$. In this case, we need to route m/s flow from points in Q''_i to c_i and $(|P'_i| - |Q'_i| \cdot m/s)$ amount of flow from c' to c_i. Let $p_{\min}$ and $p_{\max}$ be the nearest and farthest points in P'_i from c_i. The total cost is,

$$\begin{aligned}
&\sum_{p \in Q'_i} (m/s) \cdot \mathrm{dist}(p, c_i)^2 + (|P'_i| - |Q'_i| \cdot m/s) \cdot \mathrm{dist}(c', c_i)^2 \\
&\leq \sum_{p \in Q'_i} (m/s) \cdot \mathrm{dist}(p_{max}, c_i)^2 + (|P'_i| - |Q'_i| \cdot m/s) \cdot \mathrm{dist}(c', c_i)^2 \\
&\leq |P'_i| \cdot \max\{\mathrm{dist}(p_{\max}, c_i)^2, \mathrm{dist}(c', c_i)^2\} \\
&\leq |P'_i| \cdot (\mu' + \mathrm{dist}(c', c_i))^2 \\
&\leq |P'_i| \cdot (2\mu' + \mathrm{dist}(p_{\min}, c_i))^2
\end{aligned}$$

The first inequality follows by replacing the squares of the distances by their maximum. The third inequality follows by noting that $\mathrm{dist}(p_{\max}, c_i) \leq \mathrm{dist}(p_{\max}, c') + \mathrm{dist}(c', c_i) \leq \mu' + \mathrm{dist}(c', c_i)$. The last inequality follows by noting that $\mathrm{dist}(c', c_i) \leq \mathrm{dist}(c', p_{\min}) + \mathrm{dist}(p_{\min}, c_i)$.

Next, we upper bound the above expression. We consider two subcases. The first one is $\mathrm{dist}(p_{\min}, c_i) \leq 2\mu'/\varepsilon$. In this subcase,

$$\begin{aligned}
|P'_i| \cdot (2\mu' + \mathrm{dist}(p_{\min}, c_i))^2 &\leq \frac{\varepsilon^3 m}{100k} \cdot (2\mu' + 2\mu'/\varepsilon)^2 \\
&\leq \frac{\varepsilon^3 m}{25k} \cdot {\mu'}^2 (1 + 1/\varepsilon)^2 \\
&= \frac{O(\varepsilon m)}{k} \cdot {\mu'}^2.
\end{aligned}$$

In the other subcase $\mathrm{dist}(p_{\min}, c_i) > 2\mu'/\varepsilon$.

$$\begin{aligned}
|P'_i| \cdot (2\mu' + \mathrm{dist}(p_{\min}, c_i))^2 &\leq |P'_i| \cdot (\varepsilon\, \mathrm{dist}(p_{\min}, c_i) + \mathrm{dist}(p_{\min}, c_i))^2 \\
&\leq |P'_i| \cdot \mathrm{dist}(p_{\min}, c_i)^2 \cdot (1 + \varepsilon)^2 \\
&= (1 + O(\varepsilon)) \sum_{p \in P'_i} \mathrm{dist}(p, c_i)^2
\end{aligned}$$

The last inequality follows, as $\mathrm{dist}(p, c_i) \leq \mathrm{dist}(p_{\min}, c_i)$ for all $p \in P'_i$. Now, we consider the second case: $|Q'_i| > |P'_i| \cdot s/m$. We need to route the flow from points in Q''_i to c_i. Additionally, we need to route at least $|P'_i| - |Q''_i| \cdot m/s$ flow from one point p^* to c_i, as $|Q''_i| = \lfloor |P'_i| \cdot s/m \rfloor$, and m/s flow for each point in $Q'_i \setminus Q''_i$ to w. The sum of the first two costs is at most,

$$\begin{aligned}
&\sum_{p\in Q_i''} (m/s)\cdot \mathrm{dist}(p,c_i)^2 + (|P_i'| - |Q_i''|\cdot m/s)\cdot \mathrm{dist}(p^*,c_i)^2 \\
&\leq \sum_{p\in Q_i''} (m/s)\cdot \mathrm{dist}(p_{max},c_i)^2 + (|P_i'| - |Q_i''|\cdot m/s)\cdot \mathrm{dist}(p_{max},c_i)^2 \\
&\leq |P_i'|\cdot \mathrm{dist}(p_{\max},c_i)^2 \\
&= \frac{O(\varepsilon m)}{k}\cdot {\mu'}^2 + (1+O(\varepsilon))\sum_{p\in P_i'} \mathrm{dist}(p,c_i)^2
\end{aligned}$$

The last equality follows in the same way as in the first case. The remaining cost in the second case can be bounded by,

$$\begin{aligned}
&\sum_{p\in (Q_i'\setminus Q_i'')} (m/s)\cdot \mathrm{dist}(p,c')^2 \\
&\leq (|Q_i'| - |Q_i''|)\cdot (m/s)\cdot {\mu'}^2 \\
&\leq |Q_i'|\cdot (m/s)\cdot {\mu'}^2 \\
&\leq (\varepsilon s/(50k))\cdot (m/s)\cdot {\mu'}^2 \\
&= (\varepsilon m/(50k))\cdot {\mu'}^2
\end{aligned}$$

Thus, the total cost in both the cases is bounded by,

$$\frac{O(\varepsilon m)}{k}\cdot {\mu'}^2 + (1+O(\varepsilon))\sum_{p\in P_i'} \mathrm{dist}(p,c_i)^2$$

General Upper Bound on the Cost. By merging the cost in both cases, we obtain the common upper bound,

$$\begin{aligned}
&(1+O(\varepsilon))\sum_{p\in P_i'} \mathrm{dist}(p,c_i)^2 + \frac{19\varepsilon\cdot |P_i'|}{40}{\mu'}^2 + \frac{O(\varepsilon m)}{k}\cdot {\mu'}^2 \\
&= (1+O(\varepsilon))\sum_{p\in P_i'} \mathrm{dist}(p,c_i)^2 + O(\varepsilon\cdot |P_i'|)\cdot {\mu'}^2 + \frac{O(\varepsilon m)}{k}\cdot {\mu'}^2
\end{aligned}$$

Summing over all the centers in C, we obtain,

$$\begin{aligned}
wcost(W', C, \mathbf{M}) &\leq (1+O(\varepsilon))\cdot cost(P, C, \mathbf{M}) + O(\varepsilon\cdot |P'|)\cdot {\mu'}^2 + O(\varepsilon)\cdot m\cdot {\mu'}^2 \\
&= (1+O(\varepsilon))\cdot cost(P, C, \mathbf{M}) + O(\varepsilon)\cdot m\cdot {\mu'}^2.
\end{aligned}$$

Summing the cost over all rings gives us,

$$|cost(P,C,\mathbf{M}) - wcost(W',C,\mathbf{M})| \leq \sum_{(i,j)} O(\varepsilon)\cdot|P'_{i,j}|\cdot 2^j\mu^2 + O(\varepsilon k\log n)\cdot cost(P,C,\mathbf{M})$$
$$= O(\varepsilon k\log n)\cdot cost(P,C,\mathbf{M})$$

Note that the coreset size for each ring and for each color was $\mathcal{O}(k\log n/\varepsilon^5)$. To obtain the desired ε error, we need to scale ε by a factor of $\Theta(k\log n)$. Thus, the required size of the coreset becomes $\mathcal{O}((k\log n)^6/\varepsilon^5)$. Summing over all rings and colors we obtain the desired bound of $\mathcal{O}(\ell(k\log n)^7/\varepsilon^5)$ on our coreset size.

This proves the disjoint case of Theorem 7.2 for k-means. The coreset construction algorithm for k-means in the overlapping group case is again the same as that for k-median, except the bound on sample size. From the above analysis and the analysis for k-median, we obtain the desired result. This proves the overlapping case of Theorem 7.2 for k-means.

In the Euclidean case, the extension of the analysis for k-median to k-means is trivial. We obtain the following generic theorem.

Theorem 7.37. *Given a collection of ℓ possibly overlapping groups consisting of n points in total in a metric space, there is an $\mathcal{O}(n(k+\ell))$ time randomized algorithm that w.p. at least $1-1/n$, computes a universal coreset for k-means clustering of size $\mathcal{O}(\Gamma(k\log n)^7/\varepsilon^5)$. In the Euclidean case, the size of the coreset is $O\left(\frac{\Gamma}{\varepsilon^5}\cdot k^7(\log n)^6(\log n + d\log(1/\varepsilon))\right)$, and the running time is $\mathcal{O}(nd(k+l))$.*

7.3 Assignment Problem for (α,β)-Fair Clustering

Recall that we are given ℓ groups $\{P_i\}$ of P, and $P^1,\ldots,P^\Gamma$ are the point equivalence classes. Also, I_t is the set of indexes of the groups corresponding to P^t, for each $t\in[\Gamma]$. We aim to solve (α,β)-Fair Clustering on our coreset W instead of on the original points. Suppose we are given the optimal set of centers C for (α,β)-Fair Clustering. Let $\mathcal{M}$ be the collection of coloring constraints that express the assignment restriction of (α,β)-Fair Clustering. Since W is a universal coreset, computing the minimum $\text{wcost}(W,\mathbf{M},C)$ over all $k\times\Gamma$ matrix $\mathbf{M}\in\mathcal{M}$ would give us the optimal cost of fair clustering, modulo a $(1\pm\varepsilon)$ factor. Now, recall that, for k-median, $\text{wcost}(W,\mathbf{M},C)$ is the minimum value $\sum_{x\in P,c_j\in C}\psi(x,c_j)\cdot\text{dist}(x,c_j)$ over all assignments $\psi: P\times C\to\mathbb{R}_{\geq 0}$ such that

(i) For each $x\in P$, $\sum_{c_j\in C}\psi(x,c_j)=w(x)$.

(ii) For each $c_j\in C$ and class $1\leq t\leq\Gamma$, $\sum_{x\in P^t}\psi(x,c_j)=\mathbf{M}_{jt}$.

Thus, given an $\mathbf{M}$, we can compute $\text{wcost}(W,\mathbf{M},C)$ by solving a minimum cost flow problem. But, as the size of $\mathcal{M}$ can be sufficiently large, we cannot try out

all possible $\mathbf{M}$. Note that as the optimal $\mathbf{M} \in \mathcal{M}$ represents a fair partition of the equivalence classes $\{P^t\}$ between the centers , ψ automatically satisfies the fairness properties:

$$\sum_{x \in P_i} \psi(x, c_j) \leq \alpha_i \cdot \sum_{x \in P} \psi(x, c_j), \quad \forall c_j \in C, \forall i \in [\ell],$$
$$\sum_{x \in P_i} \psi(x, c_j) \geq \beta_i \cdot \sum_{x \in P} \psi(x, c_j), \quad \forall c_j \in C, \forall i \in [\ell].$$

Now, as the optimal $\mathbf{M}$ has all integer entries, the optimal cost assignment ψ must also be integral. Here we assume that the coreset points have integer weights. We note that our construction can be slightly modified to obtain coreset with integer weights (e.g, see Chen's adaptation [53]). Thus, given W and C it is sufficient to compute a minimum cost integral assignment that satisfies the above two inequalities and the constraint: For each $x \in P$, $\sum_{c_j \in C} \psi(x, c_j) = w(x)$. We refer to this assignment problem as Weighted Fair Assignment. Our main theorem of this section provides an algorithm with running time $(k\Gamma)^{O(k\Gamma)}|W|^{O(1)}$ for this problem. The general idea is to reduce the assignment problem to a linear programming problem. The unknown optimal assignment can be naturally expressed in terms of linear inequalities, along with the condition that the assignment is fair. However, the issue is that in general the optimal fractional solution to this linear programming problem is not integral, and the integrality gap could be arbitrarily large. Thus, an optimal fractional solution does not yield the desired assignment. And indeed, it was observed already by Bera et al. [23] that the assignment problem for (α, β)-Fair Clustering is NP-hard, so there is no hope to have a polynomial time assignment algorithm.

We cannot afford to make all variables integral and solve an integer linear program (ILP) instead, as the number of variables is large, of order $|W|k$, and in our construction $|W|$ is polylogarithmic in n. However, note that the optimal assignment has the property that for each $c_j \in C$ and class $1 \leq t \leq \Gamma$, $\sum_{x \in P^t} \psi(x, c_j) = \mathbf{M}_{jt}$. Thus the amount of weight assigned from each class to each center is an integer. Using this observation, we reduce our problem to a mixed-integer linear programming problem and force only $k \cdot \Gamma$ variables to be integral. These variables correspond exactly to the entries of the constraint matrix $\mathbf{M}$. Then, we show that this automatically ensures that all the other variables are integral as well, in the optimal solution.

Next, we state one of the equivalent formulations of the Mixed-Integer Linear Programming problem. The input to the problem is a matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$, a vector $\mathbf{b} \in \mathbb{R}^m$, a vector $\mathbf{c} \in \mathbb{R}^d$, and a parameter p, $0 \leq p \leq d$. The goal is to find a vector $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ such that $x_1, \ldots, x_p \in \mathbb{Z}$, $\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$, and the value $\mathbf{c} \cdot \mathbf{x}$ is minimized across all vectors satisfying the above.

By the celebrated result of Lenstra [142], Mixed-Integer Linear Programming is solvable in FPT time when parameterized by the number of integer variables p. We use the following commonly employed version of this result, following the improvements to the original Lenstra's algorithm given by Kannan [127], and Frank

and Tardos [93].

Proposition 7.38 ([142], [127], [93])**.** *There is an algorithm solving* Mixed-Integer Linear Programming *in time* $\mathcal{O}(p^{2.5p+o(p)}d^4L)$ *and space polynomial in* L*, where* L *is the bitsize of the given instance.*

Now we present the assignment algorithm itself. Note that it is sufficient to consider only the points in W for the purpose of computing an assignment, as the other points in P have zero weights. For simplicity, we denote $|W|$ by n. There is practically no difference between the cases of k-median and k-means concerning the assignment problem, and thus we state it for both cases.

Theorem 7.39. *There is an algorithm that given an instance of* Weighted Fair Assignment*, i.e, a weighted set* W *of* n *points and a set* $C = \{c_1, \dots, c_k\}$ *of* k *centers, computes an optimal assignment of* W *with the set of centers* C*. That is, the output is a minimum cost assignment* $\psi : P \times C \to \mathbb{Z}_{\geq 0}$ *that corresponds to* (α, β)-Fair Clustering*. The running time of the algorithm is* $(k\Gamma)^{O(k\Gamma)}n^{O(1)}L$*, where* L *is the total number of bits in the encoding of distances and weights in the instance.*

Proof. We reduce Weighted Fair Assignment to Mixed-Integer Linear Programming. The formulation of our problem itself follows the natural way of treating a clustering assignment problem as a flow problem. Let $W = \{(p_1, w(p_1)), \cdots, (p_n, w(p_n))\}$. For every point p_i and center c_j introduce a variable f_{ij} corresponding to how much weight from the i-th point is assigned to the j-th center. Also, for every center c_j and point equivalence class $t \in \{1, \dots, \Gamma\}$ introduce a variable g_{tj}, corresponding to how much weight from points of the class t the j-th center gets. The following constraints express that $\{f_{ij}\}$ and $\{g_{tj}\}$ define a fair clustering:

$$f_{ij} \geq 0 \qquad \forall i \in \{1, \dots, n\},\ j \in \{1, \dots, k\}, \tag{7.1}$$

$$g_{tj} \in \mathbb{Z}_{\geq 0} \qquad \forall j \in \{1, \dots, k\},\ t \in \{1, \dots, \Gamma\}, \tag{7.2}$$

$$\sum_{1 \leq j \leq k} f_{ij} = w(p_i) \qquad \forall i \in \{1, \dots, n\}, \tag{7.3}$$

$$\sum_{i \in [n] : p_i \in P^t} f_{ij} = g_{tj} \qquad \forall j \in \{1, \dots, k\},\ t \in \{1, \dots, \Gamma\}, \tag{7.4}$$

$$\sum_{i \in [n] : p_i \in P_q} f_{ij} \geq \beta_q \sum_{i \in [n]} f_{ij} \qquad \forall j \in \{1, \dots, k\},\ \forall q \in \{1, \dots, \ell\}, \tag{7.5}$$

$$\sum_{i \in [n] : p_i \in P_q} f_{ij} \leq \alpha_q \sum_{i \in [n]} f_{ij} \qquad \forall j \in \{1, \dots, k\},\ \forall q \in \{1, \dots, \ell\}. \tag{7.6}$$

Note that for a color $q \in \{1, \dots, \ell\}$, $\sum_{i \in [n] : p_i \in P_q} f_{ij}$ is precisely the weight assigned from points of color q to the center j, and $\sum_{i \in [n]} f_{ij}$ is the total weight assigned to

the center j. Thus Constraints (7.5) and (7.6) ensure that the assignment is indeed fair. Finally, the objective function is

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{k} d_{ij} f_{ij}, \tag{7.7}$$

where $d_{ij} = \text{dist}(p_i, c_j)$ in the case of k-median, and $d_{ij} = \text{dist}(p_i, c_j)^2$ in the case of k-means.

We solve the Mixed-Integer Linear Programming defined above by using Proposition 7.38. We require that the variables $\{g_{tj}\}$ take integral values, while we do not impose this restriction on the variables $\{f_{ij}\}$. Thus, in time $(k\Gamma)^{O(k\Gamma)} n^{O(1)} L$ we find the optimal solution $\{f_{ij}\}$, $\{g_{tj}\}$.

Clearly, Constraints (7.1)–(7.6) ensure that the assignment defined by $\{f_{ij}\}$ corresponds to Weighted Fair Assignment, except for the fact that some of $\{f_{ij}\}$ might be fractional. We now show that the integrality of $\{g_{tj}\}$ guarantees that there exists an optimal solution to (7.1)–(7.7) that is integral. For every equivalence class $t \in \{1, \ldots, \Gamma\}$ consider the following flow network. The network is essentially a restriction of (7.1)–(7.4) to the class t assuming that the values $\{g_{tj}\}$ are fixed. There is a node associated with every point $p_i \in P^t$ that has a supply of $w(p_i)$, and there is a node associated with every $c_j \in C$ that has a demand of g_{tj}. There is an edge e_{ij} between each point $p_i \in P^t$ and every center $c_j \in C$ that has an unlimited capacity and the cost d_{ij}. In this network, there is a maximum flow of minimal cost $\{f'_{ij}\}$ that has only integral values, since all the supplies, demands and capacities in the network are integers. Now we replace the respective values of $\{f_{ij}\}$ with the obtained $\{f'_{ij}\}$ that are integral and still satisfy (7.1)–(7.4). The cost is unchanged since $\{f_{ij}\}$ induces a maximum flow in the network as well. Thus the old cost can only be larger, but also $\{f_{ij}\}$ is an optimal solution to the Mixed-Integer Linear Programming instance, so the new cost cannot be smaller. After we perform the above for every class, the whole assignment is integral, now satisfying the statement of the theorem completely. Finally, note that $\{f'_{ij}\}$ can be found in $n^{O(1)} L$ time with the known values of $\{g_{tj}\}$ by any polynomial time minimum-cost flow algorithm. □

The algorithm in Theorem 7.39 allows us to solve (α, β)-Fair Clustering on the original points as well, as long as we know a suitable set of k centers. However, the running time would have a heavy dependence on n, roughly n^5. So to obtain a near-linear time algorithm, we cannot use Theorem 7.39 directly on the original points, even if we know the centers. Instead, in the approximation algorithms we present, we first compute a universal coreset of the original set of points, and then solve all the arising instances of the assignment problem on the coreset, thus inflicting only polylogarithmic in n time. Still, at the end we have to output a low-cost fair assignment of the original points, and again we cannot directly use Theorem 7.39. So we show how to compute the assignment in near-linear time with the help of the coreset. The idea is to run Theorem 7.39 on the coreset and then use the optimal

solution there to find a good assignment of the original points in a simpler way. Namely, knowing how many points from each equivalence class are assigned to each center, the assignment problem boils down to finding a minimum-cost flow in a bipartite network where one of the parts is small. First, we recall a suitable minimum-cost flow result by Ahuja et al. [6].

Proposition 7.40 (Theorem 7.3 in [6])**.** *The minimum-cost flow problem on a bipartite network is solvable in time $\mathcal{O}((n_1 m + n_1^3)\log(n_1 D))$, where n_1 is the size of the smaller part in the network, m is the number of edges, and D is the maximum cost of an edge in the network.*

Now we prove a general lemma that allows us to transfer any fair assignment from the coreset to a fair assignment on the original points in polynomial time, while losing only a factor of $(1+\varepsilon)$ in the cost.

Lemma 7.41. *There is an algorithm that given a set of points P with the ℓ groups P_1, ..., P_ℓ, a coreset W of P, a set of k centers C, a fair assignment $\psi : P \times C \to \mathbb{Z}_{\geq 0}$, and a value $0 < \varepsilon \leq 1$, computes a fair assignment of the points of P to the centers of C with the cost at most $(1+\varepsilon)\cdot\mathrm{cost}(\psi)$ in time $\mathcal{O}(\Gamma \cdot k^3/\varepsilon^{O(1)} \cdot n \log n)$. This holds for both (α,β)-Fair k-median and (α,β)-Fair k-meanss in general metric, provided that W satisfies*

$$\mathrm{cost}(P^t, \mathbf{M}, C) \leq (1+\varepsilon/3)\,\mathrm{wcost}(W^t, \mathbf{M}, C),$$

for every column constraint matrix $\mathbf{M} \in \mathbb{Z}^k$, where by W^t we denote the restriction of W to the points of the equivalence class P^t. In the Euclidean case, the running time is multiplied by d.

Proof. For the assignment ψ, consider the values $\{g_{tj}\}_{t\in[\Gamma],\, j\in[k]}$, using the notation in Theorem 7.39, where g_{tj} denotes how many points from the t-th class are assigned to the j-th center by ψ, and the values $\{A_t\}_{t\in[\Gamma]}$, where A_t is the cost of ψ restricted to the t-th class. Now for each class P^t in the original point set P, solve the following assignment problem: assign points of P^t to centers in C such that there are exactly g_{tj} points assigned to the j-th center, and the cost of the assignment is minimum among all such assignments. We naturally view this problem as a minimum-cost flow problem, and we solve it by running the algorithm given by Proposition 7.40. Note that the resulting network has $\mathcal{O}(k)$ vertices in the part corresponding to the centers C, and $\mathcal{O}(nk)$ edges in total. Finally, the resulting fair assignment φ from P to C is the union of assignments from P^t to C for all $t \in [\Gamma]$. Clearly, the obtained assignment is fair, since the fairness condition is completely determined by the numbers $\{g_{tj}\}$. This is true, as in the Constraints 7.5 and 7.6, $\sum_{i\in[n]:p_i\in P_q} f_{ij}$ can be expressed by $\sum_{t\in[\Gamma]:q\in I_t} g_{tj}$ and $\sum_{i\in[n]} f_{ij}$ by $\sum_{t'=1}^{\Gamma} g_{t'j}$. We now argue about the cost. By construction, the cost of the resulting assignment is $\sum_{t=1}^{\Gamma} \mathrm{cost}(P^t, (g_{tj})_{j=1}^k, C)$. Now,

$$\mathrm{cost}(P^t, (g_{tj})_{j=1}^k, C) \leq (1+\varepsilon/3)\,\mathrm{wcost}(W^t, (g_{tj})_{j=1}^k, C) \leq (1+\varepsilon/3)A_t, \quad \forall\, t \in [\Gamma].$$

Summing over all $t \in [\Gamma]$, we obtain

$$\operatorname{cost}(\varphi) = \sum_{t=1}^{\Gamma} \operatorname{cost}(P^t, (g_{tj})_{j=1}^{k}, C) \leq (1+\varepsilon/3) \sum_{t=1}^{\Gamma} A_t = (1+\varepsilon/3)\operatorname{cost}(\psi).$$

By Proposition 7.40 it takes time $\mathcal{O}(\Gamma \cdot k^3/\varepsilon^2 \cdot n \log n)$ to run the minimum-cost flow algorithm Γ times, where we assume that $\log D = O(\log n/\varepsilon^2)$. Finally, we justify the latter by a standard argument reducing the ratio of maximum distance in the instance to the minimum distance. In the network flow instance that we construct from P^t and C, tweak slightly the costs on the edges. Set $\varepsilon_0 = \varepsilon/6$, if an edge costs more than $D_{max} := 2A_t$, replace its cost by D_{max}, and if an edge costs less than $D_{min} := \varepsilon_0 A_t/(2n)$, replace its cost by D_{min}. For all the other edges, round up their cost to the closest value of the form $(1+\varepsilon_0)^q D_{min}$, where q is an integer. In the modified network, the cost scaling part then induces a factor of $\log_{1+\varepsilon_0}(D_{max}/D_{min}) = \log_{1+\varepsilon_0}(4n/\varepsilon_0) = O(\log n/\varepsilon^2)$, instead of $\mathcal{O}(\log D)$.

Now we argue about how this change influences the cost. Consider an optimal assignment $\varphi : P^t \to C$ in the modified network, obtained by the network flow algorithm. Its cost is at most $(1+2\varepsilon_0) \cdot (1+\varepsilon/3)A_t$, since the cost of an optimal assignment φ^* in the original network is at most $(1+\varepsilon/3)A_t$ by the argument above, and the cost of φ^* in the new network is at most $(1+2\varepsilon_0)$ times the cost in the original network. The latter holds since φ^* never uses edges of cost more than $D_{max} = 2A_t$, for the edges between D_{min} and D_{max} the cost increase is at most a factor of $(1+\varepsilon_0)$, and for the edges with the cost less than D_{min}, their total contribution in the new network is at most $n \cdot \varepsilon_0 A_t/n = \varepsilon_0 A_t$. The algorithm outputs the optimal assignment φ in the modified network, and its cost in the original network is at most its cost in the modified network, since edges with cost at least D_{max} are never used, and the cost of all the other edges is less in the original network. Thus, we have shown that the cost of the assignment we constructed is at most $(1+2\varepsilon_0)(1+\varepsilon/3)A_t = (1+\varepsilon/3)(1+\varepsilon/3)A_t \leq (1+\varepsilon)A_t$. From this point, the cost analysis above proceeds, and summing over all $t \in [\Gamma]$ we obtain $\operatorname{cost}(\varphi) \leq (1+\varepsilon)\operatorname{cost}(\psi)$.

Observe that in the Euclidean case we compute distances between the points from their respective d-dimensional vectors, thus taking an extra factor of d in the running time. □

Note that the condition on W in Lemma 7.41 is satisfied by the coresets obtained from Theorem 7.26 and Theorem 7.37 with a suitable error parameter, since the coreset construction samples points in each equivalence class independently, and thus approximately preserves the cost with respect to any column matrix constraint on each of them. Now we show that any instance of the assignment problem can be approximately solved in near-linear FPT time with the help of our coreset construction, Theorem 7.39, and Lemma 7.41.

Lemma 7.42. *Given a set of points P with the ℓ groups $P_1, \ldots, P_\ell$, a set of k centers C, and a parameter $0 < \varepsilon \leq 1$, a fair assignment of the points of P to the*

centers of C with the cost at most $(1+\varepsilon)\,\mathrm{faircost}(P,C)$ can be computed in time $(k\Gamma)^{O(k\Gamma)}(\log n/\varepsilon)^{O(1)} + O(\Gamma \cdot k^3/\varepsilon^2 \cdot n \log n)$ w.h.p. This holds for both (α,β)-Fair k-median and (α,β)-Fair k-meanss in general metric. In the Euclidean case, the running time is multiplied by d.

Proof. The algorithm proceeds as follows. First, we compute a coreset W from the point set P using Theorem 7.26 or Theorem 7.37, depending on the problem, with the error parameter ε_0 to be defined later. Then we compute an optimal fair assignment ψ from W to the centers C by applying Theorem 7.39. Finally, we invoke Lemma 7.41 on the assignment ψ to obtain a fair assignment $\varphi : P \to C$ with the cost at most $(1+3\varepsilon_0)\,\mathrm{cost}(\psi)$. The algorithm returns φ, and in what follows we bound the cost of this assignment. Denote by $\mathbf{M} \in \mathbb{Z}^{k\times\ell}$ the constraint matrix corresponding to the assignment ψ, i.e. $\mathbf{M}_{ij}$ is equal to how many points from the j-th group ψ sends to the i-th center, and by $\mathbf{M}^*$ the constraint matrix corresponding to an optimal fair assignment from P to C. By the choice of $\mathbf{M}$ and $\mathbf{M}^*$, and the fact that W is a universal coreset of P, we obtain

$$\begin{aligned}\mathrm{cost}(\psi) = \mathrm{wcost}(W,\mathbf{M},C) &\le \mathrm{wcost}(W,\mathbf{M}^*,C)\\ &\le (1+\varepsilon_0)\,\mathrm{cost}(P,\mathbf{M}^*,C) = (1+\varepsilon_0)\,\mathrm{faircost}(P,C).\end{aligned}$$

Thus, $\mathrm{cost}(\varphi)$ is at most $(1+3\varepsilon_0)(1+\varepsilon_0)\,\mathrm{faircost}(P,C)$, and setting ε_0 such that $(1+3\varepsilon_0)(1+\varepsilon_0) \le (1+\varepsilon)$ finishes the proof.

As for the running time, the $\mathcal{O}((k+l)\cdot n)$ is for the coreset construction, $\mathcal{O}((k\Gamma)^{O(k\Gamma)}(k\log n/\varepsilon)^{O(1)})$ is for solving the assignment problem on the coreset, and $\mathcal{O}(\Gamma \cdot k^3/\varepsilon^{O(1)} \cdot n\log n)$ is for restoring φ by Lemma 7.41. Not that the coreset construction time is dominated by the last term.

Finally, in the Euclidean case we compute distances between the points from their respective d-dimensional vectors, thus taking an extra factor of d in the running time. Note that we still use the general metric case in Theorems 7.26 and 7.37 for coreset construction, since we only need to preserve the objective with respect to the given set of centers C.

□

7.4 $(1+\varepsilon)$-Approximation in the Euclidean Space

In this section, we present a near-linear time $(1+\varepsilon)$-approximation algorithm for Euclidean (α,β)-Fair k-median and (α,β)-Fair k-meanss. For that purpose, we combine our coreset construction (Theorem 7.26 and Theorem 7.37), our assignment algorithm (Theorem 7.39), and the linear-time constrained clustering algorithm of Bhattacharya et al. [27].

We denote the cost of clustering $C_1, \ldots, C_k$ with the centers $C = (c_1,\ldots,c_k)$ by $\mathrm{cost}_C(C_1,\ldots,C_k)$. By $\mathrm{cost}(C_1,\ldots,C_k)$ we denote $\min_C \mathrm{cost}_C(C_1,\ldots,C_k)$, where

the minimum is over all possible k centers C. It is well-known that in the case of k-means the optimal center for a cluster C_i is its mean $\mu(C_i) := 1/|C_i| \sum_{x \in C_i} x$, thus $\operatorname{cost}(C_1, \dots, C_k) = \operatorname{cost}_{(\mu(C_1),\dots,\mu(C_k))}(C_1, \dots, C_k)$.

Next, we formally restate the result of Bhattacharya et al.

Proposition 7.43 ([27], Theorem 1)**.** *Given a set of n points $P \subset \mathbb{R}^d$, parameters k and ε, there is a randomized algorithm that outputs a list $\mathcal{L}$ of $2^{\tilde{O}(k/\varepsilon)}$ sets of centers of size k such that for any clustering $\{C_1^*, \dots, C_k^*\}$ of P, the following event happens with probability at least $1/2$: there is a set $C \in \mathcal{L}$ such that*

$$\operatorname{cost}_C(C_1^*, \dots, C_k^*) \le (1+\varepsilon) \operatorname{cost}(C_1^*, \dots, C_k^*),$$

where cost *is with respect to the k-means clustering objective. The running time of the algorithm is $nd \cdot 2^{\tilde{O}(k/\varepsilon)}$, where $\tilde{O}$ notation hides a $\mathcal{O}(\log \frac{k}{\varepsilon})$ factor. The same statement holds for k-median, except the size of the list $\mathcal{L}$ becomes $2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}$, and the running time becomes $nd \cdot 2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}$.*

Note that Proposition 7.43 together with our assignment algorithm from Theorem 7.39 already implies $(1+\varepsilon)$-approximation algorithm, as stated in the next claim.

Claim 7.44. *There exists a $(1+\varepsilon)$-approximation algorithm solving (α, β)-*Fair Clustering *in $\mathbb{R}^d$ in time $2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(k\Gamma)^{\mathcal{O}(k\Gamma)} n^{\mathcal{O}(1)} d$ with high probability. The algorithm also extends to the weighted version of the problem.*

Proof. The proof is by solving the assignment problem with the help of Theorem 7.39 on each set of centers in the list returned by Proposition 7.43. We run Proposition 7.43 $\Theta(\log n)$ times to succeed with high probability, and thus run Theorem 7.39 on $2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})} \log n$ candidate sets of centers.

For the weighted version, observe that the algorithm of Proposition 7.43 trivially extends to the case where the input points have weight, since the only step where all the input points are used is to perform D^2-sampling, and there the sampling probabilities just need to be multiplied by the respective weights. Theorem 7.39 holds in the weighted case by definition. □

However, the running time of Claim 7.44 has a high-degree polynomial dependency on n. To achieve a near-linear time algorithm, we use the help of our coreset construction. Observe that Proposition 7.43 together with Lemma 7.42 already imply an algorithm of this form. Nevertheless, we proceed with a variation of this scheme that leads to a slightly better running time, in particular, avoiding a $n \log^2 n$ factor.

The general idea of our algorithm is as follows. First, we obtain a list of candidate sets of centers by Proposition 7.43. Then we compute a universal coreset from the input points such that the objective is preserved with respect to all the computed sets of centers. For each set of k centers in the list we run our assignment algorithm

on the coreset to determine the set of centers with the best cost. The algorithm of Bhattacharya et al. and the coreset computation take linear time, and the assignment problem is solved on the coreset, thus taking time polylogarithmic in n. Finally, we run Lemma 7.41 on the best set of centers to construct a fair assignment on the original points. We state and prove the theorem formally next.

Theorem 7.45. *There is a randomized algorithm that given an instance P of (α, β)-*Fair Clustering *and a parameter $0 < \varepsilon \leq 1$ outputs a set of k centers C and a fair assignment $\varphi : P \to C$ satisfying cost$(\varphi) \leq (1+\varepsilon)\,\mathrm{faircost}(P)$ with high probability. The running time of the algorithm is*

$$2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(k\Gamma)^{\mathcal{O}(k\Gamma)}nd\log n.$$

Proof. First, we run the algorithm given by Proposition 7.43 to obtain a list $\mathcal{L}$ of $2^{\tilde{O}(k/\varepsilon_0^{\mathcal{O}(1)})}$ candidate sets of centers, using the error parameter $\varepsilon_0 < \varepsilon$ to be defined later. To increase the probability of success, we repeat this $\Theta(\log n)$ times concatenating all the obtained lists, to form a list $\mathcal{L}$ of $2^{\tilde{O}(k/\varepsilon_0^{\mathcal{O}(1)})}\log n$ candidate sets of centers.

For (α, β)-Fair k-median, we then compute a universal coreset W of size

$$O(\frac{\Gamma}{\varepsilon_0^3}k^2(\log(n + k2^{\tilde{O}(k/\varepsilon_0^{\mathcal{O}(1)})}\log n))^2) = \Gamma(k/\varepsilon_0 \log n)^{\mathcal{O}(1)},$$

using Theorem 7.26, again with the error parameter ε_0. We use the general metric case of the theorem with respect to the points P and the possible centers contained in the list $\mathcal{L}$. For (α, β)-Fair k-meanss, we employ instead Theorem 7.37 to obtain a universal coreset W, its size is also $\Gamma(k/\varepsilon_0 \log n)^{\mathcal{O}(1)}$. For the rest of the proof, there is no difference between the two problems.

For each set of k centers in $\mathcal{L}$ we run the assignment algorithm given by Theorem 7.39, and select the set of centers C with the best cost. Now we bound faircost(P, C). Denote by $\mathbf{M}$ the color constraint matrix that corresponds to an optimal fair assignment from W to C, it holds that

$$\mathrm{faircost}(P, C) \leq \mathrm{cost}(P, \mathbf{M}, C) \leq \frac{1}{1-\varepsilon_0}\,\mathrm{wcost}(W, \mathbf{M}, C),$$

where the last inequality is by the definition of a universal coreset. By Proposition 7.43 with probability $1 - (1/2)^{\Theta(\log n)} = 1 - (1/n)^{\Theta(1)}$ there is a set $\tilde{C}$ in $\mathcal{L}$ such that faircost$(P, \tilde{C} \leq (1+\varepsilon_0)\,\mathrm{faircost}(P)$. Denote by $\tilde{\mathbf{M}}$ the color constraint matrix that corresponds to an optimal fair assignment from W to $\tilde{C}$, since C achieves the lowest cost of fair clustering for W among $\mathcal{L}$, wcost$(W, \mathbf{M}, C) \leq$ wcost$(W, \tilde{\mathbf{M}}, \tilde{C})$. Denote by $\tilde{\mathbf{M}}^*$ the constraint matrix achieving faircost$(P, \tilde{C}) = \mathrm{cost}(P, \tilde{\mathbf{M}}^*, \tilde{C})$, by the choice of $\tilde{\mathbf{M}}$ we have that

$$\mathrm{wcost}(W, \tilde{\mathbf{M}}, \tilde{C}) \leq \mathrm{wcost}(W, \tilde{\mathbf{M}}^*, \tilde{C}) \leq (1+\varepsilon_0)\,\mathrm{cost}(P, \tilde{\mathbf{M}}^*, \tilde{C}),$$

where the last inequality is because W is a universal coreset of P. And since $\operatorname{cost}(P, \tilde{\mathbf{M}}^*, \tilde{C}) = \operatorname{faircost}(P, \tilde{C}) \leq (1+\varepsilon_0)\operatorname{faircost}(P)$ by the choice of $\tilde{\mathbf{M}}^*$ and $\tilde{C}$, we have the following bound:

$$\operatorname{faircost}(P, C) \leq \frac{1+\varepsilon_0}{1-\varepsilon_0} \operatorname{cost}(P, \tilde{\mathbf{M}}^*, \tilde{C}) \leq \frac{(1+\varepsilon_0)^2}{1-\varepsilon_0} \operatorname{faircost}(P).$$

Finally, we compute a fair assignment from P to C running the algorithm from Lemma 7.42, using the error parameter ε_0. The computed assignment has cost at most $(1+\varepsilon_0)\operatorname{faircost}(P, C)$, which by the above is at most $\frac{(1+\varepsilon_0)^3}{1-\varepsilon_0}\operatorname{faircost}(P)$. Setting ε_0 such that $1+\varepsilon \geq \frac{(1+\varepsilon_0)^3}{1-\varepsilon_0}$ concludes the proof.

The running time of the algorithm is the sum of the $2^{\tilde{\mathcal{O}}(k/\varepsilon^{\mathcal{O}(1)})}nd$ running time of Proposition 7.43 multiplied by $\mathcal{O}(\log n)$, the $\mathcal{O}((k+l)nd)$ running time given by Theorem 7.26/Theorem 7.37, $2^{\tilde{\mathcal{O}}(k/\varepsilon^{\mathcal{O}(1)})}$ times the $(k\Gamma)^{\mathcal{O}(k\Gamma)}(\log n)^{\mathcal{O}(1)}d$ running time of the assignment algorithm given by Theorem 7.39 on the coreset, and finally the running time of Lemma 7.41. All of these terms are dominated by $2^{\tilde{\mathcal{O}}(k/\varepsilon^{\mathcal{O}(1)})}(k\Gamma)^{\mathcal{O}(k\Gamma)}nd\log n$. □

Note that the algorithm in Theorem 7.45 is in a sense a non-typical use of a coreset: we first do the heavy part of running Proposition 7.43 on the original points, and only then use the coreset to speed up the assignment problem. We can also devise a true reductive algorithm, where we first construct a universal coreset from the input data, and then do everything on the coreset, both Proposition 7.43 and selection of the best centers. We show this algorithm in the next subsection.

7.4.1 Reduction to a Small-sized Instance

In fact, we show a general reduction result: that the original instance of (α, β)-Fair Clustering could be replaced by a small-sized one, such that any approximate solution could be lifted from the reduced instance with an extra error factor of $(1+\varepsilon)$. Moreover, it can be done in polynomial time that is near-linear in n and linear in d. Essentially, this result is a combination of the universal coreset property and Lemma 7.41, and it shows that our universal coreset construction can indeed be used for data compression wrt. (α, β)-Fair Clustering. In the next theorem, we state and prove the result formally.

Theorem 7.46. *There is a randomized algorithm that given an instance P of (α, β)-Fair Clustering in $\mathbb{R}^d$ outputs a reduced weighted instance W of size $d(k/\varepsilon \log n)^{\mathcal{O}(1)}$ in the same space. W.h.p. it holds that for any $\gamma \geq 1$, and for any set of k centers C in $\mathbb{R}^d$ and a fair assignment ψ from W to C such that $\operatorname{cost}(\psi) \leq \gamma \operatorname{faircost}(W)$, there exists a fair assignment $\varphi : P \to C$ with the cost at most $(1+\varepsilon)\gamma \operatorname{faircost}(P)$ that can be restored from ψ and C. Both constructing W from P and restoring φ from ψ and C take time $\mathcal{O}(\Gamma k^3/\varepsilon^2 nd \log n)$.*

Proof. The algorithm to construct W from P is simply the algorithm from Theorem 7.26 constructing a universal coreset for (α, β)-Fair k-median in the Euclidean case (Theorem 7.37 for (α, β)-Fair k-median). We invoke the coreset construction algorithm with the error parameter $\varepsilon_0 < \varepsilon$ to be defined later, the $\mathcal{O}((k+l)nd)$ running time is dominated by $\mathcal{O}(\Gamma k^3/\varepsilon^2 nd \log n)$. The reduced weighted instance W is exactly the obtained coreset. Its size is $d(k/\varepsilon \log n)^{\mathcal{O}(1)}$, and w.h.p. for any set C of k centers in $\mathbb{R}^d$ and any constraint matrix $\mathbf{M} \in \mathbb{Z}^{k \times \Gamma}$ it holds that

$$(1-\varepsilon) \cdot \operatorname{cost}(P, \mathbf{M}, C) \leq \operatorname{wcost}(W, \mathbf{M}, C) \leq (1+\varepsilon) \cdot \operatorname{cost}(P, \mathbf{M}, C).$$

Now consider a particular $\gamma > 1$, a set of k centers C and a fair assignment $\psi : P \times C \to \mathbb{Z}_{\geq 0}$ of the coreset W such that $\operatorname{cost}(\psi) \leq \gamma \operatorname{faircost}(W)$. Observe that $\operatorname{faircost}(W) \leq (1+\varepsilon_0) \operatorname{faircost}(P)$ since for the set of centers C^* and the constraint matrix $\mathbf{M}^*$ achieving $\operatorname{faircost}(P) = \operatorname{cost}(P, \mathbf{M}, C)$, it holds that

$$\operatorname{faircost}(W) \leq \operatorname{wcost}(W, \mathbf{M}^*, C^*) \leq (1+\varepsilon_0) \operatorname{cost}(P, \mathbf{M}^*, C^*) = (1+\varepsilon_0) \operatorname{faircost}(P).$$

Thus, $\operatorname{cost}(\psi) \leq (1+\varepsilon_0)\gamma \operatorname{faircost}(P)$. To construct the fair assignment φ, we invoke Lemma 7.41 on the assignment ψ. By Lemma 7.41, the cost of φ is at most

$$(1+3\varepsilon_0) \operatorname{cost}(\psi) \leq (1+3\varepsilon_0)(1+\varepsilon_0)\gamma \operatorname{faircost}(P).$$

Finally, we set ε_0 such that $(1+3\varepsilon_0)(1+\varepsilon_0) \leq (1+\varepsilon)$ to obtain the desired bound.

The running time of Lemma 7.41 is exactly $\mathcal{O}(\Gamma k^3/\varepsilon^2 nd \log n)$, and this dominates the $\mathcal{O}((k+l)nd)$ running time required by the coreset construction. □

Theorem 7.46 allows for any exact or approximate algorithm for (α, β)-Fair Clustering to be run on the small-sized coreset instead of the original points. By plugging in the $(1+\varepsilon)$-approximation algorithm given by Claim 7.44, we obtain the following theorem.

Theorem 7.47. *There is an algorithm solving* (α, β)-Fair Clustering *in time*

$$O(\Gamma k^3/\varepsilon^2 nd \log n) + 2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})} (k\Gamma)^{\mathcal{O}(k\Gamma)} (d \log n)^{\mathcal{O}(1)}$$

with high probability, for any given $0 < \varepsilon \leq 1$.

Proof. Set $\varepsilon_0 = \varepsilon/3$. Invoke Theorem 7.46 with the error parameter ε_0 to obtain a reduced weighted instance W from the input points P. Run the algorithm from Claim 7.44 on W to obtain the set of k centers C and a fair assignment ψ from W to C of cost at most $(1+\varepsilon_0) \operatorname{faircost}(W)$. Finally, by the second part of Theorem 7.46 compute a fair assignment $\varphi : P \to C$. The assignment φ is the output of the algorithm, and its cost is at most $(1+\varepsilon_0)^2 \operatorname{faircost}(P) \leq (1+\varepsilon) \operatorname{faircost}(P)$.

Both algorithms from Theorem 7.46 run in time $\mathcal{O}(\Gamma k^3/\varepsilon^2 nd \log n)$, and running the algorithm from Claim 7.44 on the input of size $n' := d(k/\varepsilon \log n)^{\mathcal{O}(1)}$ amounts to the time complexity of

$$2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})} (k\Gamma)^{\mathcal{O}(k\Gamma)} (n')^{\mathcal{O}(1)} d = 2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})} (k\Gamma)^{\mathcal{O}(k\Gamma)} (d \log n)^{\mathcal{O}(1)}.$$

□

In the running time of Theorem 7.47, observe that the exponential term is just polylogarithmic in n, compared to Theorem 7.45. However, the dependency on d in Theorem 7.47 is a high-degree polynomial. That is since we invoke the Euclidean case of Theorem 7.26/Theorem 7.37 so that coreset preserves the objective with respect to every k points in $\mathbb{R}^d$, and that requires an additional factor of d in the coreset size.

7.4.2 Dimensionality Reduction

In the case of k-means, we show how to apply the recent dimensionality reduction tools to effectively replace the dimension d by $\mathcal{O}(k/\varepsilon)$, thus making the algorithm from Theorem 7.47 linear in d too, and independent of d after the computation of the coreset. At the end, dimensionality reduction and our coreset construction effectively compress the instance to just $(k/\varepsilon \log n)^{\mathcal{O}(1)}$ real numbers, providing a stronger variant of Theorem 7.46.

In the proof of the theorem, we employ the notion and results on projection-cost preserving sketches, introduced in Section 4.1.

Theorem 7.48. *There is a randomized algorithm that given an instance P of (α,β)-Fair Clustering in $\mathbb{R}^d$ outputs a reduced weighted instance W of size $(k/\varepsilon \log n)^{\mathcal{O}(1)}$ in a low-dimensional space $\mathbb{R}^m$, where $m = O(k/\varepsilon)$. W.h.p. for any $\gamma \geq 1$, and for any set of k centers $\tilde{C}$ in $\mathbb{R}^m$ and a fair assignment ψ from W to $\tilde{C}$ such that $\operatorname{cost}(\psi) \leq \gamma \operatorname{faircost}(W)$, there exists a set of k centers C in $\mathbb{R}^d$ and a fair assignment $\varphi : P \to C$ with the cost at most $(1+\varepsilon)\gamma \operatorname{faircost}(P)$ that can be restored from ψ and $\tilde{C}$. Both constructing W from P and restoring (C,φ) from $(\tilde{C},\psi)$ takes time $\mathcal{O}(\Gamma k^3/\varepsilon^2 nd \log n)$.*

Proof. Fix a value $0 < \varepsilon_0 < \varepsilon$ to be defined later. Represent the given points P as a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, where each row corresponds to a point. Set $s = \lceil k/\varepsilon_0 \rceil$ and run the algorithm given by Theorem 4.8 on the matrix $\mathbf{A}$ and the parameter s to obtain a matrix $\mathbf{S} \in \mathbb{R}^{d\times s}$. By Markov inequality, it holds with probability at least $1 - \frac{1+\varepsilon_0}{1+2\varepsilon_0} = \Omega(\varepsilon_0)$ that

$$||\mathbf{A} - \mathbf{A}\mathbf{S}\mathbf{S}^T||_F^2 \leq (1+2\varepsilon_0)||\mathbf{A} - \mathbf{A}_\mathbf{s}||_F^2.$$

By invoking Theorem 4.8 $\mathcal{O}(\varepsilon_0^{-1} \log n)$ times and picking $\mathbf{S}$ with the smallest value of $||\mathbf{A} - \mathbf{A}\mathbf{S}\mathbf{S}^T||_F^2$, we achieve that the bound above holds with high probability. Then, the sketch $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{S}$ is a projection-cost preserving sketch, i.e. it holds that

$$||\mathbf{A} - \mathbf{M}\mathbf{A}||_F^2 \leq ||\tilde{\mathbf{A}} - \mathbf{M}\tilde{\mathbf{A}}||_F^2 + c \leq (1+3\varepsilon_0)||\mathbf{A} - \mathbf{M}\mathbf{A}||_F^2,$$

for any rank k orthogonal projection matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$ and some constant c independent of $\mathbf{M}$. Consider the corresponding to $\tilde{\mathbf{A}}$ set of points $\tilde{P}$ in $\mathbb{R}^m$. Recall that any k-means clustering of the rows of $\mathbf{A}$ may be represented by a particular orthogonal

projection matrix $\mathbf{M}$, such that the cost of the clustering is equal to $||\mathbf{A} - \mathbf{MA}||_F^2$. Thus, for any clustering $C_1, \ldots, C_k$ of P and the corresponding clustering $\tilde{C}_1, \ldots, \tilde{C}_k$ of $\tilde{P}$, it holds that

$$\operatorname{cost}(C_1, \ldots, C_k) \leq \operatorname{cost}(\tilde{C}_1, \ldots, \tilde{C}_k) + c \leq (1 + 3\varepsilon_0) \operatorname{cost}(C_1, \ldots, C_k). \tag{7.8}$$

In particular, if we equip $\tilde{P}$ with the same l groups as P, (7.8) holds for any fair clustering.

We run the algorithm given by Theorem 7.46 on $\tilde{P}$ to obtain a reduced weighted instance W in $\mathbb{R}^m$, using the error parameter ε_0. Now, consider a set of k centers $\tilde{C}$ in $\mathbb{R}^m$, and an assignment ψ from W to $\tilde{C}$ that has the cost of at most $\gamma \operatorname{faircost}(W)$. By Theorem 7.46, ψ can be lifted to a fair assignment $\tilde{\varphi}$ from $\tilde{P}$ to $\tilde{C}$ with the cost of at most $(1+\varepsilon_0)\gamma \operatorname{faircost}(\tilde{P})$. Consider the clustering $\{\tilde{C}_1, \ldots, \tilde{C}_k\}$ of $\tilde{P}$ that corresponds to the assignment $\tilde{\varphi}$. Consider also the clustering $\{C_1, \ldots, C_k\}$ of P that corresponds to $\{\tilde{C}_1, \ldots, \tilde{C}_k\}$, i.e. for each $i \in [k]$, C_i contains exactly the preimages of points in $\tilde{C}_i$ under sketching. The resulting set of centers $C = \{c_1, \ldots, c_k\}$ is the set of means of the clusters $C_1, \ldots, C_k$, that is, for each $i \in [k]$, $c_i = \mu(C_i)$. The resulting assignment φ sends C_i to c_i, for each $i \in [k]$. Clearly, φ is a fair assignment since it clusters together exactly the same points as $\tilde{\varphi}$, and $\tilde{\varphi}$ is a fair assignment by Theorem 7.46. Now we bound the cost of φ, by (7.8),

$$\operatorname{cost}(\varphi) = \operatorname{cost}(C_1, \ldots, C_k) \leq \operatorname{cost}(\tilde{C}_1, \ldots, \tilde{C}_k) + c \leq (1 + \varepsilon_0)\gamma \operatorname{faircost}(\tilde{P}) + c.$$

To bound $\operatorname{faircost}(\tilde{P})$ in terms of $\operatorname{faircost}(P)$, consider an optimal clustering $C_1^*, \ldots, C_k^*$ of P, and the corresponding clustering $\tilde{C_1}^*, \ldots, \tilde{C_k}^*$ of $\tilde{P}$. By (7.8),

$$\operatorname{faircost}(\tilde{P}) + c \leq \operatorname{cost}(\tilde{C_1}^*, \ldots, \tilde{C_k}^*) + c \leq (1+\varepsilon_0) \operatorname{cost}(C_1^*, \ldots, C_k^*) = (1+\varepsilon_0) \operatorname{faircost}(P).$$

Combining it with the earlier bound on $\operatorname{cost}(\varphi)$, we obtain

$$\operatorname{cost}(\varphi) \leq (1 + \varepsilon_0)^2 \gamma \operatorname{faircost}(P).$$

Finally, setting ε_0 such that $(1 + \varepsilon_0)^2 \leq (1 + \varepsilon)$ shows that φ satisfies the statement of the theorem.

The running time of the algorithm reducing P to W is $\mathcal{O}((k/\varepsilon^2)nd \log n)$ from Proposition 4.7 and $\mathcal{O}(\Gamma k^3/\varepsilon^2 nd \log n)$ from Theorem 7.46. The algorithm computing (C, φ) from $(\tilde{C}, \psi)$ runs in time $\mathcal{O}(\Gamma k^3/\varepsilon^2 nd \log n)$ by Theorem 7.46, plus an additional $\mathcal{O}(ndk)$ time required to compute ϕ and C from $\tilde{\varphi}$. Clearly, $\mathcal{O}(\Gamma k^3/\varepsilon^2 nd \log n)$ dominates the total running time. □

As Theorem 7.46, Theorem 7.48 allows to speed up any approximate algorithm for weighted (α, β)-Fair k-means by running it on the small-sized coreset in the low-dimensional space instead of the original points. In particular, we obtain an analogue of Theorem 7.47.

Theorem 7.49. *There is a randomized algorithm that given an instance P of (α,β)-Fair k-meanss and a parameter $0<\varepsilon\leq 1$ outputs a set of k centers C and a fair assignment $\varphi: P\to C$ such that $\mathrm{cost}(\varphi)\leq(1+\varepsilon)\,\mathrm{faircost}(P)$ with high probability. The running time of the algorithm is*

$$O(\Gamma k^3/\varepsilon^2 nd\log n)+2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(k\Gamma)^{\mathcal{O}(k\Gamma)}(\log n)^{\mathcal{O}(1)}.$$

Proof. The proof is identical to the proof of Theorem 7.47, the only difference is that Theorem 7.48 is used to reduce the instance, instead of Theorem 7.46. □

The benefit of the algorithm in Theorem 7.49 compared to Theorem 7.45 is that only the "simple" steps like sketching, sampling the coreset, and running the flow to restore the assignment, are applied to the "big" original data. On the other hand, the "heavy" part of the algorithm that has an exponential dependency on the parameters, deals exclusively with the compressed instance, with the size independent of the dimension d, and polylogarithmic in the number of points n. It might be said that the combination of the dimensionality reduction and our coreset construction in the proof of Theorem 7.48 obtains a coreset of size $\mathcal{O}((k\log n/\varepsilon)^{\mathcal{O}(1)})$ for fair k-means in the Euclidean case. However, since after reducing the dimension the points lie in a different low-dimensional space, our definition of a universal coreset could not be applied to the coreset with respect to the original points. Therefore we do not state Theorem 7.48 as a coreset result, but rather as a reduction procedure.

Finally, note that we only implement the dimensionality reduction for k-means, since for k-median the reduction techniques are more limiting. In particular, the correspondence between clusterings and particular orthogonal projection operators does not hold. It is an open question whether it is possible to achieve the analogue of Theorem 7.48 for k-median.

7.5 $(3+\varepsilon)$- and $(9+\varepsilon)$-Approximations in General Metric

In this section, we show a $(3+\varepsilon)$-approximation algorithm for fair k-median in general metric, and $(9+\varepsilon)$-approximation for fair k-means in general metric. After computing the coreset by Theorem 7.26, the strategy is essentially identical to that used in [64] and [66]: from each of the clusters in an optimal solution on the coreset we guess the closest point to the center, called a *leader* of that cluster. We also guess a suitably discretized distance from each leader to the center of the corresponding cluster. Finally, selecting any center that has roughly the guessed distance to the leader provides us with a $(3+\varepsilon)$-approximation, in the case of k-median. Now we state formally the main result of the section.

Theorem 7.50. *For any $1 \geq \varepsilon > 0$, there exists a $(3+\varepsilon)$-approximation algorithm for (α, β)-Fair k-median, and $(9+\varepsilon)$-approximation algorithm for (α, β)-Fair k-means. Both algorithms run in time*

$$(k\Gamma)^{\mathcal{O}(k\Gamma)}/\varepsilon^{\mathcal{O}(k)} \cdot n + \Gamma k^3/\varepsilon^2 n \log n.$$

Note that the distance guessing step of our algorithm requires that the *aspect ratio* of the instance is bounded by a polynomial in n, where the aspect ratio is the ratio of the maximum distance between the points to the minimum distance. As opposed to the case of capacitated clustering studied in [66], achieving polynomial aspect ratio is less straightforward for fair clustering, since there was no previously known true approximation algorithm for the general version of fair clustering. We refer the reader to the introduction for the discussion on assumptions and limitations in previous works. So, for the ease of presentation, we first prove Theorem 7.50 under the polynomial aspect ratio assumption, and later show how to achieve this assumption for any instance.

Claim 7.51. *The statement of Theorem 7.50 holds in the case when the aspect ratio of the input instance is bounded by $n^{\mathcal{O}(1)}$.*

Proof. For now, focus on the case of k-median. Fix a small positive number $\varepsilon_0 < \varepsilon$ that will be defined later. We start by computing a universal coreset W of size $\mathcal{O}(\Gamma(k \log n)^2 \varepsilon_0^{-3})$ by Theorem 7.26, applied with the error parameter ε_0. Then we try all possible sets of k points $l_1, \ldots, l_k$ out of the points in the coreset W. We also try all possible sets of k values $R_1, \ldots, R_k$, where each R_i ranges from the minimum distance between the points in the space to the maximum distance, taking values that are powers of $(1+\varepsilon_0)$ times the minimum distance. Thus, there are $|W|^k$ choices of $l_1, \ldots, l_k$, and $(\log n/\varepsilon_0)^{\mathcal{O}(k)}$ choices for $R_1, \ldots, R_k$, since the ratio of maximum distance to minimum distance is at most $n^{\mathcal{O}(1)}$. Now, for every choice of $l_1, \ldots, l_k$ and $R_1, \ldots, R_k$, we take a tuple of k centers $C = (c_1, \ldots, c_k)$ such that $\operatorname{dist}(l_i, c_i) \in [R_i, (1+\varepsilon_0)R_i)$ for every $i \in [k]$. If for $i \in [k]$ there are multiple choice of c_i, we take any one of them. If for some $i \in [k]$ there is no suitable c_i, we continue to the next choice of $l_1, \ldots, l_k$, and $R_1, \ldots, R_k$. After the centers are fixed, we run the assignment algorithm given by Theorem 7.39 on the coreset W and the centers C. Out of all considered tuples of centers, we select the one with the lowest cost of the assignment. Then we compute a fair assignment from P to these centers with the help of Lemma 7.42, and return the assignment and the centers. This concludes the algorithm.

For the proof of correctness, consider an optimal solution $C^* = \{c_1^*, \ldots, c_k^*\}$. Since W is a universal coreset of P, $\operatorname{faircost}(P, C^*) \leq (1+\varepsilon_0)\operatorname{faircost}(W, C^*)$. Consider an optimal assignment φ from W to C^* achieving the cost of $\operatorname{faircost}(W, C^*)$. Take l_1^*, $\ldots$, l_k^* such that l_i^* is the closest point to c_i^* among the points in $\varphi^{-1}(c_i^*)$, for each

$i \in [k]$. Here by $\varphi^{-1}(c_i^*)$ we mean the set of points in W such that φ sends positive weight from them to c_i^*. Take $R_1^*, \ldots, R_k^*$ such that for each $i \in [k]$, $R_i^* = (1+\varepsilon_0)^t m$ for a certain nonnegative integer t, where m is the minimum distance between the points, and $R_i^* \le \operatorname{dist}(l_i^*, c_i^*) < (1+\varepsilon_0)R_i^*$. At some point, the algorithm considers the choice of $l_1^*, \ldots, l_k^*$ and $R_1^*, \ldots, R_k^*$, take the tuple of centers $C = (c_1, \ldots, c_k)$ obtained by the algorithm at this iteration. We know that C exists since $(c_1^*, \ldots, c_k^*)$ is one of the possible choices for C. Consider the assignment ψ from W to C that behaves in the same way as φ: for each $i \in [k]$, ψ sends to c_i exactly the same weight from the same points in W, as φ does to c_i^*. Clearly, ψ is a fair assignment since the composition of each cluster is exactly the same as for φ. Now we bound the cost of ψ, for each point x in the coreset W and each center c_i such that a positive weight is assigned from x to c_i by ψ, it holds that

$$\begin{aligned}\operatorname{dist}(x, c_i) \le \operatorname{dist}(x, l_i^*) + \operatorname{dist}(l_i^*, c_i) &\le \operatorname{dist}(x, c_i^*) + \operatorname{dist}(c_i^*, l_i^*) + \operatorname{dist}(l_i^*, c_i) \\ &\le \operatorname{dist}(x, c_i^*) + (2+\varepsilon_0)\operatorname{dist}(c_i^*, l_i^*).\end{aligned}$$

The first two inequalities are by triangle inequality, and the last is since $\operatorname{dist}(l_i^*, c_i^*)$ is at least R_i^*, and $\operatorname{dist}(l_i^*, c_i)$ is at most $(1+\varepsilon_0)R_i^*$. Moreover, l_i^* is chosen in a way that $\operatorname{dist}(l_i^*, c_i^*) \le \operatorname{dist}(x, c_i^*)$, thus $\operatorname{dist}(x, c_i) \le (3+\varepsilon_0)\operatorname{dist}(x, c_i^*)$. Now, the total cost of ψ is

$$\begin{aligned}\sum_{x\in W}\sum_{i=1}^{k}\psi(x,c_i)\cdot\operatorname{dist}(x,c_i) &\le \sum_{x\in W}\sum_{i=1}^{k}(3+\varepsilon_0)\psi(x,c_i)\cdot\operatorname{dist}(x,c_i^*) \\ &= (3+\varepsilon_0)\sum_{x\in W}\sum_{i=1}^{k}\varphi(x,c_i^*)\cdot\operatorname{dist}(x,c_i^*) = (3+\varepsilon_0)\cdot\operatorname{faircost}(W,C^*) \\ &\le (3+\varepsilon_0)(1+\varepsilon_0)\operatorname{faircost}(P,C^*).\end{aligned}$$

Observe that $\operatorname{faircost}(P,C) \le \frac{1}{1-\varepsilon_0}\operatorname{faircost}(W,C)$ since W is a universal coreset. The assignment ψ is a particular fair assignment form W to C, thus its cost is at least $\operatorname{faircost}(W,C)$, and finally we get

$$\operatorname{faircost}(P,C) \le \frac{1}{1-\varepsilon_0}(3+\varepsilon_0)(1+\varepsilon_0)\operatorname{faircost}(P,C^*).$$

Recall that $\operatorname{faircost}(P,C^*)$ is the cost of an optimal solution, and that Lemma 7.42 returns a fair assignment of cost at most $(1+\varepsilon_0)\operatorname{faircost}(P,C)$. Thus setting ε_0 small enough such that $(3+\varepsilon_0)\frac{(1+\varepsilon_0)^2}{1-\varepsilon_0}$ is at most $(3+\varepsilon)$, provides the desired approximation.

For the running time, recall that first we compute the coreset in time $\mathcal{O}(n(k+l))$, and then we consider

$$|W|^k(\log n/\varepsilon_0)^{\mathcal{O}(k)} = (\Gamma(k\log n)^2/\varepsilon_0^3)^k(\log n/\varepsilon_0)^{\mathcal{O}(k)} = (k\Gamma\log n/\varepsilon_0)^{\mathcal{O}(k)}$$

tuples of k centers, and for each of them we run the assignment algorithm in time $(k\Gamma)^{\mathcal{O}(k\Gamma)}(\log n/\varepsilon_0)^{\mathcal{O}(1)}$. Thus, the total running time is

$$n(k+l) + (k\Gamma)^{\mathcal{O}(k\Gamma)}(\log n)^{\mathcal{O}(k)}/\varepsilon^{\mathcal{O}(k)}.$$

Note that for any constant $c > 0$, $(\log n)^{\mathcal{O}(k)}$ might be upper-bounded by $n^c + k^{O_c(k)}$, and we can bound the total running time required to find the best centers by $(k\Gamma)^{\mathcal{O}(k\Gamma)}/\varepsilon^{\mathcal{O}(k)} \cdot n$. Finally, an additional term of $\mathcal{O}(\Gamma k^3/\varepsilon^2 n \log n)$ is from Lemma 7.41.

Now to the case of fair k-means. The algorithm and analysis are essentially the same, up to a few minor details. For the coreset construction here we use Theorem 7.37 that constructs a universal coreset with respect to the k-means objective. The size of the coreset is still bounded by $\Gamma(k \log n/\varepsilon)^{\mathcal{O}(1)}$. Now the only difference is the bound on the cost of the assignment ψ. It becomes

$$\begin{aligned}\sum_{x\in W}\sum_{i=1}^{k}\psi(x,c_i)\cdot\operatorname{dist}(x,c_i)^2 &\le \sum_{x\in W}\sum_{i=1}^{k}(3+\varepsilon_0)^2\psi(x,c_i)\cdot\operatorname{dist}(x,c_i^*)^2\\ &= (3+\varepsilon_0)^2\sum_{x\in W}\sum_{i=1}^{k}\varphi(x,c_i^*)\cdot\operatorname{dist}(x,c_i^*)^2 = (3+\varepsilon_0)^2\cdot\operatorname{faircost}(W,C^*)\\ &\le (3+\varepsilon_0)^2(1+\varepsilon_0)\operatorname{faircost}(P,C^*).\end{aligned}$$

Analogously, we obtain

$$\operatorname{faircost}(P,C) \le \frac{1}{1-\varepsilon_0}(3+\varepsilon_0)^2(1+\varepsilon_0)\operatorname{faircost}(P,C^*),$$

and we set ε_0 small enough such that $(3+\varepsilon_1)^2\frac{(1+\varepsilon_0)^2}{1-\varepsilon_0} \le 9+\varepsilon$ to finally get the desired $(9+\varepsilon)$ approximation. □

7.5.1 Polynomial Aspect Ratio

We follow the standard trick to reduce the aspect ratio of the instance, see e.g. [66]. For that, we require an estimate of the cost of the optimal solution. So we start with showing a $\mathcal{O}(n)$-factor approximation algorithm for (α,β)-Fair Clustering. This algorithm combines the simple linear time $\mathcal{O}(n)$-approximation to the vanilla clustering problem, then the argument due to Bera et al. [23] that a set of centers that provides a good approximation w.r.t. vanilla clustering objective is also good enough for the purpose of fair clustering, and finally our assignment algorithm given by Theorem 7.39. We state a slight modification of the result of Bera et al. [23] first.

Proposition 7.52 (Lemma 3 in [23])**.** *Assume we are given a ρ-approximation algorithm $\mathcal{A}$ for k-median. Run $\mathcal{A}$ on the input set of points P and denote by C the*

returned set of centers. It holds that $\text{faircost}(P, C)$ *is at most* $(\rho+2)$ *times the cost of an optimal solution to* (α, β)*-Fair k-median on* P*. The same holds for k-means and* (α, β)*-Fair k-means, only the cost factor is* $(\rho+2)^2$*.*

Proof. The statement for k-median is exactly a special case of Lemma 3 in [23] where we only restrict to k-median and k-means, and the assignment algorithm has no violation of the constraints. For k-means, Lemma 3 in [23] holds for the same k-means and (α, β)-Fair k-means we consider in this paper, with the only difference that their objective function is the square root of the sum of squared distances. Thus, from their lemma we immediately get that the square root of the cost of the approximate solution is at most $(\rho+2)$ times the square root of the cost of the optimal solution. Squaring both sides provides the approximation factor of $(\rho+2)^2$. □

To achieve a linear-time algorithm, we are rather restricted in what kind of algorithm we can use to get the initial approximation for the vanilla clustering. Thus we use a simple $\mathcal{O}(n)$-approximation given by the classical k-center algorithm that is enough for our purposes. We also need to use the coreset construction as an intermediate step, so that computing the fair assignment takes time sublinear in n. Note that in this result, we do not aim to return the actual fair assignment, just the approximation to the cost.

Lemma 7.53. *There exists a* $\mathcal{O}(n)$*-factor approximation algorithm for computing the optimal cost in both* (α, β)*-Fair k-median and* (α, β)*-Fair k-means, with the running time of* $(k\Gamma)^{\mathcal{O}(k\Gamma)} \cdot n$*.*

Proof. For k-median, we start with computing the initial approximation using the min-max algorithm for k-center [102] in time $\mathcal{O}(nk)$. It is well-known that this gives a $\mathcal{O}(n)$-approximation of the k-median objective. For the obtained set C of k centers, by Proposition 7.52 it holds that $\text{faircost}(P, C)$ is at most $\mathcal{O}(n)$ times the optimal fair clustering cost of P. So it only remains to run the assignment algorithm. First, we compute a universal coreset W of P of size $\mathcal{O}(\Gamma(k \log n)^{\mathcal{O}(1)})$ by Theorem 7.26, using a constant error parameter. Then we run the assignment algorithm given by Theorem 7.39 on the weighted points W and the centers C. By definition of a universal coreset, the cost is changed by at most a constant factor, thus the cost of the fair assignment achieved by the algorithm is an $\mathcal{O}(n)$-approximation of the optimal solution on the original points P. The total running time of the algorithm is $\mathcal{O}(nk + n(k+l) + (k\Gamma)^{\mathcal{O}(k\Gamma)} \cdot (\log n)^{\mathcal{O}(1)})$.

For k-means, the only difference is that we run the k-center min-max algorithm using distances given by $d'(u, v) = \text{dist}(u, v)^2$ for all $u, v \in \mathcal{X}$. This is not a metric, but it holds that $d'(u, v) \leq 2(d'(u, w) + d'(w, v))$ for all $u, v, w \in \mathcal{X}$, since $\text{dist}(u, v)^2 \leq (\text{dist}(u, w) + \text{dist}(w, v))^2 \leq 2(\text{dist}(u, w)^2 + \text{dist}(w, v)^2)$. The min-max algorithm still obtains a $\mathcal{O}(1)$-approximation for the k-center objective with such a relaxed triangle inequality, and thus a $\mathcal{O}(n)$-approximation for the k-means objective with the original distances given by d. The rest is the same, but we invoke Theorem 7.37 to obtain the coreset. □

Finally, we show how to reduce an arbitrary instance of (α, β)-Fair Clustering to an equivalent one that has polynomial aspect ratio by modifying the distances.

Lemma 7.54. *Given an instance of* (α, β)-Fair Clustering *in the metric space with the distance function* dist*, we can construct a distance function* dist$'$ *such that the cost of any* n^{10}*-approximate solution changes by at most a factor of* $1 + 1/n$*. The distance function* dist$'$ *has polynomial aspect ratio. This requires the preprocessing time of* $(k\Gamma)^{\mathcal{O}(k\Gamma)} \cdot n$*.*

Proof. We state first how dist$'$ is obtained from dist. By Lemma 7.53 compute D that is a $\mathcal{O}(n)$-approximation of the cost of an optimal solution. Set $D_{max} = 2n^{10}D$ and $D_{min} = \alpha D/n^3$, for a sufficiently small constant α. For all the distances that are larger than D_{max}, set them to D_{max}. Then increase the distance between every pair of points by D_{min}. Clearly, the distances still form a metric.

No solution to the instance of (α, β)-Fair Clustering that has the cost of at most n^{10} times the optimal cost uses distances that are set to D_{max}, since $D_{max} = 2n^{10}D$ and D is at least the cost of the optimal solution. Thus the decreasing large distances to D_{max} does not affect the instance. Due to increasing by D_{min}, the cost of any solution is increased by at most a factor of $(1+1/n)$, since $n \cdot D_{min} = \alpha D/n^2$. This is at most $1/n$ times the cost of the optimal solution, since D is a $\mathcal{O}(n)$-approximation. Now the aspect ratio is at most $D_{max}/D_{min} = O(n^{13})$.

Note that running Lemma 7.53 incurs the preprocessing time of $(k\Gamma)^{\mathcal{O}(k\Gamma)} \cdot n$. After D_{max} and D_{min} are computed, the distance oracle for dist$'$ is obtained from the distance oracle for dist by an extra constant time per query. □

Finally, we prove Theorem 7.50 by combining Lemma 7.53 and Claim 7.51.

Proof of Theorem 7.50. Apply Lemma 7.53 to obtain the new instance of (α, β)-Fair Clustering with polynomial aspect ratio. For every solution that is $3 + \varepsilon$ (or $9 + \varepsilon$ in the case of k-means), the cost w.r.t. the new instance is at least the cost w.r.t. the old instance, and at most $(1 + 1/n)$ of that cost. Observe that $1/\varepsilon = o(n)$, otherwise all possible sets of centers can be trivially enumerated in time $n^k = (1/\varepsilon)^{\mathcal{O}(k)}$. Thus $(1 + 1/n) \leq (1 + \varepsilon/3)$, and invoking Claim 7.51 with the error parameter $\varepsilon/3$ finishes the proof. □

7.6 Algorithms for Other Clustering Problems

We note that the algorithms for fair clustering in general metrics suggest a generic algorithm for any clustering problem with constraints, such that the constraints can be represented by a set of matrices. Here we state this algorithm. Let $\mathcal{D}$ be the set of all possible distinct distances. Also, let $D_{\min}$ and $D_{\max}$ be the minimum and maximum distances in $\mathcal{D}$, respectively. This algorithm has the following steps.

· Compute a universal coreset W.

- For every pair of tuples $(l_1, \ldots, l_k)$ and $(R_1, \ldots, R_k)$ such that $l_i \in W$ for all i and $R_j \in \mathcal{D}$ for all j, do the following.
  - Select a set $C = \{c_1, \ldots, c_k\}$ of centers such that $c_i \in F$ and $\text{dist}(l_i, c_i) \in [R_i, (1+\varepsilon)R_i]$ for all i. If no such set C exists, probe the next choice.
  - Find an assignment of the points in P to the centers in C of the minimum cost that satisfies the respective clustering constraints (assignment problem).
- Return the set of centers and the assignment that minimizes the cost over all choices.

From the analysis for fair clustering, we have the following theorem.

Theorem 7.55. *Consider any clustering problem with constraint K, such that the constraint can be represented by a set of matrices, and suppose the aspect ratio $D_{\max}/D_{\min}$ is bounded by Δ for all instances. Moreover, suppose the universal coreset can be computed in $T_1(n, k, \ell)$ time and the assignment problem for the clustering problem can be solved in $T_2(n, k)$ time. Then one can obtain, w.h.p, a $(3+\varepsilon)$- (resp. $(9+\varepsilon)$-) approximation for k-median (resp. k-means) with constraint K in time $T_1(n, k, \ell) + (\varepsilon^{-1} k \Gamma \log(n+\Delta))^{\mathcal{O}(k)} \cdot T_2(n, k)$.*

From the above theorem, it is sufficient to (i) show that the aspect ratios of instances are bounded and (ii) design an efficient algorithm for the assignment problem, to obtain constant approximations for a clustering problem with constraints. We will use this theorem on various clustering problems.

For the Euclidean version of clustering problems with constraints, we have the following generic algorithm.

- Compute a universal coreset W.
- Apply the algorithm mentioned in Proposition 7.43 to find the list $\mathcal{L}$ of candidate sets of centers.
- For every set $C = \{c_1, \ldots, c_k\}$ of centers in $\mathcal{L}$, do the following.
  - Find an assignment of the points in W to the centers in C of the minimum cost that satisfies the respective clustering constraints (assignment problem).
- Let C' be the set of centers that minimizes the cost over all choices. Find an assignment of the points in P to the centers in C' of the minimum cost that satisfies the respective clustering constraints.

From the analysis for fair clustering and Proposition 7.43, we know that C' is an approximately optimal set of centers with constant probability. By repeating step 3 of the above algorithm $\mathcal{O}(\log n)$ times, we obtain this w.h.p. Hence, we have the following theorem.

Theorem 7.56. *Consider any Euclidean clustering problem with constraint K, such that the constraint can be represented by a set of matrices. Moreover, suppose the universal coreset can be computed in $T_1(n, k, \ell, d)$ time and the assignment problem for the clustering problem can be solved in $T_2(n, k)$ time. Then one can obtain, w.h.p, a $(1+\varepsilon)$-approximation for k-median (resp. k-means) with constraint K in time $T_1(n, k, \ell, d) + 2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})} \cdot (nd + \log n \cdot T_2(|W|, k)) + T_2(n, k)$.*

From the above theorem, it is sufficient to design an efficient algorithm for the assignment problem, to obtain constant approximations for a clustering problem with constraints. In the following, we will use this theorem on various clustering problems.

7.6.1 Lower-Bounded Clustering

In the lower-bounded clustering problem, we are given a lower bound parameter L and the size of each cluster must be at least L. In the Euclidean version of the problem the set of points $P \subset \mathbb{R}^d$ and the set of centers $F = \mathbb{R}^d$.

First, we note that the lower bound constraint can be represented by a set of $k \times 1$ column matrices such that each of the entries is at least L and at most n.

To apply Theorem 7.55, we need to show two things as mentioned before. To show the bounded aspect ratio, we can argue in the same way as we did for fair clustering. The assignment problem for lower-bounded clustering can be modeled as a minimum cost network flow problem that can be solved in polynomial time. Indeed, the modeling is similar to the one in the proof of Lemma 7.41. We construct a bipartite network where on one side we have the n points of P and on the other side the k centers of C and an additional node w. Source s is connected to all points of P. Sink t is connected to all centers through edges of capacity L. t is also connected to w through an edge of capacity $n - kL$. w is connected to all points through an edge. The cost of the edges between points and centers are their respective distances. The cost between a point p and w is $\operatorname{dist}(p, C)$. The idea is to route L flow to each center and $n - kL$ flow to w. This is equivalent to assigning at least L points to each center using the minimum cost and assigning the remaining points to their closest neighbor in C. We also scale the distances, as mentioned in the proof of Lemma 7.41. Then, the cost of the minimum cost flow in this network (with n flow) is a $(1+\varepsilon)$-approximation of the minimum assignment cost. Hence, by Proposition 7.40, the assignment problem in this case can be solved in time $nk^2\varepsilon^{-\mathcal{O}(1)} \log n$. Hence, the constant approximations follow for this problem in general metrics in time

$$\mathcal{O}(nk) + (\varepsilon^{-1}k \log n)^{\mathcal{O}(k)} nk^2 \varepsilon^{-\mathcal{O}(1)} \log n = (\varepsilon^{-1}k \log n)^{\mathcal{O}(k)} n = (k/\varepsilon)^{\mathcal{O}(k)} n^{\mathcal{O}(1)}.$$

From the above discussion, one can also apply Theorem 7.56. However, to solve the assignment problem on coreset, we do not want to spend $nk^2\varepsilon^{-\mathcal{O}(1)}\log n$ time. We would like to obtain an algorithm that is polynomial in the size of the coreset. We do the same as we did for fair clustering. We solve a mixed ILP that has $|W| \times k$ unconstrained variables and k integer variables. The construction is much easier compared to fair clustering. The running time is $k^{\mathcal{O}(k)}|W|^{\mathcal{O}(1)}$. Hence, we obtain $(1+\varepsilon)$-approximation for the Euclidean version in time

$$\begin{aligned}&2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(nd + \log n \cdot k^{\mathcal{O}(k)}(k\log n/\varepsilon)^{\mathcal{O}(1)}) + nk^2\varepsilon^{-\mathcal{O}(1)}\log n\\ =&2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(nd + (d\log n)^{\mathcal{O}(1)}) + nk^2\varepsilon^{-\mathcal{O}(1)}\log n.\end{aligned}$$

Theorem 7.57. *For any $\varepsilon > 0$, there exists a $(3+\varepsilon)$- and a $(9+\varepsilon)$-approximation algorithm for lower-bounded k-median and lower-bounded k-means, respectively, that runs in time $(k/\varepsilon)^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$. In $\mathbb{R}^d$, there are improved $(1+\varepsilon)$-approximation algorithms for both of the problems that run in time $2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(nd + (d\log n)^{\mathcal{O}(1)}) + nk^2\varepsilon^{-\mathcal{O}(1)}\log n$.*

7.6.2 Capacitated Clustering

Here we study the Euclidean capacitated clustering where the set of points $P \subset \mathbb{R}^d$ and the set of centers $F = \mathbb{R}^d$. Additionally, we are given a capacity parameter U and the size of each cluster must be at most U.

First, we note that the capacity constraint can be represented by a set of $k \times 1$ column matrices such that each of the entries is at least 0 and at most U.

Now, the assignment problem for capacitated clustering can be modeled as a minimum cost network flow problem that can be solved in polynomial time. Again, the modeling is similar to the one in the proof of Lemma 7.41. We construct a complete bipartite network where on one side we have the n points of P and on the other side the k centers of C. Source s is connected to all points of P. Sink t is connected to all centers with edges of capacity U. The cost of the edges between points and centers are their respective distances. We also scale the distances, as mentioned in the proof. Then, the cost of the minimum cost flow in this network (with n flow) is a $(1+\varepsilon)$-approximation of the minimum assignment cost. We note that one can assume that the aspect ratio of the input instance is bounded by $(n/\varepsilon)^{\mathcal{O}(1)}$. The assumption can be removed in the same way as in the case of fair clustering. Hence, by Proposition 7.40, the assignment problem in this case can be solved in time $nk^2\varepsilon^{-\mathcal{O}(1)}\log n$.

We solve the assignment problem on coreset in the same way mentioned for lower-bounded clustering. A $(1+\varepsilon)$-approximation follows for the Euclidean version in time

$$
\begin{aligned}
&2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(nd + \log n \cdot k^{\mathcal{O}(k)}(k \log n/\varepsilon)^{\mathcal{O}(1)}) + nk^2\varepsilon^{-\mathcal{O}(1)} \log n \\
=&2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(nd + (d \log n)^{\mathcal{O}(1)}) + nk^2\varepsilon^{-\mathcal{O}(1)} \log n.
\end{aligned}
$$

Theorem 7.58. *For any $\varepsilon > 0$, there exists $(1+\varepsilon)$-approximation algorithms for capacitated k-median and capacitated k-means that run in time $2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(nd + (d \log n)^{\mathcal{O}(1)}) + nk^2\varepsilon^{-\mathcal{O}(1)} \log n$.*

7.6.3 ℓ-Diversity Clustering

In the ℓ-Diversity clustering problem, $P = \cup_{i=1}^{\tilde{n}} P_i$ is a set of n colored points such that all points in P_i have the same color, and each cluster must have no more than a fraction $1/\ell$ (for some constant $\ell > 1$) of its points sharing the same color. Thus, for each cluster A and $i \in [\ell]$, $|A \cap P_i| \leq |A|/\ell$. We note that each point can have only one color. Ding and Xu [75] gave a $(1+\varepsilon)$-approximation for this problem in $\mathbb{R}^d$ with time complexity $\mathcal{O}(n^2(\log n)^{k+2}(t+1)^k d)$, where $t = \max_{1 \leq i \leq \tilde{n}} |P_i|$.

We note that ℓ-Diversity clustering is a special case of (α, β)-fair clustering without the lower bound constraints involving parameter β, and $\alpha_i = 1/\ell$ for all i. Thus, we obtain algorithms for this problem with bounds same as for (α, β)-fair clustering, including a $(1+\varepsilon)$-approximation that significantly improves the time complexity of the one in [75].

Theorem 7.59. *For any $\varepsilon > 0$, there exists a $(3+\varepsilon)$- and a $(9+\varepsilon)$-approximation algorithm for ℓ-Diversity k-median and ℓ-Diversity k-means, respectively, that runs in time $(k\ell)^{\mathcal{O}(k\ell)}/\varepsilon^{\mathcal{O}(k)} \cdot n + \ell k^3/\varepsilon^2 n \log n$. In $\mathbb{R}^d$, there are improved $(1+\varepsilon)$-approximation algorithms for both of the problems that run in time*

$$2^{\tilde{O}(k/\varepsilon^{\mathcal{O}(1)})}(k\ell)^{\mathcal{O}(k\ell)} nd \log n.$$

7.6.4 Chromatic Clustering

In chromatic clustering, again $P = \cup_{i=1}^{\tilde{n}} P_i$ is a set of n colored points such that all points in P_i have the same color, and each cluster contains at most one point from P_i for each i. Ding and Xu [75] obtained a linear time $(1+\varepsilon)$-approximation for this problem in $\mathbb{R}^d$. To the best of our knowledge the metric version of the problem was not studied before. Thus, we give the first constant approximation for this version.

First, we note that the chromatic constraint can be represented by a set of $k \times \ell$ matrices with 0/1 entries.

Ding and Wu [75] showed that the assignment problem for chromatic clustering can be modeled as a bipartite matching problem that can be solved in $\mathcal{O}(k^3 n)$ time. However, it is not clear how to bound the aspect ratio for this problem. Hence, the constant approximations follow for this problem in general metrics in time $\mathcal{O}(nk) + (\varepsilon^{-1}k\ell \log(n+\Delta))^{\mathcal{O}(k)} n$.

Theorem 7.60. *For any $\varepsilon > 0$, there exists a $(3+\varepsilon)$- and a $(9+\varepsilon)$-approximation algorithm for chromatic k-median and chromatic k-means, respectively, that runs in time $\mathcal{O}(nk) + (\varepsilon^{-1}k\ell \log(n+\Delta))^{\mathcal{O}(k)}n$.*

7.7 Streaming Universal Coreset

Here we describe a streaming algorithm for maintaining universal coreset for k-median. The algorithm can be trivially extended to k-means with a slightly larger space complexity. Our algorithm is based on the merge and reduce framework of Bentley and Saxe [22], which was first applied in the context of clustering in [2]. Indeed, in streaming setting this is a standard technique, which have been applied in many works [110, 53, 1]. We mainly follow the approach of Har-Peled and Mazumdar [110], which was further refined by Chen [53] for randomized coresets. In the following we describe how to maintain a small size coreset in each step. Let us refer to a universal coreset as an ε-coreset, where ε is the corresponding error parameter. Our approach is based on composability of coresets.

Lemma 7.61. *Suppose S_1 and S_2 are the ε-coresets of the points in P_1 and P_2, respectively. Then, $S_1 \cup S_2$ is an ε-coreset of the points in $P_1 \cup P_2$.*

The proof of this lemma follows by definition of universal coresets and can be found in [173]. The proof assumes that the coreset points have integer weights. We note that our construction can be slightly modified to obtain coreset with integer weights (e.g, see Chen's adaptation [53]). Next, we have an observation that again follows by the definition of coresets.

Observation 7.62. *Suppose S_1 is an ε-coreset of the points in S_2 and S_2 is an δ-coreset of the points in S_3. Then, S_1 is an $((1+\varepsilon)(1+\delta)-1)$-coreset of the points in S_3.*

Let λ be the confidence probability parameter for the coreset we want to construct. Suppose the points arrive in the order $p_1, p_2, \ldots$ and let $P = (p_1, \ldots, p_n)$ be the set of points arrived so far. We partition P into $t+1$ subsets $P_0, P_1, \ldots, P_t$ such that $|P_i| = 2^i T$, where $T = \lceil \ell k^2/\varepsilon^3 \rceil$.

Let $\rho_j = \varepsilon/(b\cdot(j+1)^2)$ for a sufficiently large constant b, and $1+\delta_j = \Pi_{i=0}^{j}(1+\rho_i)$ for $j = 1, \ldots, \lceil \log n \rceil$. It is not hard to verify that $1+\delta_j \leq 1+\varepsilon$ for all j. We maintain a δ_j-coreset Q_j for the points in P_j, where $Q_0 = P_0$. Thus, by composability of coresets, $\cup_{j\geq 0} Q_j$ is an ε-coreset for points in P.

When a new point p_m arrives, we add it to Q_0. If Q_0 contains fewer than T points, we are done. Otherwise, let $r \geq 1$ be the minimum index such that Q_r is empty. We compute a ρ_r-coreset Q'_r of $\cup_{j=0}^{r-1} Q_j$ with confidence parameter $\lambda_m = \lambda/m^2$ and set Q'_r to be Q_r. We also make all the sets Q_j empty for $1 \leq j \leq r-1$. It is not hard to verify that the total weight of the points in Q_r is $2^{r-1}T$.

Note that here we need to compute coresets of weighted points. We can trivially extend our coreset construction algorithm in the sequential setting to handle points with integer weights. For example, for a point p with weight w, now we treat it as the point p with w copies. Instead of using the algorithm of Indyk at the start, we use our algorithm in Section 7.4. Thus the algorithm can be implemented where the space complexity is linear in the number of points.

Next we claim that $Q = \cup_{i\geq 0} Q_i$ is an ε-coreset of the points received so far w.p. at least $1-\lambda$. First, note that Q_r is constructed by computing a ρ_r-coreset of $\cup_{j=0}^{r-1} Q_j$. By applying Observation 7.62 repetitively, Q_r is a $(\Pi_{i=0}^{r}(1+\rho_i)-1)$-coreset of the corresponding subset of the input points w.p. at least $1-\lambda/m^2$. Now for $m \geq T$, when p_m arrives, our computation fails w.p. at most λ/m^2. The failure probability over all iterations is at most $\sum_{m=T}^{n} \lambda/m^2 \leq \lambda$ for $T \geq 2$. As $\Pi_{i=0}^{r}(1+\rho_i) - 1 = 1 + \delta_r \leq 1 + \varepsilon$, the claim follows by composability.

Now, we bound the size of individual coreset Q_i. Note that $|Q_0| \leq T = \lceil \ell k^2/\varepsilon^3 \rceil$. Also for $i \geq 1$, Q_i is constructed for a subset of input points of size $2^{i-1}T$ and with error parameter $\rho_i = \varepsilon/(b \cdot (i+1)^2)$. Thus by Theorem 7.26, the size of Q_i is

$$\mathcal{O}(\ell k^2 \log(2^i + T)(\log(2^i + T) + d\log(1/\varepsilon)) \cdot i^6/\varepsilon^3) = \mathcal{O}(d\ell k^2 (\log n)^8/\varepsilon^4).$$

Hence, the total size of the coreset $Q = \cup_{i=0}^{\lceil \log n \rceil} Q_i$ is bounded by $\mathcal{O}(d\ell k^2 (\log n)^9/\varepsilon^4)$. In $\mathbb{R}^d$, we need $\mathcal{O}(d)$ space for storing each point, and thus we obtain the following theorem.

Theorem 7.63. *In one pass streaming model, a universal coreset for k-median clustering of size $\mathcal{O}(d^2 \ell k^2 (\log n)^9/\varepsilon^4)$ can be computed w.h.p.*

7.8 Conclusions and Open Questions

In this chapter, we studied the widely popular fair clustering problem with k-median and k-means objectives. Our universal coreset construction allows us to obtain the first coreset for fair clustering in general metric spaces. The coreset size is comparable to the best-known bound in the vanilla case. For Euclidean spaces, we obtain the first coreset for this problem whose size does not depend exponentially on the dimension. In the vanilla case, it is possible to construct coresets of size $(k/\varepsilon)^{\mathcal{O}(1)}$. Thus, an interesting open question is to remove the dependence on d and $(\log n)^{\mathcal{O}(1)}$ completely from our coreset size.

The new coreset construction helps to design improved FPT constant-approximations for a wide range of problems including fair clustering in general metrics and $(1+\varepsilon)$-approximations in the Euclidean metric. However, for fair clustering, it is not trivial to find an optimal solution on a coreset like in the case of other popular clustering problems. This is true, as the assignment problem is not easy to solve in this case. We give a novel algorithm for this problem that runs in time FPT parameterized by k and

Γ. We note that for (t, k)-fair clustering the factor of $(k\ell)^{\mathcal{O}(k\ell)}$ in the running time of our algorithms can be improved to only $(k\ell)^{\mathcal{O}(k)}$, as in this case the assignment problem can be solved in time FPT parameterized by only k. Designing a polynomial time constant-approximation for fair clustering still remains an open question. Our $(1 + \varepsilon)$-approximation algorithms in the Euclidean case run in near-linear time. It would be interesting to see if one can obtain similar $(1 + \varepsilon)$-approximation in linear time matching the bound of the vanilla case.

8

Sampling Approach: Clustering with Missing Entries

In this chapter, we consider a generalization of the fundamental Euclidean k-means clustering for data with incomplete or corrupted entries. It is a common occurrence in practical applications that some features of data can be missing or unspecified. Since missing pieces could significantly affect the information retrieved from the data, handling such data is a pervasive challenge. Various heuristic, greedy, convex optimization, statistical, or even ad hoc methods were proposed throughout the years in different practical domains to handle missing data [8, 185].

Gao, Langberg, and Schulman in [96] proposed the following geometric approach to the clustering of incomplete data. A d-dimensional data object that misses Δ entries corresponds to a Δ-dimensional affine subspace in $\mathbb{R}^d$. This subspace is parallel to coordinate axes corresponding to the missing coordinates. We call such affine subspaces Δ-*points*. With this notation, a regular point in $\mathbb{R}^d$ is a 0-point. The distance between a Δ-point x and a point y is naturally defined as the minimum distance between y and a point from x. In this setting, the classical k-means and other clustering problems like k-median and k-center, can be defined on a set of Δ-points. The only difference is that we minimize the corresponding objective function based on the distances between the center of the cluster and the Δ-points from the cluster. Note that in this setting the center of the cluster is a regular d-dimensional vector with no missing entries. Gao et al.'s geometric model has the following explanation: It provides the values of missing entries that are most suitable for clustering objective. In particular, under the assumption that the set of "complete" data objects is well-clustered, this approach yields the correct clustering.

From the computational perspective, the Δ-point clustering models are way more challenging than their vanilla clustering counterparts. Most of the clustering algorithms crucially exploit the fact that clustering occurs in a metric space. The major obstacle of using these algorithms for the more general clustering problem is that

the "distances" between Δ-points do not satisfy the triangle inequality. As Gao et al. [96] wrote: "*This problem defeats many existing algorithmic approaches for "clustering"-type tasks, and for good reason—the geometry seems, in a genuine sense, to be absent.*"

While the definition of clustering of Gao et al. [96] is applicable to k-center, k-means, and k-median versions of clustering, most of the work in this direction concentrated on k-center. (In k-center clustering the objective is to minimize the distance r such that every point is within distance r from at least one of the k centers.) Only very recently the first approximation algorithm for k-means Δ-point clustering was given by Marom and Feldman [155] for the special case of $\Delta = 1$. We discuss in details the literature relevant to our work in the next subsection.

The main result of this chapter is the following theorem, which is the first step in the study of the computational complexity of k-means Δ-point clustering beyond $\Delta \in \{0, 1\}$.

Theorem 8.1. *The problem of k-means clustering of Δ-points in $\mathbb{R}^d$ admits an $(1+\varepsilon)$-approximation algorithm with running time $2^{\mathcal{O}(\frac{\Delta^7 k^3}{\varepsilon}\log\frac{k\Delta}{\varepsilon})}n^2 d$.*

Related Work Naturally, Theorem 8.1 provides a generalization of the seminal result by Kumar, Sabharwal, and Sen [137] to clustering of Δ-points. Recall that there Kumar et al. gave a $(1+\varepsilon)$-approximation algorithm for Euclidean k-means that runs in time $2^{(k/\varepsilon)^{\mathcal{O}(1)}}nd$. If we consider Δ to be a parameter together with k, our algorithm is an FPT approximation scheme with the same kind of running time as their result, Moreover, the general shape of our algorithm stems from their approach as well, although clustering of Δ-points presents considerable challenges compared to regular points in $\mathbb{R}^d$. We discuss these connections in more detail later.

Clustering of Δ-points was defined by Gao, Langberg, and Schulman [96]. (They call Δ-points axis-parallel Δ-flats.) In [96] and the consecutive work [97], Gao et al. developed several constant factor approximation algorithms for k-center clustering of Δ-points and lines. Lee and Schulman [141] gave a $(1+\varepsilon)$-approximation algorithm for k-center Δ-point clustering that runs in time $2^{\mathcal{O}(\Delta k\log k(1+1/\varepsilon^2))}nd$. On the negative side, they show that even if one of k or Δ (but not both) is a fixed constant greater that 3, it is NP-hard to decide whether there is a k-center Δ-point clustering of value 0. This implies that there is no approximation algorithm running in time polynomial in $n+d+k$, respectively polynomial in $n+d+\Delta$, for any approximation factor for k-center as well as k-median and k-means clustering of Δ-points. Eiben et al. [81] provide a thorough study of different variants of k-center with incomplete information.

A number of results on k-means and k-median clustering of lines can be found in the literature. Ommer and Malik [163] studied k-median clustering of lines in $\mathbb{R}^3$. Their algorithm does not have any approximation guarantee and can run for unbounded time. Perets [168] gave an algorithm that in time $n(\log n/\varepsilon)^{\mathcal{O}(k)}d$ finds a

$(1+\varepsilon)$-approximate solution for k-median line clustering in $\mathbb{R}^2$. Finally, Marom and Feldman in [155] gave the first PTAS for k-means clustering of lines by providing an $(1+\varepsilon)$-approximation algorithm of running time $f(k,d,\varepsilon)n\log n$. The algorithm of Marom and Feldman follows from the construction of a coreset of size $dk^{\mathcal{O}(k)}\log n/\varepsilon^2$. Comparing Theorem 8.1 with the result of Marom and Feldman, since every 1-point is a line, their result implies a PTAS for k-means clustering of Δ-points for $\Delta = 1$. However, Theorem 8.1 implies PTAS only for axis-parallel lines. To the best of our knowledge, no approximation algorithm was known for k-means for $\Delta > 1$.

Overview of the Algorithm In order to describe our algorithm, let us recall roughly the argument of Kumar et al. [137] for the case when $\Delta = 0$ and $k = 2$. Let P denote the set of points in the instance, let (P_1, P_2) be an optimal partition of P such that $|P_1| \geq |P_2|$, and let $(\mathbf{c}_1, \mathbf{c}_2)$ be the optimal cluster centers for this partition. The algorithm starts by picking at random some $s = s(\varepsilon)$ points $S \subseteq P$. Because $|P_1| \geq |P_2|$, it means that with constant probability, all these points belong to P_1 and the center $\mathbf{c}'_1$ of S gives a good approximation of $\mathbf{c}_1$. Once this is achieved, the algorithm tries to sample inside P_2 in order to get an approximation of $\mathbf{c}_2$. Because $|P_1|$ can be very large compared to $|P_2|$, the algorithm needs to remove some elements of P from the sampling pool. What Kumar et al. show is that, if t denote the distance between $\mathbf{c}_1$ and $\mathbf{c}_2$, then the ball B of radius $t/4$ around $\mathbf{c}'_1$ contains only elements of P_1. Moreover, they show that either P_2 is large compared to $P_1 - B$ or the solution containing only one cluster with center $\mathbf{c}'_1$ is a good enough approximation. Therefore, by guessing an approximation of B, the algorithm is able to sample inside P_2 with constant probability and thus obtain a good estimate $\mathbf{c}'_2$ for $\mathbf{c}_2$.

Let us now explain some of the difficulties encountered while trying to generalize this argument to Δ-points. Let P denote an instance of 2-clustering with Δ-points and let (P_1, P_2) be an optimal clustering with centers $(\mathbf{c}_1, \mathbf{c}_2)$. The first problem we encounter is that sampling elements of P_1 might not give a good approximation for $\mathbf{c}_1$. Indeed, suppose, for example, that for almost all the points in P_1, the first coordinate value is missing. In that case, almost surely, a constant number of randomly sampled elements of P_1 will be such that for all of them the value of the first coordinate is missing, and we get no information about the value of $\mathbf{c}_1$ on this coordinate. However, we can show that the number of such "bad" coordinates is at most Δ. This means that we can obtain a good approximation of $\mathbf{c}_1$ on some set of coordinates I_1 such that $|I_1| \geq d - \Delta$. Let us call this approximation u_1. Moreover, a large portion of the elements of P_1 will have all their missing values outside of I_1. For these points, u_1 contains all the information necessary to decide whether they belong to P_1 or not. So the algorithm will then guess these points and remove them from the sampling pool (we will come back to how exactly this is done later). Afterward, either P_2 is large enough so that we can sample inside this set with good probability, or what remains of P_1 is still larger than P_2. In the last case, we can sample in P_1 to obtain information on the value of $\mathbf{c}_1$ outside of I_1. The first time we sample inside P_1

we get information about $d - \Delta$ coordinates, and with each of the subsequent set of samples, learn at least one new coordinate. Hence we only have to do $2(\Delta + 1)$ sampling steps to obtain an estimate of all the coordinates of $\mathbf{c}_1$ and $\mathbf{c}_2$.

However, the major problem that we face is how to find the elements of P_1 with missing values outside of I_1 to remove from the sampling pool. The triangular inequality does not hold for Δ-points, and in particular, if t is the distance between $\mathbf{c}_1$ and $\mathbf{c}_2$, it is not true that the ball of radius $t/4$ around $\mathbf{c}_1$ does not contain any point of P_2. What we are able to prove is that the above holds if we exclude a certain small set of coordinates from the computation. Namely, if $I_{1,2}$ is the set of indices obtained from $[d]$ by removing the Δ indices that maximize $|\mathbf{c}_1[r] - \mathbf{c}_2[r]|$ over $r \in [d]$, then for $t' = \text{dist}^{I_{1,2}}(\mathbf{c}_1, \mathbf{c}_2)$ (*i.e.*, the distance between $\mathbf{c}_1$ and $\mathbf{c}_2$ when considering only coordinates in $I_{1,2}$), no point of P_2 is at distance less than $t'/4$ from $\mathbf{c}_1$. Moreover, we can also show that either the ball of radius $t'/4$ around $\mathbf{c}_1$ removes enough points from the sampling pool, or the coordinates of $I_{1,2}$ are "useless", meaning that we can set the two centers to be equal on these coordinate and still obtain a good approximation. The main difficulty here is that we do not know the coordinates $I_{1,2}$ and guessing this set would add a factor $\binom{d}{\Delta}$ to the running time, which we can not afford. We will show how to deal with this problem in Section 8.2, which is the main technical part of our proof.

8.1 Notations and Preliminaries

As the problem of clustering Δ-points is considerably different from the regular Euclidean k-Means, we introduce a variety of notations that will help to present our algorithm in a concise manner. We list these chapter-specific definitions in this section, together with some simple observations about the structure of our problem.

Points with Missing Coordinates

As explained above, the goal of this chapter is to study clustering of points in $\mathbb{R}^d$ with missing entries in some coordinates. Let us denote the missing entry value by "?" and let $\mathbb{H}^d$ denote the set of elements of $\mathbb{R}^d$ where we allow some coordinates to take value "?". We say that a point $\mathbf{x} = (x[1], \ldots, x[d])$ in $\mathbb{H}^d$ is a Δ-point, if at most Δ of the coordinates $x[i]$ of $\mathbf{x}$ have value "?".

Definition 8.2 (**Domain of a point**). *For an element $\mathbf{x} \in \mathbb{H}^d$, we call the* domain *of $\mathbf{x}$, denoted by* $\text{Dom}(\mathbf{x})$*, the set of coordinates $i \in [d]$ such that $x[i] \neq ?$.*

Definition 8.3 (FD **and** PD **points**). *For a set S of elements of $\mathbb{H}^d$ and a set I of indices in $[d]$, corresponding to coordinates, let* $\text{FD}(S, I)$ *denote the set of points in S that are* fully defined on I, i.e. $\mathbf{x} \in S$ *such that* $\text{Dom}(\mathbf{x}) \subseteq I$. *Formally,*

$$\text{FD}(S, I) = \{\mathbf{x} \in S \mid \text{Dom}(\mathbf{x}) \subseteq I\}.$$

By $\mathrm{PD}(S, I)$*, we denote the set of points in* S *that are* partially defined on I*,* i.e. $\mathbf{x} \in S$ *such that* $\mathrm{Dom}(\mathbf{x}) \cap I \neq \emptyset$*. Formally,*

$$\mathrm{PD}(S, I) = \{\mathbf{x} \in S \mid \mathrm{Dom}(\mathbf{x}) \cap I \neq \emptyset\}.$$

With a slight abuse of notation, in all the definitions here and next that concern a particular set of indices $I \subset [d]$, we might use $i \in [d]$ instead of $\{i\}$, *i.e.*, $\mathrm{PD}(S, i) = \mathrm{PD}(S, \{i\})$.

For elements $\mathbf{x}, \mathbf{y} \in \mathbb{H}^d$ and a set of coordinates in $I \subseteq [d]$, we define

$$\mathrm{dist}^I(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i \in I} (\mathbf{x}[i] - \mathbf{y}[i])^2},$$

where by convention if either $y =?$ or $x =?$, then $(x - y)^2 = 0$. When $I = [d]$, we let $\mathrm{dist}(\mathbf{x}, \mathbf{y}) = \mathrm{dist}^I(\mathbf{x}, \mathbf{y})$. Note that $\mathrm{dist}(\mathbf{x}, \mathbf{y})$ corresponds to the standard Euclidean distance when $\mathbf{x}$ and $\mathbf{y}$ are elements of $\mathbb{R}^d$. For a set P of elements of $\mathbb{H}^d$ and a set I of coordinates, we denote by $\mathbf{c}^I(P)$ the "mean" of P on the coordinates of I. That is, $\mathbf{c}^I(P)$ is the element of $\mathbb{H}^d$ such that for every $i \in I$,

$$\mathbf{c}^I(P)_i = \begin{cases} ? & \text{if } \mathrm{PD}(P, i) \text{ is empty,} \\ \frac{\sum_{\mathbf{x} \in \mathrm{PD}(P,i)} \mathbf{x}[i]}{|\mathrm{PD}(P,i)|} & \text{otherwise.} \end{cases}$$

When I contains all elements of $[d]$, we let $\mathbf{c}(P) = \mathbf{c}^I(P)$. For an element $\mathbf{y} \in \mathbb{H}^d$, a set X of elements of $\mathbb{H}^d$ and a set I of coordinates in $[d]$, let us define

$$f_2^I(X, \mathbf{y}) = \sum_{\mathbf{x} \in X} (\mathrm{dist}^I(\mathbf{x}, \mathbf{y}))^2.$$

Note that if I_1 and I_2 are disjoint sets of coordinates, then $f_2^{I_2 \cup I_1}(X, \mathbf{y}) = f_2^{I_1}(X, \mathbf{y}) + f_2^{I_2}(X, \mathbf{y})$. For $I = [d]$, we write $f_2(X, \mathbf{y}) = f_2^I(X, \mathbf{y})$.

For a set $(P_i)_{i \in [k]}$ of subsets of $\mathbb{H}^d$ and a set of points $(\mathbf{c}_i)_{i \in [k]}$, we set

$$\mathrm{val}\big((P_i)_{i \in [k]}, (\mathbf{c}_i)_{i \in [k]}\big) = \sum_{i \in [k]} f_2(P_i, \mathbf{c}_i).$$

Lemma 8.4. *For every point* $\mathbf{x} \in \mathbb{H}^d$*, set of points* $P \subseteq \mathbb{H}^d$*, and set of coordinates* $I \subseteq [d]$*, it holds that*

$$f_2^I(P, \mathbf{x}) = f_2^I(P, \mathbf{c}(P)) + \sum_{i \in I} |\mathrm{PD}(P, i)| (\mathbf{x}[i] - \mathbf{c}(P)[i])^2.$$

In particular,

$$f_2^I(P, \mathbf{x}) \leq f_2^I(P, \mathbf{c}(P)) + |\mathrm{PD}(P, I)| \, \mathrm{dist}(\mathbf{x}, \mathbf{c}(P))^2.$$

Proof. For an index $i \in I$, we have

$$\begin{aligned} f_2^i(P, \mathbf{x}) &= \sum_{\mathbf{v} \in \mathrm{PD}(P,i)} (\mathbf{v}[i] - \mathbf{x}[i])^2 \\ &= \sum_{\mathbf{v} \in \mathrm{PD}(P,i)} (\mathbf{v}[i] - \mathbf{c}(P)[i] + \mathbf{c}(P)[i] - \mathbf{x}[i])^2 \\ &= \sum_{\mathbf{v} \in \mathrm{PD}(P,i)} (\mathbf{v}[i] - \mathbf{c}(P)[i])^2 + \sum_{\mathbf{v} \in \mathrm{PD}(P,i)} (\mathbf{x}[i] - \mathbf{c}(P)[i])^2, \end{aligned}$$

because $\sum_{\mathbf{v} \in \mathrm{PD}(P,i)} (\mathbf{v}[i] - \mathbf{c}(P)[i]) = 0$ by definition of $\mathbf{c}(P)$. This means that $f_2^i(P, \mathbf{x}) = f_2^i(P, \mathbf{c}^i(P)) + |\mathrm{PD}(P, i)|(\mathbf{x}[i] - \mathbf{c}(P)[i])^2$. We conclude by summing over all $i \in I$. □

k-Means Clustering of Δ-Points

Let us define the k-Means problem for Δ-points. Given an instance P of n Δ-points in $\mathbb{H}^d$, the task is to partition P into k sets $(P_1, \ldots, P_k)$, which we will refer to as *clusters*. A solution also consists of a set of centers $(\mathbf{c}_1, \ldots, \mathbf{c}_k)$ and the objective is to minimize $\sum_{i \in [k]} f_2(P_i, \mathbf{c}_i)$. Note that, by Lemma 8.4, for a given cluster P_i the optimal center is exactly $\mathbf{c}(P_i)$, and we can equivalently minimize the objective value $\sum_{i \in [k]} f_2(P_i, \mathbf{c}(P_i))$ over all partitions $(P_1, \ldots, P_k)$. Furthermore, note that if $\mathrm{Dom}(\mathbf{x}) = \emptyset$, then $\mathbf{x}$ always contributes zero to $\sum_{i \in [k]} f_2(P_i, \mathbf{c}_i)$, so we can assume that $\mathrm{Dom}(\mathbf{x}) \neq \emptyset$ for all $\mathbf{x} \in P$, and, consequently, $\Delta < d$.

From now on we fix an instance of the Δ-point k-Means problem, and denote by P the corresponding set of Δ-points in $\mathbb{H}^d$.

Partial clustering. Suppose $(P_1, \ldots, P_k)$ together with centers $(\mathbf{c}_1, \ldots, \mathbf{c}_k)$ is an optimal solution. As explained earlier, the goal of our algorithm is to discover the centers of the clusters step by step, while assigning some elements of P to some clusters. For this purpose we define the notion of *partial clustering*. We say that integers $(n_1, \ldots, n_k)$, sets $(H_1, \ldots, H_k)$, $(I_1, \ldots, I_k)$ and points $(\mathbf{u}_1, \ldots, \mathbf{u}_k)$ form a *partial clustering* if for every $i \in [k]$:

- H_i is a set of elements of P,
- n_i is an integer in $[\Delta + 1]$,
- I_i is a set of indices in $[d]$,
- if $n_i > 0$, then $|I_i| \geq d - \Delta + (n_i - 1)$,
- if $n_i = 0$, then I_i and H_i are empty, and
- $\mathbf{u}_i$ is a point of $\mathbb{H}^d$ such that $\mathrm{Dom}(\mathbf{u}_i) = I_i$.

Intuitively, for every $i \in [k]$, H_i is a set of points which are already assigned to the cluster i, $\mathbf{u}_i$ is a partially discovered center of the cluster, and I_i represents the coordinates where $\mathbf{u}_i$ is already specified. As we will incur some error each time we are performing a sampling step, the values n_i represent the number of sampling steps that has been done for guessing $\mathbf{u}_i$ on I_i, and the fact that $I_i \geq d - \Delta + (n_i - 1)$ is used to show that the number of sampling steps performed before reaching a point where $I_i = [d]$ for each i is just $\mathcal{O}(\Delta \cdot k)$. Let R denote the set $P - (\bigcup_{i\in[k]} H_i)$ of the points in P that are not yet assigned to a cluster.

For a partial clustering $\mathcal{P} = \{(n_i)_{i\in[k]}, (H_i)_{i\in[k]}, (I_i)_{i\in[k]}, (\mathbf{u}_i)_{i\in[k]}\}$, an *extension* is a partition $(P'_i)_{i\in[k]}$ of the elements of P such that for every $i \in [k]$, $H_i \subseteq P'_i$. We say that the points $(\mathbf{c}'_i)_{i\in[k]}$ are the centers *associated* with $(P'_i)_{i\in[k]}$

if for every $i \in [k]$, $\mathbf{c}'_i$ and $\mathbf{u}_i$ are equal on the coordinates of I_i and $\mathbf{c}'_i$ is equal to $\mathbf{c}^{[d]-I_i}(P'_i)$ on $[d] - I_i$. The *value* of an extension $(P'_i)_{i\in[k]}$ with associated centers $(\mathbf{c}'_i)_{i\in[k]}$, denoted $\mathrm{val}\big((P'_i)_{i\in[k]}\big)$, is equal to $\mathrm{val}\big((P'_i)_{i\in[k]}, (\mathbf{c}'_i)_{i\in[k]}\big)$. The *value* of a partial clustering $\mathcal{P}$, denoted $\mathsf{OPT}(\mathcal{P})$, is the minimum value of an extension $(P'_i)_{i\in[k]}$ of $\mathcal{P}$. We call the extension optimizing this value *optimal*.

Observation 8.5. *Let $\mathcal{P} = \{(n_i)_{i\in[k]}, (H_i)_{i\in[k]}, (I_i)_{i\in[k]}, (\mathbf{u}_i)_{i\in[k]}\}$ be a partial clustering and $\mathbf{x} \in R$ such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$ for all $i \in [k]$ and $f_2(\mathbf{x}, \mathbf{u}_j)$ is minimal among all the $f_2(\mathbf{x}, \mathbf{u}_i)$. The partial clustering obtained from $\mathcal{P}$ by putting $\mathbf{x}$ in the set H_j has the same value as $\mathcal{P}$.*

Proof. It follows from the fact that, for any extension with associated centers $(\mathbf{c}'_i)_{i\in[k]}$, $f_2(\mathbf{x}, \mathbf{c}'_i) = f_2(\mathbf{x}, \mathbf{u}_i)$ for every $i \in [k]$. □

Therefore from now on, we can assume that no point of $\mathbf{x} \in R$ is such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$ for all $i \in [k]$. The previous statement and the conditions on $|I_i|$ imply the following remark:

Observation 8.6. *If $\mathcal{P} = \{(n_i)_{i\in[k]}, (H_i)_{i\in[k]}, (I_i)_{i\in[k]}, (\mathbf{u}_i)_{i\in[k]}\}$ is a partial clustering such that $\sum_{i\in[k]} n_i = k(\Delta + 1)$, then $\bigcup_{i\in[k]} H_i = P$.*

Let $\mathcal{P} = \{(n_i)_{i\in[k]}, (H_i)_{i\in[k]}, (I_i)_{i\in[k]}, (\mathbf{u}_i)_{i\in[k]}\}$ be a partial clustering, and let $(P'_i)_{i\in[k]}$ with centers $(\mathbf{c}'_i)_{i\in[k]}$ be its optimal extension. Let us denote $R_i = R \cap P'_i$. The goal of the algorithm is to sample some of the elements of R in order to guess, for some $i \in [k]$, coordinates of $\mathbf{c}'_i$ which are not in I_i. To do so we need to make sure that our sampling avoids elements $\mathbf{x}$ of R_i such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$ ($\mathbf{x} \in \mathrm{FD}(R_i, I_i)$) as these elements do not provide any information about $[d]-I_i$. The goal of the following section will be to cluster some of the elements of $\mathrm{FD}(R_i, I_i)$ in order to make this set small compared to R. Note that we might need to consider an extension which is not an optimal one.

8.2 Finding a Proper Partial Clustering

This section is devoted to the proof of the following lemma:

Lemma 8.7. *Let $\mathcal{P} = \{(n_i)_{i\in[k]}, (H_i)_{i\in[k]}, (I_i)_{i\in[k]}, (\mathbf{u}_i)_{i\in[k]}\}$ be a partial clustering. For every constant $\alpha \in \mathbb{R}_{\geq 0}$, $0 < \alpha < 1$, there exists an algorithm running in time $\mathcal{O}(ndk)$, that with probability at least $(\frac{1}{3k^2\Delta\log(n)^k})$ either:*

- *Returns a partial clustering $\mathcal{P}' = \{(n'_i)_{i\in[k]}, (H'_i)_{i\in[k]}, (I'_i)_{i\in[k]}, (\mathbf{u}'_i)_{i\in[k]}\}$ with $\mathsf{OPT}(\mathcal{P}') \leq (1+\alpha)\mathsf{OPT}(\mathcal{P})$ and $\sum_{i\in[k]} n'_i > \sum_{i\in[k]} n_i$; or*
- *Finds a set B of elements of R such that there exists an extension $(P'_i)_{i\in[k]}$ of $\mathcal{P}$ with value at most $(1+\alpha)\mathsf{OPT}(\mathcal{P})$ and such that $B \subseteq \bigcup_{i\in[k]} \mathrm{FD}(P'_i, I_i)$ and for every $i \in [k]$, there is an index $j \in [k]$ such that $|\mathrm{PD}(P'_j \cap R, I_i - I_j)| \geq \frac{\alpha}{32\cdot 6^{(\Delta+1)k}}|\mathrm{FD}(P'_i \cap R, I_i) - B|$.*

The Lemma 8.7 will serve as the base for one step of our algorithm in Section 8.3. The basic idea behind our main algorithm is to iteratively extend the partial clustering, until we get a clustering of P. In each step it computes a partial clustering $\mathcal{P}' = \{(n'_i)_{i\in[k]}, (H'_i)_{i\in[k]}, (I'_i)_{i\in[k]}, (\mathbf{u}'_i)_{i\in[k]}\}$ with $\mathsf{OPT}(\mathcal{P}') \leq (1+\alpha)\mathsf{OPT}(\mathcal{P})$ and $\sum_{i\in[k]} n'_i > \sum_{i\in[k]} n_i$ with high enough probability. Observations 8.5 and 8.6 then allow us to conclude in at most $k(\Delta+1)$ steps. Note that the first case of the Lemma 8.7 is precisely what we want. On the other hand, the second case, together with the fact that there are at most k clusters and the pigeonhole principle, will allow us to show that there is an index $r \in [k]$ such that $|(P'_r \cap R) - \mathrm{FD}(P'_r \cap R, I_r)| \geq h(k, \Delta, \alpha)|R - B|$ for some function h depending only on k, Δ, and α and $(P'_r \cap R) - \mathrm{FD}(P'_r \cap R, I_r) \subseteq R - B$. Hence we can with sufficiently high probability, by sampling in $R - B$, obtain some points from $(P'_r \cap R) - \mathrm{FD}(P'_r \cap R, I_r)$ and a good approximation of the center $\mathbf{c}_r$ on some coordinate outside of I_r.

8.2.1 Overview of the Proof

Before presenting our quite technical proof in the next subsection, let us first explain some of the ideas and difficulties encountered. Let

$$\mathcal{P} = \{(n_i)_{i\in[k]}, (H_i)_{i\in[k]}, (I_i)_{i\in[k]}, (\mathbf{u}_i)_{i\in[k]}\}$$

be a partial clustering and $(P_i)_{i\in[k]}$ be an optimal extension with the associated centers $(\mathbf{c}'_i)_{i\in[k]}$. Let us denote $R_i = P_i \cap R$ for every $i \in [k]$.

Suppose first that $\Delta = 0$ and let $(\mathbf{c}_i)_{i\in[k]}$ be the centers of an optimal extension. Kumar et al. [137] proved a similar statement as Lemma 8.7, where the first condition is replaced by the statement that two centers can be equal. To do that, assuming that they have a good approximation $\mathbf{u}_i$ of one center $\mathbf{c}_i$, they consider the ball B with center $\mathbf{u}_i$ and radius $t/4$ where t is the minimum over all $j \in [k]$, $j \neq i$ of the distance of $\mathbf{c}_i$ to the other centers $\mathbf{c}_j$. Because $\mathbf{u}_i$ is a good approximation, B contains only elements of P_i. Moreover, they are able to show that either $|P_j|$ is large enough compared to $|P_i - B|$, or putting all the points of P_j in P_i gives a good

approximation. Unfortunately, this property is not true anymore for Δ-points with $\Delta > 0$.

First note that in the case when $\Delta > 0$, as opposed to Kumar et al. [137] where all approximate centers $\mathbf{u}_i$ are either in $\mathbb{R}^d$ or not set at all, we have approximate centers that are partially set. Now, if we have some two centers $\mathbf{u}_i$ and $\mathbf{u}_j$ and we want to extend $\mathbf{u}_j$ to some coordinates in $\mathrm{Dom}(\mathbf{u}_i) - \mathrm{Dom}(\mathbf{u}_j) = I_i - I_j$ then even if $t = \mathrm{dist}^{I_i - I_j}(\mathbf{u}_i, \mathbf{c}'_j)$, it might not be true that the ball with center $\mathbf{u}_i$ and diameter $t/4$ contains no elements of P_j which makes the previous argument considerably more difficult to make. To overcome this problem we will consider the coordinates r in $I_i - I_j$ where $\mathrm{dist}^r(\mathbf{u}_i, \mathbf{c}'_j)$ is large, separately one by one.

Let us now fix an index i for the rest of this subsection. For every $j \in [k]$, let $J_j = I_i - I_j$, let $i^j_1, \ldots, i^j_{|J_j|}$ be the coordinates in J_j, and let $d^j_r = \mathrm{dist}^{i^j_r}(\mathbf{u}_i, \mathbf{c}'_j)$ for all $r \in [|J_j|]$. Without loss of generality, we can assume that $d^j_1 \geq d^j_2 \geq \cdots \geq d^j_{|J_j|}$. We distinguish two cases depending on whether $|J_j| \geq \Delta + 1$ or not.

Case 1: $|J_j| \geq \Delta + 1$**.** Let us denote by J'_j the set $\{i^j_{\Delta+1}, \ldots, i^j_{|J_j|}\}$ and let $d^j = \mathrm{dist}^{J'_j}(\mathbf{u}_i, \mathbf{c}'_j)$. Note that in this case it follows from the definition of partial clustering that $I_j = \emptyset$, $n_j = 0$, and, consequently, $J_j = I_i$.

Lemma 8.8. *For every $j \in [k] \setminus \{i\}$ such that $I_j = \emptyset$ and every $\mathbf{x} \in R_j$ such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$, it holds that $\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) > d^j/4$.*

Proof. First observe that if $d^j = 0$, then the lemma trivially holds and we can assume for the rest of the proof that $d^j > 0$. Now, note that $|I_i - \mathrm{Dom}(\mathbf{x})| \leq \Delta$ and $|I_i - J'_j| = \Delta$. Because both sets ($\mathrm{Dom}(\mathbf{x})$ and J'_j) are subsets of I_i, we have $|\mathrm{Dom}(\mathbf{x}) - J'_j| \geq |J'_j - \mathrm{Dom}(\mathbf{x})|$. Moreover, by the definition of J'_j and because $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$ we have that $\mathrm{Dom}(\mathbf{x}) - J'_j \subseteq \{i^j_1, \ldots, i^j_\Delta\}$. Since $d^j_1, \ldots, d^j_\Delta$ are larger than any d^j_r for $r > \Delta$, we have that $\mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{u}_i, \mathbf{c}'_j) \geq d^j$.

For the sake of contradiction, let us assume for the remainder of the proof that $\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) \leq \frac{d^j}{4}$. Since $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$, we have $\mathrm{dist}(\mathbf{x}, \mathbf{c}'_i) = \mathrm{dist}(\mathbf{x}, \mathbf{u}_i)$ and $\mathrm{dist}(\mathbf{x}, \mathbf{c}'_i) \leq \frac{d^j}{4}$. Moreover, because $(P_i)_{i \in [k]}$ is optimal, we have that $\mathrm{dist}(\mathbf{x}, \mathbf{c}'_j) \leq \mathrm{dist}(\mathbf{x}, \mathbf{c}'_i) \leq \frac{d^j}{4}$. Finally, since $\mathbf{x}$, $\mathbf{u}_i$, and $\mathbf{c}'_j$ are points without any "?" on $\mathrm{Dom}(\mathbf{x})$, the triangle inequality implies that $\mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{u}_i, \mathbf{c}'_j) \leq \mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{u}_i, x) + \mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{x}, \mathbf{c}'_j) \leq 2\frac{d^j}{4}$, which contradicts $\mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{u}_i, \mathbf{c}'_j) \geq d^j$. □

Given the above, we show in the following lemma that the set of elements in $\mathrm{FD}(R, I_i)$ at distance at most $d^j/4$ from $\mathbf{u}_i$ is basically what is sufficient to include in the set B of elements in R for the index i to satisfy the second case of Lemma 8.7 with a caveat that if it is not the case, then we can actually set $\mathbf{u}_j$ to be the same as $\mathbf{u}_i$ on the coordinates of J'_j without introducing too large error.

Lemma 8.9. *For every constant $c > 0$ and every index $j \in [k] \setminus \{i\}$ such that $I_i - I_j > \Delta$, if B denotes the set of all elements in $\mathrm{FD}(R, I_i)$ at distance at most $d^j/4$ from $\mathbf{u}_i$, then:*

- *Either* $|\mathrm{PD}(R_j, J'_j) - B| \geq c|P_i - B|$;
- *or* $f_2^{J'_j}(P_j, \mathbf{u}_i) - f_2^{J'_j}(P_j, \mathbf{c}'_j) \leq 16c f_2(P_i, \mathbf{u}_i)$.

Proof. Recall that $I_i - I_j > \Delta$ implies that $I_j = \emptyset$ and by the definition of partial clustering we have $n_j = 0$, $H_j = \emptyset$ and consequently $P_j = R_j$. Suppose that $|\mathrm{PD}(R_j, J'_j) - B| < c|P_i - B|$ (otherwise we are done). We know that $f_2(P_i, \mathbf{u}_i) \geq |P_i - B|(d^j/4)^2$ as all the points of $P_i - B$ are at distance at least $(d^j/4)$ from $\mathbf{u}_i$. By Lemma 8.8, we have that

$$|\mathrm{PD}(R_j, J'_j)| = |\mathrm{PD}(R_j, J'_j) - B| \leq c|P_i - B|.$$

However, by Lemma 8.4,

$$f_2^{J'_j}(R_j, \mathbf{u}_i) - f_2^{J'_j}(R_j, \mathbf{c}'_j) \leq |\mathrm{PD}(R_j, J'_j)|(d^j)^2,$$

which implies that

$$f_2^{J'_j}(P_j, \mathbf{u}_i) - f_2^{J'_j}(P_j, \mathbf{c}'_j) = f_2^{J'_j}(R_j, \mathbf{u}_i) - f_2^{J'_j}(R_j, \mathbf{c}'_j) \leq 16c f_2(P_i, \mathbf{u}_i).$$

□

Now let j be the index that minimizes d^j among all indices j' in $[k] \setminus \{i\}$ for which $I_{j'} = \emptyset$ (i.e., an index j such that $d^j = \min_{j' \in [k]\setminus\{i\}, I_{j'} = \emptyset}\{d^{j'}\}$). Then the set $B = \{\mathbf{x} \in \mathrm{FD}(R, I_i) \mid \mathrm{dist}(\mathbf{x}, \mathbf{u}_i) \leq d^j/4\}$, *i.e.*, the set of all the elements in $\mathrm{FD}(R, I_i)$ at distance at most $d^j/4$ from $\mathbf{u}_i$, does not contain any element in $\bigcup_{j' \in [k]\setminus i, I_{j'} = \emptyset} R_{j'}$ (that is, any element of $R_{j'}$ for every $j' \in [k] \setminus \{i\}$ with $I_{j'} = \emptyset$). Furthermore, we have that

- either $|\mathrm{PD}(R_j, J_j) - B| \geq c|R_i - B|$,
- or $f_2^{J'_j}(R_j, \mathbf{u}_1) - f_2^{J'_j}(R_j, \mathbf{c}'_2) \leq 4\alpha\Delta f_2(R_1, \mathbf{u}_1)$.

When the first inequality occurs, this is the good case. Basically it means that $\mathrm{PD}(R_j, J'_j)$ is large enough so that sampling in $R - B$ avoids $\mathrm{FD}(R_i, I_i)$ with constant probability. Note that even though we do not know d^j, we can get a superset of $R - B$ of size at most $2|R - B|$ with probability $\frac{1}{\log n}$ by taking $\frac{n}{2^r}$ furthest points from $\mathbf{u}_i$ for some $r \in [\lfloor \log n \rfloor]$. To deal with the case when the second inequality holds, let us first show the following lemma, which will be useful throughout the paper.

Lemma 8.10. *Let $\mathcal{P} = \{(n_i)_{i\in[k]}, (H_i)_{i\in[k]}, (I_i)_{i\in[k]}, (\mathbf{u}_i)_{i\in[k]}\}$ be a partial clustering, $(P_i)_{i\in[k]}$ be an optimal extension of $\mathcal{P}$ with associated centers $(\mathbf{c}'_i)_{i\in[k]}$, and $C \in \mathbb{R}_{\geq 0}$. If there exist two indices $i, j \in [k]$ and $I' \subseteq I_i$ such that I_j is empty, $|I'| \geq d - \Delta$, and $f_2^{I'}(P_j, \mathbf{u}_i) - f_2^{I'}(P_j, \mathbf{c}'_j) \leq C f_2(P_i, \mathbf{u}_i)$, then the partial clustering $\mathcal{P}'$ obtained from $\mathcal{P}$ by setting $I_j = I'$, $\mathbf{u}_j$ the element of $\mathbb{H}^d$ equal to $\mathbf{u}_i$ on the coordinates of I' and "?" on the rest and $n_j := 1$ is a partial clustering of value at most $(1 + C)\mathsf{OPT}(\mathcal{P})$.*

Proof. Indeed, $(P_i)_{i\in[k]}$ is an extension of $\mathcal{P}'$. Let C_j be the point of $\mathbb{H}^d$ such that

$$(C_j)_r = \begin{cases} (\mathbf{u}_i)_r & \text{if } r \in I', \\ (\mathbf{c}'_j)_r & \text{otherwise,} \end{cases} \tag{8.1}$$

and let $C_s = \mathbf{c}'_s$ for $s \in [k] \setminus \{j\}$. Then

$$\begin{aligned} \mathrm{val}\big((P_i)_{i\in[k]}, (C_i)_{i\in[k]}\big) &\leq \mathrm{val}\big((P_i)_{i\in[k]}, (\mathbf{c}'_i)_{i\in[k]}\big) + C f_2(P_i, \mathbf{u}_i) \\ &\leq (1 + C)\mathsf{OPT}(\mathcal{P}). \end{aligned}$$

□

This means that the extension $\mathcal{P}'$ obtained by applying the previous Lemma with $I' = J'_j$ satisfies the first property of Lemma 8.7. A major problem is that we do not know J'_j and in the worst case there are d^Δ possibilities for J'_j, so guessing a feasible set I' for Lemma 8.10 is not an option here. We postpone dealing with this problem for later and switch our focus to the second case.

Case 2: $|J_j| \leq \Delta$. Let $r = |J_j|$. Recall that $i^j_1, \ldots, i^j_r$ denote the coordinates of $J_j = I_i - I_j$ and r is such that d^j_r is minimal. Let us denote by B_j the ball of elements of $\mathrm{FD}(R, I_i)$ at distance at most d^j_r from $\mathbf{u}_i$. We show now that we can adapt Lemmas 8.8 and 8.9 to this setting as well.

Lemma 8.11. *Suppose $j \in [k]$ is such that $I_i - I_j \leq \Delta$, if $\mathbf{x} \in \mathrm{PD}(R_j, I_i - I_j)$, then* $\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) \geq d^j_r/4$.

Proof. Let i' be the index of $I_i - I_j$ such that $i' \in \mathrm{Dom}(\mathbf{x})$. By the choice of i^j_r, we have that $\mathrm{dist}^{i'}(\mathbf{u}_i, \mathbf{c}'_j) > d^j_r$, and the triangle inequality allows us to conclude. □

Note that this time the set of elements of $\mathrm{FD}(R, I_i)$ at distance at most $d^j_r/4$ from $\mathbf{u}_i$ can contain some elements of R_j, but only those such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_j$, which are the "useless" ones for our sampling as they do not give any information for the coordinates outside of I_j. An analogous proof to the one of Lemma 8.9 gives the following result:

Lemma 8.12. *For every constant $c \in \mathbb{R}_{\geq 0}$ and $j \in [k]$ such that $r = |I_i - I_j| \leq \Delta$, if we denote by B the set of elements of $\mathrm{FD}(R, I_i)$ at distance at most $d^j_r/4$ from $\mathbf{u}_1$, then:*

- *Either* $|\mathrm{PD}(R_j, i_r^j) - B| \geq c|R_i - B|$;
- *or* $f_2^{i_r^j}(R_j, \mathbf{u}_1) - f_2^{i_r^j}(R_j, \mathbf{c}'_2) \leq 16cf_2(P_1, \mathbf{u}_1)$.

Again the first case is the good one, as B then provides a set such that drawing a sample from $R - B$ has constant probability to avoid $\mathrm{FD}(R_i, I_i)$. In the second case, however, it means that setting $I_j = I_i \cup i_r^j$ and $(\mathbf{u}_j)_r := (\mathbf{u}_i)_r$ gives a partial clustering of value at most $(1 + 16c)\mathsf{OPT}(\mathcal{P})$. In that case, we can guess the index i_r^j uniformly among $I_i - I_j$ and succeed with probability at least $1/\Delta$. Since we do this only when I_j is non-empty, and thus of size at least $d - \Delta$, the number of times we can do this for each $j \in [k]$ is at most Δ. In total, it means that we will perform this guessing only $k\Delta$ times, and it will only contribute $(\Delta)^{k\Delta}$ to the time complexity.

Therefore, the main problem we have to overcome is the case $|I_j| = 0$ and $f_2^{J'_j}(P_j, \mathbf{u}_i) - f_2^{J'_j}(P_j, \mathbf{c}'_j) \leq 16cf_2(P_i, \mathbf{u}_i)$ where we do not know J'_j. The main idea is the following. Recall that j is the index of $[k]\setminus\{i\}$ such that I_j is empty and d^j is minimal among all such indices. Suppose $f_2^{J_j}(P_j, \mathbf{u}_i) - f_2^{J'_j}(P_j, \mathbf{c}'_j) \leq 16cf_2(P_i, \mathbf{u}_i)$ and now consider the next smallest distance $d^{j'}$ and B' the set of elements of $\mathrm{FD}(R, I_i)$ at distance at most $d^{j'}/4$ from $\mathbf{u}_i$. If $|\mathrm{PD}(R_{j'}, J'_{j'}) - B'| \geq c|P_i - B'|$, then we almost have the set that we want, except that some elements of $\mathrm{FD}(P_j, I_i)$ can belong to B'. The idea will be to move these elements to P_i so that B' satisfies the desired properties for this new extension. While we are able to control the value of the new extension, as it increases by at most $16cf_2(P_i, \mathbf{u}_i)$, the fact that it stops being an optimal one creates some problems. Moreover, since we want to do this for each $i \in [k]$, we have to also impose some control over the centers associated with these extensions. This is the goal of the next subsection.

8.2.2 Extensions and Useless Sets of Coordinates

Let us fix some partial clustering $\mathcal{P} = \{(n_i)_{i\in[k]}, (H_i)_{i\in[k]}, (I_i)_{i\in[k]}, (\mathbf{u}_i)_{i\in[k]}\}$. By Observation 8.5, we can assume that for every index $i \in [k]$ such that $I_i - I_j$ is empty for every $j \in [k]$, no point $\mathbf{x}$ satisfying $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$ belongs to R. For the rest of this section, $\mathcal{P}$ will be fixed, and any extension $(P'_i)_{i\in[k]}$ will refer to an extension of $\mathcal{P}$ unless stated otherwise. We say that an extension is *safe* if for every $\mathbf{x} \in R$ and every $i, j \in [k]$ such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$ and $\mathrm{Dom}(\mathbf{x}) \subseteq I_j$, $\mathbf{x} \in P'_i$ implies that $\mathrm{dist}(\mathbf{x}, \mathbf{c}'_i) \leq \mathrm{dist}(\mathbf{x}, \mathbf{c}'_j)$.

As explained before, the hard case is when $|I_i - I_j| > \Delta$, which means that I_j is empty. To deal with this case, let us first introduce the following notion of useless sets of coordinates. Let $t \in \mathbb{R}_{\geq 0}$, $i, j \in [k]$ such that $I_j = \emptyset$ and $I_i \neq \emptyset$, let $(P'_i)_{i\in[k]}$ be an extension of $\mathcal{P}$, and finally set $R'_i = P'_i \cap R$. A set of indices $Z_{i,j} \subseteq I_i$ is a *t-useless set of coordinates* for $(P'_i)_{i\in[k]}$ if

- $Z_{i,j}$ is either empty or of size at least $d - \Delta$, and

· $f_2^{Z_{i,j}}(R'_j, \mathbf{u}_i) - f_2^{Z_{i,j}}(R'_j, \mathbf{c}'_j) \leq t \cdot \mathsf{OPT}(\mathcal{P})$.

To simplify the quantifications over $i, j \in [k]$ where $Z_{i,j}$ appears, we define t-useless sets of coordinates to be empty sets when $I_i = \emptyset$ or $I_j \neq \emptyset$. Intuitively, $Z_{i,j}$ corresponds to a set of coordinates such that setting $\mathbf{u}_j$ to be equal to $\mathbf{u}_i$ on these coordinates still gives a good partial clustering. The whole argument revolves around modifying the extension by "moving" some elements of some R'_i into some R'_j. However, by doing this we might change the values of $|f_2^{Z_{i,j}}(R'_j, \mathbf{u}_i) - f_2^{Z_{i,j}}(R'_j, \mathbf{c}'_j))|$, especially we might change the centers associated with the extension. The next lemma allows us to have some control of what happens for t-useless sets of coordinates when the changes are "small".

Lemma 8.13. *Let $(P'_i)_{i\in[k]}$ be an extension with associated centers $(\mathbf{c}'_i)_{i\in[k]}$ of value at most $(1+t_1)\cdot\mathsf{OPT}(\mathcal{P})$ and $(Z_{i,j})_{i,j\in[k]}$ be t-useless sets of coordinates for $(P'_i)_{i\in[k]}$. Let $(P^1_i)_{i\in[k]}$ be another extension of $\mathcal{P}$, and denote by X the set of points of P belonging to different clusters in $(P^1_i)_{i\in[k]}$ and $(P'_i)_{i\in[k]}$. For every $\mathbf{x} \in X$ such that $\mathbf{x} \in P'_r$ and $\mathbf{x} \in P^1_s$ for some $r, s \in [k]$, let $F(\mathbf{x}) = f_2(\mathbf{x}, \mathbf{c}'_s) - f_2(\mathbf{x}, \mathbf{c}'_r)$. If*

· *for every $\mathbf{x} \in X$ and $r \in [k]$ such that $\mathbf{x} \in P^1_r$, $\mathrm{Dom}(\mathbf{x}) \subseteq I_r$ and*

· *$\sum_{x\in X} F(\mathbf{x}) \leq t_2 \cdot \mathsf{OPT}(\mathcal{P})$,*

then $(Z_{i,j})_{i,j\in[k]}$ are $(t + t_1 + t_2)$-useless sets of coordinates for $(P^1_i)_{i\in[k]}$.

Proof. Let $(\mathbf{c}^1_i)_{i\in[k]}$ be the centers associated with the extension $(P^1_i)_{i\in[k]}$ and for each $r \in [k]$ let $R^1_r = R \cap P^1_r$. Note that by the definition of X, for every $r \in [k]$, every element $\mathbf{x}$ of $P^1_r - P'_r$ is such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_r$. In particular, if for some $r \in [k]$ it holds that $I_r = \emptyset$, then $P^1_r \subseteq P'_r$ (as we assume that no point $\mathbf{x}$ in P is such that $\mathrm{Dom}(\mathbf{x}) = \emptyset$).

Moreover, the t-useless sets of coordinates $Z_{i,j}$, for some $i, j \in [k]$, are only defined in the case that $I_j = \emptyset$. This implies that, for all $i, j \in [k]$ such that $Z_{i,j}$ is defined, it holds that:

$$f_2^{Z_{i,j}}(P^1_j, \mathbf{u}_i) \leq f_2^{Z_{i,j}}(P'_j, \mathbf{u}_i) \quad \text{and} \quad f_2^{[d]-Z_{i,j}}(P^1_j, \mathbf{u}_i) \leq f_2^{[d]-Z_{i,j}}(P'_j, \mathbf{u}_i). \tag{8.2}$$

Suppose now that for some $i, j \in [k]$ such that $Z_{i,j}$ is defined, it holds that $f^{Z_{i,j}}(P^1_j, \mathbf{c}^1_j) < f^{Z_{i,j}}(P'_j, \mathbf{c}'_j) - (t_1 + t_2)\mathsf{OPT}(\mathcal{P})$. We want to reach a contradiction by showing that in this case the value of $(P^1_i)_{i\in[k]}$ is smaller than $\mathsf{OPT}(\mathcal{P})$. For this purpose, let C_j be the element of $\mathbb{R}^d$ equal to $\mathbf{c}^1_j$ on the coordinates of $Z_{i,j}$ and to $\mathbf{c}'_j$ on the rest of coordinates, and let $C_r = \mathbf{c}'_r$ for $r \in [k] \setminus \{j\}$. By the definitions of X,

$F(\bullet)$, $f_2(\bullet,\bullet)$, and by (8.2), we have that

$$\sum_{r\in[k],r\neq j} f_2(P_r^1,C_r)+f_2^{[d]-Z_{i,j}}(P_j^1,C_j) \leq \sum_{r\in[k],r\neq j} f_2(P_r',C_r)+\sum_{x\in X}F(\mathbf{x})+f_2^{[d]-Z_{i,j}}(P_j',C_j). \quad (8.3)$$

Note that

$$\mathrm{val}\big((P_r^1)_{r\in[k]},(C_r)_{r\in[k]}\big)=\sum_{r\in[k],r\neq j} f_2(P_r^1,C_r)+f_2^{[d]-Z_{i,j}}(P_j^1,C_j)+f_2^{Z_{i,j}}(P_j^1,\mathbf{c}_j^1)$$

and

$$\mathrm{val}\big((P_r')_{r\in[k]},(\mathbf{c}_r')_{r\in[k]}\big)=\sum_{r\in[k],r\neq j} f_2(P_r',C_r)+f_2^{[d]-Z_{i,j}}(P_j',C_j)+f_2^{Z_{i,j}}(P_j',\mathbf{c}_j').$$

Hence the inequality (8.3) together with our assumption implies that

$$\begin{aligned}\mathrm{val}((P_r^1)_{r\in[k]},(C_r)_{r\in[k]}) &< \mathrm{val}((P_r')_{r\in[k]},(\mathbf{c}_r')_{r\in[k]})+t_2\cdot\mathsf{OPT}(\mathcal{P})-(t_1+t_2)\mathsf{OPT}(\mathcal{P})\\ &< \mathsf{OPT}(\mathcal{P}),\end{aligned}$$

a contradiction. This means that

$$\Big(f_2^{Z_{i,j}}(P_j',\mathbf{c}_j')-f_2^{Z_{i,j}}(P_j^1,\mathbf{c}_j')\Big)\leq(t_1+t_2)\mathsf{OPT}(\mathcal{P}),$$

and thus

$$\begin{aligned}f_2^{Z_{i,j}}(P_j^1,\mathbf{u}_i)-f_2^{Z_{i,j}}(P_j^1,\mathbf{c}_j') &\leq f_2^{Z_{i,j}}(P_j',\mathbf{u}_i)-f_2^{Z_{i,j}}(P_j^1,\mathbf{c}_j')\\ &\leq \Big(f_2^{Z_{i,j}}(P_j',\mathbf{u}_i)-f_2^{Z_{i,j}}(P_j',\mathbf{c}_j')\Big)\\ &\quad+\Big(f_2^{Z_{i,j}}(P_j',\mathbf{c}_j')-f_2^{Z_{i,j}}(P_j^1,\mathbf{c}_j')\Big)\\ &\leq (t+t_1+t_2)\cdot\mathsf{OPT}(\mathcal{P}).\end{aligned}$$

□

An important part of the proof of Lemmas 8.8 and 8.11 is that the extension we consider is optimal. This allowed us to say that picking balls of small radius around $\mathbf{u}_i$ will contain only elements of P_i'. Since we have to handle extensions which are not optimal, we have to require similar properties for them. This is the role of the next definition.

For an extension $(P_i')_{i\in[k]}$ with associated centers $(\mathbf{c}_j')$ and a t-useless set of coordinates $Z_{i,j}$ for some $i,j\in[k]$, we say that $Z_{i,j}$ is *compatible* with $(P_i')_{i\in[k]}$ if

· there is no element $\mathbf{x} \in P'_j$ such that $\mathrm{Dom}(\mathbf{x}) \not\subseteq I_j$ and $\mathrm{Dom}(\mathbf{x}) \subseteq Z_{i,j}$; and

· there is no element $\mathbf{x} \in P'_j$ such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$ and $\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) < \frac{\mathrm{dist}(\mathbf{x}, \mathbf{c}'_j)}{2}$.

Note that the factor $1/2$ here might seems strange, since if $\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) \leq \mathrm{dist}(\mathbf{x}, \mathbf{c}'_j)$, putting $\mathbf{x}$ in P'_i decreases the value of the extension. However, the problem with the definition without the factor $1/2$ would be the following: suppose we have an extension $(P'_i)_{i\in[k]}$, a set of t-useless sets of coordinates $Z_{i,j}$ and the goal is to make $Z_{i,j}$ compatible with $(P'_i)_{i\in[k]}$. To do so, suppose we move iteratively elements $\mathbf{x} \in P'_j$ such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$ and $\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) < \mathrm{dist}(\mathbf{x}, \mathbf{c}'_j)$ to P'_i (where $\mathbf{c}'_j$ is the updated center), until there are no such elements. The value of the extension can only decrease after this modification. However, we have to show that the $(Z_{i,j})$ are still t' useless for some bounded t'. As we have seen in Lemma 8.13, if we have some upper bound on the sum of $f_2(\mathbf{x}, \mathbf{c}'_r) - f_2(\mathbf{x}, \mathbf{c}'_j)$ over the elements $\mathbf{x}$ which are moved, then we achieve our goal. If $\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) < \mathrm{dist}(\mathbf{x}, \mathbf{c}'_j)$, it doesn't seem like we can have control over these values, however if $\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) < \frac{\mathrm{dist}(\mathbf{x}, \mathbf{c}'_j)}{2}$ every time we move an element $\mathbf{x}$, the value of the extension decrease by at least $f_2(\mathbf{x}, \mathbf{c}'_r)/2$. This means that if we start with an extension of value $(1 + t_1)\mathsf{OPT}(\mathcal{P})$, then the sum of $f_2(\mathbf{x}, \mathbf{c}'_r)$ over the elements $\mathbf{x}$ which are moved is bounded by $2t_1\mathsf{OPT}(\mathcal{P})$ and we can apply Lemma 8.13. This will be the main argument of the next lemma.

Lemma 8.14. *Suppose $(P'_i)_{i\in[k]}$ is a safe extension of value at most $(1+t_1)\mathsf{OPT}(\mathcal{P})$ with associated centers $(\mathbf{c}'_j)$ and $(Z_{i,j})_{i,j\in[k]}$ are compatible t_2-useless sets of coordinates for $(P'_i)_{i\in[k]}$. If there exist $i, j \in [k]$ with $I_i \neq \emptyset$ and $I_j = \emptyset$ as well as a set $I_{i,j}$ of coordinates in I_i such that $|I_{i,j}| \geq d - \Delta$, $I_{i,j} \not\subseteq Z_{i,j}$, and $f_2^{I_{i,j}}(P'_j, \mathbf{u}_i) - f_2^{I_{i,j}}(P'_j, \mathbf{c}'_j) \leq t_3 \cdot \mathsf{OPT}(\mathcal{P})$, then there exists a safe extension $(P^1_i)_{i\in[k]}$ of $\mathcal{P}$ with compatible $3t_1 + 2t_3 + t_2$-useless sets of coordinates $(Z^1_{i,j})_{i,j}$ such that the value of $(P^1_i)_{i\in[k]}$ is at most $(1 + t_1 + t_3)\mathsf{OPT}(\mathcal{P})$ and $\sum_{i,j\in[k]} |Z^1_{i,j}| > \sum_{i,j\in[k]} |Z'_{i,j}|$.*

Proof. Define $Z^1_{i,j} = Z_{i,j} \cup I_{i,j}$, and note that $|Z^1_{i,j}| > |Z_{i,j}|$ and $|Z^1_{i,j}| \geq |d - \Delta|$. Let $Z^1_{i',j'} = Z_{i',j'}$ for any other pair $i', j' \in [k]$. Note that $Z^1_{i',j'}$ is still a compatible t-useless set of coordinates for $(P'_i)_{i\in[k]}$. Because $f_2^{I_{i,j}}(P'_j, \mathbf{u}_i) - f_2^{I_{i,j}}(P'_j, \mathbf{c}'_j) \leq t_3 \cdot \mathsf{OPT}(\mathcal{P})$ and $Z_{i,j}$ is t_2-useless, we get that $Z^1_{i,j}$ is a $t_3 + t_2$ useless set of coordinates for $(P'_i)_{i\in[k]}$.

Consider now the extension $(P^1_r)_{r\in[k]}$ obtained from $(P'_r)_{r\in[k]}$ by putting in P'_i the set X_1 of elements $\mathbf{x} \in P'_j$ such that $\mathrm{Dom}(\mathbf{x}) \not\subseteq I_j$ and $\mathrm{Dom}(\mathbf{x}) \subseteq Z^1_{i,j}$. Let us define a sequence $(P^2_r)_{r\in[k]}, \ldots (P^q_r)_{r\in[k]}$ of extensions with the associated centers $(\mathbf{c}^2_r)_{r\in[k]}$, $\ldots$, $(\mathbf{c}^q_r)_{r\in[k]}$. For each $s \in [q-1]$, we obtain the extension $(P^{s+1}_r)_{r\in[k]}$ from $(P^s_r)_{r\in[k]}$ as follows. If there is an element in P^s_j such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_{i'}$ for some $i' \in [k]$ and $\mathrm{dist}(\mathbf{x}, \mathbf{u}_{i'}) \leq \frac{\mathrm{dist}(\mathbf{x}, \mathbf{c}^s_j)}{2}$ then we get $(P^{s+1}_r)_{r\in[k]}$ from $(P^s_r)_{r\in[k]}$ by moving $\mathbf{x}$ from P^s_j to $P^s_{i'}$. If there are multiple choices, we simply take any i' such that $\mathrm{dist}(\mathbf{x}, \mathbf{u}_{i'})$ is

minimal. Note that throughout this process, we are only removing elements from P_j^s to put it in another $P_{i'}^s$, which means that this process has to stop after at most $|P_j^1|$ steps. Let $(P_r'')_{r\in[k]}$ be the final extension of this process. Note as well that since we only add to $P_{i'}^s$ elements $\mathbf{x}$ such that $\mathrm{Dom}(\mathbf{x}) \in I_{i'}$, it means that $\mathbf{c}_{i'}^s = \mathbf{c}_{i'}'$ for every i' different from j. Denote by X_2 the set of all elements we moved between $(P_r^1)_{r\in[k]}$ and $(P_r'')_{r\in[k]}$, that is, $X_2 = P_j^1 - P_j''$. For $\mathbf{x} \in X_2$ such that $\mathbf{x} \in P_r''$, let $f(\mathbf{x}) = f_2(\mathbf{x}, \mathbf{c}_r')$.

Claim 8.15. $\sum_{x\in X_2} f(\mathbf{x}) \leq 2(t_1 + t_3)\mathsf{OPT}(\mathcal{P})$.

Proof. Since $f_2^{I_{i,j}}(P_j', \mathbf{u}_i) - f_2^{I_{i,j}}(P_j', \mathbf{c}_j') \leq t_3 \cdot \mathsf{OPT}(\mathcal{P})$, we have that $(P_r^1)_{r\in[k]}$ has value at most $(1 + t_1 + t_3)\mathsf{OPT}(\mathcal{P})$. However, suppose $\mathbf{x}$ is such that $\mathbf{x} \in P_j^s$ and $\mathbf{x} \in P_r^{s+1}$ for some $r \in [k] \setminus j$. It means that $\mathrm{dist}(\mathbf{x}, \mathbf{u}_r) \leq \frac{\mathrm{dist}(\mathbf{x}, \mathbf{c}(P_j^s))}{2}$ and:

$$\begin{aligned}\mathrm{val}((P_r^{s+1})_{r\in[k]}) &= \mathrm{val}((P_r^s)_{r\in[k]}) + (\mathrm{dist}(\mathbf{x}, \mathbf{u}_r)^2 - \mathrm{dist}(\mathbf{x}, \mathbf{c}(P_j^s))^2) \\ &\leq val((P_r^s)_{r\in[k]}) - \frac{3f(\mathbf{x})}{4}.\end{aligned}$$

Since $\mathrm{val}((P_i'')_{i\in[k]}) \geq \mathsf{OPT}(\mathcal{P})$, this ends the proof of the claim. □

Overall, if we set $X := X_1 \cup X_2$, then we can apply Lemma 8.13 to show that each $Z_{i',j'}^1$ is a $3t_1 + 2t_3 + t_2$-useless set of coordinates for $i', j' \in [k]$. Moreover, since we only move elements of P_j' and $\mathbf{c}_r' = \mathbf{c}_r''$ for every other $r \in [k] \setminus j$, the extension $(P_r'')_{r\in[k]}$ remains safe. Finally we can verify that the $(Z_{i',j'}^1)$ are compatible with $(P_r'')_{r\in[k]}$ as well. Indeed, for every $i' \in [k]$, $Z_{i',j}^1$ is compatible with $(P_r'')_{r\in[k]}$ by the definition of X. For every $j' \neq j$ and $i' \in [k]$, we have that $Z_{i',j'}^1 = Z_{i',j'}$ and thus since $Z_{i',j'}$ is compatible with $(P_i')_{i\in[k]}$ and the elements $\mathbf{x}$ of $P_{j'}'' - P_{j'}'$ are such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_{j'}$, there is no element $\mathbf{x} \in P_{j'}''$ such that $\mathrm{Dom}(\mathbf{x}) \not\subseteq I_{j'}$ and $\mathrm{Dom}(\mathbf{x}) \subseteq Z_{i',j'}^1$. Moreover, since $\mathbf{c}_{j'}'' = \mathbf{c}_{j'}'$, we have $\mathrm{dist}(\mathbf{x}, \mathbf{c}_{j'}'') = \mathrm{dist}(\mathbf{x}, \mathbf{c}_{j'}')$, which means that there is no element of P_j' such that $\mathrm{dist}(\mathbf{x}, \mathbf{u}_{i'}) \leq \frac{\mathrm{dist}(\mathbf{x}, \mathbf{c}_j'')}{2}$. Finally for every $\mathbf{x} \in P_{j'}'' - P_{j'}'$, $\mathrm{dist}(\mathbf{x}, \mathbf{c}_{j'}'') = \mathrm{dist}(\mathbf{x}, \mathbf{u}_{j'})$, which ends the proof, as we chose j' as the index such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_{j'}$ and $\mathrm{dist}(\mathbf{x}, \mathbf{u}_{j'})$ is minimal. □

The next lemma is the main technical part of this subsection.

Lemma 8.16. *Let $t \in \mathbb{R}_{\geq 0}$, let $(P_i')_{i\in[k]}$ be a safe extension of $\mathcal{P}$ with value at most $(1 + t)\mathsf{OPT}(\mathcal{P})$ and for every $i, j \in [k]$ let $Z_{i,j}$ be a compatible t-useless set of coordinates, such that $(Z_{i,j})$ are compatible with $(P_i')_{i\in[k]}$. One of the following holds:*

- *Either $Z_{i,j} = I_i$ for one pair $i, j \in [k]$ such that $I_i \neq \emptyset$ and $I_j = \emptyset$; or*
- *There exists $i, j \in [k]$ such that I_i and I_j are nonempty and $f_2^{i_1^j}(P_j', \mathbf{u}_i) - f_2^{i_1^j}(P_j', \mathbf{c}_j') \leq t \cdot f_2(R_i', \mathbf{u}_i)$ for some index $i_1^j \in I_i - I_j$; or*

- *There is an algorithm running in time* $\mathcal{O}(nkd)$ *that returns for every* $i \in [k]$ *a set* T'_i *such that* $T'_i \subseteq \mathrm{FD}(R'_i, I_i)$ *and* $|\mathrm{PD}(P'_j, I_i - I_j) - T'_i| \geq \frac{t}{32}|\mathrm{FD}(R'_i, I_i) - T'_i|$ *for some* $j \in [k]$ *with probability at least* $(\frac{1}{\log(n)})^k$*; or*

- *There exists an extension* $(P^1_i)_{i\in[k]}$ *of* $\mathcal{P}$ *with compatible* $(5t + t^2)$*-useless sets of coordinates* $(Z^1_{i,j})_{i,j\in[k]}$ *such that the value of* $(P^1_i)_{i\in[k]})$ *is at most* $(1 + \frac{3t+t^2}{2})\mathsf{OPT}(\mathcal{P})$ *and* $\sum_{i,j\in[k]} |Z^1_{i,j}| > \sum_{i,j\in[k]} |Z_{i,j}|$.

Proof. Let $c = t/32$. If $Z_{i,j} = I_i$ for one pair $i, j \in [k]$ such that $I_j = \emptyset$ and $I_i \neq \emptyset$, then nothing needs to be done, so let us assume this is not the case. Let F be the set of indices $i \in [k]$ such that I_i is empty and $G = [k] - F$.

Let i be some index of G. For any $j \in F$, let us denote $i_{j,min}$ the element of $I_i - Z_{i,j}$ such that $\mathrm{dist}^{i_{j,min}}(\mathbf{u}_i, \mathbf{c}(P'_j))$ is minimal. Let us now denote by $r_{1,j}, \ldots, r_{\Delta,j}$ the Δ coordinates of $I_i - i_{j,min}$ such that the $\mathrm{dist}^{r_{s,j}}(\mathbf{u}_i, \mathbf{c}(P'_j))$ for $s \in [\Delta]$ are the Δ maximal values among all $\mathrm{dist}^{r}(\mathbf{u}_i, \mathbf{c}(P'_j))$ for $r \in I_i - i_{j,min}$, and set $I_{i,j} = I_i - \{r_1, \ldots, r_\Delta\}$. Let $d^j = \mathrm{dist}^{I_{i,j}}(\mathbf{u}_i, \mathbf{c}(P'_j))$. The following claim is the analogue of Lemma 8.8, but using the fact that $Z_{i,j}$ is compatible with $(P'_r)_{r\in[k]}$ instead of the extension being optimal.

Claim 8.17. *For any* $\mathbf{x} \in P'_j$ *such that* $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$, $d(\mathbf{x}, \mathbf{u}_i) > d^j/4$.

Proof. Let $\mathbf{x}$ be an element of P'_j such that $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$ and note that $\mathrm{Dom}(\mathbf{x}) - I_i < \Delta$. Because $Z_{i,j}$ is compatible with $(P'_i)_{i\in[k]}$, we have that $\mathrm{Dom}(\mathbf{x}) \cap (I_i - Z_{i,j})$ is nonempty, and thus, by the choice of $I_{i,j}$, we have that $\mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{u}_i, \mathbf{c}'_j) \geq \mathrm{dist}^{I_{i,j}}(\mathbf{u}_i, \mathbf{c}'_j)$. However, because $Z_{i,j}$ is compatible with $(P'_i)_{i\in[k]}$ and $\mathrm{Dom}(\mathbf{x}) \subseteq I_i$, we have that $\mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{x}, \mathbf{c}'_j) \leq 2\,\mathrm{dist}(\mathbf{x}, \mathbf{u}_i)$. Therefore, if

$$\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) \leq \mathrm{dist}^{I_{i,j}}(\mathbf{u}_i, \mathbf{c}'_j)/4 \leq \mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{u}_i, \mathbf{c}'_j)/4,$$

then the triangle inequality gives us

$$\mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{u}_i, \mathbf{c}'_j) \leq \mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{u}_i, x) + \mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{x}, \mathbf{c}'_j) < \mathrm{dist}^{\mathrm{Dom}(\mathbf{x})}(\mathbf{u}_i, \mathbf{c}'_j),$$

where the triangle inequality applies because no coordinates of $\mathbf{x}, \mathbf{u}_i$ and $\mathbf{c}'_j$ have "?" on $\mathrm{Dom}(\mathbf{x})$. This is a contradiction and thus ends the proof. □

We can also show the following.

Claim 8.18. *For every constant* $c \in \mathbb{R}_{\geq 0}$ *and* $j \in F$*, the set* B_j *of elements of* $\mathrm{FD}(R, I_i)$ *at distance at most* $d^j/4$ *from* $\mathbf{u}_i$ *is such that one of the following properties is satisfied:*

- $|P'_j - B_j| \geq c|\mathrm{FD}(R'_i, I_i) - B_j|$*; or*

- $f_2^{I_{i,j}}(P'_j, \mathbf{u}_i) - f_2^{I_{i,j}}(P'_j, \mathbf{c}'_j) \leq 16c(1 + t)\mathsf{OPT}(\mathcal{P})$.

Proof. Indeed, suppose that $|P'_j - B_i| < c|\mathrm{FD}(R'_i, I_i) - B_j|$. We know that $f_2(P'_i, \mathbf{u}_i) \geq |\mathrm{FD}(R'_i, I_i) - B|(d^j/4)^2$ as all the points of $\mathrm{FD}(R'_i, I_i) - B_i$ are at distance at least $d^j/4$ from $\mathbf{u}_i$. Because of Claim 8.17, we have that $|P'_j - B_j| = |P'_j| \leq c|\mathrm{FD}(R'_i, I_i) - B_i|$. Moreover, since I_j is empty, it means that $\mathbf{c}'_j = \mathbf{c}(P'_j)$ and thus, by Lemma 8.4, we have that $f_2(P'_j, \mathbf{u}_i) - f_2(P'_j, \mathbf{c}'_j) \leq |P'_j| \operatorname{dist}(\mathbf{u}_i, \mathbf{c}'_j)^2$, which implies that $f_2(P'_j, \mathbf{u}_i) - f_2(P'_j, \mathbf{c}'_j) \leq 16c \cdot f_2(P'_i, \mathbf{u}_i) \leq 16c(1+t)\mathsf{OPT}(\mathcal{P})$. □

If $f_2^{I_{i,j}}(P'_j, \mathbf{u}_i) - f_2^{I_{i,j}}(P'_j, \mathbf{c}'_j) \leq 16c(1+t)\mathsf{OPT}(\mathcal{P})$, then Lemma 8.14 gives us the existence of an extension and some sets of coordinates satisfying the fourth property of the lemma. Therefore, from now on we assume that for every $j \in F$, $|P'_j - B_j| \geq c|\mathrm{FD}(R'_i, I_i) - B_j|$.

For any $j \in G$ different from i such that $I_i - I_j \neq \emptyset$, let us define $i^j_1, \ldots, i^j_{|I_i - I_j|}$ as the coordinates of $I_i - I_j$, and set $d^j_r = \mathrm{dist}^{i^j_r}(\mathbf{u}_i, \mathbf{c}'_j)$ for all $r \in [|I_i - I_j|]$. Without loss of generality, we can assume d^j_1 is minimum, and let $d^j = d^j_1$. Using analogous proofs as those of Lemmas 8.11 and 8.12, we can use the fact that $(P'_i)_{i \in [k]}$ is safe to prove the following two claims.

Claim 8.19. *For $j \in G$, if $\mathbf{x} \in \mathrm{PD}(P'_j, I_i - I_j)$, then $\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) \geq d^j/4$.*

Claim 8.20. *For every constant $c \in \mathbb{R}_{\geq 0}$ and $j \in G$, if we denote by B_j the set of elements of $\mathrm{FD}(R, I_i)$ at distance at most d^j from $\mathbf{u}_1$, then:*

- *Either $|\mathrm{PD}(P'_j, i^j_1) - B_j| \geq c|R'_i - B|$; or*
- *$f_2^{i^j_1}(P'_j, \mathbf{u}_i) - f_2^{i^j_1}(P'_j, \mathbf{c}'_j) \leq 16c f_2(P'_i, \mathbf{u}_i)$.*

Note that if there exists j such that $f_2^{i^j_1}(P'_j, \mathbf{u}_i) - f_2^{i^j_1}(P'_j, \mathbf{c}'_j) \leq 16c f_2(R'_i, \mathbf{u}_i)$, then the second property of the lemma is satisfied, so we can assume that $|\mathrm{PD}(P'_j, i^j_1) - B_j| \geq c|R'_i - B_j|$ for all $j \in G$.

Let us consider now the index $j \in G \cup F$ such that d^j is defined and minimal. Note that if no d^j is defined, it means that $I_i - I_j$ is empty for every j and thus because of Observation 8.5, we can assume that $\mathrm{FD}(R'_i, I_i)$ is empty. Let us denote by B_i the set of elements $\mathbf{x}$ of $\mathrm{FD}(R, I_i)$ at distance at most $d^j/4$ from $\mathbf{u}_i$ and such that, if $\mathrm{Dom}(\mathbf{x}) \subseteq I_j$ for some $j \in [k]$, then $\mathrm{dist}(\mathbf{x}, \mathbf{u}_i) \leq \mathrm{dist}(\mathbf{x}, \mathbf{u}_j)$. By combining Claims 8.17 and 8.19, the choice of j and the assumptions we made on the results of Claims 8.20 and 8.18 we get the following claim:

Claim 8.21. *B_i contains only elements of R'_i and $|\mathrm{PD}(P'_j, I_1 - I_j) - B_i| \geq c|\mathrm{FD}(R'_i, I_i) - B_i|$.*

Consider the integer r such $\frac{n}{2^r} \leq |\mathrm{FD}(R, I_i) - B_i| \leq \frac{n}{2^{r-1}}$, let H be the set of $\frac{n}{2^{r-1}}$ points of $\mathrm{FD}(R, I_i)$ that are the farthest away from $\mathbf{u}_1$ and let $B'_i = \mathrm{FD}(R, I_i) - H$. By the definition of H, $B'_i \subseteq B_i$ and $|R_i - B'| \leq 2|R_i - B|$ which means that

$|\mathrm{PD}(P'_j, I_i - I_j) - B'_i| \geq \frac{c|R_i - B'|}{2}$. Therefore, if the algorithm selects uniformly at random an integer in $[\log(n)]$, then with probability $1/\log(n)$ this integer is equal to r, and the algorithm is then able to find the set $T'_i := B'_i$. Note that once r is selected, the set B' can be found in $\mathcal{O}(nd)$ time. We finish the proof by repeating this for every i such that d^j can be defined. For the other indices i, as explained, $\mathrm{FD}(R'_i, I_i)$ is empty and thus $B'_i := \emptyset$ has the required properties. □

A very important remark here is that in the case where there is an algorithm running in time $\mathcal{O}(nkd)$ that returns for every $i \in [k]$ a set T'_i such that $T'_i \subseteq \mathrm{FD}(R'_i, I_i)$ and $|\mathrm{PD}(P'_j, I_i - I_j) - T'_i| \geq \frac{t}{32}|\mathrm{FD}(R'_i, I_i) - T'_i|$ for some $j \in [k]$ with probability at least $(\frac{1}{\log(n)})^k$, the algorithm does not need to know the extension $(P'_i)_{i\in[k]}$, as the only thing that matters are the distances of the elements of P to the point $\mathbf{u}_i$ that are given in $\mathcal{P}$.

8.2.3 Proof of Lemma 8.7

We are now ready to prove the main result of this section, Lemma 8.7.

Proof of Lemma 8.7. Let $t_1 = \frac{\alpha}{6^{(\Delta+1)k}}$ and note that $6^{(\Delta+1)k}t_1 = \alpha$. Let $(P^1_i)_{i\in[k]}$ be an optimal extension of $\mathcal{P}$ and $Z^1_{i,j} = \emptyset$ for every pair $i, j \in [k]$. Note that $(P^1_i)_{i\in[k]}$ is safe and the $(Z^1_{i,j})$ are compatible t_1-useless sets of coordinates. By applying Lemma 8.16 to $(P^1_i)_{i\in[k]}$ and $(Z^1_{i,j})_{i,j\in[k]}$, we have that, denoting $R^1_i = P^1_i \cap R$:

- Either $Z_{i,j} = I_i$ for one pair $i, j \in [k]$ such that $I_i \neq \emptyset$ and $I_j = \emptyset$; or
- There exists $i, j \in [k]$ such that I_i and I_j are non empty and $f_2^{i^j_1}(P^1_j, \mathbf{u}_i) - f_2^{i^j_1}(P^1_j, \mathbf{c}'_j) \leq t_1 \cdot f_2(R^1_i, \mathbf{u}_i)$ for some index $i^j_1 \in I_i - I_j$; or
- There is an algorithm that returns for every $i \in [k]$ a set T'_i such that $T'_i \subseteq \mathrm{FD}(R^1_i, I_i)$ and $|\mathrm{PD}(P^1_j, I_i - I_j) - T'_i| \geq \frac{t_1}{32}|\mathrm{FD}(R^1_i, I_i) - T'_i|$ for some $j \in [k]$ with probability at least $(\frac{1}{\log(n)})^k$; or
- There exists an extension $(P^2_i)_{i\in[k]}$ of $\mathcal{P}$ with some compatible $5t_1 + t_1^2$-useless set of coordinates $(Z^2_{i,j})_{i,j}$ such that the value of $(P^2_i)_{i\in[k]}$ is at most $(1 + \frac{3t_1+t_1^2}{2})\mathsf{OPT}(\mathcal{P}) \leq (1 + 5t_1 + t_1^2)\mathsf{OPT}(\mathcal{P})$ and $\sum_{i,j\in[k]} |Z^2_{i,j}| > \sum_{i,j\in[k]} |Z^1_{i,j}|$.

In the first case, the partial clustering $\mathcal{P}'$ obtained from $\mathcal{P}$ by setting $I_j := I_i$, $\mathbf{u}_j := \mathbf{u}_i$, and $n_j := n_i$, has value at most

$$\mathsf{OPT}(\mathcal{P}) + \left(f_2^{Z_{i,j}}(P'_j, \mathbf{u}_i) - f_2^{i^j_1}(P'_j, \mathbf{c}'_j)\right) \leq (1 + t_1)\mathsf{OPT}(\mathcal{P})$$

by definition of t_1-useless sets of coordinates, and therefore $\mathcal{P}'$ satisfies the first property of Lemma 8.7. In the second case, the partial clustering $\mathcal{P}'$ obtained from

$\mathcal{P}$ by setting $I_j := I_j \cup i_1^j$, $(\mathbf{u}_j)_{i_1^j} := (\mathbf{u}_i)_{i_1^j}$ and $n_j := n_j + 1$ is a partial clustering of value at most

$$\mathsf{OPT}(\mathcal{P}) + \left(f_2^{i_1^j}(P'_j, \mathbf{u}_i) - f_2^{i_1^j}(P'_j, \mathbf{c}'_j)\right) \leq (1 + t_1)\mathsf{OPT}(\mathcal{P}),$$

and therefore $\mathcal{P}'$ satisfies the first property of Lemma 8.7. In the third case, then the set $B = \bigcup_{i\in[k]} T'_i$ satisfies the second property of Lemma 8.7.

In the last case, we can again apply Lemma 8.16 to $(P_i^2)_{i\in[k]}$ and $(Z_{i,j}^2)_{i,j\in[k]}$ with $t_2 = 5t_1 + t_1^2$. By repeating this process until we arrive to an application of Lemma 8.16 where one of the first three cases is satisfied, we can define a sequence of extensions $(P_i^s)_{i\in[k]}$, $(Z_{i,j}^s)_{i,j\in[k]}$ and $t_s = 5t_{s-1} + t_{s-1}^2$ such that the $(Z_{i,j}^s)_{i,j\in[k]}$ are compatible t_s-useless sets of coordinates for $(P_i^s)_{i\in[k]}$. Note that if $t_{s-1} \leq 1$, then $t_s \leq 6t_{s-1}$ and recall that $\alpha < 1$. Moreover, this process has to stop after at most $(\Delta + 1)k$ iterations, because $\sum_{i,j\in[k]} |Z_{i,j}^{s+1}| > \sum_{i,j\in[k]} |Z_{i,j}^s|$, and if $Z_{i,j}^r$ increases more than Δ times, then $|Z_{i,j}^r| = d$, and we are in the first case. Therefore, by the choice of t_1, there exists a safe extension $(P'_i)_{i\in[k]}$ with compatible α-useless sets $(Z'_{i,j})_{i,j\in[k]}$ such that the application of Lemma 8.16 falls into one of the first three cases, and we can conclude.

The desired algorithm proceeds as follows. First, it guesses with probability $1/3$ in which of the above cases it falls. In the first case, guessing the pair i, j allows us to conclude with probability $1/k^2$. In the second, guessing i, j and $i_1^j \in I_i - I_j$ allows us to conclude with probability $\frac{1}{k^2\Delta}$ (remember that $|I_i - I_j| \leq \Delta$). Finally in the last case, the algorithm succeeds if the algorithm of Lemma 8.16 succeeds, so the probability of that is $(\frac{1}{\log(n)})^k$.

Overall, the probability of success is at least $\frac{1}{3k^2\Delta\log(n)^k}$, and the algorithm runs in time $\mathcal{O}(nkd)$. □

8.3 The Algorithm

Now that we have Lemma 8.7, we can describe our algorithm. Let us first recall the following lemma, which is a direct consequence of the definition of variance (see Lemma 1 of Inaba et al. [118]).

Lemma 8.22. *Let $x_1, \ldots, x_m$ be a set of reals with average μ and $s_1, \ldots, s_t$ be a set of elements obtained by t independent and uniform draws among the x_i. The random variable $s = \sum_{i\in t} s_i/t$ is such that $\mathrm{E}(|s - \mu|^2) \leq \frac{\sum_{i\in[m]} |x_i - \mu|^2}{tm}$.*

The main element of our proof is the following lemma providing one step of the algorithm.

Lemma 8.23. *For every constant $\alpha \in \mathbb{R}_{\geq 0}$, there exists an algorithm that, given a partial clustering $\mathcal{P} = \{(n_i)_{i\in[k]}, (H_i)_{i\in[k]}, (I_i)_{i\in[k]}, (\mathbf{u}_i)_{i\in[k]}\}$ outputs in time $\mathcal{O}(nkd)$*

with probability at least

$$\min\{\frac{1}{2^{\mathcal{O}(\frac{\Delta^3 k}{\alpha}\log\frac{1}{\alpha})}(\log n)^k}, \frac{1}{2^{\mathcal{O}(\Delta^6 k\log\Delta)}(\log n)^k}\}$$

a partial clustering $\mathcal{P}' = \{(n'_i)_{i\in[k]}, (H'_i)_{i\in[k]}, (I'_i)_{i\in[k]}, (\mathbf{u}'_i)_{i\in[k]}\}$ *such that* $\sum_{i\in[k]} n'_i > \sum_{i\in[k]} n_i$ *and* $\mathsf{OPT}(\mathcal{P}') \leq (1+\alpha)\mathsf{OPT}(\mathcal{P})$.

Proof. Let us fix sufficiently small $q \in \mathbb{R}_{\geq 0}$ such that $(1+q)^2 \leq (1+\alpha)$ and $\exp(-\frac{1}{4\Delta q}) \leq \frac{q}{4\Delta}$. The choice of q will be clear later in the proof. For now just notice that $q < 1$ and setting $q = \min\{\frac{\alpha}{3}, \frac{1}{128\Delta^3}\}$ satisfies both conditions. Additionally, let $q' = \frac{q}{32\cdot 6^{(\Delta+1)k}}$. By applying Lemma 8.7 with the constant q, we have an algorithm that runs in time $\mathcal{O}(nd)$ and with probability at least $(\frac{1}{2\log(n)})^k$ returns either:

- A partial clustering $\mathcal{P}' = \{(n'_i)_{i\in[k]}, (H'_i)_{i\in[k]}, (I'_i)_{i\in[k]}, (\mathbf{u}'_i)_{i\in[k]}\}$ with $\mathsf{OPT}(\mathcal{P}') \leq (1+q)\mathsf{OPT}(\mathcal{P})$ and $\sum_{i\in[k]} n'_i > \sum_{i\in[k]} n_i$; or
- a set B of elements of R such that there exists an extension $(P'_i)_{i\in[k]}$ of $\mathcal{P}$ with value at most $(1+q)\mathsf{OPT}(\mathcal{P})$ and such that $B \subseteq \bigcup_{i\in[k]} \mathrm{FD}(P'_i \cap R, I_i)$ and for every $i \in [k]$, there is an index $j \in [k]$ such that $|\mathrm{PD}(P'_j \cap R, I_i - I_j) - B| \geq q'|\mathrm{FD}(P'_i \cap R, I_i) - B|$.

In the former case, nothing needs to be done as $\mathcal{P}'$ satisfies all the properties of the lemma. Therefore, from now on we assume that we are in the latter case and we are given a set $B \subseteq R$ satisfying all the conditions of the second case of Lemma 8.7. Let $(P'_i)_{i\in[k]}$ be the hypothetical extension with value at most $(1+q)\mathsf{OPT}(\mathcal{P})$ whose existence is guaranteed. Let $R'_i = P'_i \cap (R - B)$ for all $i \in [k]$ and let $(\mathbf{c}'_i)_{i\in[k]}$ be the centers associated with $(P'_i)_{i\in[k]}$. Let i be the index such that $|R'_i|$ is maximal, meaning $|R'_i| \geq |R-B|/k$. Now either $|\mathrm{FD}(R'_i, I_i)| \geq |R'_i|/2$, in which case we know that there exists an index $j \in [k]$ such that $|\mathrm{PD}(R'_j, I_i - I_j)| \geq q'|\mathrm{FD}(R'_i, I_i)| \geq \frac{q'|R-B|}{2k}$, or $|R'_i - \mathrm{FD}(R'_i, I_i)| \geq \frac{|R-B|}{2k}$. Note that $\mathrm{PD}(R'_j, I_i - I_j) \cap \mathrm{FD}(R'_j, I_j) = \emptyset$ which means that there exists an index $r \in \{i, j\}$ such that $|R'_r - \mathrm{FD}(R'_r, I_r)| \geq \frac{q'|R-B|}{2k}$. The goal of the algorithm now will be to sample points inside $R'_r - \mathrm{FD}(R'_r, I_r)$ in order to obtain a good estimate of some coordinates of $\mathbf{c}'_r$ that are not yet in I_r. We will consider two different cases, depending on whether $|I_r| \geq d - \Delta$ or not.

Case 1: $|I_r| < d - \Delta$**.** Note that in this case, by the definition of a partial clustering, $n_r = 0$ and $H_r \cup \mathrm{FD}(R'_r, I_r)$ is empty. This implies that $R'_r = P'_r$. Let $\delta = \frac{1}{2\Delta}$ and note that $(1-\delta)^\Delta \geq 1/2$. We claim that there exists a set L_r of at most Δ coordinates (possibly empty) such that there exists a set $F_r \subseteq R'_r$, $|F_r| \geq |R'_r|/2$ where every element $\mathbf{x}$ of F_r is such that $\mathbf{x}_j = ?$ on every $j \in L_r$ and for every $i \in [d] - L_r$, there are at most $(1-\delta)|F_r|$ points $\mathbf{x}$ in F_r with $\mathbf{x}[i] = ?$. We can obtain the set F_r from

R'_r as follows. Start with $L_r = \emptyset$ and $F_r = R'_r$. As long as there exists a coordinate $i \in [d] - L_r$ where a $(1-\delta)$ fraction of points in F_r has value ? on the coordinate i, then set $L_r := L_r \cup \{i\}$ and F_r as the set of points of F_r with value ? on the coordinate i. This process has to stop after Δ steps as P consists of Δ-points. This means that, at the end, $|F_r| \geq (1-\delta)^{\Delta}|R'_r|$, which ends the proof of the claim. For $i \in L_r$, let $R^i_r = \mathrm{PD}(R'_r, i)$ and $p_i = |R^i_r|/|R'_r|$. The previous discussion shows that $p_i \geq \frac{\delta}{2}$ for all $i \notin L_r$.

Intuitively, L_r corresponds to the set of coordinates such that, if we sample in R'_r, then we might not get a good estimate for $\mathbf{c}'_r$ on these coordinates. Suppose now we pick uniformly at random an element $\mathbf{x}$ of $R - B$. Because $|F_r| \geq \frac{|R'_r|}{2} \geq \frac{q'|R-B|}{4k}$, with probability at least $\frac{q'}{4k}$, $\mathbf{x} \in F_r$. Assume from now on this is the case, and let $J_r = \mathrm{Dom}(x)$. Note that $L_r \subseteq [d] - J_r$ and $|J_r| \geq d - \Delta$.

Let $t = \frac{8}{q\delta}$. From the choice of q and because $p_i \geq \frac{\delta}{2}$ one can show that $exp(-tp_i^2/4) \leq \frac{2}{tp_i}$. Consider $X = \{x_1, \ldots, x_t\}$, a (multi)set of t elements in $R - B$ obtained by doing t independent and uniform draws. With probability at least $(\frac{q'}{2k})^t$, all these points belong to R'_r. From now on we assume this is the case, and all the probabilities computed will be conditioned by that fact. Note that the set $\{x_1, \ldots, x_t\}$ then follows exactly the same distribution as the one obtained by doing t independent and uniform draws in R'_r. Let J'_r be the set of coordinates e of J_r for which there exists an element $\mathbf{x}_s \in X$ such that $\mathbf{x}_s[e] \neq ?$, note that $|J_r - J'_r| \leq \Delta$. Let $\mathbf{u}' = \mathbf{c}^{J'_r}(X)$. For every $i \in J'_r$, denote by X_i the random variable counting the number of the points $\mathbf{x}_j$ with $j \in [t]$ such that $\mathbf{x}_j[i] \neq ?$. Note that X_i follows the binomial distribution with parameters t and p_i. By using standard tail bounds for the binomial distribution (see, e.g., Theorem 1 of [113]) we can show the following claim:

Claim 8.24. *For every $i \in J'_r$, $\Pr[X_i \leq tp_i/2] \leq exp(-tp_i^2/4)$.*

Moreover, if we condition by the event that $X_i = p$, then the distribution followed by the p values of $\mathbf{x}_s[i]$ is exactly the same as the one obtained by doing p uniform and random draws among all the v_i for $v \in \mathrm{PD}(R'_r, i)$. This implies the following result.

Claim 8.25. *For every $i \in J'_r$, we have $\mathrm{E}\big(|\mathbf{c}'_r[i] - \mathbf{u}'[i]|^2\big) \leq \frac{\sum_{\mathbf{x} \in R^i_r} |\mathbf{x}[i] - \mathbf{c}'_r[i]|^2}{|R^i_r|} \cdot \frac{4}{tp_i}$.*

Proof. By the definition of a partial clustering and an extension, and because I_r is empty, we know that $\mathbf{c}'_r[i]$ is equal to the average of $\mathbf{x}[i]$ over all elements $\mathbf{x}$ of R^i_r. By applying Lemma 8.22, we get that, for every $s \leq t$,

$$\mathrm{E}\big(|\mathbf{c}'_r[i] - \mathbf{u}'[i]|^2 | X_i = s\big) \leq \frac{\sum_{\mathbf{x} \in R^i_r} |\mathbf{x}[i] - \mathbf{c}_r[i]|^2}{|R^i_r| s}.$$

This means that:

$$\begin{aligned}
\mathrm{E}\big(|\mathbf{c}'_r[i]-\mathbf{u}'[i]|^2\big) &= \mathrm{E}\big(|\mathbf{c}'_r[i]-\mathbf{u}'[i]|^2 \mid X_i \le tp_i/2\big)\Pr[X_i \le tp_i/2] \\
&+ \mathrm{E}\big(|\mathbf{c}'_r[i]-\mathbf{u}'[i]|^2 \mid X_i > tp_i/2\big)\Pr[X_i > tp_i/2] \\
&\le \frac{\sum_{\mathbf{x}\in R_r^i}|\mathbf{x}[i]-\mathbf{c}_r[i]|^2}{|R_r^i|}\cdot\Pr[X_i \le tp_i/2] \\
&+ \frac{\sum_{\mathbf{x}\in R_r^i}|\mathbf{x}[i]-\mathbf{c}_r[i]|^2}{|R_r^i|tp_i/2} \\
&\le \frac{\sum_{\mathbf{x}\in R_r^i}|\mathbf{x}[i]-\mathbf{c}_r[i]|^2}{|R_r^i|}\left(exp(-tp_i^2/4)+\frac{2}{tp_i}\right).
\end{aligned}$$

Which ends the proof as we chose q such that $exp(-tp_i^2/4) \le \frac{2}{tp_i}$. □

Now for every index $a \in J_r - J'_r$, consider $A_a = \{\mathbf{x}_1^a, \ldots, \mathbf{x}_t^a\}$ a new (multi)set of t elements in $R-B$ obtained by doing uniform and independent draws. With probability at least $(\frac{q'\delta}{4k})^t$, all these elements belong to R_r^a. From now on, let us assume that it is the case for every $a \in J_r - J'_r$. Setting $\mathbf{u}'[a] = \mathbf{c}^a(A_a)$, an analogous proof as the one of Claim 8.25 would yield:

Claim 8.26. *For every $a \in (J_r - J'_r)$, we have* $\mathrm{E}\big(|\mathbf{c}'_r[a]-\mathbf{u}'[a]|^2\big) \le \frac{\sum_{\mathbf{x}\in R_r^a}|\mathbf{x}[a]-\mathbf{c}'_r[i]|^2}{|R_r^a|}\cdot\frac{4}{tp_a}$.

By summing over all coordinates of J_r, we obtain the following result.

Claim 8.27. *With probability at least* $1/2$, $f_2^{J_r}(R'_r,\mathbf{u}') \le (1+\frac{8}{tp_i})(f_2^{J_1}(R_r,\mathbf{c}'_r))$.

Proof. Indeed, $\mathrm{E}(f_2^{J_r}(R'_r,\mathbf{u}')-f_2^{J_r}(R_r,\mathbf{c}'_r)) = \sum_{i\in J_r}|R_r^i|\cdot\mathrm{E}(|\mathbf{c}'_r[i]-\mathbf{u}'[i]|^2)$ by Lemma 8.4, which is smaller than $\frac{4}{tp_i}(f_2^{J_r}(R_r,\mathbf{c}'_1))$ by the previous claims. Markov's inequality allows us to conclude. □

Therefore, by choosing the following at random:

- an index $r \in [k]$ such that $|R'_r - \mathrm{FD}(R'_r, I_r)| \ge \frac{q'|R-B|}{2k}$,
- an element $\mathbf{x} \in R-B$,
- t elements $\mathbf{x}_1, \ldots, \mathbf{x}_t$ in $R-B$, and
- t elements $\mathbf{x}_1^a, \ldots, \mathbf{x}_1^a$ in $R-B$ for each $a \in J_r - J'_r$

we find, with probability at least $\frac{1}{k}\cdot\frac{q'}{4k}\cdot(\frac{q'}{2k})^{(t-1)}\cdot(\frac{q'\delta}{4k})^{t\Delta}\cdot 1/2 \ge \frac{(q'\delta)^{(\Delta+1)t}}{(4k)^{(\Delta+1)t+1}} = \frac{1}{2^{\mathcal{O}(\frac{\Delta^3 k}{q}\log\frac{1}{q})}}$, a point $\mathbf{u}_1 \in \mathbb{H}^d$ such that $f_2^{J_r}(R'_r,\mathbf{u}') \le (1+\frac{8}{tp_i})(f_2^{J_r}(R_r,\mathbf{c}'_r))$. Now consider the set of points $(\mathbf{c}_i)_{i\in[k]}$ defined as follows: if $i \ne r$, then $\mathbf{c}_i = \mathbf{c}'_i$ and $\mathbf{c}_r$ is

the point obtained from $\mathbf{c}'_r$ by setting $(\mathbf{c}_r)_j = \mathbf{u}'_j$ on all the coordinates of J_r. We have that:

$$\begin{aligned}\mathrm{val}((P'_i)_{i\in[k]}, (\mathbf{c}_i)_{i\in[k]}) &\leq \mathrm{val}((P'_i)_{i\in[k]}, (\mathbf{c}'_i)_{i\in[k]}) + \frac{8}{tp_i}(f_2^{J_1}(R_r, \mathbf{c}'_r)) \\ &\leq (1+q)^2\mathsf{OPT}(\mathcal{P}) \\ &\leq (1+\alpha)\mathsf{OPT}(\mathcal{P}).\end{aligned}$$

Therefore, it means that the partial clustering $\mathcal{P}'$ obtained from $\mathcal{P}$ by setting $n_r = 1$, $\mathbf{u}_r = \mathbf{u}'$ and $I_r = J_r$ satisfies the property of the lemma. Indeed, the partition $(P'_i)_{i\in[k]}$ is an extension of $\mathcal{P}'$ with value at most $\mathrm{val}\big((P'_i)_{i\in[k]}, (\mathbf{c}_i)_{i\in[k]}\big)$ and $|J_r| \geq d - \Delta$.

Case 2: $|I_r| \geq d-\Delta$. Let $S = [d]-I_r$ and note that, by the definition of $\mathrm{FD}(R'_r, I_r)$, for every element $y \in R'_r - \mathrm{FD}(R'_r, I_r)$, there is an index $j \in S$ such that $j \in \mathrm{Dom}(y)$. In particular it means that there exists an index $j \in S$ such that $j \in \mathrm{Dom}(y)$ for at least $|R'_r - \mathrm{FD}(R'_r, I_r)|/\Delta$ of the elements of $R'_r - \mathrm{FD}(R'_r, I_r)$. Because $|S| \leq \Delta$, by picking a random element of S, with probability at least $1/\Delta$, we can assume that we know this index j. Our main goal now will be to guess $\mathbf{c}'_r[j]$. Let $t' = \frac{2}{q}$ and suppose that $\mathbf{s}_1, \ldots, \mathbf{s}_{t'}$ is a (multi)set of elements of $R-B$ obtained by t' uniform and independent draws. With probability at least $(\frac{q'}{2k\Delta})^{t'}$, all the $\mathbf{s}_i$ belong to $\mathrm{PD}(R'_r, j)$. From now on, let us assume that this is the case. Note that in this case, the random set $(\mathbf{s}_1, \ldots, \mathbf{s}_{t'})$ follows the same distribution as one obtained by doing t uniform and independent draws in $\mathrm{PD}(R'_r, j)$. Let $a_j = \sum_{i\in[t]} \mathbf{s}_i[j]/t'$, a proof similar to the one of Claim 8.27 gives the following claim:

Claim 8.28. *With probability at least* $1/2$, $f_2^j(R'_r, a_j) \leq (1+q)(f_2^j(R'_r, \mathbf{c}'_r))$.

In that case, the partial clustering $\mathcal{P}'$ obtained from $\mathcal{P}$ by setting $\mathbf{u}_r[j] = a_j$ and $n_r := n_r + 1$ satisfies the desired properties.

By considering both cases, we obtain an algorithm running in time $\mathcal{O}(knd)$ and succeeding with probability at least the probability that the algorithm of Lemma 8.7 succeeds times the minimum of $\frac{(q'\delta)^{(\Delta+1)t}}{(4k)^{(\Delta+1)t+1}}$ and $\frac{1}{\Delta}\cdot(\frac{q'}{2k\Delta})^{t'}$, which is at least

$$\frac{1}{2^{\mathcal{O}(\frac{\Delta^3 k}{q}\log\frac{1}{q})}(\log n)^k}.$$

Note that if α is sufficiently small, then this is $\frac{1}{2^{\mathcal{O}(\frac{\Delta^3 k}{\alpha}\log\frac{1}{\alpha})}(\log n)^k}$ and else it is $\frac{1}{2^{\mathcal{O}(\Delta^6 k\log\Delta)}(\log n)^k}$, finishing the proof. □

Finally, by applying Lemma 8.23 at most $k(\Delta+1)$ times we obtain Theorem 8.1.

Theorem 8.1. *The problem of k-means clustering of Δ-points in $\mathbb{R}^d$ admits an $(1+\varepsilon)$-approximation algorithm with running time $2^{\mathcal{O}(\frac{\Delta^7 k^3}{\varepsilon}\log\frac{k\Delta}{\varepsilon})}n^2 d$.*

Proof. Let P be a instance of the k-MEANS CLUSTERING problem consisting of Δ-points. Fix $\alpha = ((1+\varepsilon)^{1/k(\Delta+1)} - 1)$, note that $\alpha \geq \frac{\varepsilon}{3k(\Delta+1)}$, and let $\mathcal{P} = \{(n_i)_{i\in[k]}, (H_i)_{i\in[k]}, (I_i)_{i\in[k]}, (\mathbf{u}_i)_{i\in[k]}\}$ be the partial clustering such that for each $i \in [k]$, $n_i = 0$, $H_i = \emptyset$, $I_i = \emptyset$ and $\mathbf{u}_i$ is the point of $\mathbb{H}^d$ with only "?" entries. Note that $\mathsf{OPT}(\mathcal{P})$ is equal to the optimal value of the instance.

The algorithm consists of applying inductively Lemma 8.23 with the constant α and Observation 8.5 until $\bigcup_{i\in[k]}(H_i) = P$. Since $\sum_{i\in[k]} n_i$ increases in every application of Lemma 8.23, we get by Observation 8.6 that this process stops after at most $k(\Delta+1)$ steps. The probability that all the steps succeed is at least $(\frac{g(\alpha,k,\Delta)}{\log(n)^k})^{k(\Delta+1)}$, where $g(\alpha,k,\Delta) = \min\{\frac{1}{2^{\mathcal{O}(\frac{\Delta^3 k}{\alpha}\log\frac{1}{\alpha})}}, \frac{1}{2^{\mathcal{O}(\Delta^6 k\log\Delta)}}\}$, and if it holds then the partial clustering $\mathcal{P}' = \{(n'_i)_{i\in[k]}, (H'_i)_{i\in[k]}, (I'_i)_{i\in[k]}, (\mathbf{u}'_i)_{i\in[k]}\}$ obtained at the end is such that $\mathsf{OPT}(\mathcal{P}') \leq (1+\alpha)^{k(\Delta+1)}$. Since $\bigcup_{i\in[k]}(H_i) = P$, this gives us indeed a $(1+\varepsilon)$ approximation. Note that we can obtain a center for H_i simply by either taking $\mathbf{u}_i$ or computing $c(H_i)$. The running time is $\mathcal{O}(k^2\Delta nd)$ and the probability of success is at least $(\frac{g(\alpha,k,\Delta)}{\log(n)^k})^{k(\Delta+1)}$, which means that running the previous algorithm $\mathcal{O}((\frac{\log(n)^k}{g(\alpha,k,\Delta)})^{k(\Delta+1)})$ times allows us to find the approximation with constant probability. Finally, it is well-known that for any constant C, $\log(n)^C \leq n + C^{\mathcal{O}(C)}$ which gives the total running time of

$$\max\{2^{\mathcal{O}(\frac{\Delta^3 k}{\alpha}\log\frac{1}{\alpha})}, 2^{\mathcal{O}(\Delta^6 k\log\Delta)}\}^{k(\Delta+1)}2^{\mathcal{O}(k^2\Delta)\log(k\Delta)}k^2\Delta n^2 d.$$

It can be simplified to

$$\max\{2^{\mathcal{O}(\frac{\Delta^5 k^3}{\varepsilon}\log\frac{k\Delta}{\varepsilon})}, 2^{\mathcal{O}(\Delta^7 k^2\log(k\Delta))}\}n^2 d,$$

finishing the proof. □

8.4 Concluding Remarks

In this paper we gave the first PTAS for k-means clustering of Δ-points when $\Delta > 1$ running in time $2^{\mathcal{O}(\frac{\Delta^7 k^3}{\varepsilon}\log\frac{k\Delta}{\varepsilon})}n^2 d$ based on iteratively sampling points to discover new coordinates of some center. We believe that the study of clustering problems of Δ-points is an interesting research direction and there is still a lot to be discovered. We conclude with concrete open questions.

Arguable, the most popular clustering objectives are k-center, k-means, and k-median. For k-center clustering of Δ-points a PTAS was obtained by Lee and Schulman in [141]. However, for k-median clustering of Δ-points, the question whether

the problem admits a PTAS, remains open. We would like to remark here that the algorithm of Kumar et al. [137] for clustering of points in $\mathbb{R}^d$ works not only for k-means, but also for k-medial clustering.

Since Δ-points are basically Δ-dimensional axis-parallel subspaces, another interesting question would be whether it is possible to extend our result to clustering of arbitrary Δ-dimensional affine subspaces in $\mathbb{R}^d$. This is a very natural computational geometry problem which complexity, to the best of our knowledge, is widely open.

Following the coreset construction for k-means clustering of lines by Marom and Feldman [155], it is a natural open question whether it is possible do design a coresets of small size for clustering of Δ-points for $\Delta > 1$. For lines, the size of coreset of Marom and Feldman is $dk^{\mathcal{O}(k)} \log n/\varepsilon^2$. In particular, whether $\log n$ can be removed even for $\Delta = 1$, is open.

Finally, we do not know how tight is the running time of our algorithm. While it is plausible to suggest that the dependency in k and Δ is not optimal, to design a faster algorithm we need new ideas. It ls also an interesting open question whether one can improve the dependency on n from quadratic to linear.

Part IV

Low-Rank Approximation

9

Algebraic Geometry Approach for Robust PCA

In this chapter, our primary focus is the following algorithmic question about low-rank approximation with outliers. For a set of n points in $\mathbb{R}^d$, how to learn a subset of points, say 1% of the total number of points, such that the remaining part of the points is best fit into some unknown r-dimensional subspace? This is the PCA WITH OUTLIERS problem introduced in Subsection 3.2.1, and we recall its formal definition next.

PCA WITH OUTLIERS

Input: Data matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, integer parameters r and k.

Task:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{A} - \mathbf{L} - \mathbf{N}\|_F^2 \\ \text{subject to} \quad & \mathbf{L}, \mathbf{N} \in \mathbb{R}^{n \times d}, \\ & \operatorname{rank}(\mathbf{L}) \leq r, \text{ and} \\ & \mathbf{N} \text{ has at most } k \text{ non-zero rows.} \end{aligned}$$

Even though PCA WITH OUTLIERS was assumed to be NP-hard, to the best of our knowledge, this has never been studied formally. For an in-depth discussion about previous works on low-rank approximation problems, we refer to Section 3.2. While NP-hardness is a serious strike against the tractability of the problem, on the other hand, it only says that in the worst case the problem is not tractable. But since the complexity of the problem could be governed by several parameters like

the rank r of $\mathbf{L}$, the number of outliers k or the dimension d of $\mathbf{A}$, it is natural to ask how these parameters influence the complexity of the problem. For example, when k is a small constant, we can guess which points are outliers and run PCA for the remaining points. This will bring us to n^k calls of PCA which is polynomial for constant k and is exponential when k is a fraction of n.

In this chapter we give an algorithm solving PCA WITH OUTLIERS roughly in time $|\mathbf{A}|^{\mathcal{O}(rd)}$, where $|\mathbf{A}|$ is the size of the input matrix $\mathbf{A}$. Thus for fixed dimension d, the problem is solvable in polynomial time. Note that the algorithms works in polynomial time for any number of outliers k. Our algorithm strongly relies on the tools developed in computational algebraic geometry, in particular, for handling arrangements of algebraic surfaces in $\mathbb{R}^d$ defined by polynomials of bounded degree.

We complement this algorithmic result by a complexity lower bound. Our lower bound not only implies that the problem is NP-hard when dimension d is part of the input, it also rules out a possibility of certain type of approximation algorithms for PCA WITH OUTLIERS. More precisely, assuming the Exponential Time Hypothesis (ETH), we show that for *any* value α, constant or depending on the input size, PCA WITH OUTLIERS cannot be α-approximated in time $f(d)|\mathbf{A}|^{o(d)}$, for any function f of d only. The reduction holds in fact for the ROBUST SUBSPACE RECOVERY problem, which is the special case of deciding whether for an instance of PCA WITH OUTLIERS the value of the optimal solution is 0 or not. Equivalently, the input to ROBUST SUBSPACE RECOVERY is an instance $(\mathbf{A}, r, k)$ of PCA WITH OUTLIERS, and the task is to decide whether there exist matrices $\mathbf{L}$ and $\mathbf{N}$ such that $\mathbf{A} = \mathbf{L} + \mathbf{N}$, the rank of $\mathbf{L}$ is at most r, and $\mathbf{N}$ has at most k non-zero rows.

Our algorithm is, foremost, of theoretical interest, especially in the presence of the nearly-matching lower bound showing that doing something essentially better is next to impossible. In practice, PCA is often applied to reduce high-dimensional datasets, and for this task the running time exponential in d is not practical. However, there are cases where such an algorithm could still be useful. One example could be the visualization of low-dimensional data, where the number of dimensions, even if it is small already, needs to be lowered down to two to actually draw the dataset. Another example could be when we suspect a small subset of features to be highly correlated, and we want to reduce them to one dimension in order to get rid of the redundancy in data. This potential application is well illustrated by the popular PCA tutorial [175], where essentially one-dimensional movement of a spring-mass is captured by three cameras, resulting in 6 features.

9.1 Polynomial-time Algorithm for Bounded Dimension

First, we emphasize on a folklore observation that geometrically the low-rank approximation matrix $\mathbf{L}$ is defined as orthogonal projection of rows of $\mathbf{A}$ on some

r-dimensional subspace of $\mathbb{R}^d$. For the proof see e.g. [29].

Proposition 9.1. *Given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$, the task of finding a matrix $\mathbf{L}$ of rank at most r which minimizes $||\mathbf{A}-\mathbf{L}||_F^2$ is equivalent to finding an r-dimensional subspace of $\mathbb{R}^d$ which minimizes the total squared distance from rows of $\mathbf{A}$ treated as points in $\mathbb{R}^d$:*

$$\min_{\substack{\mathbf{L}\in\mathbb{R}^{n\times d}\\ \mathrm{rank}\mathbf{L}\le r}} ||\mathbf{A}-\mathbf{L}||_F^2 = \min_{\substack{U\subset\mathbb{R}^d\\ U \text{ is a linear subspace of } \dim r}} \sum_{i=1}^{n} ||\mathbf{a}_i - \mathrm{proj}_U \mathbf{a}_i||_F^2,$$

where $\mathrm{proj}_U \mathbf{x}$ is the orthogonal projection of $\mathbf{x}$ on U for $\mathbf{x}\in\mathbb{R}^d$.

By Proposition 9.1, if we fix an r-dimensional subspace U containing the span of rows of $\mathbf{L}$, then the outliers are automatically defined as k farthest points from U among $\{\mathbf{a_i}\}_{i=1}^n$. In the next proposition, we give a precise statement of this.

Proposition 9.2. *The optimization objective of* PCA WITH OUTLIERS *for a given matrix $\mathbf{A}\in\mathbb{R}^{n\times d}$ with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$ can be equivalently redefined as follows.*

$$\min_{\substack{\mathbf{L},\mathbf{N}\in\mathbb{R}^{n\times d}\\ \mathrm{rank}\mathbf{L}\le r\\ \mathbf{N} \text{ has at most } k \text{ non-zero rows}}} ||\mathbf{A}-\mathbf{L}-\mathbf{N}||_F^2 = \min_{\substack{U\subset\mathbb{R}^d\\ U \text{ is a linear subspace of } \dim r}} ||\mathbf{A}-\mathbf{L}_U-\mathbf{N}_U||_F^2,$$

where $\mathbf{N}_U$ has k non-zero rows which are k rows of $\mathbf{A}$ with the largest value of $||\mathbf{a}_i - \mathrm{proj}_U \mathbf{a}_i||_F^2$, and $\mathbf{L}_U$ is the orthogonal projection of the rows of $(\mathbf{A}-\mathbf{N}_U)$ on U.

$$\mathbf{N}_U = \begin{pmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_k \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{pmatrix}, \quad \mathbf{L}_U = \begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathrm{proj}_U \mathbf{a}_{k+1} \\ \vdots \\ \mathrm{proj}_U \mathbf{a}_n \end{pmatrix},$$

assuming that rows of $\mathbf{A}$ are ordered by the descending value of $||\mathbf{a}_i - \mathrm{proj}_U \mathbf{a}_i||_F^2$.

So for a fixed U we may determine $\mathbf{S}_U$ easily and then solve the classical PCA for the matrix $(\mathbf{A}-\mathbf{S}_U)$. The intuition behind our algorithm is that the set of k farthest points is the same for many subspaces, and solving PCA for $(\mathbf{A}-\mathbf{S}_U)$ treats all these subspaces. The crucial point is to bound the number of different matrices $\mathbf{S}_U$ we have to consider. There is of course a trivial bound of n^k since $\mathbf{S}_U$ is always obtained by choosing k rows of $\mathbf{A}$. But the number of different $\mathbf{S}_U$ is also geometrically limited, and exploiting this we are able to obtain another bound of $n^{O(rd)}$, resulting in the following theorem.

Theorem 9.3. *Solving* PCA WITH OUTLIERS *is reducible to solving*

$$\binom{n}{2}^{\min(rd,(d-r)d)} 2^{\mathcal{O}(d)} = n^{\mathcal{O}(d^2)}$$

instances of PCA. This reduction can be computed in the number of operations over $\mathbb{R}$ *bounded by the expression above.*

First, a note about the statement of Theorem 9.3. Our algorithm relies on solving the classical PCA, and since only iterative algorithms for PCA and SVD exist, we could not claim that our algorithm solves PCA WITH OUTLIERS in some fixed number of operations. However, if we are only interested in solving the problem up to some constant precision, for example machine epsilon, then PCA is solvable in polynomial number of operations and so by Theorem 9.3, PCA WITH OUTLIERS is solvable in $n^{\mathcal{O}(d^2)}$ operations.

Proof of Theorem 9.3. We start with associating r-dimensional subspaces of $\mathbb{R}^d$ with points of a certain algebraic set. Consider the matrix space $\mathbb{R}^{(d-r)\times d}$, and for an element $\mathbf{V} \in \mathbb{R}^{(d-r)\times d}$, $\mathbf{V} = (\mathbf{V}_{ij})_{i,j}$, the following polynomial conditions:

$$Q^O_{i,j}(\mathbf{V}) := \sum_{\ell=1}^{d} \mathbf{V}_{i\ell}\mathbf{V}_{j\ell} = 0, \text{ for } 1 \le i < j \le (d-r),$$
$$Q^N_j(\mathbf{V}) := \left(\sum_{\ell=1}^{d} \mathbf{V}^2_{j\ell}\right) - 1 = 0, \text{ for } 1 \le j \le (d-r),$$

where the condition $Q^O_{i,j}(\mathbf{V}) = 0$ requires rows i, j of $\mathbf{V}$ to be pairwise orthogonal and the condition $Q^N_j(\mathbf{V}) = 0$ requires row j of $\mathbf{V}$ to have length one. We may write all these conditions as a single polynomial condition $Q(\mathbf{V}) = 0$ by taking the sum of squares:

$$Q(\mathbf{V}) = \sum_{1\le i<j\le(d-r)} (Q^O_{i,j}(\mathbf{V}))^2 + \sum_{j=1}^{d-r} (Q^N_j(\mathbf{V}))^2.$$

Thus $Q(\mathbf{V}) = 0$ if and only if each of $Q^O_{i,j}(\mathbf{V}) = 0$ and each of $Q^N_j(\mathbf{V}) = 0$.

Consider an algebraic set $W \subset \mathbb{R}^{(d-r)\times d}$ defined as the zero set of $Q(\mathbf{V})$. For any $\mathbf{V} \in W$ with rows $\mathbf{v}_1, \ldots, \mathbf{v}_{d-r}$, consider the r-dimensional subspace $\mathrm{comp}(\mathbf{V}) := \mathrm{span}(\{\mathbf{v}_1, \cdots, \mathbf{v}_{d-r}\})^\perp \subset \mathbb{R}^d$ which is the orthogonal complement of the span of the rows of $\mathbf{V}$. Since $Q(\mathbf{V}) = 0$, the rows of $\mathbf{V}$ are pairwise orthogonal and are of length one. Thus, the dimension of $\mathrm{comp}(\mathbf{V})$ is r and for any point $\mathbf{x} \in \mathbb{R}^d$ the squared distance from $\mathbf{x}$ to $\mathrm{comp}(\mathbf{V})$ is equal to

$$\sum_{i=1}^{d-r} (\mathbf{v}_i \cdot \mathbf{x})^2 = ||\mathbf{V}\mathbf{x}^\top||^2_F,$$

assuming that $\mathbf{x}$ is a row vector.

Each $\mathbf{V} \in W$ defines an r-dimensional subspace $\mathrm{comp}(\mathbf{V}) \subset \mathbb{R}^d$ and each r-dimensional subspace $U \subset \mathbb{R}^d$ is of this form for some $\mathbf{V} \in W$ since there exists an orthonormal basis of the orthogonal complement of U. Then we can reformulate Proposition 9.2 in terms of elements of W as follows.

$$\min_{\substack{\mathbf{L},\mathbf{N}\in\mathbb{R}^{n\times d}\\ \mathrm{rank}\mathbf{L}\le r\\ \mathbf{N}\text{ has at most }k\text{ non-zero rows}}} ||\mathbf{A}-\mathbf{L}-\mathbf{N}||_F^2 = \min_{\mathbf{V}\in W} ||\mathbf{A}-\mathbf{N}_{\mathrm{comp}(\mathbf{V})}-\mathbf{L}_{\mathrm{comp}(\mathbf{V})}||_F^2, \quad (9.1)$$

where $\mathbf{N}_{\mathrm{comp}(\mathbf{V})}$ and $\mathbf{L}_{\mathrm{comp}(\mathbf{V})}$ are defined in accordance with notation in Proposition 9.2. Let $\mathbf{a}_1, \ldots, \mathbf{a}_n$ be the rows of the input matrix $\mathbf{A}$; $\mathbf{N}_{\mathrm{comp}(\mathbf{V})}$ has k non-zero rows which are the k rows of $\mathbf{A}$ with the largest value of $||\mathbf{a}_i - \mathrm{proj}_{\mathrm{comp}(\mathbf{V})}\,\mathbf{a}_i||_F^2 = ||\mathbf{V}\mathbf{a}_i^\top||_F^2$, and $\mathbf{L}_{\mathrm{comp}(\mathbf{V})}$ is the orthogonal projection of the rows of $(\mathbf{A}-\mathbf{N}_{\mathrm{comp}(\mathbf{V})})$ on $\mathrm{comp}(\mathbf{V})$. Denote $\mathbf{N}_{\mathrm{comp}(\mathbf{V})}$ by $\mathbf{N}_\mathbf{V}$ and $\mathbf{L}_{\mathrm{comp}(\mathbf{V})}$ by $\mathbf{L}_\mathbf{V}$.

Now, consider the set of polynomials $\mathcal{P} = \{P_{i,j}\}_{1\le i<j\le n}$ defined on W, where

$$P_{i,j}(\mathbf{V}) = ||\mathbf{V}\mathbf{a}_i^\top||_F^2 - ||\mathbf{V}\mathbf{a}_j^\top||_F^2.$$

Consider the partition $\mathcal{C}$ of W on cells over $\mathcal{P}$. For each cell C, the sign condition with respect to $\mathcal{P}$ is constant over C, meaning that for every pair $1 \le i < j \le n$, the sign of

$$||\mathbf{V}\mathbf{a}_i^\top||_F^2 - ||\mathbf{V}\mathbf{a}_j^\top||_F^2$$

is the same for all $\mathbf{V} \in C$. So the relative order on $\{||\mathbf{V}\mathbf{a}_i^\top||_F^2\}_{i=1}^n$ is also the same for all $\mathbf{V} \in C$. Since $||\mathbf{V}\mathbf{a}_i^\top||_F^2$ is exactly the squared distance from $\mathbf{a}_i$ to $\mathbf{V}$, the k rows of $\mathbf{A}$ which are the farthest are also the same for all $\mathbf{V} \in C$. Then $\mathbf{N}_\mathbf{V}$ is constant over $\mathbf{V} \in C$, denote this common value by $\mathbf{N}_C$. We can rewrite (9.1) as

$$\min_{\mathbf{V}\in W} ||\mathbf{A}-\mathbf{N}_\mathbf{V}-\mathbf{L}_\mathbf{V}||_F^2 = \min_{C\in\mathcal{C}}\min_{V\in C} ||\mathbf{A}-\mathbf{N}_\mathbf{V}-\mathbf{L}_\mathbf{V}||_F^2 = \min_{C\in\mathcal{C}}\min_{\mathbf{V}\in C} ||(\mathbf{A}-\mathbf{N}_C-\mathbf{L}_\mathbf{V}||_F^2.$$

Note that

$$\min_{C\in\mathcal{C}}\min_{\mathbf{V}\in C} ||(\mathbf{A}-\mathbf{N}_C)-\mathbf{L}_\mathbf{V}||_F^2 = \min_{C\in\mathcal{C}}\min_{\mathbf{V}\in W} ||(\mathbf{A}-\mathbf{N}_C)-\mathbf{L}_\mathbf{V}||_F^2, \quad (9.2)$$

as for any $C \in \mathcal{C}$, $\min_{\mathbf{V}\in C} ||(\mathbf{A}-\mathbf{N}_C)-\mathbf{L}_\mathbf{V}||_F^2 \ge \min_{\mathbf{V}\in W} ||(\mathbf{A}-\mathbf{N}_C)-\mathbf{L}_\mathbf{V}||_F^2$ since $C \subset W$. Also, any $(\mathbf{N}_C, \mathbf{L}_\mathbf{V})$ in the right-hand side of (9.2) is still a valid choice of $(\mathbf{N}, \mathbf{L})$ for the original problem, and the optimum of the original problem is equal to the left-hand side of (9.2).

For a fixed $C \in \mathcal{C}$ computing right-hand side of (9.2) is equivalent to solving an instance $(\mathbf{A}-\mathbf{N}_C, r)$ of the classical PCA by Proposition 9.1:

$$\min_{\mathbf{V}\in W} ||(\mathbf{A}-\mathbf{N}_C)-\mathbf{L}_\mathbf{V}||_F^2 = \min_{\mathbf{L}\in\mathbb{R}^{n\times d},\,\mathrm{rank}(\mathbf{L})\le r} ||(\mathbf{A}-\mathbf{N}_C)-\mathbf{L}||_F^2.$$

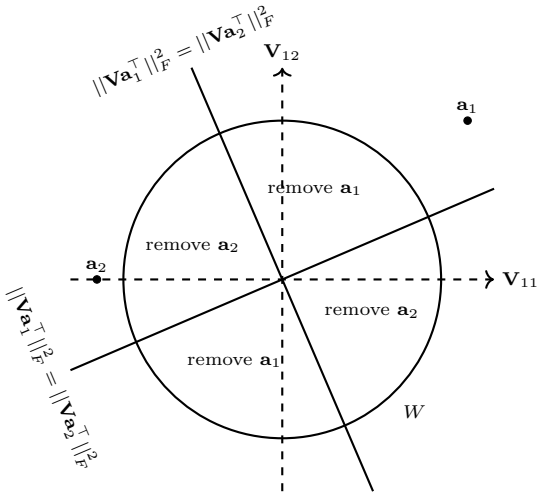


Figure 9.1: An example with $d = 2$, $n = 2$, $r = 1$, $k = 1$. Two rows of the input matrix $\mathbf{A}$ are represented as two points $\mathbf{a}_1$ and $\mathbf{a}_2$ on the plane. The same plane represents the choice of 1-dimensional approximation subspace through the selection of a vector $\mathbf{V}$ orthogonal to it. The algebraic set W is the unit circle since $\mathbf{V}$ must be of length one. The diagonal lines mark the values of $\mathbf{V}$ for which $\mathbf{a}_1$ and $\mathbf{a}_2$ are equidistant. They split W into four one-dimensional and four zero-dimensional cells. For each of the one-dimensional cells it is shown in the corresponding sector which of the points is the outlier and hence is removed.

By the reasoning above, the optimum of the original instance of PCA WITH OUTLIERS is reached on one of the constructed instances $\{(\mathbf{A} - \mathbf{N}_C, r)\}_{C \in \mathcal{C}}$ of PCA. A toy example of an algebraic set W and its partitioning is shown in Figure 9.1.

Putting all together, our algorithm proceeds as follows.

(i) Using the algorithm from Theorem 4.12, obtain a point $\mathbf{V}_C$ from each cell C of W over $\mathcal{P}$.

(ii) For each $\mathbf{V}_C$, compute the optimal $\mathbf{N}_{\mathbf{V}_C}$: select the k rows of $\mathbf{A}$ with the largest value of $||\mathbf{V}_C \mathbf{a}_i^\top||_F^2$. Construct the instance $(\mathbf{A} - \mathbf{N}_{\mathbf{V}_C}, r)$ of PCA.

(iii) The solution to the original instance of PCA WITH OUTLIERS is the best solution among the solutions of all the constructed PCA instances.

Since degrees of Q and polynomials from $\mathcal{P}$ are at most 4, $|\mathcal{P}| = \binom{n}{2}$, and the real dimension of W is at most $(d - r)d$, which is the dimension of $\mathbb{R}^{(d-r)\times d} \supset W$, the algorithm of Theorem 4.12 does at most

$$t = \binom{n}{2}^{(d-r)d} 2^{\mathcal{O}(d)}$$

operations and produces at most t matrices $\mathbf{V}_C$, and our algorithm produces one instance of PCA for each computed matrix.[1]

We are also able to obtain a reduction to

$$\binom{n}{2}^{rd} 2^{\mathcal{O}(d)}$$

instances of PCA by proceeding in the same manner for a slightly different characterization of r-dimensional subspaces. Intuitively, now points on the algebraic set define the orthonormal basis of the subspace itself, and not of its orthogonal complement as in the previous part.

Now the matrix space is $\mathbb{R}^{r\times d}$, the conditions that an element $\mathbf{V} \in \mathbb{R}^{r\times d}$ defines an orthonormal basis of size r are analogous:

$$\bar{Q}^O_{i,j}(\mathbf{V}) := \sum_{\ell=1}^{d} \mathbf{V}_{i\ell}\mathbf{V}_{j\ell} = 0, \text{ for } 1 \le i < j \le r,$$
$$\bar{Q}^N_{j}(\mathbf{V}) := \left(\sum_{\ell=1}^{d} \mathbf{V}^2_{j\ell}\right) - 1 = 0, \text{ for } 1 \le j \le r.$$

Again, we may write them as a single polynomial condition $\bar{Q}(\mathbf{V}) = 0$ where

$$\bar{Q}(\mathbf{V}) = \sum_{1\le i<j\le r} (\bar{Q}^O_{i,j}(\mathbf{V}))^2 + \sum_{j=1}^{r} (\bar{Q}^N_j(\mathbf{V}))^2,$$

Consider an algebraic set $\bar{W} \subset \mathbb{R}^{r\times d}$ defined as the set of zeroes of $\bar{Q}(\mathbf{V})$. Similarly, any $\mathbf{V} \in \bar{W}$ defines an r-dimensional subspace $U \in \mathbb{R}^d$ which is the span of the rows of $\mathbf{V}$. Since the rows of $\mathbf{V}$ form an orthonormal basis of U, for any point $\mathbf{x} \in \mathbb{R}^d$ the squared distance from $\mathbf{x}$ to U is equal to

$$||\mathbf{x}||^2_F - \sum_{i=1}^{r} (\mathbf{v}_i \cdot \mathbf{x})^2 = ||\mathbf{x}||^2_F - ||\mathbf{V}\mathbf{x}^\top||^2_F.$$

The new distance formula leads to a slightly different set of polynomials $\bar{\mathcal{P}} = \{\bar{P_{i,j}}\}_{1\le i<j\le n}$ on W, comparing the distance from $\mathbf{a}_i$ and from $\mathbf{a}_j$,

$$\bar{P_{i,j}}(\mathbf{V}) = (||\mathbf{a}_i||^2_F - ||\mathbf{V}\mathbf{a}_i^\top||^2_F) - (||\mathbf{a}_j||^2_F - ||\mathbf{V}\mathbf{a}_j^\top||^2_F).$$

Again, the k farthest points and the matrix $\mathbf{N}_\mathbf{V}$ are the same over any cell in the partition of $\bar{W}$ over $\bar{\mathcal{P}}$. So by the same reasoning as in the first part, it suffices to

[1] As W is restricted by $Q(V) = 0$, its dimension is actually smaller. It could be bounded more precisely as $(d-r)(d+r-1)/2$, but we omit the calculation in order not to unnecessarily complicate the text.

take a point $\mathbf{V}$ from each cell, compute the outlier matrix $\mathbf{N}_\mathbf{V}$ and solve the vanilla PCA for $(\mathbf{A} - \mathbf{N}_\mathbf{V}, r)$.

As we can choose the most efficient of the two subspace representations, we can reduce PCA WITH OUTLIERS to

$$\binom{n}{2}^{\min(rd,(d-r)d)} 2^{\mathcal{O}(d)}$$

instances of PCA.[2] □

9.2 Hardness of Robust Subspace Recovery

Recall that ROBUST SUBSPACE RECOVERY is the special case of PCA WITH OUTLIERS where the objective value is zero. In other words, the input matrix $\mathbf{A}$ has to be exactly represented as the sum of a low-rank matrix $\mathbf{L}$ and an outlier matrix $\mathbf{N}$. In this section, we prove our hardness result for ROBUST SUBSPACE RECOVERY, and hardness of approximation for PCA WITH OUTLIERS with any approximation factor follows immediately from that. We state the result formally in the next theorem.

Theorem 9.4. *There is no algorithm solving* ROBUST SUBSPACE RECOVERY *in time $f(d) \cdot n^{o(d)}$ for any computable function f of d, unless ETH fails.*

First, we give a high-level overview of the proof. We show a parameter-preserving reduction from CLIQUE. Assume that we are given an instance (G, r) of CLIQUE with $|V(G)| = n$, $|E(G)| = m$. Set $d = r + 1$, one larger than the size of the clique to find, and the target rank will be exactly r. With each vertex v_i of G, we associate a vector $\mathbf{w}_i$ in $\mathbb{R}^d$, such that the resulting set $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ of n vectors satisfies a certain generality condition, in particular that any d of the vectors are linearly independent. One may think of W as a set of n random vectors in $\mathbb{R}^d$, though the construction may be made discrete and deterministic. Now consider the matrix $\mathbf{A} \in \mathbb{R}^{m \times d}$ where the rows correspond to the edges of G,

$$\mathbf{A} = (\mathbf{w}_i + \mathbf{w}_j \mid \{v_i, v_j\} \in E(G)) .$$

The matrix $\mathbf{A}$ is the input matrix to ROBUST SUBSPACE RECOVERY, the target rank is r, the number of outliers k is set to $m - \binom{r}{2}$. We claim that (G, r) is a yes-instance of CLIQUE if and only if the constructed instance of ROBUST SUBSPACE RECOVERY is a yes-instance.

To reformulate the objective of ROBUST SUBSPACE RECOVERY, an instance $(\mathbf{A}, r, k)$ is a yes-instance if and only if there is a subset of $n - k = \binom{r}{2}$ rows of

[2] As with W, the dimension of $\bar{W}$ could be bounded more precisely as $r(2d - r - 1)/2$, and with these dimension bounds PCA WITH OUTLIERS reduces to $\binom{n}{2}^{\min(r(2d-r-1)/2,(d-r)(d+r-1)/2)} 2^{\mathcal{O}(d)}$ instances of PCA.

$\mathbf{A}$ with rank at most r. Crucial to the proof is the following claim that shows how to identify the rank just by the structure of the edges corresponding to the selected rows.

Claim 9.5. *The rank of any submatrix $\mathbf{A}' \subset \mathbf{A}$ obtained by deleting rows from $\mathbf{A}$ is*

$$\max(d, |V(G(\mathbf{A}'))| - \textit{number of bipartite graphs among } \mathsf{cc}(G(\mathbf{A}'))),$$

where $G(\mathbf{A}')$ is the subgraph of G such that its edges are exactly the edges corresponding to the rows of $\mathbf{A}'$, and its vertex set is the set of all endpoints of the edges, and $\mathsf{cc}(G(\mathbf{A}'))$ is the set of connected components of $G(\mathbf{A}')$.

Intuitively, the claim holds since for each connected component C of $G(\mathbf{A}')$ one can find a basis for the corresponding row space going along a spanning tree of the component. The size of the basis is $|V(C)|-1$ if C is bipartite (e.g. for a single edge, the space will be spanned by the corresponding row), and $|V(C)|$ otherwise.

Now the role that Claim 9.10 plays in the reduction is as follows. We need to keep exactly $\binom{r}{2}$ rows while minimizing the rank, and by the claim this boils down to making the corresponding graph as dense as possible. A computation shows that only an r-clique can achieve the rank of r on $\binom{r}{2}$ edges, thus the subspace recovery is possible if and only if the given graph contains an r-clique, which proves the correctness of the reduction. By [51], see also [70], assuming ETH, there is no algorithm solving Clique in time $f(r) \cdot n^{o(r)}$ where r is the size of the clique, for any computable function f of r. In our reduction d is equal to $r+1$, thus the theorem follows.

Finally we note that since Robust Subspace Recovery is the zero-valued restriction of PCA with Outliers, the hardness of approximation for the latter easily follows from Theorem 9.4.

Corollary 9.6. *Assuming ETH, there is no algorithm approximating* PCA with Outliers *with any multiplicative guarantee in time $f(d) \cdot n^{o(d)}$.*

Now we move to the formal proof.

Proof of Theorem 9.4. We show a reduction from Clique. First, we need a set of points satisfying a certain generality condition.

Definition 9.7. *For $d < n$, let us say that a set $W \subset \mathbb{R}^d$ of size n is* in a forest-general position *if for any forest F such that $V(F) \subset W$ and $|E(F)|$ plus the number of isolated vertices in F is exactly d, the set*

$$\operatorname{vect} F := \{\mathbf{u}+\mathbf{v} \mid \mathbf{uv} \in E(F)\} \cup \{\mathbf{w} \in V(F) \mid \mathbf{w} \textit{ is isolated in } F\},$$

is linearly independent and of size d.

Note that this definition extends the common notion of vectors in a general linear position, which requires every d vectors to be linearly independent, since F can also be an empty forest on d vertices. The next claim extends the behavior in Definition 9.7 to forests of any size.

Claim 9.8. *If a set $W \subset \mathbb{R}^d$ is in a forest-general position, for any forest F such that $V(F) \subset W$,*

$$\operatorname{rank}(\operatorname{vect}(F)) = \min(d, |\operatorname{vect}(F)|).$$

Proof. If $|\operatorname{vect}(F)| = d$, the claim is by definition.

If $|\operatorname{vect}(F)| > d$, obtain a subforest F' of F such that $|\operatorname{vect}(F')| = d$. This is always possible since removing either an isolated vertex from F or a leaf of a tree in F together with the incident edge decreases the size of $\operatorname{vect}(F)$ by one. By Definition 9.7, $\operatorname{vect} F'$ is linearly independent, and since $\operatorname{vect}(F') \subset \operatorname{vect}(F)$, they both have rank d.

If $|\operatorname{vect}(F)| < d$, obtain F' by adding isolated vertices or edged in F such that $\operatorname{vect}(F') = d$. This is always possible since $d < n$. By Definition 9.7, $\operatorname{vect}(F')$ is linearly independent, and since $\operatorname{vect}(F) \subset \operatorname{vect}(F')$, $\operatorname{vect}(F)$ has full rank. □

Finally, to use Definition 9.7 in our reduction, we need to construct a corresponding set of points.

Claim 9.9. *For any n and d, there exists a set of n vectors in $\mathbb{N}^d$ in a forest-general position such that the total bit-length of the coordinates is bounded by $\mathsf{poly}(n + d)$.*

Proof. Let $W = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n\}$ be a set of n vectors in $\mathbb{R}^d$. Assume that Definition 9.7 does not hold for W and a particular forest F. Then there exists a linear combination of $\operatorname{vect}(F)$ which is a zero vector. Without loss of generality, $V(F) = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_t\}$, let $\mathbf{a} = (a_i)_{i=1}^t$ be the zero linear combination of $\operatorname{vect}(F)$, treated as a linear combination of vectors in $V(F)$. We claim that $\mathbf{M} \cdot \mathbf{a}$ is a zero vector for the $t \times t$ matrix $\mathbf{M}$ defined as follows. The first d rows are the coordinates of $\mathbf{w}_1$, $\mathbf{w}_2$, …, $\mathbf{w}_t$, written as columns. Since $\mathbf{a}$ is the zero linear combination of these vectors, clearly each of these rows times $\mathbf{a}$ is zero. Next, for every connected component C of F append a row $\mathbf{r}$ such that $\mathbf{r}_i = 0$ if $\mathbf{w}_i \notin C$, and $\mathbf{r}_i = \pm 1$ if $\mathbf{w}_i \in C$, and for every edge of C its endpoints have different signs, this encodes a 2-coloring of C. The orthogonality to $\mathbf{a}$ follows from observing how $\mathbf{a}$ on the coordinates of C is obtained from the original linear combination of $\operatorname{vect}(F)$. Note that now we have exactly t rows. Thus, Definition 9.7 does not hold if and only if $\mathbf{M}$ is non-invertible.

Now, the way to construct the required set is to take the rows of a Vandermonde matrix where the generating elements are selected to be in a certain general position regarding the differences between them, in this case it is possible to show that every matrix of the form discussed above is invertible. For the sake of brevity we omit

this technical argument, and instead explain the simple randomized procedure of generating the set, which also shows why sets in a forest-general position are common.

Let W be a set of n vectors in $\mathbb{N}^d$ where each coordinate of each vector is sampled independently and uniformly from the set $\{1, \cdots, N\}$, where N is a value we fix later. The number of matrices which must be invertible by the observation above is at most $d \cdot \binom{n}{2d} \cdot d^{2d} \cdot 2^{2d}$, since the matrix up to permutations is defined by the number of components in F (at most d), the choice of t vectors in W (at most $\binom{n}{2d}$), the partition of them into components (at most d^{2d}), and the 2-coloring on each component (at most 2^{2d}). We bound the probability that a fixed matrix is non-invertible, and then apply union bound. For a fixed matrix $\mathbf{M}$, we can treat the process as follows. First we are given the $t - d$ rows obtained from the components of the forest, and then we sample one by one the d rows each composed out of t coordinates of vectors in W. For the first row, probability of falling into the span of already existing rows is at most $1/N^d$, for the second it is $1/N^{d-1}$, and so on. Then the probability of success for $\mathbf{M}$ is at least $(1-1/N) \cdot (1-1/N^2) \cdots$, which could be lower-bounded by $1 - 1/(N-1)$. By taking N sufficiently large compared to the number of matrices, the union bound is satisfied. Note that $\log N$ is polynomially bounded in n and d, and thus the total bit-length is also polynomially bounded. □

Now we are ready to show the reduction. Assume that we are given an instance (G, r) of Clique with $|V(G)| = n$, $|E(G)| = m$. Set $d = r + 1$, one larger than the size of the clique to find. By Claim 9.9, obtain a set $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ of n vectors in $\mathbb{R}^d$ in a forest-general position.

Vectors from W are associated with vertices of G. Now consider the matrix $\mathbf{A}$ where rows correspond to the edges of G,

$$\mathbf{A} = (\mathbf{w}_i + \mathbf{w}_j \mid \{v_i, v_j\} \in E(G)) .$$

The matrix $\mathbf{A}$ is the input matrix to Robust Subspace Recovery, the target rank is r, the number of outliers k is set to $m - \binom{r}{2}$. We claim that (G, r) is a yes-instance of Clique if and only if the constructed instance of Robust Subspace Recovery is a yes-instance.

To reformulate the objective of Robust Subspace Recovery, an instance $(\mathbf{A}, r, k)$ is a yes-instance if and only if there is a subset of $n - k = \binom{r}{2}$ rows of $\mathbf{A}$ with rank at most r. The following claim shows how to identify the rank just by the structure of the edges corresponding to the selected rows.

Claim 9.10. *The rank of any submatrix $\mathbf{A}' \subset \mathbf{A}$ obtained by deleting rows from $\mathbf{A}$ is*

$$\max(d, |V(G(\mathbf{A}'))| - \textit{number of bipartite graphs among } \mathsf{cc}(G(\mathbf{A}'))),$$

where $G(\mathbf{A}')$ is the subgraph of G such that its edges are exactly the edges corresponding to the rows of $\mathbf{A}'$, and its vertex set is the set of all endpoints of the edges, and $\mathsf{cc}(G(\mathbf{A}'))$ is the set of connected components of $G(\mathbf{A}')$.

Proof. We will modify $\mathbf{A}'$ to $\operatorname{vect}(F)$ for a certain forest F, by using elemental row operations which do not change rank. Then Claim 9.8 finishes the proof.

The modification is performed on each connected component of $G(\mathbf{A}')$ independently. Consider a set of rows $\mathbf{C}$ which corresponds to the edges of a connected component in $G(\mathbf{A}')$. If this component contains an odd cycle, then from the rows of $\mathbf{C}$ we can obtain all the underlying elements of W by elimination: start from the row corresponding to one edge of a cycle, iteratively subtract/add subsequent edges of the cycle. In the end we are left with $2w$, where w is a vector in W which corresponds to the vertex of the cycle we started the elimination from. After multiplication by $1/2$ we have one of the vertex vectors. Now by consequently subtracting from the edge vectors along a spanning tree of $G(\mathbf{C})$ we can obtain all the vertex vectors of $V(G(\mathbf{C}))$. After zeroing out the remaining edge vectors $\mathbf{C}$ is $(\mathbf{w}_i | v_i \in V(G(\mathbf{C})))$, up to permuting the rows and appending zero rows.

In the other case, if $G(\mathbf{C})$ is bipartite, consider the matrix $\mathbf{S} \subset \mathbf{C}$ which corresponds to the edges of a spanning tree T of $G(\mathbf{C})$. Let uv be an edge of $G(\mathbf{C})$ which is not in T. Since $G(\mathbf{C})$ is bipartite, u and v are connected by a path in T with odd number of edges. Then by consequently adding/subtracting the edge vectors of this path we can obtain the edge vector corresponding to uv, in the same fashion as in the previous case. Thus we can zero out all rows of $\mathbf{C}$ except for $\mathbf{S}$.

After dealing with each component, consider the forest F where $V(F) = V(G(\mathbf{A}'))$, and $E(F)$ is the union of the edges of all spanning trees picked during the modification in the bipartite components. Thus, vertices of all the non-bipartite components are isolated in F. We claim that the set of non-zero rows of matrix $\mathbf{B}$ obtained from $\mathbf{A}'$ by the modifications above is exactly $\operatorname{vect}(F)$. Indeed, for each non-bipartite component we obtained all the vertex vectors while zeroing out everything else, and for each bipartite component we kept only the edge vectors of the corresponding spanning tree. Now, since modifications are rank-preserving,

$$\operatorname{rank}(\mathbf{A}') = \operatorname{rank}(\mathbf{B}) = \operatorname{rank}(\operatorname{vect}(F)) = \min(d, |\operatorname{vect}(F)|)$$

by Claim 9.8. The size of $\operatorname{vect}(F)$ by definition is the number of isolated vertices in F plus the number of edges in F, and that is equal to the number of vertices in F minus the number of non-trivial connected components in F, since F is a forest. Finally, we observe that $|V(F)| = |V(G(\mathbf{A}'))|$, and bipartite components of $G(\mathbf{A}')$ are in one-to-one correspondence with non-empty components of F, finishing the proof of Claim 9.10.

□

With Claim 9.10 proven, we show that the only way to keep at least $\binom{r}{2}$ edge vectors in such a way that the rank is at most r, is to select the rows corresponding to the edges of an r-clique. For any $\mathbf{A}' \subset \mathbf{A}$, let $\kappa(\mathbf{A}') = \frac{|\mathbf{A}'|}{\operatorname{rank}\mathbf{A}'}$, the number of edges per unit of rank. We claim that κ is strictly maximized on an r-clique over all $\mathbf{A}' \subset \mathbf{A}$ which have rank at most r.

For a matrix $\mathbf{K}$ corresponding to an r-clique, $\kappa(\mathbf{K}) = \frac{r-1}{2}$ by Claim 9.10. Consider any $\mathbf{A}' \subset \mathbf{A}$ such that $G(\mathbf{A}')$ is connected. Since $\text{rank}(\mathbf{A}')$ is at most r, there are two possibilities by Claim 9.10. If $G(\mathbf{A}')$ is a non-bipartite graph on at most r vertices, it has less edges than an r-clique, and so $\kappa(\mathbf{A}') < \frac{r-1}{2}$. Otherwise $G(\mathbf{A}')$ is bipartite, and $|V(G(\mathbf{A}'))|$ must be $r+1$. Then there are at most $\left(\frac{r+1}{2}\right)^2$ edges, so $\kappa(\mathbf{A}') < \frac{r-1}{2}$ for $r \geq 4$.

To prove the statement for any $\mathbf{A}' \subset \mathbf{A}$ such that $\text{rank}(\mathbf{A}') \leq r$, we do an induction on the number of connected components in $G(\mathbf{A}')$. The base case when there is only one connected component is already proven. Now, consider $\mathbf{A}' = \mathbf{B} \cup \mathbf{C}$ where $G(\mathbf{C})$ is connected. By Claim 9.10, $\text{rank}(\mathbf{B} \cup \mathbf{C}) = \text{rank}(\mathbf{B}) + \text{rank}(\mathbf{C})$, and also $|\mathbf{A}'| = |\mathbf{B}| + |\mathbf{C}|$. By the induction, $|\mathbf{B}|/\text{rank}(\mathbf{B}) < \frac{r-1}{2}$, and $|\mathbf{C}|/\text{rank}(\mathbf{C}) < \frac{r-1}{2}$, so

$$\kappa(\mathbf{A}') = \frac{|\mathbf{B}| + |\mathbf{C}|}{\text{rank}(\mathbf{B}) + \text{rank}(\mathbf{C})} < \frac{r-1}{2}.$$

Thus, the rows selected have rank r if and only if they correspond to an r-clique, which proves the correctness of the reduction.

By [51], see also [70], assuming ETH, there is no algorithm solving Clique in time $f(r) \cdot n^{o(r)}$ where r is the size of the clique, for any computable function f of r. Since in our reduction $d = r + 1$, the theorem follows.[3] □

Finally, for completeness we give the proof of Corollary 9.6, restated here.

Corollary 9.6. *Assuming ETH, there is no algorithm approximating* PCA with Outliers *with any multiplicative guarantee in time* $f(d) \cdot n^{o(d)}$.

Proof. An algorithm described in the statement could distinguish between $OPT \leq D$ and $OPT > \alpha \cdot D$ for given D, where α is the approximation guarantee which may depend on the input instance. Then this algorithm could also distinguish between $OPT = 0$ and $OPT > 0$, violating Theorem 9.4. □

[3]The same reduction also shows that Robust Subspace Recovery is W[1]-hard when parameterized by $(d, n-k)$.

10

Dimensionality Reduction for Robust PCA

In this chapter, we tackle again the PCA WITH OUTLIERS problem. Complementing the algebraic geometry methods of the previous chapter, here we focus on various dimensionality reduction techniques. In contrast to the "exact" algorithm given by Theorem 9.3, here we obtain either $(1+\varepsilon)$-approximation, or exact algorithms under certain separating conditions on outliers. As opposed to Theorem 9.3, the running time of all the algorithms presented in this chapter depends only polynomially on d, while the exponential dependence is, instead, in r.

First, we show that PCA WITH OUTLIERS admits a randomized Polynomial Time Approximation Scheme (PTAS) when the dimension r of the solution subspace is a fixed constant.

Theorem 10.1. *For every $\varepsilon > 0$, an $(1+\varepsilon)$-approximate solution to* PCA WITH OUTLIERS *can be found in time $n^{\mathcal{O}(\frac{r \log r}{\varepsilon^2})} \cdot d^{\mathcal{O}(1)}$.*

Since $r \leq d$, Theorem 9.4 shows that, in particular, a running time of $n^{\Omega(r)}$ is essential for any approximation algorithm. Thus up to the $\log r$ factor in the exponent of n, the running time of our PTAS is tight.

We also provide algorithms for solving PCA WITH OUTLIERS exactly. While the nature of outliers can be elusive, we make two natural assumptions on outliers. It appears that these assumptions can be very useful from the algorithmic perspective. The intuition behind the first assumption is that every outlier is further from some optimal solution than any inlier. The second assumption is stronger, it assumes that the squared distance from any outlier to the solution subspace is larger than the sum of all inliers' squared distances to the subspace.

Definition 10.2 (α**-gap and** α**-heavy assumptions**). *For $\alpha > 0$, the* α-gap assumption *about instance $(\mathbf{A}, r, k)$ of* PCA WITH OUTLIERS *is that there is an optimal solution $(\mathbf{L}, \mathbf{N})$ with the following properties. Denote the rows of $\mathbf{A}$ by $\mathbf{a}_1, \ldots, \mathbf{a}_n$, let O be the indices of outliers, that is, the indices of non-zero rows of $\mathbf{N}$, and $I = [n] \setminus O$,*

the indices of inliers. Then for every $i \in O$ and $j \in I$,

$$\operatorname{dist}^2(\mathbf{a}_i, \mathbf{V}^*) > (1+\alpha) \cdot \operatorname{dist}^2(\mathbf{a}_j, \mathbf{V}^*),$$

where $\mathbf{V}^*$ is the r-dimensional subspace which spans the rows of $\mathbf{L}$. Similarly, the α-heavy assumption *about $(\mathbf{A}, r, k)$ is that for every $i \in O$,*

$$\operatorname{dist}^2(\mathbf{a}_i, \mathbf{V}^*) > (1+\alpha) \sum_{j \in I} \operatorname{dist}^2(\mathbf{a}_j, \mathbf{V}^*)$$

For example, if all inliers lie in $\mathbf{V}^*$, as in Robust Subspace Recovery, then such an instance satisfies both α-gap and α-heavy assumptions. Thus both "α-assumptions" create parameterized classes of problems "between" PCA with Outliers (no assumptions on the outliers) and Robust Subspace Recovery (all inliers belong to a subspace and all outliers do not).

Theorem 10.3. *For constant α, there exists a randomized algorithm for* PCA with Outliers *that in time*

$$2^{\mathcal{O}(r(\log k + \log\log n)(r + \log n + \log(1/\delta)))}(nd)^{\mathcal{O}(1)}$$

under α-gap assumption outputs an optimal solution with success probability $1-\delta$.

Theorem 10.4. *For constant α, there exists a randomized algorithm for* PCA with Outliers *that in time*

$$2^{\mathcal{O}(r^2(\log k + \log\log n)(\log k + \log\log n + \log(1/\delta)))}(nd)^{\mathcal{O}(1)}$$

under α-heavy assumption outputs an optimal solution with success probability $1-\delta$
.

Observe that the running time of our nearly-tight $(1+\varepsilon)$-approximation algorithm for PCA with Outliers given by Theorem 10.1 can be represented as $2^{\mathcal{O}_\varepsilon(r \log r \log n)}$. Essentially, Theorem 10.3 gives an analogue of that for solving the problem exactly in the α-gap case. The running time of Theorem 10.4 is also similar in the general case. Moreover, if k grows sufficiently slower than n, then this running time is asymptotically better than that of Theorem 10.1, as the former depends only polynomially on $\log k$ and $\log\log n$ in the exponent. Thus, in the regime where the α-heavy assumption holds and k is small enough, it is possible to overcome the lower bound given by Theorem 9.4.

The proofs of both Theorems 10.3 and 10.4 are based on the following strategy: apply randomized dimensionality reduction (sketching), and then use the methods of algebraic geometry to compute the exact solution. The difficulty is that in general, the dimensionality reduction distorts distances between the points. The main technical contribution of this chapter is the proof that under α-gap and α-heavy assumptions, carefully selected sketches still can be used to obtain exact solutions. We believe that these ideas could find applications beyond robust PCA problems.

Finally, for Robust Subspace Recovery we prove the following theorem.

Theorem 10.5. Robust Subspace Recovery *is solvable in time*

$$2^{\mathcal{O}(k(\log r+\log k))}(nd)^{\mathcal{O}(1)}.$$

Note that when k is small, this running time improves considerably all the theorems above in this special case of PCA with Outliers. However, when k grows, the performance of Theorem 10.5 gets worse, as its running time depends exponentially on k, while for Theorems 10.3 and 10.4 the exponential dependency is only on $\mathsf{poly}(\log k)$. The proof of Theorem 10.5 is essentially by a combination of Theorem 10.1 and known kernelization methods for the Matrix Rigidity problem given by Fomin et al. [91].

10.1 Approximation Scheme for PCA with Outliers

In this section, we present the $n^{\mathcal{O}(r\log r\varepsilon^{-2})}$ time algorithm for solving the general case of PCA with Outliers with $(1+\varepsilon)$-factor approximation, as claimed by Theorem 10.1. First, we give a high-level overview of the algorithm. The general idea is to observe that the unknown rows of $\mathbf{A}$ that form the inlier submatrix can be well approximated by a small-sized sample of them, in terms of low-rank approximation. This follows from established results saying that one can obtain a projection-cost preserving sketch of a matrix by sampling and reweighting a few of its rows in proportion to a certain modification of their leverage scores. In particular, we employ the result of [61] stating that sampling $\mathcal{O}(r\log r\varepsilon^{-2})$ rows in accordance with their ridge leverage scores provides a $(1+\varepsilon)$ approximation for any rank-r orthogonal projection.

Note that we do not know the actual inlier matrix to compute the scores and to perform the sampling from, as an arbitrary set of k rows of the given matrix might be outliers. However, we can guess the particular rows from a successful sample and also guess the approximated ridge leverage scores so that the optimal low-rank approximation of the resulting small matrix will also approximate well the unknown inlier matrix. Here we use crucially that constant-factor overestimates of the ridge leverage scores still suffice for the result of [61]. After this, it is only a matter of greedily selecting the rows of $\mathbf{A}$ that are the closest to the computed low-dimensional approximation space. The above summarizes the intuition behind Theorem 10.1, and the detailed proof follows next. We restate the theorem here for convenience.

Theorem 10.1. *For every $\varepsilon>0$, an $(1+\varepsilon)$-approximate solution to* PCA with Outliers *can be found in time $n^{\mathcal{O}(\frac{r\log r}{\varepsilon^2})}\cdot d^{\mathcal{O}(1)}$.*

First, we recall the known results about row sampling. We will use the ridge leverage score construction due to [61]. For a matrix $\mathbf{A}\in\mathbb{R}^{n\times d}$ and an index $i\in[n]$ the *i-th ridge leverage score of* $\mathbf{A}$ is given as

$$\tau_i(A)=\mathbf{a}_i(\mathbf{A}^\top\mathbf{A}+\lambda\mathbf{I}_d)^+\mathbf{a}_i^\top,$$

where $\lambda = ||\mathbf{A} - \mathbf{A}_r||_F^2/k$, and $+$ denotes the Moore–Penrose pseudoinverse of a matrix. The following statement about sampling w.r.t. to ridge leverage scores is proven in [61].

Theorem 10.6 (Theorem 6 in [61])**.** *For $i \in [n]$, let $\tilde{\tau}_i \geq \tau_i(\mathbf{A})$ be an overestimate for the i-th ridge leverage score. Let $p_i = \frac{\tilde{\tau}_i}{\sum_i \tilde{\tau}_i}$. Let $t = \frac{c \log(r/\delta)}{\varepsilon^2} \sum_i \tilde{\tau}_i$ for any $\varepsilon > 0$ and some sufficiently large constant c. Construct $\mathbf{C}$ by sampling t rows of $\mathbf{A}$, each set to $\frac{1}{\sqrt{tp_i}}\mathbf{a}_i$ with probability p_i. With probability $1 - \delta$, for any rank r orthogonal projection $\mathbf{X}$,*

$$(1-\varepsilon)||\mathbf{A} - \mathbf{A}\mathbf{X}||_F^2 \leq ||\mathbf{C} - \mathbf{C}\mathbf{X}||_F^2 \leq (1+\varepsilon)||\mathbf{A} - \mathbf{A}\mathbf{X}||_F^2.$$

Observe that Theorem 10.6 essentially provides a projection-cost preserving sketch for $\mathbf{A}$, as per Definition 4.5. The only difference is that for the purpose of this chapter, it is more convenient to sketch the matrix from the left, as we need to select out rows.

Our objective now is to guess certain ridge leverage score overestimates. For that, we will need a lemma bounding the range of ridge leverage scores, following from [61].

Lemma 10.7.

$$\frac{1}{2} \leq \sum_{i=1}^{n} \tau_i(\mathbf{A}) \leq 2r.$$

Proof. The upper bound is precisely given by Lemma 4 in [61]. For the lower bound,

$$\begin{aligned}
\sum_{i=1}^{n} \tau_i(\mathbf{A}) &= \sum_{i=1}^{n} \frac{\sigma_i(\mathbf{A})^2}{\sigma_i(\mathbf{A})^2 + \frac{||\mathbf{A}-\mathbf{A}_r||_F^2}{r}} \\
&= \sum_{i=1}^{n} \frac{\sigma_i(\mathbf{A})^2}{\sigma_i(\mathbf{A})^2 + \frac{1}{r}\sum_{j=r+1}^{n} \sigma_j(\mathbf{A})^2} \\
&\geq \sum_{i=r+1}^{n} \frac{\sigma_i(\mathbf{A})^2}{\left(1+\frac{1}{r}\right)\sum_{j=r+1}^{n} \sigma_j(\mathbf{A})^2} \geq \frac{1}{1+\frac{1}{r}} \geq \frac{1}{2},
\end{aligned}$$

where the first equality is given by the proof of Lemma 4 in [61], and then we lower bound the first r terms of the sum by zero. □

Now we are ready to prove the main result of this section.

Proof of Theorem 10.1. The algorithm proceeds as follows. Set $T = 6r$, a sufficiently small $\varepsilon_0 < \varepsilon$ (to be defined later), and $t = \frac{c \log(2r)}{\varepsilon_0^2} T$ where c is a sufficiently large constant from the statement of Theorem 10.6. First, guess t indices $\{i_1, \ldots i_t\}$ in $[n]$,

each corresponding to a row in $\mathbf{A}$. For each index i in $\{i_1, \ldots, i_t\}$, guess a value $\tilde{\tau}_i$ from the set $\mathcal{T} = \{\frac{2r}{2^q}, \frac{2r}{2^{p-1}}, \ldots, 2r\}$, where q is the smallest integer such that $2^q \geq n$. Compose the matrix $\mathbf{C} \in \mathbb{R}^{t \times d}$ from the rows of $\mathbf{A}$: for each $j \in [t]$, take the i_j-th row of $\mathbf{A}$ multiplied by $\frac{1}{\sqrt{t\tilde{\tau}_i}}$. By using the standard PCA algorithm, find in polynomial time the optimal low-rank approximation of $\mathbf{C}$, i.e. the rank r orthogonal projection matrix $\mathbf{X} \in \mathbb{R}^{d \times d}$ minimizing $||\mathbf{C} - \mathbf{C}\mathbf{X}||_F^2$. Construct the matrix $\mathbf{N} \in \mathbb{R}^{n \times d}$ such that it contains k rows of $\mathbf{A}$ maximizing the distance to the r-dimensional subspace corresponding to $\mathbf{X}$, at the respective positions of these rows in $\mathbf{A}$, and all the other rows of $\mathbf{N}$ are zero rows. Set $\mathbf{L}$ to be $(\mathbf{A} - \mathbf{N})\mathbf{X}$. Finally, return $\mathbf{L}$ and $\mathbf{N}$ minimizing the value $||\mathbf{A} - \mathbf{L} - \mathbf{N}||_F^2$ over all guesses performed by the algorithm.

Correctness of the algorithm. Clearly, the matrices $\mathbf{L}$ and $\mathbf{N}$ returned by the algorithm are subject to the constraints that rank$\mathbf{L} \leq r$ and $\mathbf{N}$ contains at most k non-zero rows, thus it only remains to prove that the cost of the returned solution is at most $(1 + \varepsilon)$ times the cost of the optimal solution. Fix an optimal solution $(\mathbf{L}^*, \mathbf{N}^*)$. Denote by $\mathbf{A}^* = \mathbf{A} - \mathbf{N}^*$ the optimal inlier matrix, that is, the matrix $\mathbf{A}$ where the outlier rows are replaced by zero rows. Recall that for $i \in [n]$, $\tau_i(\mathbf{A}^*)$ is the i-th ridge leverage score of $\mathbf{A}^*$. For each $i \in [n]$, denote by $\tilde{\tau}_i^*$ the smallest element of $\mathcal{T}$ that is at least $\tau_i(\mathbf{A}^*)$, $\tilde{\tau}_i^*$ is well-defined since $\tau_i(\mathbf{A}^*)$ is at most $2r$ by Lemma 10.7, and the set $\mathcal{T}$ contains $2r$. We show now that $\sum_{i=1}^n \tilde{\tau}_i$ is at most T, and thus the number of the sampled rows $t = \frac{c \log(r/\delta)}{\varepsilon_0^2} T$ is sufficiently large to apply Theorem 10.6. For each $i \in [n]$ consider two cases. First, if $\tau_i(\mathbf{A}^*)$ is at least the smallest element of $\mathcal{T}'$, then $\tilde{\tau}_i \leq 2\tau_i(\mathbf{A}^*)$, as elements of the set $\mathcal{T}'$ are at factor two from each other. Over all such indices, $\sum_i \tilde{\tau}_i \leq 2\sum_{i=1}^n \tau_i(\mathbf{A}^*) \leq 4r$ by Lemma 10.7. Second, if $\tau_i(\mathbf{A}^*)$ is less than $\frac{2r}{2^q}$, then $\tilde{\tau}_i$ is set to this value, and the sum of all such $\tilde{\tau}_i$ is at most $2r$ as there are $n \leq 2^q$ values in total. Summing the bounds for both cases, we get that $\sum_{i=1}^n \tilde{\tau}_i \leq 6r = T$.

Now denote $\delta = 1/2$ and invoke Theorem 10.6 for $\mathbf{A}^*$ with the set values of δ, t, and $\tilde{\tau}_i$ for each $i \in [n]$, and with the error parameter ε_0. With probability $1 - \delta$, the sampling procedure described in the statement of Theorem 10.6 succeeds. Since this probability is positive, there exists a particular selection of t rows that produces the desired matrix. Thus the matrix $\mathbf{C}^*$ composed of these rows and reweighted according to Theorem 10.6, satisfies

$$(1 - \varepsilon_0)||\mathbf{A}^* - \mathbf{A}^*\mathbf{X}||_F^2 \leq ||\mathbf{C}^* - \mathbf{C}^*\mathbf{X}||_F^2 \leq (1 + \varepsilon_0)||\mathbf{A}^* - \mathbf{A}^*\mathbf{X}||_F^2, \tag{10.1}$$

for any rank r orthogonal projection matrix $\mathbf{X} \in \mathbb{R}^{d \times d}$. Denote the indices of these rows by $i_1^*, \ldots, i_t^*$. In one of the branches, our algorithm considers the values $i_1 = i_1^*$, $\ldots$, $i_t = i_t^*$, and $\tilde{\tau}_i = \tilde{\tau}_i^*$ for all $i \in \{i_1, \cdots, i_t\}$. Thus the matrix $\mathbf{C}$ constructed by our algorithm at this step is exactly the matrix $\mathbf{C}^*$ where every entry is multiplied by $1/\sqrt{\sum_{i=1}^n \tilde{\tau}_i}$. Consider the orthogonal projection matrix $\mathbf{X} \in \mathbb{R}^{d \times d}$ that provides the optimal rank k approximation of $\mathbf{C}$, and also of $\mathbf{C}^*$ since these two matrices are identical up to multiplying by a constant. Let $\mathbf{X}^*$ be the projection matrix of the optimal solution, that is, $\mathbf{L}^* = \mathbf{A}^*\mathbf{X}^*$. Then by (10.1), and because $\mathbf{X}$ gives the best

low-rank approximation for $\mathbf{C}^*$, we have that

$$\begin{aligned} ||\mathbf{A}^* - \mathbf{A}^*\mathbf{X}||_F^2 &\leq \frac{1}{1-\varepsilon_0}||\mathbf{C}^* - \mathbf{C}^*\mathbf{X}||_F^2 \\ &\leq \frac{1}{1-\varepsilon_0}||\mathbf{C}^* - \mathbf{C}^*\mathbf{X}^*||_F^2 \\ &\leq \frac{1+\varepsilon_0}{1-\varepsilon_0}||\mathbf{A}^* - \mathbf{A}^*\mathbf{X}^*||_F^2 \leq \frac{1+\varepsilon_0}{1-\varepsilon_0}OPT. \end{aligned} \tag{10.2}$$

Finally, denote $\mathbf{A}' = \mathbf{A} - \mathbf{N}$. Let us note that both $\mathbf{A}'$ and $\mathbf{A}^*$ contain certain $(n-k)$ rows of $\mathbf{A}$, but $\mathbf{A}'$ contains precisely the $(n-k)$ rows that incur the smallest loss w.r.t. $\mathbf{X}$. Therefore,

$$||\mathbf{A}' - \mathbf{A}'\mathbf{X}||_F^2 \leq ||\mathbf{A}^* - \mathbf{A}^*\mathbf{X}||_F^2. \tag{10.3}$$

Since $\mathbf{L}$ is exactly $\mathbf{A}'\mathbf{X}$, by (10.2) and (10.3), we have

$$||\mathbf{A} - \mathbf{L} - \mathbf{N}||_F^2 \leq \frac{1+\varepsilon_0}{1-\varepsilon_0}OPT.$$

Setting $\varepsilon_0 = \Theta(\varepsilon)$ so that $\frac{1+\varepsilon_0}{1-\varepsilon_0}$ is at most $1+\varepsilon$, concludes the proof of correctness.

Running time. The algorithm consider n choices for each of the t values $i_1, \ldots, i_t$, and $\mathcal{O}(\log n)$ choices for each of the t values $\tilde{\tau}_{i_1}, \ldots, \tilde{\tau}_{i_t}$, where $t = O(r \log r/\varepsilon^2)$. For each choice, the optimal low-rank approximation and the outliers are found in polynomial time. Thus, the total running time of the algorithm is upper-bounded by

$$(n \log n)^{\mathcal{O}(t)}(nd)^{\mathcal{O}(1)} = n^{\mathcal{O}(r \log r/\varepsilon^2)}d^{\mathcal{O}(1)}.$$

□

10.2 Robust Subspace Recovery Algorithm

In this section we prove Theorem 10.5 that Robust Subspace Recovery is solvable in time $2^{\mathcal{O}(\min\{k, r\log r\}\cdot(\log r + \log k))}\cdot(nd)^{\mathcal{O}(1)}$. The algorithm we give is almost identical to the algorithm of [91] for the Matrix Rigidity problem. We provide here full details for completeness.

Let us remind that in Robust Subspace Recovery, we are given a matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$, whose rows correspond to the data points, and integers r, k. The question is whether there are matrices $\mathbf{L}$ and $\mathbf{N}$ such that $\mathbf{A} = \mathbf{L} + \mathbf{N}$, the rank of $\mathbf{L}$ is at most r, and $\mathbf{N}$ has at most k non-zero rows. Equivalently, the question is whether it is possible to delete at most k rows of $\mathbf{A}$ such that the resulting matrix is of rank at most r.

We need the following observation: For every set X of $r+1$ independent rows of matrix $\mathbf{A}$, at least one row from X is an outlier. In other words,

Proposition 10.8. *Let X be a set of indices of $r+1$ independent rows of $\mathbf{A}$. Then for every optimal solution $(\mathbf{L}, \mathbf{N})$, at least one index from X is the index of a non-zero row of $\mathbf{N}$.*

Proof. The rank of $\mathbf{L}$ is at most r, thus $\mathbf{L}$ cannot contain more than r independent rows. □

The crux of the algorithm is in the procedure that for an input $(\mathbf{A}, r, k)$ of Robust Subspace Recovery in polynomial time constructs an equivalent instance $(\tilde{\mathbf{A}}, r, k)$, with matrix $\tilde{\mathbf{A}}$ containing at most $(r+1)(k+1)$ rows. We assume that the rank of $\mathbf{A}$ is more than r, otherwise $\mathbf{L} = \mathbf{A}$ is trivially a solution. We also assume that $n > (r+1)(k+1)$ because otherwise we can put $\tilde{\mathbf{A}} = \mathbf{A}$. The procedure runs in two steps.

First, we find pairwise disjoint sets $R_1, \ldots, R_t$ of rows in $\mathbf{A}$. Each set R_i consists of $r+1$ independent rows. We construct such sets greedily by picking a set of $r+1$ independent rows and deleting them from $\mathbf{A}$ until the rank of the remaining rows will be at most r. By Proposition 10.8, each of these sets R_i contains at least one outlier. Thus if $t > k$, the rank of $\mathbf{A}$ cannot be reduced to r by deleting k rows, hence $(\mathbf{A}, r, k)$ is a no-instance.

Second, from the remaining rows of $\mathbf{A}$, that is, the rows that are not in $R_1 \cup \cdots \cup R_t$, we select pairwise disjoint sets $R_{t+1}, \ldots, R_{k+1}$ as follows. For $i \geq t+1$ let M_i be the rows of $\mathbf{A}$ that are not in $R_1 \cup \cdots \cup R_{i-1}$. Then R_i is a subset of M_i forming its basis. Note that $|R_i| \leq r$ for $i \geq t+1$.

Finally, the matrix $\tilde{\mathbf{A}}$ is the matrix whose rows are $R_1 \cup \cdots \cup R_{k+1}$. At every step of the construction of $\tilde{\mathbf{A}}$ we find an independent set of rows and thus the total running time is polynomial. Since every set R_i contains at most $r+1$ row, matrix $\tilde{\mathbf{A}}$ contains at most $(r+1) \cdot (k+1)$ rows. Thus what remains is to show the equivalence of both instances.

Lemma 10.9. *$(\mathbf{A}, r, k)$ is a yes-instance of* Robust Subspace Recovery *if and only if $(\tilde{\mathbf{A}}, r, k)$ is a yes-instance.*

Proof. In one direction the proof is trivial. If the rank of $\mathbf{A}$ can be reduced to r by deleting at most k rows, the same is true for $\tilde{\mathbf{A}}$.

For the opposite direction. Let O be the set of outliers, that is, the set of rows of $\tilde{\mathbf{A}}$ of size at most k whose removal decreases the rank of the matrix down to r. The rows of matrix $\mathbf{A}$ are partitioned into the rows of $\tilde{\mathbf{A}}$ and the remaining rows, which we denote by M. (By slightly abusing notation, we do not distinguish a matrix and a set of rows forming this matrix.) We claim that removing rows of O from $\mathbf{A}$ also reduces its rank to r. Targeting a contradiction, let us assume that this is not true. Then $\mathbf{A} \setminus O$ contains a set X of $r+1$ linearly independent rows.

The rows of $\tilde{\mathbf{A}}$ are $R_1 \cup \cdots \cup R_t \cup \cdots \cup R_{k+1}$. Because $|O| \leq k$, by the pigeonhole principle, there is R_i that does not contain a row from O. For $i \leq t$, each R_i consists of $r+1$ independent rows, and by Proposition 10.8 must contain at least one row

from O. This means that at least one R_i, $i > t$, contains no row from O. But by the construction of sets R_i for $i > t$, the set of rows M is in the span of R_i. Hence the set $X' = (X \cap \tilde{\mathbf{A}}) \cup R_i \subseteq \tilde{\mathbf{A}} \setminus O$ contains $r+1$ linearly independent rows of $\tilde{\mathbf{A}} \setminus O$, which is a contradiction. □

Finally, to prove Theorem 10.5, for input $(\mathbf{A}, r, k)$, we construct an equivalent instance $(\tilde{\mathbf{A}}, r, k)$, where $\tilde{\mathbf{A}}$ contains at most $(r+1)(k+1)$ rows. For each subset of rows of $\tilde{\mathbf{A}}$ of size k, we check whether removal of this set results in a matrix of rank at most r. If we found such a set O, by Lemma 10.9, the same rows are outliers for $\mathbf{A}$ as well. If we did not find a set of outliers for $\tilde{\mathbf{A}}$, we can safely conclude that $(\mathbf{A}, r, k)$ is a no-instance. Construction of the reduced instance can be done in polynomial time, and the number of all subsets of rows of $\tilde{\mathbf{A}}$ of size k, does not exceed $\binom{(r+1)(k+1)}{k} = 2^{\mathcal{O}(k(\log r + \log k))}$. Hence the total running time is $2^{\mathcal{O}(k(\log r+\log k))} \cdot (nd)^{\mathcal{O}(1)}$.

Alternatively, instead of trying all subsets of k rows of $\tilde{\mathbf{A}}$, we can run the algorithm of Theorem 10.1 on the instance $(\tilde{\mathbf{A}}, r, k)$ with the error parameter ε set to an arbitrary constant. Recall that by Theorem 10.1 an $(1+\varepsilon)$-approximate solution to PCA WITH OUTLIERS can be found in time $n^{\mathcal{O}(\frac{r \log r}{\varepsilon^2})} \cdot d^{\mathcal{O}(1)}$. Since an instance of ROBUST SUBSPACE RECOVERY is a yes-instance if and only if it has the objective value of zero as an instance of PCA WITH OUTLIERS, a constant-factor approximation for PCA WITH OUTLIERS suffices for solving ROBUST SUBSPACE RECOVERY exactly. Thus the whole algorithm for ROBUST SUBSPACE RECOVERY finishes in time $2^{\mathcal{O}(r \log r(\log r+\log k))}(nd)^{\mathcal{O}(1)}$, as there are at most $(r+1)(k+1)$ rows in the matrix $\tilde{\mathbf{A}}$. The running time of $2^{\mathcal{O}(\min\{k, r\log r\}\cdot(\log r+\log k))} \cdot (nd)^{\mathcal{O}(1)}$ follows by combining the above two algorithms.

10.3 α-heavy and α-gap PCA with Outliers

In this section, we present Theorems 10.3 and 10.4, their proofs are done in two steps. First, we show an $2^{\mathcal{O}(\log k+\log\log n)rd}\mathsf{poly}(n, d, 1/\delta)$ subspace-sampling algorithm for α-gap and α-heavy instances of PCA WITH OUTLIERS which succeeds with probability $1-\delta$, building upon the $n^{\mathcal{O}(rd)}$ algorithm given by Theorem 9.3. Second, we get rid of the exponential dependence on d by using dimensionality reduction techniques.

10.3.1 Subspace-sampling Algorithm

We start with briefly recalling the idea of the algorithm from Theorem 9.3. One can parameterize the unknown r-dimensional subspace by $r \times d$ variables, and then for every pair of points construct a polynomial in these variables such that its sign determines the farthest point from a subspace. Thus the signs of all the $\binom{n}{2}$ polynomials determine exactly which k points are the farthest from the subspace, and

so are the outliers. By enumerating all possible sign conditions via Theorem 4.12 in time $n^{\mathcal{O}(rd)}$, we get all potential sets of outliers.

The main idea for the new algorithm is as follows. We show that it is possible to replace the trivial $\binom{n}{2}$-sized polynomial system by a much smaller one that still allows to detect the outliers, provided that either the α-gap or the α-heavy assumption holds. Intuitively, we first partition the points into $m = \Theta(k)$ buckets such that there is at most one outlier in each bucket w.h.p. Then, we compose $\binom{m}{2}$ polynomials to determine the k buckets containing the outliers, and for each bucket we also construct $\log n$ polynomials to detect the outlier in the bucket. Thus, our system contains only $\mathsf{poly}(k \log n)$ polynomials, giving the desired running time.

Now we give the proof in full detail. Denote the rows of $\mathbf{A}$ by $\mathbf{a}_1, \ldots, \mathbf{a}_n$. The algorithm, later denoted as Subspace-sampling algorithm, proceeds as follows.

(i) Partition the rows of $\mathbf{A}$ (the points) into m buckets using perfect hashing. Let B_1, B_2,...,$B_m \subset [n]$ be the indices of points in each bucket.

(ii) For each bucket B_i, $i \in [m]$, construct the set of polynomials $\mathcal{P}_i = \{P_i^j\}_{1 \le j \le \log_2 n}$,

$$P_i^j(\mathbf{V}) = \sum_{\substack{\ell \in B_i \\ \ell_j = 1}} (\mathrm{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}))^b - \sum_{\substack{\ell \in B_i \\ \ell_j = 0}} (\mathrm{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}))^b$$

where ℓ_j is the j-th bit in the binary representation of ℓ and $b = \Theta(\frac{\log n}{\log(1+\alpha)})$ for α-gap instances and $b = 1$ for α-heavy instances. Note that $\mathbf{V}$ can be parameterized by $r \times d$ variables such that for each $\ell \in [n]$, $\mathrm{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V})$ can be expressed as a constant-degree polynomial in these variables. See Section 9.1 for details.

(iii) Consider also the set of polynomials $\mathcal{Q} = \{Q_{i,j}\}_{1 \le i < j \le m}$, where

$$Q_{i,j}(\mathbf{V}) = \sum_{\ell \in B_i} (\mathrm{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}))^b - \sum_{\ell \in B_j} (\mathrm{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}))^b$$

and let $\mathcal{P}$ be the collection of all the polynomials defined above, i.e., $\mathcal{P} = \bigcup_i \mathcal{P}_i \cup \mathcal{Q}$.

(iv) Using Theorem 4.12, enumerate all possible sign conditions of $\mathcal{P}$ on $\mathcal{X}$, the space of all r-dimensional subspaces, i.e. compute the set $\mathcal{T} = \mathrm{Sample}(\mathbf{A}, r, k)$ where

$$\mathrm{Sample}(\mathbf{A}, r, k) = \{(S, \mathbf{V}_S) | S \text{ is a sign condition of } \mathcal{P} \text{ on } \mathcal{X} \text{ and } \mathbf{V}_S \text{ realizes } S\}$$

(v) For each $(S, \mathbf{V}_S) \in \mathcal{T}$, do the following:

(a) Note that the signs of $\mathcal{Q}$ give an ordering on $[m]$. Take the top k indices from this ordering and let WLOG this set be $[k]$.

(b) For each B_i, $i \in [k]$, choose $p \in B_i$ such that p_j is set to 1 if $P_i^j(\mathbf{V}_S) > 0$ and 0 otherwise. This gives us k rows of $\mathbf{A}$, one from each bucket, let $\mathbf{N} \in \mathbb{R}^{n\times d}$ be the matrix containing these k rows at their respective positions in $\mathbf{A}$.

(c) Find the optimal r-dimensional projection $\mathbf{L}$ of $\mathbf{A} - \mathbf{N}$ via the vanilla PCA algorithm.

(vi) Return $\mathbf{N}$ and $\mathbf{L}$ from step 5 with the minimum cost of projection.

Running time. The sampling step 4 dominates the running time. Since $|\mathcal{P}| = m \log n + \binom{m}{2} = O(k \log n + k^2)$ and the degree of all the polynomials involved in $\mathcal{P}$ is bounded by $\mathcal{O}(\frac{\log n}{\log(1+\alpha)})$, Theorem 4.12 gives the time $2^{\mathcal{O}(\log k + \log\log n - \log\log(1+\alpha))rd}$, and this bound also holds for the size of $\mathcal{T}$. Also in step 1, in time $\mathcal{O}(n/\delta)$ one can construct a perfect hash function with success probability $1 - \delta$. All the other steps of the algorithm take $\mathsf{poly}(n, d)$ time. So the total running time is

$$2^{\mathcal{O}(\log k + \log\log n - \log\log(1+\alpha))rd}\mathsf{poly}(n, d, 1/\delta).$$

Correctness of the algorithm. Assume that in the optimal solution the outlier matrix is $\mathbf{N}^*$, and the low-rank matrix is $\mathbf{L}^*$. Denote by $\mathbf{V}^*$ the r-dimensional subspace corresponding to $\mathbf{L}^*$. The following claim shows that the sign condition corresponding to $\mathbf{V}^*$ allows the algorithm to restore $\mathbf{N}^*$.

Claim 10.10. *In step 5 of the algorithm, with high probability, the outlier matrix* $\mathbf{N}$ *generated on the sign condition* S^* *which comes from evaluating* $\mathcal{P}$ *on* $\mathbf{V}^*$ *is equal to* $\mathbf{N}^*$.

Proof. **Getting buckets with outliers, Step 5(a)**: Note that in Step 1 of the algorithm we map points to $\mathcal{O}(k)$ buckets in order to ensure that each bucket has at most one outlier. So some buckets will end up having no outliers at all. We show that in step 5(a) of the algorithm for the sign condition S^* we correctly find the k buckets containing outlier points. Specifically, for a bucket B_i with an outlier point p and a bucket B_j with no outliers we show that $Q_{i,j}(\mathbf{V}^*) > 0$. For α-gap instances we have

$$\begin{aligned} Q_{i,j}(\mathbf{V}^*) &= \sum_{\ell\in B_i} (\mathrm{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*))^b - \sum_{\ell\in B_j} (\mathrm{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*))^b \\ &\geq (\mathrm{dist}^2(\mathbf{a}_p^\top, \mathbf{V}^*))^b - n \max_{\ell\in I} (\mathrm{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*))^b \\ &\geq ((1+\alpha)^b - n) \max_{\ell\in I} (\mathrm{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*))^b > 0. \end{aligned}$$

Similarly, for α-heavy instances we have

$$\begin{aligned} Q_{i,j}(\mathbf{V}^*) &= \sum_{\ell \in B_i} \text{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*) - \sum_{\ell \in B_j} \text{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*) \\ &\geq \text{dist}^2(\mathbf{a}_p^\top, \mathbf{V}^*) - \sum_{\ell \in I} \text{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*) \\ &\geq \alpha \sum_{\ell \in I} \text{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*) > 0. \end{aligned}$$

Therefore, taking the top k buckets by the ordering induced by the signs of $\mathcal{Q}(\mathbf{V}^*)$ will give us precisely the k buckets containing the outlier points. WLOG, let $B_1, B_2, ..., B_k$ be those buckets, later referred to as the outlier buckets.

Extracting outliers from the outlier buckets, Step 5(b): Fix an $i \in [k]$, denote the sole outlier point in the bucket B_i by p. We show that for each $j \in [\log n]$, $p_j = 1$ iff $P_i^j(\mathbf{V}^*) > 0$ and $p_j = 0$ iff $P_i^j(\mathbf{V}^*) < 0$. For α-gap instances, if $p_j = 1$ then we have

$$\begin{aligned} P_i^j(\mathbf{V}) &= \sum_{\substack{\ell \in B_i \\ \ell_j = 1}} (\text{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}))^b - \sum_{\substack{\ell \in B_i \\ \ell_j = 0}} (\text{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}))^b \\ &\geq (\text{dist}^2(\mathbf{a}_\mathbf{p}^\top, \mathbf{V}^*))^b - n \max_{\ell \in I} (\text{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*))^b \\ &\geq ((1+\alpha)^b - n) \max_{\ell \in I} (\text{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*))^b \\ &> 0. \end{aligned}$$

Analogously, if $p_j = 0$, then $P_i^j(\mathbf{V}^*) < 0$. Similar analysis works for α-heavy instances with $b = 1$.

Thus the outlier matrix $\mathbf{N}$ generated on the sign condition S^* in step 5 is precisely the optimal $\mathbf{N}^*$. Success of the algorithm relies on perfect hashing of the outlier points into m buckets in step 1 and for $m = \Theta(k)$ one can find in time $\mathcal{O}(n/\delta)$ a perfect hash function with success probability $1 - \delta$. □

Since in our algorithm we go over all possible sign conditions of $\mathcal{P}$ on $\mathcal{X}$, the sign condition S^* will also be considered, and will provide the optimal outlier matrix. Once we have the optimal $\mathbf{N}^*$, computing the optimal rank-r projection of $\mathbf{A} - \mathbf{N}^*$ as the matrix $\mathbf{L}$ will give us the optimal cost. Thus, Claim 10.10 implies the correctness of the algorithm.

10.3.2 Dimensionality Reduction

In this subsection we improve upon the d in the exponent of the running time of the algorithm from the previous section. We achieve this by observing that for α-gap and α-heavy instances we only need the approximate distances of points to a subspace instead of the exact ones. The new algorithm proceed as follows:

(i) Sample a Normal Transform matrix $\mathbf{S} \in \mathbb{R}^{d\times t}$, where $t = \mathcal{O}(r+\log n+\log(1/\delta))$ for α-gap instances and $t = \mathcal{O}(r(\log k + \log\log n + \log(1/\delta))$ for α-heavy instances.

(ii) Sketch the input matrix, $\mathbf{A}$, from the right, $\widetilde{\mathbf{A}} = \mathbf{AS}$.

(iii) Find the optimal set of outliers for $\mathbf{AS}$ using the algorithm from the previous subsection.

(iv) Construct the matrix $\mathbf{N}$ from the corresponding rows of $\mathbf{A}$, and return $\mathbf{N}$ together with the optimal rank-r projection $\mathbf{L}$ of $\mathbf{A} - \mathbf{N}$.

Running Time. Clearly, step 3 dominates the running time. Since the ambient dimension is reduced from d to t, the running time of the new algorithm is

$$2^{O(r(\log k+\log\log n)(r+\log n+\log(1/\delta)))}(nd)^{\mathcal{O}(1)}$$

for α-gap instances and

$$2^{\mathcal{O}(r^2(\log k+\log\log n)(\log k+\log\log n+\log(1/\delta))}(nd)^{\mathcal{O}(1)}$$

for α-heavy instances.

Correctness of the algorithm. Correctness of the algorithm relies on the following two lemmas, handling α-gap instances and α-heavy instances respectively. Intuitively, we prove that a suitable embedding preserves the set of the optimal outliers. For the α-gap case, we use ε-embeddings (Definition 4.1).

Lemma 10.11. *Let $\mathbf{A} \in R^{n\times d}$ and integer parameters r and k be an α-gap instance of* PCA WITH OUTLIERS. *Let $\mathbf{S} \in R^{d\times t}$ be an ε-embedding for*

$$\mathrm{col}([\mathbf{V}^{*\top}|\mathbf{a}_1]), \mathrm{col}([\mathbf{V}^{*\top}|\mathbf{a}_2]), \ldots, \mathrm{col}([\mathbf{V}^{*\top}|\mathbf{a}_n]),$$

simultaneously, for a small enough constant ε. Here $\mathbf{V}^* \in R^{r\times d}$ is the r-dimensional linear subspace which spans the rows of the optimal rank-r matrix $\mathbf{L}^*$. Let $\widetilde{\mathbf{A}} = \mathbf{AS}$ and $\widetilde{\mathcal{T}} = \mathrm{Sample}(\widetilde{\mathbf{A}}, r, k)$ where Sample *is the procedure from step 4 of the Subspace-sampling algorithm. Then there exists $(\widetilde{C}, \widetilde{\mathbf{U}}) \in \widetilde{\mathcal{T}}$ such that the outlier matrix $\widetilde{\mathbf{N}}$ generated on $(\widetilde{C}, \widetilde{\mathbf{U}})$ in step 5 of the Subspace-sampling algorithm is same as the optimal outlier matrix $\mathbf{N}^*$*

Proof. We begin by observing that the distances of the rows of $\mathbf{AS}$ from $\mathrm{row}(\mathbf{V}^*\mathbf{S})$ are the same as the distances of rows of $\mathbf{A}$ from $\mathrm{row}(\mathbf{V}^*)$, up to a constant factor distortion. Since $\mathbf{S}$ is a ε-embedding for $\mathrm{col}([\mathbf{V}^{*\top}|\mathbf{a}_1]), \mathrm{col}([\mathbf{V}^{*\top}|\mathbf{a}_2]), \ldots, \mathrm{col}([\mathbf{V}^{*\top}|\mathbf{a}_n])$ simultaneously, using Theorem 4.3 we have for each $i \in [n]$

$$(1-\varepsilon)\,\mathrm{dist}^2(\mathbf{a}_i^\top, \mathbf{V}^*) \le \mathrm{dist}^2(\mathbf{a}_i^\top\mathbf{S}, \mathbf{V}^*\mathbf{S}) \le (1+\varepsilon)\,\mathrm{dist}^2(\mathbf{a}_i^\top, \mathbf{V}^*). \tag{10.4}$$

Now let $\widetilde{\mathcal{P}} = \cup\widetilde{\mathcal{P}^i} \cup \widetilde{\mathcal{Q}}$ be the collection of polynomials same as $\mathcal{P}$, but defined on the smaller space, i.e. the space of all r-dimensional subspaces in $\mathbb{R}^t$. Let $(\widetilde{C}, \mathbf{V}^*\mathbf{S}) \in \widetilde{T}$ be the sign condition of $\widetilde{\mathcal{P}}$ on $(\mathbf{V}^*\mathbf{S}; \mathbf{AS})$. We claim that the outlier matrix, $\mathbf{N}$, generated on $(\widetilde{C}, \mathbf{V}^*\mathbf{S})$ in the step 5 of the Subspace-sampling algorithm is the optimal outlier matrix $\mathbf{N}^*$. To see this, first we we claim that in step 5(a) the top k indices obtained from ordering on $[m]$ given by signs of $\mathbf{V}^*\mathbf{S}$ on $\widetilde{\mathcal{Q}}$ give us k buckets containing outlier points. Note that to prove this it suffices to show that $\widetilde{Q_{i,j}}(\mathbf{V}^*\mathbf{S}; \mathbf{AS}) > 0$ for a bucket B_i with an outlier point p and a bucket B_j with no outlier point. Starting with the definition of $\widetilde{Q_{i,j}}(\mathbf{V}^*\mathbf{S}; \mathbf{AS})$,

$$\begin{aligned}
&\sum_{\ell \in B_i} (\mathrm{dist}^2(\mathbf{a}_\ell^\top \mathbf{S}, \mathbf{V}^*\mathbf{S}))^b - \sum_{\ell \in B_j} (\mathrm{dist}^2(\mathbf{a}_\ell^\top \mathbf{S}, \mathbf{V}^*\mathbf{S}))^b \\
&\geq \sum_{\ell \in B_i} ((1-\varepsilon)\,\mathrm{dist}^2(\mathbf{a}_\ell, \mathbf{V}^*))^b - \sum_{\ell \in B_j} ((1+\varepsilon)\,\mathrm{dist}^2(\mathbf{a}_\ell, \mathbf{V}^*))^b \\
&\geq (1-\varepsilon)^b (\mathrm{dist}^2(\mathbf{a}_p, \mathbf{V}^*))^b - (1+\varepsilon)^b n \max_{\ell \in I} (\mathrm{dist}^2(\mathbf{a}_\ell, \mathbf{V}^*))^b \\
&\geq ((1-\varepsilon)^b (1+\alpha)^b - n(1+\varepsilon)^b) \max_{\ell \in I} (\mathrm{dist}^2(\mathbf{a}_\ell, \mathbf{V}^*))^b \\
&> 0 \quad \text{(For an appropriate } \varepsilon\text{)}
\end{aligned}$$

where we have used (10.4) in the first inequality and the α-gap property in the third inequality.

Similarly the signs of $\widetilde{\mathcal{P}}$ on $\mathbf{V}^*\mathbf{S}$ in the Subspace-sampling algorithm will be able to retrieve the outlier points $\mathbf{N}^*$. The analysis is analogous to the above. □

While in the α-gap case it suffices to use simple subspace embeddings, for the α-heavy case we require a stronger sketch to achieve the improved dimension bound. We are going to employ the notion of an affine embedding given by Definition 4.4 that we recall next.

Definition 4.4 (Affine embedding)**.** *Let $\mathbf{U} \in \mathbb{R}^{r \times d}$ and $\mathbf{A} \in \mathbb{R}^{n \times d}$, then $\mathbf{S} \in \mathbb{R}^{d \times s}$ is an ε-affine embedding for $(\mathbf{U}, \mathbf{A})$ if for every $\mathbf{X} \in \mathbb{R}^{n \times r}$, we have*

$$\|(\mathbf{A} - \mathbf{XU})\mathbf{S}\|_2^2 = (1 \pm \varepsilon)\,\|\mathbf{A} - \mathbf{XU}\|_2^2 .$$

Lemma 10.12. *Let $\mathbf{A} \in R^{n \times d}$ and integer parameters r and k be a α-heavy instance of* PCA WITH OUTLIERS. *After bucketing points in step 1 of the Subspace-sampling algorithm let $I_{i,j}^1$ and $I_{i,j}^0$ be sets of indices of points in bucket B_i whose j-bit is 1 and 0 respectively. Let $\mathbf{S} \in R^{d \times t}$ be a ε-affine embedding for $\{(\mathbf{V}^{*\top}, \mathbf{A}[I_{i,j}^k :]^\top)\}_{\substack{1 \leq i \leq m \\ 1 \leq j \leq \log n \\ 0 \leq k \leq 1}}$ simultaneously, where ε is a sufficiently small constant. Here $\mathbf{V}^* \in R^{r \times d}$ is the r-dimensional linear subspace which spans the rows of optimal $\mathbf{L}$. Let $\widetilde{\mathbf{A}} = \mathbf{AS}$ and $\widetilde{\mathcal{T}} = \mathrm{Sample}(\widetilde{\mathbf{A}}, r, k)$ where* Sample *is the procedure from step 4 of the Subspace-sampling algorithm. Then there exist $(\widetilde{C}, \widetilde{\mathbf{U}}) \in \widetilde{\mathcal{T}}$ such that the outlier matrix $\widetilde{\mathbf{N}}$*

generated on $(\widetilde{C}, \widetilde{\mathbf{U}})$ *in step 5 of the Subspace-sampling algorithm is same as the optimal outlier matrix* $\mathbf{N}^*$

Proof. The proof is similar to Lemma 10.11. We start by observing that since $\mathbf{S}$ is a ε-affine embedding for $\{\mathbf{V}^{*\top}, \mathbf{A}[I^k_{i,j} :]^\top\}_{\substack{1\le i\le m\\ 1\le j\le \log n\\ 0\le k\le 1}}$ simultaneously, we have that for all $i \in [m]$, $j \in [\log n]$ and $k \in \{0, 1\}$

$$\begin{aligned}(1-\varepsilon)\sum_{\substack{\ell\in B_i\\ \ell_j=k}} \operatorname{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*) &\le \sum_{\substack{\ell\in B_i\\ \ell_j=k}} \operatorname{dist}^2(\mathbf{a}_\ell^\top S, \mathbf{V}^*\mathbf{S})\\ &\le (1+\varepsilon)\sum_{\substack{\ell\in B_i\\ \ell_j=k}} \operatorname{dist}^2(\mathbf{a}_\ell^\top, \mathbf{V}^*).\end{aligned} \tag{10.5}$$

Now let $\widetilde{\mathcal{P}} = \cup\widetilde{\mathcal{P}^i} \cup \widetilde{\mathcal{Q}}$ be the collection of polynomials same as $\mathcal{P}$ but defined on smaller space i.e. all r-dimensional subspaces of t-dimensional space. Let $(\widetilde{C}, \mathbf{V}^*\mathbf{S}) \in \widetilde{T}$ be the sign condition of $\widetilde{\mathcal{P}}$ on $(\mathbf{V}^*\mathbf{S}; \mathbf{AS})$. We claim that the outlier matrix, $\mathbf{N}$, generated on $(\widetilde{C}, \mathbf{V}^*\mathbf{S})$ in the step 5 of the Subspace-sampling algorithm is the optimal outlier matrix $\mathbf{N}^*$. To see this note that that in step 5(a) the top k indices obtained from ordering on $[s]$ given by signs of $\mathbf{V}^*\mathbf{S}$ on $\widetilde{\mathcal{Q}}$ gives us k-buckets with outlier points. To prove this it suffices to show that $\widetilde{Q_{i,j}}(\mathbf{V}^*\mathbf{S}; \mathbf{AS}) > 0$ for a bucket B_i with an outlier point p and a bucket B_j with no outlier point. Starting with the definition of $\widetilde{Q_{i,j}}(\mathbf{V}^*\mathbf{S}; \mathbf{AS})$,

$$\begin{aligned}&\sum_{\ell\in B_i}(\operatorname{dist}^2(\mathbf{a}_\ell\mathbf{S}, \mathbf{V}^*\mathbf{S})) - \sum_{\ell\in B_j}(\operatorname{dist}^2(\mathbf{a}_\ell\mathbf{S}, \mathbf{V}^*\mathbf{S}))\\ &\ge (1-\varepsilon)\sum_{\ell\in B_i}(\operatorname{dist}^2(\mathbf{a}_\ell, \mathbf{V}^*)) - (1+\varepsilon)\sum_{\ell\in B_j}(\operatorname{dist}^2(\mathbf{a}_\ell, \mathbf{V}^*))\\ &\ge (1-\varepsilon)(\operatorname{dist}^2(\mathbf{a_p}, \mathbf{V}^*)) - (1+\varepsilon)\sum_{\ell\in I}(\operatorname{dist}^2(\mathbf{a}_\ell, \mathbf{V}^*))\\ &\ge ((1-\varepsilon)(1+\alpha) - (1+\varepsilon))\sum_{\ell\in I}(\operatorname{dist}^2(\mathbf{a}_\ell, \mathbf{V}^*))\\ &> 0 \quad \text{(For appropriate } \varepsilon)\end{aligned}$$

where we have used (10.5) in the first inequality and the α-heavy property in the third inequality.

Next we claim that using signs of $\widetilde{\mathcal{P}}$ on $\mathbf{V}^*\mathbf{S}$ the Subspace-sampling algorithm will be able to retrieve the outlier points $\mathbf{N}^*$. Analysis is analogous to the above. □

Lemma 10.11 and 10.12 prove the correctness of the algorithm, given that the sketching matrix $\mathbf{S}$ satisfies the conditions in the lemmas. Next we prove that the embedding dimension t is large enough for that to happen with probability at least $1 - \delta$.

Claim 10.13. *Let $\varepsilon, \delta \in (0,1)$ and $\mathbf{S} = \frac{1}{\sqrt{s}}\mathbf{G} \in R^{d\times s}$ where the entries of $\mathbf{G}$ are independent standard normal random variables with $s = \mathcal{O}((r+p+\log(1/\delta))\varepsilon^{-2})$ and $p = \min(\text{rank}(\mathbf{A}), \log n)$. Then $\mathbf{S}$ is a ε-embedding simultaneously for $\text{col}([\mathbf{U}^\top|\mathbf{a}_1])$, $\text{col}([\mathbf{U}^\top|\mathbf{a}_2])$, ..., $\text{col}([\mathbf{U}^\top|\mathbf{a}_n])$, for fixed $\mathbf{U} \in R^{r\times d}$, with probability at least $1-\delta$.*

Proof. Consider the following two arguments.

(i) By Theorem 4.2, for a fixed $i \in \{1, \ldots, n\}$, $\mathbf{S}$ is an ε-subspace embedding for $\text{col}([\mathbf{U}^\top|\mathbf{a}_i])$ with probability at least $1 - \frac{\delta}{2^p}$. By the union bound, $\mathbf{S}$ is the desired ε-embedding with probability at least $1 - \frac{\delta}{2^p}n$.

(ii) Let $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{\text{rank}(\mathbf{A})}\}$ be the row basis for $\mathbf{A}$ and let $\mathcal{V}'$ be the column space of $[\mathbf{U}^\top|\mathbf{b}_1|\mathbf{b}_2|\cdots|\mathbf{b}_{\text{rank}(\mathbf{A})}]$. Since $\text{col}([\mathbf{U}^\top|\mathbf{a}_i]) \subseteq \mathcal{V}'$ for each $i \in [n]$, and $\dim(\mathcal{V}') \leq r + \text{rank}(\mathbf{A})$, by Theorem 4.2, we have that $\mathbf{S}$ is an ε-embedding for $\mathcal{V}$ with probability at least $1 - \frac{\delta}{2^p}2^{\text{rank}(\mathbf{A})}$.

Combining the above two arguments we have that $\mathbf{S}$ is ε-embedding for each $\text{col}([\mathbf{U}^\top|\mathbf{a}_i])$, $i \in [n]$, with probability at least $1-\delta$.

□

Affine Embedding for the α-heavy case

Before proving the analogue of Claim 10.13 for the α-heavy case, we present a construction for ε-affine embedding based on the known results. The work [59] introduces several oblivious (data independent) constructions for affine embeddings like sparse embedding matrices, fast JL matrices, etc. These construction vary in dimension they embed into, dependence of embedding dimension on the failure probability, and time it takes to apply them. For our results, we would need an oblivious construction with embedding dimension as small as possible and $\log(\frac{1}{\delta})$ dependence on the failure probability δ. The embedding dimension in various constructions of [59] are optimized while keeping the time taken to apply them small and with only $\frac{1}{\delta}$ dependence on the failure probability. For our purposes, the time taken to apply the sketch is not the bottleneck and thus we show the following theorem, which gives the optimal dependence on the embedding dimension.

Theorem 10.14. *Let $0 < \varepsilon, \delta < 1$ and $\mathbf{S} = \frac{1}{\sqrt{s}}\mathbf{G} \in \mathbb{R}^{d\times s}$, where the entries of the matrix $\mathbf{G}$ are independent standard normal random variables. If $s = \Theta(r\log(1/\delta)\varepsilon^{-2})$, then for every fixed $\mathbf{U} \in \mathbb{R}^{r\times d}$ and $\mathbf{A} \in \mathbb{R}^{n\times d}$, with probability at least $1-\delta$, $\mathbf{S}$ is an ε-affine embedding for $(\mathbf{U}, \mathbf{A})$.*

Proof. First we state the following theorem from [59], see Theorem 39, which gives sufficient conditions for $\mathbf{S}$ to be an ε-affine embedding.

Theorem 10.15. *Let $\mathbf{U} \in \mathbb{R}^{r\times d}$ and $\mathbf{A} \in \mathbb{R}^{n\times d}$, then $\mathbf{S} \in \mathbb{R}^{d\times s}$ is a 3ε-affine embedding for $(\mathbf{U}, \mathbf{A})$ if the following events occur simultaneously.*

(i) Subspace Embedding: $\mathbf{S}$ *is a subspace embedding for column space of* $\mathbf{U}$.

(ii) Approximate Matrix Multiplication: For given matrices $\mathbf{Y}$ *and* $\mathbf{Z}$ *with* n *rows and* d *columns we have*

$$\left\|\mathbf{YSS}^\top\mathbf{Z}^\top - \mathbf{YZ}^\top\right\|_F^2 \leq \frac{\varepsilon^2}{r}\left\|\mathbf{Y}\right\|_F^2\left\|\mathbf{Z}\right\|_F^2.$$

(iii) Approximate Matrix Norm: For a given matrix $\mathbf{W} \in \mathbb{R}^{n\times d}$,

$$\left\|\mathbf{WS}\right\|_F^2 = (1\pm\varepsilon)\left\|\mathbf{W}\right\|_F^2$$

.

The next two lemmas show that random Gaussian matrices satisfy conditions (ii) and (iii) of above theorem.

Lemma 10.16. *Fix matrices* $\mathbf{Y}, \mathbf{Z} \in \mathbb{R}^{n\times d}$. *Let* $0 < \varepsilon, \delta < 1$ *and* $\mathbf{S} = \frac{1}{\sqrt{s}}\mathbf{G} \in R^{d\times s}$ *where the entries of matrix* $\mathbf{G}$ *are independent standard normal random variables. Then for* $s = \mathcal{O}(\frac{1}{\varepsilon^2}\log(1/\delta))$

$$\Pr_{\mathbf{S}}\left(\left\|\mathbf{YSS}^\top\mathbf{Z}^\top - \mathbf{YZ}^\top\right\|_F^2 \leq \varepsilon^2\left\|\mathbf{Y}\right\|_F^2\left\|\mathbf{Z}\right\|_F^2\right) \geq 1-\delta$$

Proof. The proof follows from Theorem 6.2 and Remark 6.3 from [126]. □

For the next lemma, we use the following inequality form [108].

Theorem 10.17. *(Hanson-Wright Inequality) Let* $\mathbf{g} \in \mathbb{R}^n$ *be a vector of standard normal random variables. There exists a constant* $C > 0$ *such that for all* $\varepsilon > 0$ *and for any matrix* $\mathbf{Q} \in \mathbb{R}^{n\times n}$,

$$\Pr_{\mathbf{g}}\left(\left|\mathbf{g}^\top\mathbf{Qg} - \mathrm{E}[\mathbf{g}^\top\mathbf{Qg}]\right| > \varepsilon\right) \leq e^{-C\varepsilon^2/\|\mathbf{Q}\|_F^2} + e^{-C\varepsilon/\|\mathbf{Q}\|_2}.$$

Lemma 10.18. *Fix a matrix* $\mathbf{W} \in \mathbb{R}^{n\times d}$. *Let* $0 < \varepsilon, \delta < 1$ *and* $\mathbf{S} = \frac{1}{\sqrt{s}}\mathbf{G} \in R^{d\times s}$ *where the entries of matrix* $\mathbf{G}$ *are independent standard normal random variables. Then for* $s = \Theta(\frac{1}{\varepsilon^2}\log(1/\delta))$,

$$\Pr_{\mathbf{S}}\left(\left|\left\|\mathbf{WS}\right\|_F^2 - \left\|\mathbf{W}\right\|_F^2\right| \leq \varepsilon\left\|\mathbf{W}\right\|_F^2\right) \geq 1-\delta.$$

Proof. First we show that $\mathrm{E}[\|\mathbf{WS}\|_F^2] = \|\mathbf{W}\|_F^2$.

$$\begin{aligned}\mathrm{E}_{\mathbf{S}}[\|\mathbf{WS}\|_F^2] &= \mathrm{E}_{\mathbf{S}}[\mathrm{Tr}(\mathbf{WSS}^\top\mathbf{W}^\top)]\\ &= \mathrm{Tr}(\mathbf{W}\mathrm{E}_{\mathbf{S}}[\mathbf{SS}^\top]\mathbf{W})\\ &= \mathrm{Tr}(\mathbf{WI_n W}^\top) = \left\|\mathbf{W}\right\|_F^2\end{aligned}$$

Now let $\mathbf{g}_i \in \mathbb{R}^d$ denote the i-th column of $\mathbf{S}$. Introduce random variables $\mathbf{X}_i = \mathbf{g}_i^\top \mathbf{W}^\top \mathbf{W} \mathbf{g}_i$ for each $i \in [s]$, and $\mathbf{X} = \frac{1}{s}\sum_{i=1}^{s} \mathbf{X}_i$. Then observe that

$$\|\mathbf{WS}\|_F^2 = \frac{1}{s}\sum_{i=1}^{s} \mathbf{g}_i^\top \mathbf{W}^\top \mathbf{W} \mathbf{g}_i = \frac{1}{s}\sum_{i=1}^{s} \mathbf{X}_i = \mathbf{X}.$$

By Hanson-Wright Inequality, we have that

$$\Pr_{\mathbf{g}_i}(|\mathbf{g}_i^\top \mathbf{W}^\top \mathbf{W} \mathbf{g}_i - \mathrm{E}[\mathbf{g}_i^\top \mathbf{W}^\top \mathbf{W} \mathbf{g}_i]| > \varepsilon \|\mathbf{W}\|_F^2) \leq e^{-C\varepsilon^2} + e^{-C\varepsilon} \leq 2e^{-C\varepsilon^2}.$$

Which tell us that $\mathbf{X_i}$ is a sub-Gaussian random variable and that $\mathbf{X} = \|\mathbf{WS}\|_F^2$ is an average of s independent sub-Gaussian random variables. Using Chernoff tail-inequality for sub-Gaussian random variables, we get

$$\Pr_{\mathbf{S}}(|\, \|\mathbf{WS}\|_F^2 - \|\mathbf{S}\|_F^2 \,| > \varepsilon \|\mathbf{S}\|_F^2) = \Pr_{\mathbf{S}}(|\mathbf{X} - \mathrm{E}[\mathbf{X}]| > \varepsilon \|\mathbf{A}\|_F^2) \leq e^{\mathcal{O}(-s\varepsilon^2)} \leq \delta.$$

□

Combining the above two lemmas with Theorem 4.2 completes the proof of Theorem 10.14. □

Now we can conclude the dimensionality reduction for α-heavy instances with the help of the ε-affine embedding given by Theorem 10.14.

Claim 10.19. *Let $\varepsilon, \delta \in (0,1)$ and $\mathbf{S} = \frac{1}{\sqrt{t}}\mathbf{G} \in R^{d\times t}$ where the entries of $\mathbf{G}$ are independent standard normal random variables with $t = \Theta(r(\log k + \log\log n + \log(1/\delta))\varepsilon^{-2})$. Then $\mathbf{S}$ is a ε-affine embedding simultaneously for $\{\mathbf{V}^{*\top}, \mathbf{A}[I_{i,j}^k :]^\top\}_{\substack{1\leq i\leq m \\ 1\leq j\leq \log n \\ 0\leq k\leq 1}}$, for fixed $\mathbf{V} \in R^{r\times d}$, with probability $1-\delta$.*

Proof. Follows from Theorem 10.14 and union bound. □

Finally, we observe that we have two sources of error in our algorithm. One is coming from the Subspace-sampling algorithm and the other from the dimensionality reduction. Setting failure probability to $\delta/2$ in each of them and applying union bound gives us $1-\delta$ success probability.

11

Low-Rank Approximation in Column-sum Norm

In this chapter, we present a polynomial-time approximation scheme for the L_1-Rank-r Approximation over $\mathsf{GF}(2)$. Recall that in this problem, given a binary $n \times d$ matrix $\mathbf{A}$ and a positive integer constant r, one seeks a binary matrix $\mathbf{L}$ of rank at most r, minimizing the column-sum norm $\|\mathbf{A} - \mathbf{L}\|_1$, defined as

$$\|\mathbf{A}\|_1 = \sup_{\|\mathbf{x}\|_1 \neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_1}{\|\mathbf{x}\|_1} = \max_{1 \leq j \leq d} \sum_{i=1}^{n} |\mathbf{A}_{ij}|.$$

More precisely, we prove the following theorem.

Theorem 11.1. *For every $\varepsilon \in (0, 1)$, there is a randomized $(1+\varepsilon)$-approximation algorithm for L_1-Rank-r Approximation over $\mathsf{GF}(2)$ of running time $n^{\mathcal{O}(2^{4r} \cdot \varepsilon^{-4})} d^{\mathcal{O}(1)}$.*

In order to prove Theorem 11.1 we obtain a PTAS for a more general problem that we call Binary Constrained k-Center. This problem is very similar to Constrained Clustering with Outliers that was the key technical problem in Chapter 6. Only, in Binary Constrained k-Center there are no outliers, and the cost of clustering is the maximum Hamming distance between a point and the closest center. Binary Constrained k-Center has as much expressive power as Constrained Clustering with Outliers, and we use it to obtain PTASes for a number of problems related to L_1-Rank-r Approximation over $\mathsf{GF}(2)$. For example, for the variant, when the rank of the matrix $\mathbf{L}$ is not over $\mathsf{GF}(2)$ but is Boolean. Or a variant of clustering, where we want to partition binary vectors into groups, minimizing the maximum distance in each of the group to some subspace of small dimension. We provide discussions of other applications of our work in Section 11.4.

Our approach

The usual toolbox of techniques to handle NP-hard variants of low-rank matrix approximation problems like sketching [189], sampling, and dimensionality reduction [29] is based on randomized linear algebra. It is very unclear whether any of these techniques can be used to solve even the simplest case of L_1-Rank-r Approximation over GF(2) with $r = 1$. For example for sampling, the presence of just one outlier outside of a sample, makes all information we can deduce from the sample about the column sum norm of the matrix, completely useless. This is exactly the reason why approximation algorithms for Closest String do not rely on such techniques. On the other hand, randomized dimension reduction appears to be very helpful as a "preprocessing" procedure whose application allows us to solve L_1-Rank-r Approximation over GF(2) by applying linear programming techniques similar to the ones developed for the Closest String. From a very general perspective, our algorithm consists of three steps. While each of these steps is based on the previous works, the way to combine these steps, as well as the correctness proof, is a non-trivial task. We start with a high-level description of the steps and then provide more technical explanations.

Step 1. In order to solve L_1-Rank-r Approximation over GF(2), we encode it as the Binary Constrained k-Center problem. This initial step is almost identical to the encoding used in [88] for Low GF(2)-Rank Approximation. Informally, Binary Constrained k-Center is defined as follows. For a given set of binary vectors X, a positive integer k, and a set of constraints, we want to find k binary vectors $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ satisfying the constraints and minimizing $\max_{\mathbf{x} \in X} d_H(\mathbf{x}, C)$, where $d_H(\mathbf{x}, C)$ is the Hamming distance between $\mathbf{x}$ and the closest vector from C. For example, when $k = 1$ and there are no constraints, then this is just the Closest String problem over binary alphabet.

In the technical description below we give a formal definition of this encoding and in Section 11.4 we prove that L_1-Rank-r Approximation over GF(2) is a special case of Binary Constrained k-Center. From now on, we work with the Binary Constrained k-Center problem.

Step 2. We give an approximate Turing reduction which allows to find a partition of vector set X into clusters $X_1, \ldots, X_k$ such that if we find a tuple of vectors $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ satisfying the constraints and minimizing $\max_{1 \leq i \leq k, \mathbf{x} \in X_i} d_H(\mathbf{x}, \{\mathbf{c}_i\})$, then the same tuple C will be a good approximation to Binary Constrained k-Center. In order to obtain such a partition, we use the dimension reduction technique of Ostrovsky and Rabani [165]. While this provides us with important structural information, we are not done yet. Even with a given partition, the task of finding the corresponding tuple of "closest strings" C satisfying the constraints, is non-trivial.

Step 3. In order to find the centers, we implement the approach used by Li, Ma, and Wang in [144] to solve Closest String. By brute-forcing, it is possible to reduce the solution of the problem to special instances that, loosely speaking, have a large optimum. Moreover, Binary Constrained k-Center has an Integer Programming (IP) formulation. Similar to [144], for the reduced instance of Binary Constrained k-Center (which has a "large optimum") it is possible to prove that the randomized rounding of the corresponding Linear Program (LP) relaxation of this IP, provides a good approximation.

11.1 Overview of the Algorithm

Now we give a more technical description of the algorithm.

Step 1. Binary Constrained k-Center**.** Note that the Binary Constrained k-Center problem is nearly identical to Binary Constrained Clustering defined in [88], except for the cost function, and similar to the Constrained Clustering with Outliers problem introduced in Chapter 6. Still, for completeness we define Binary Constrained k-Center formally next. First, we need to recall some notations from Chapter 6. A k-ary relation R is a set of binary k-tuples with elements from $\{0,1\}$. A k-tuple $t=(t_1,\ldots,t_k)$ *satisfies* R, we write $t\in R$, if t is equal to one of the k-tuples in R.

Definition 11.2 (Vectors satisfying $\mathcal{R}$)**.** *Let $\mathcal{R}=(R_1,\ldots,R_d)$ be a tuple of k-ary relations. We say that a tuple $C=(\mathbf{c}_1,\mathbf{c}_2,\ldots,\mathbf{c}_k)$ of binary d-dimensional vectors* satisfies *$\mathcal{R}$ and write $<C,\mathcal{R}>$, if $(\mathbf{c}_1[i],\ldots,\mathbf{c}_k[i])\in R_i$ for all $i\in\{1,\ldots,d\}$.*

Let us recall that the *Hamming distance* between two vectors $\mathbf{x},\mathbf{y}\in\{0,1\}^d$, where $\mathbf{x}=(x_1,\ldots,x_d)^\intercal$ and $\mathbf{y}=(y_1,\ldots,y_d)^\intercal$, is $d_H(\mathbf{x},\mathbf{y})=\sum_{i=1}^{d}|x_i-y_i|$ or, in words, the number of positions $i\in\{1,\ldots,d\}$ where x_i and y_i differ. Recall that for a set of vectors $C\subseteq\{0,1\}^d$ and a vector $\mathbf{x}\in\{0,1\}^d$, $d_H(\mathbf{x},C)=\min_{\mathbf{c}\in C}d_H(\mathbf{x},\mathbf{c})$. For sets $X,C\subset\{0,1\}^d$, we define $\mathrm{cost}(X,C)=\max_{\mathbf{x}\in X}d_H(\mathbf{x},C)$.

Now we define Binary Constrained k-Center formally.

Binary Constrained k-Center

Input:	A set $X\subseteq\{0,1\}^d$ of n vectors, a positive integer k, and a tuple of k-ary relations $\mathcal{R}=(R_1,\ldots,R_d)$.
Task:	Among all tuples $C=(\mathbf{c}_1,\ldots,\mathbf{c}_k)$ of vectors from $\{0,1\}^d$ satisfying $\mathcal{R}$, find a tuple C minimizing $\mathrm{cost}(X,C)$.

As in the case of Low GF(2)-Rank Approximation in [88], we prove that L_1-Rank-r Approximation over $\mathsf{GF}(2)$ is a special case of Binary Constrained

k-Center, where $k = 2^r$. For completeness, this proof and other applications of Binary Constrained k-Center are given in Section 11.4. Thus, to prove Theorem 11.1, it is enough to design a PTAS for Binary Constrained k-Center.

Theorem 11.3. *There is an algorithm for* Binary Constrained k-Center *that given an instance $J = (X, k, \mathcal{R})$ and $0 < \varepsilon < 1$, runs in time $n^{\mathcal{O}((k/\varepsilon)^4)} d^{\mathcal{O}(1)}$, and outputs a $(1+\varepsilon)$-approximate solution with probability at least $1 - 2n^{-2}$.*

By the argument above, Theorem 11.1 is an immediate corollary of Theorem 11.3.

Step 2: Dimensionality reduction. Let $J = (X, k, \mathcal{R} = (R_1, \ldots, R_d))$ be an instance of Binary Constrained k-Center and $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ be a solution to J, that is, a tuple of vectors satisfying $\mathcal{R}$. Then, the cost of C is $\mathrm{cost}(X, C)$. Given the tuple C, there is a natural way we can partition the set of vectors X into k parts $X_1 \uplus \cdots \uplus X_k$ such that

$$\mathrm{cost}(X, C) = \max_{i \in \{1,\ldots,k\}, \mathbf{x} \in X_i} d_H(\mathbf{x}, \mathbf{c}_i).$$

Thus, for each vector $\mathbf{x}$ in X_i, the closest to $\mathbf{x}$ vector from C is $\mathbf{c}_i$. We call such a partition $X_1 \uplus \cdots \uplus X_k$ the *clustering of X induced by C* and refer to the sets $X_1, \ldots, X_k$ as the *clusters corresponding to C*. We use $\mathsf{OPT}(J)$ to denote the cost of an optimal solution to J. That is, $\mathsf{OPT}(J) = \min\{\mathrm{cost}(X, C) \mid < C, \mathcal{R} >\}$. In fact, even if we know the clustering of X induced by a hypothetical optimal solution, finding a good solution is not trivial as the case when $k = 1$ is the same as the Closest String problem.

As mentioned before, our approach is to reduce to a version of Binary Constrained k-Center, where we know the partition of X, and solve the corresponding problem. That is, we design an approximation scheme for the following partitioned version of the problem.

Binary Constrained Partition Center

Input:	A positive integer k, a set $X \subseteq \{0,1\}^d$ of n vectors partitioned into $X_1 \uplus \ldots \uplus X_k$, and a tuple of k-ary relations $\mathcal{R} = (R_1, \ldots, R_d)$.
Task:	Among all tuples $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ of vectors from $\{0,1\}^d$ satisfying $\mathcal{R}$, find a tuple C minimizing $\max_{i \in \{1,\ldots,k\}, \mathbf{x} \in X_i} d_H(\mathbf{x}, \mathbf{c}_i)$.

For an instance $J' = (k, X = X_1 \uplus \ldots \uplus X_k, \mathcal{R})$ of Binary Constrained Partition Center, we use $\mathsf{OPT}(J')$ to denote the cost of an optimal solution to J'. That is,

$$\mathsf{OPT}(J') = \min_{C=(\mathbf{c}_1,\ldots,\mathbf{c}_k) \text{ s.t. } <C,\mathcal{R}>} \left\{ \max_{i \in \{1,\ldots,k\}, \mathbf{x} \in X_i} d_H(\mathbf{x}, \mathbf{c}_i) \right\}.$$

Clearly, for an instance $J = (X, k, \mathcal{R})$ of Binary Constrained k-Center and a partition of X into $X_1 \uplus \ldots \uplus X_k$, any solution to the instance $J' = (k, X = X_1 \uplus \ldots X_k, \mathcal{R})$ of Binary Constrained Partition Center, of cost D, is also a solution to J with cost at most D. We prove that there is a randomized polynomial time algorithm that given an instance $J = (X, k, \mathcal{R})$ of Binary Constrained k-Center and $0 < \varepsilon \leq \frac{1}{4}$, outputs a collection $\mathcal{I}$ of Binary Constrained Partition Center instances $J' = (k, X = X_1 \uplus \ldots \uplus X_k, \mathcal{R})$ such that the cost of at least one instance in $\mathcal{I}$ is at most $(1 + 4\varepsilon)\mathsf{OPT}(J)$ with high probability.

Lemma 11.4. *There is an algorithm that given an instance $J = (X, k, \mathcal{R})$ of* Binary Constrained k-Center, *$0 < \varepsilon \leq \frac{1}{4}$, and $\gamma > 0$, runs in time $n^{\mathcal{O}(k/\varepsilon^4)}d^2$, and outputs a collection $\mathcal{I}$ of $n^{\mathcal{O}(k/\varepsilon^4)}d$ instances of* Binary Constrained Partition Center *such that each instance in $\mathcal{I}$ is of the form $(k, X = X_1 \uplus \ldots \uplus X_k, \mathcal{R})$, and there exists $J' \in \mathcal{I}$ such that $\mathsf{OPT}(J') \leq (1 + 4\varepsilon)\mathsf{OPT}(J)$ with probability at least $1 - n^{-\gamma}$.*

To prove Lemma 11.4, we use the dimensionality reduction technique of Ostrovsky and Rabani from [165]. Loosely speaking, this technique provides a linear map ψ with the following properties. For any $\mathbf{y} \in \{0,1\}^d$, $\psi(\mathbf{y})$ is a 0-1 vector of length $\mathcal{O}(\log n/\varepsilon^4)$, and for any set Y of $n + k$ vectors, Hamming distances between any pair of vectors in $\psi(Y)$ are relatively preserved with high probability. So we assume that ψ is "a good map" for the set of vectors $X \cup C$, where $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ is a hypothetical optimal solution to J. Then, we guess the potential tuples of vectors $(\phi(\mathbf{c}_1), \ldots, \phi(\mathbf{c}_k))$ for the hypothetical optimal solution $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$, and use these choices for $(\phi(\mathbf{c}_1), \ldots, \phi(\mathbf{c}_k))$ to construct partitions of X, and thereby construct instances in $\mathcal{I}$. Lemma 11.4 is proved in Section 11.2.

Step 3: LP relaxation. Because of Lemma 11.4, to prove Theorem 11.3, it is enough to design a PTAS for Binary Constrained Partition Center, which is the most challenging part of our algorithm. For this, we prove the following lemma.

Lemma 11.5. *There is an algorithm for* Binary Constrained Partition Center *that given an instance $J = (k, X = X_1^i \uplus \ldots \uplus X_k^i, \mathcal{R})$ and $0 < \varepsilon < 1/2$, runs in time $n^{\mathcal{O}((k/\varepsilon)^4)}d^{\mathcal{O}(1)}$, and outputs a solution of cost at most $(1 + \varepsilon)\mathsf{OPT}(J)$ with probability at least $1 - n^{-2}$.*

Towards the proof of Lemma 11.5, we encode Binary Constrained Partition Center using an Integer programming (IP) formulation (see (11.4) in Section 11.3). We show that the randomized rounding using the solution of the linear programming relaxation of this IP provides a good approximation if the optimum value is large. Here we follow the approach similar to the one used by Li, Ma, and Wang in [144] to solve Closest String. We prove that there exist $Y_1 \subseteq X_1, \ldots, Y_k \subseteq X_k$, each of size $r = 1 + \frac{4}{\varepsilon}$, with the following property. Let Q be the set of positions in $\{1, \ldots, d\}$ such that for each $i \in \{1, \ldots, k\}$ and $j \in Q$, all the vectors in Y_i agree at the position

j, and for each $j \in Q$, $(\mathbf{y}_1[j], \ldots, \mathbf{y}_k[j]) \in R_j$, where $\mathbf{y}_i \in Y_i$ for all $i \in \{1, \ldots, k\}$. Then, for any solution of J such that for each $j \in Q$ the entries at the position j coincide with $(\mathbf{y}_1[j], \ldots, \mathbf{y}_k[j])$, the cost of this solution restricted to Q deviates from the cost of an optimal solution restricted to Q by at most $\frac{1}{r-1}\mathsf{OPT}(J)$. Moreover, the subproblem of J restricted to $\{1, \ldots, d\} \setminus Q$ has large optimum value and we are able to use linear programming to solve this subproblem. Lemma 11.5 is proved in Section 11.3.

Putting together. Next we explain how to prove Theorem 11.3 using Lemmata 11.4 and 11.5. Let $J = (X, k, \mathcal{R})$ be the input instance of Binary Constrained k-Center and $0 < \varepsilon < 1$ be the given error parameter. Let $\beta = \frac{\varepsilon}{8}$, since $\varepsilon < 1$, $\beta < \frac{1}{4}$. Now, we apply Lemma 11.4 on J, β, and $\gamma = 2$. As a result, we get a collection $\mathcal{I}$ of instances of Binary Constrained Partition Center such that each instance in $\mathcal{I}$ is of the form $(k, X = X_1 \uplus \ldots \uplus X_k, \mathcal{R})$, and there exists $J' \in \mathcal{I}$ such that $\mathsf{OPT}(J') \leq (1 + 4\beta)\mathsf{OPT}(J)$ with probability at least $1 - n^{-2}$. From now on, we assume that this event happened. Next, for each instance $\widehat{J} \in \mathcal{I}$, we apply Lemma 11.5 with the error parameter β, and output the best solution among the solutions produced. Let $J' \in \mathcal{I}$ be the instance such that $\mathsf{OPT}(J') \leq (1 + 4\beta)\mathsf{OPT}(J) \leq (1 + \frac{\varepsilon}{2})\mathsf{OPT}(J)$. Any solution to $\widehat{J} \in \mathcal{I}$ of cost D, is also a solution to J of cost at most D. Therefore, because of Lemmas 11.4 and 11.5, our algorithm outputs a solution of J with cost at most $(1 + \beta)\mathsf{OPT}(J') = (1 + \frac{\varepsilon}{8})(1 + \frac{\varepsilon}{2})\mathsf{OPT}(J) \leq (1 + \varepsilon)\mathsf{OPT}(J)$ with probability at least $1 - 2n^{-2}$, since both Lemmas 11.4 and 11.5 have the success probability of at least $1 - n^{-2}$. The running time of the algorithm follows from Lemmata 11.4 and 11.5.

As Theorem 11.3 is already proved using Lemmas 11.4 and 11.5, the rest of the chapter is devoted to the proofs of Lemmata 11.4 and 11.5, and to the examples of the expressive power of Binary Constrained k-Center, including L_1-Rank-r Approximation over $\mathsf{GF}(2)$. In Sections 11.2 and 11.3, we prove Lemmata 11.4 and 11.5, respectively. In Section 11.4, we give applications of Theorem 11.3.

11.2 Reducing to a Partitioned Instance

In this section we prove Lemma 11.4. The main idea is to map the given instance to a low-dimensional space while approximately preserving distances, then try all possible tuples of centers in the low-dimensional space, and construct an instance of Binary Constrained Partition Center by taking the optimal partition of the images with respect to a fixed tuple of centers back to the original vectors.

To implement the mapping, we employ the notion of (δ, ℓ, h)-distorted maps, introduced by Ostrovsky and Rabani [165]. Intuitively, a (δ, ℓ, h)-distorted map approximately preserves distances between ℓ and h, does not shrink distances larger than h too much, and does not expand distances smaller than ℓ too much. In what

follows we make the definitions formal.

Recall that a metric space is a pair $(\mathcal{X}, \text{dist})$ where $\mathcal{X}$ is a set (whose elements are called points), and dist is a distance function $\text{dist} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (called a metric), such that for every $p_1, p_2, p_3 \in \mathcal{X}$ the following conditions hold: (i) $\text{dist}(p_1, p_2) \geq 0$, (ii) $\text{dist}(p_1, p_2) = \text{dist}(p_2, p_1)$, (iii) $\text{dist}(p_1, p_2) = 0$ if and only if $p_1 = p_2$, and (iv) $\text{dist}(p_1, p_2) + \text{dist}(p_2, p_3) \geq \text{dist}(p_1, p_3)$. Condition (iv) is called the triangle inequality. The pair $(\{0,1\}^d, d_H)$, the space of binary vectors of length d equipped with the Hamming distance, is a metric space.

Definition 11.6 ([165])**.** *Let $(\mathcal{X}, \text{dist})$ and $(\mathcal{X}', \text{dist}')$ be two metric spaces. Let $X, Y \subseteq \mathcal{X}$. Let δ, ℓ, h be such that $\delta > 0$ and $h > \ell \geq 0$. A mapping $\psi : \mathcal{X} \to \mathcal{X}'$ is (δ, ℓ, h)-distorted on (X, Y) if and only if there exists $\alpha > 0$ such that for every $x \in X$ and $y \in Y$, the following conditions hold.*

(i) If $\text{dist}(x, y) < \ell$, then $\text{dist}'(\psi(x), \psi(y)) < (1 + \delta)\alpha\ell$.

(ii) If $\text{dist}(x, y) > h$, then $\text{dist}'(\psi(x), \psi(y)) > (1 - \delta)\alpha h$.

(iii) If $\ell \leq \text{dist}(x, y) \leq h$, then $(1 - \delta)\alpha \,\text{dist}(x, y) \leq \text{dist}'(\psi(x), \psi(y)) \leq (1 + \delta)\alpha \,\text{dist}(x, y)$.

If $X = Y$, then we say that ψ is (δ, ℓ, h)-distorted on X.

For any $r, r' \in \mathbb{N}$ and $\varepsilon > 0$, $\mathcal{A}_{r,r'}(\varepsilon)$ denotes a distribution over $r' \times r$ binary matrices $M \in \{0,1\}^{r' \times r}$, where entries are independent, identically distributed, random 0/1 variables with $\Pr[1] = \varepsilon$. The following theorem by Ostrovsky and Rabani [165] states that the matrices $\mathcal{A}_{r,r}(\varepsilon)$ allow to achieve distorted maps for the Hamming space.

Theorem 11.7 ([165])**.** *Let $d, \ell \in \mathbb{N}$, and let $X \subseteq \{0,1\}^d$ be a set of n vectors. For every $0 < \varepsilon \leq 1/2$, there exists a mapping $\phi : X \to \{0,1\}^{d'}$, where $d' = \mathcal{O}(\log n/\varepsilon^4)$, which is $(\varepsilon, \ell/4, \ell/2\varepsilon)$-distorted on X (with respect to the Hamming distance in both spaces). More precisely, for every $\gamma > 0$ there exists $\lambda > 0$, such that, setting $d' = \lambda \log n/\varepsilon^4$, the linear map $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is a random matrix drawn from $\mathcal{A}_{m,m'}(\varepsilon^2/\ell)$, is $(\varepsilon, \ell/4, \ell/2\varepsilon)$-distorted on X with probability at least $1 - n^{-\gamma}$.*

Now we are ready to prove Lemma 11.4. We restate it for convenience.

Lemma 11.4. *There is an algorithm that given an instance $J = (X, k, \mathcal{R})$ of* Binary Constrained k-Center*, $0 < \varepsilon \leq \frac{1}{4}$, and $\gamma > 0$, runs in time $n^{\mathcal{O}(k/\varepsilon^4)}d^2$, and outputs a collection $\mathcal{I}$ of $n^{\mathcal{O}(k/\varepsilon^4)}d$ instances of* Binary Constrained Partition Center *such that each instance in $\mathcal{I}$ is of the form $(k, X = X_1 \uplus \ldots \uplus X_k, \mathcal{R})$, and there exists $J' \in \mathcal{I}$ such that $\mathsf{OPT}(J') \leq (1 + 4\varepsilon)\mathsf{OPT}(J)$ with probability at least $1 - n^{-\gamma}$.*

Proof. Without loss of generality, we may assume $\mathsf{OPT}(J) > 0$. If $\mathsf{OPT}(J) = 0$, there are at most k distinct vectors in X, and we trivially construct a single instance of Binary Constrained Partition Center by grouping equal vectors together.

Let $n = |X|$ and $n' = n + k$. Let $\lambda = \lambda(\gamma)$ be the constant mentioned in Theorem 11.7, and $d' = \lambda \log n'/\varepsilon^4$. Then, for each $\ell \in [d]$, we construct the collection $\mathcal{I}_\ell$ of $n^{\mathcal{O}(k/\varepsilon^4)}$ Binary Constrained Partition Center instances as follows.

- Start with $\mathcal{I}_\ell := \emptyset$.
- Randomly choose a matrix $\mathbf{A}^\ell$ from the distribution $\mathcal{A}_{d,d'}(\varepsilon^2/\ell)$.
- For each choice of k vectors $\mathbf{c}'_1, \ldots, \mathbf{c}'_k \in \{0,1\}^{d'}$, construct a partition $X_1 \uplus \ldots \uplus X_k$ of X such that for each $\mathbf{x} \in X_i$, $\mathbf{c}'_i$ is one of the closest vectors to $A^\ell\mathbf{x}$ among $C' = \{\mathbf{c}'_1, \ldots, \mathbf{c}'_k\}$. Then, add $(k, X = X_1^i \uplus \ldots X_k^i, \mathcal{R})$ to $\mathcal{I}_\ell$.

Finally, our algorithm outputs $\mathcal{I} = \bigcup_{\ell \in [d]} \mathcal{I}_\ell$ as the required collection of Binary Constrained Partition Center instances. Notice that for any $\ell \in [d]$, $|\mathcal{I}_\ell| = 2^{d'k} = n^{\mathcal{O}(k/\varepsilon^4)}$. This implies that the cardinality of $\mathcal{I}$ is upper bounded by $n^{\mathcal{O}(k/\varepsilon^4)}d$, and the construction of $\mathcal{I}_\ell$ takes time $n^{\mathcal{O}(k/\varepsilon^4)}d$. Thus, the total running time of the algorithm is $n^{\mathcal{O}(k/\varepsilon^4)}d^2$.

Next, we prove the correctness of the algorithm. Let $\ell = \mathsf{OPT}(J)$ and $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ be an optimum solution of J. Let $Y_1, \ldots, Y_k$ be the clusters corresponding to C. Consider the step in the algorithm where we constructed $\mathcal{I}_\ell$. By 11.7, the map $\psi\colon \mathbf{x} \mapsto \mathbf{A}^\ell\mathbf{x}$ is $(\varepsilon, \ell/4, \ell/2\varepsilon)$-distorted on $X \cup C$ with probability at least $1 - n^{-\gamma}$. In the rest of the proof, we assume that this event happened. Let $\mathbf{c}'_1 = \mathbf{A}^\ell\mathbf{c}_1, \ldots, \mathbf{c}'_k = \mathbf{A}^\ell\mathbf{c}_k$. Consider the Binary Constrained Partition Center instance constructed for the choice of vectors $\mathbf{c}'_1, \ldots, \mathbf{c}'_k$. That is, let $X_1, \ldots, X_k$ be the partition of X such that for each $\mathbf{x} \in X_i$, $\mathbf{c}'_i$ is one of the closest vector to $\mathbf{A}^\ell\mathbf{x}$ from $C' = \{\mathbf{c}'_1, \ldots, \mathbf{c}'_k\}$. Let J' be the instance $(k, X = X_1^i \uplus \ldots X_k^i, \mathcal{R})$ of Binary Constrained Partition Center.

Now, we claim that C is a solution to J' with cost at most $(1 + 4\varepsilon)\ell = (1 + 4\varepsilon)\mathsf{OPT}(J)$. Since C satisfies $\mathcal{R}$, C is a solution of J'. To prove $\mathsf{OPT}(J') \leq (1+4\varepsilon)\ell$, it is enough to prove that for each $i \in [k]$ and $\mathbf{x} \in X_i$, $d_H(\mathbf{x}, \mathbf{c}_i) \leq (1 + 4\varepsilon)\ell$. Fix an index $i \in [k]$ and $\mathbf{x} \in X_i$. Suppose $\mathbf{x} \in Y_i$. Since C is an optimum solution of J with corresponding clusters $Y_1, \ldots Y_k$, we have that $d_H(\mathbf{y}, \mathbf{c}_i) \leq \ell$ for all $\mathbf{y} \in Y_i \cap X_i$. Thus, $d_H(\mathbf{x}, \mathbf{c}_i) \leq \ell$. So, now consider the case $\mathbf{x} \in Y_j$ for some $j \neq i$. Notice that if $d_H(\mathbf{x}, \mathbf{c}_i) \leq \ell$, then we are done. We have the following two subcases.

Case 1: $d_H(\mathbf{x}, \mathbf{c}_i) \leq \frac{\ell}{2\varepsilon}$. We know that the map $\psi\colon \mathbf{x} \mapsto \mathbf{A}^\ell\mathbf{x}$ is $(\varepsilon, \ell/4, \ell/2\varepsilon)$-distorted on $X \cup C$, and let $\alpha > 0$ be the number such that conditions of Definition 11.6 hold. Since $\mathbf{x} \in X_i$, we have that (*a*) $d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_i)) \leq d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_j))$. Since $d_H(\mathbf{x}, \mathbf{c}_j) \leq \ell$ (because $\mathbf{x} \in Y_j$) and ψ is $(\varepsilon, \ell/4, \ell/2\varepsilon)$-distorted on $X \cup C$, we have that (*b*) $d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_j)) \leq (1 + \varepsilon)\alpha\ell$. Since $\ell < d_H(\mathbf{x}, \mathbf{c}_i) \leq \frac{\ell}{2\varepsilon}$, and ψ is

$(\varepsilon, \ell/4, \ell/2\varepsilon)$-distorted on $X \cup C$, we have that (c) $(1-\varepsilon)\alpha d_H(\mathbf{x}, \mathbf{c}_i) \leq d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_i))$. The statements (a), (b), and (c) imply that

$$d_H(\mathbf{x}, \mathbf{c}_i) \leq \frac{1+\varepsilon}{1-\varepsilon}\ell \leq (1+4\varepsilon)\ell,$$

where the last inequality holds since $\varepsilon \leq 1/4$.

Case 2: $d_H(\mathbf{x}, \mathbf{c}_i) > \frac{\ell}{2\varepsilon}$**.** We prove that this case is impossible by showing a contradiction. Since $\varepsilon \leq 1/4$, in this case, we have that $d_H(\mathbf{x}, \mathbf{c}_i) > 2\ell$. Since ψ is $(\varepsilon, \ell/4, \ell/2\varepsilon)$-distorted on $X \cup C$, $d_H(\mathbf{x}, \mathbf{c}_i) > 2\ell$, and $d_H(\mathbf{x}, \mathbf{c}_j) \leq \ell$, we have that

$$(1-\varepsilon)\alpha \cdot 2\ell \leq d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_i)) \leq d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_j)) \leq (1+\varepsilon)\alpha \cdot \ell.$$

Then $2(1-\varepsilon) \leq (1+\varepsilon)$ and thus $\varepsilon \geq 1/3$, which contradicts the assumption that $\varepsilon \leq 1/4$.

This completes the proof of the lemma. □

11.3 Solving the Partitioned Instance

For a set of positions $P \subset [m]$, let us define the Hamming distance restricted to P by

$$d_H^P(\mathbf{x}, \mathbf{y}) = \sum_{i \in P} |x_i - y_i|.$$

We use the following lemma in our proof.

Lemma 11.8. *Let $Y = \{\mathbf{y}_1, \cdots, \mathbf{y}_\ell\} \subset \{0,1\}^d$ be a set of vectors and $\mathbf{c}^* \in \{0,1\}^d$ be a vector. Let $d^* = \mathrm{cost}(Y, \{\mathbf{c}^*\}) = \max_{\mathbf{y} \in Y} d_H(\mathbf{y}, \mathbf{c}^*)$. For any $r \in \mathbb{N}$, $r \geq 2$, there exist indices $i_1, \ldots, i_r$ such that for any $\mathbf{x} \in Y$*

$$d_H^P(\mathbf{x}, \mathbf{y}_{i_1}) - d_H^P(\mathbf{x}, \mathbf{c}^*) \leq \frac{1}{r-1} d^*,$$

where P is any subset of $Q_{i_1,\ldots,i_r}$ and $Q_{i_1,\ldots,i_r}$ is the set of positions where all of $\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_r}$ coincide (i.e., $Q_{i_1,\ldots,i_r} = \{j \in [m] \colon \mathbf{y}_{i_1}[j] = \mathbf{y}_{i_2}[j] = \ldots = \mathbf{y}_{i_r}[j]\}$).

Proof. For a vector $\mathbf{x} = \mathbf{y}_{\ell'} \in Y$ and $P \subseteq Q_{i_1,\ldots,i_r}$, let

$$\begin{aligned} J_P(\ell') &= \{j \in P \colon \mathbf{y}_{i_1}[j] \neq \mathbf{x}[j] \text{ and } \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}, \text{ and} \\ J(\ell') &= \{j \in Q_{i_1,\ldots,i_r} \colon \mathbf{y}_{i_1}[j] \neq \mathbf{x}[j] \text{ and } \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}. \end{aligned}$$

To prove the lemma it is enough to prove that $|J_P(\ell')| \leq \frac{1}{r-1} d^*$. Also, since $J_P(\ell') \subseteq J(\ell')$, to prove the lemma, it is enough to prove that $|J(\ell')| \leq \frac{1}{r-1} d^*$. Recall that for any $s \in [\ell]$ and $1 \leq i_1, \ldots, i_s \leq \ell$, $Q_{i_1,\ldots,i_s}$ is the set of positions where all of

$\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_s}$ coincide. For any $2 \leq s \leq r+1$ and $1 \leq i_1, \ldots, i_s \leq \ell$, let $p_{i_1,\ldots,i_s}$ be the number of mismatches between $\mathbf{y}_{i_1}$ and $\mathbf{c}^*$ at the positions in $Q_{i_1,\ldots,i_s}$. Let

$$\rho_s = \min_{1 \leq i_1,\ldots,i_s \leq n} \frac{p_{i_1,\ldots,i_s}}{d^*}.$$

Notice that for any $2 \leq s \leq r+1$, $\rho_s \leq 1$.

Claim 11.9 (Claim 2.2 [144]). [1] *For any s such that $2 \leq s \leq r$, there are indices $1 \leq i_1, i_2, \ldots, i_r \leq \ell$ such that for any $\mathbf{x} = \mathbf{y}_{\ell'} \in Y$, $|J(\ell')| \leq (\rho_s - \rho_{s+1})d^*$.*

Proof. Consider indices $1 \leq i_1, \ldots, i_s \leq \ell$ such that $p_{i_1,\ldots,i_s} = \rho_s \cdot d^*$. Next arbitrarily pick $r-s$ indices $i_{s+1}, i_{s+2}, \ldots, i_r$ from $[\ell] \setminus \{i_1, \ldots, i_s\}$. Next we prove that $i_1, i_2, \ldots, i_r$ are the required set of indices. Towards that, fix $\mathbf{x} = \mathbf{y}_{\ell'} \in Y$,

$$\begin{aligned}
J(\ell') &= |\{j \in Q_{i_1,\ldots,i_r} : \mathbf{y}_{i_1}[j] \neq \mathbf{x}[j] \text{ and } \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| \\
&\leq |\{j \in Q_{i_1,\ldots,i_s} : \mathbf{y}_{i_1}[j] \neq \mathbf{x}[j] \text{ and } \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| \\
&\qquad\qquad \text{(Because } Q_{i_1,\ldots,i_r} \subseteq Q_{i_1,\ldots,i_s}\text{)} \\
&= |\{j \in Q_{i_1,\ldots,i_s} : \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\} \setminus \{j \in Q_{i_1,\ldots,i_s} : \mathbf{y}_{i_1}[j] = \mathbf{x}[j] \wedge \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| \\
&= |\{j \in Q_{i_1,\ldots,i_s} : \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\} \setminus \{j \in Q_{i_1,\ldots,i_s,\ell'} : \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| \\
&\qquad\qquad \text{(Since } \mathbf{x} = \mathbf{y}_{\ell'}\text{)} \\
&= |\{j \in Q_{i_1,\ldots,i_s} : \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| - |\{j \in Q_{i_1,\ldots,i_s,\ell'} : \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| \qquad (11.1) \\
&= p_{i_1,\ldots,i_s} - p_{i_1,\ldots,i_s,\ell'} \qquad \text{(By definition)} \\
&\leq (\rho_s - \rho_{s+1})d^* \qquad (11.2)
\end{aligned}$$

The equality (11.1) holds since $Q_{i_1,\ldots,i_s} \supseteq Q_{i_1,\ldots,i_s,\ell'}$. The inequality (11.2) holds because $p_{i_1,\ldots,i_s} = \rho_s \cdot d^*$ by the choice of $i_1, \ldots, i_s$, and $\rho_{s+1}d^* \leq p_{i_1,\ldots,i_s,\ell'}$ by definition. □

Notice that $(\rho_2 - \rho_3) + (\rho_3 - \rho_4) + \ldots + (\rho_r - \rho_{r+1}) = (\rho_2 - \rho_{r+1}) \leq \rho_2 \leq 1$. Thus, one of $(\rho_2 - \rho_3), (\rho_3 - \rho_4), \ldots, (\rho_r - \rho_{r+1})$ is at most $1/(r-1)$. This completes the proof. □

Consider the instance $J = (k, X = X_1^i \uplus \ldots X_k^i, \mathcal{R})$ of Binary Constrained Partition Center. Let $C^* = (\mathbf{c}_1^*, \cdots, \mathbf{c}_k^*) \subset \{0,1\}^d$ be an optimum solution to J. Let $d_{opt} = \mathsf{OPT}(J) = \max_{i \in [k], \mathbf{x} \in X_i} d_H(\mathbf{x}, \mathbf{c}_i^*)$. For each $i \in [k]$ and $r \geq 2$, by Lemma 11.8, there exist r elements $\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(r)}$ of X_i such that for any $\mathbf{x} \in X_i$,

$$d_H^P(\mathbf{x}, \mathbf{x}_i^{(1)}) - d_H^P(\mathbf{x}, \mathbf{c}_i^*) \leq \frac{1}{r-1} d_{opt}, \qquad (11.3)$$

[1] We remark that Claim 2.2 in [144] is stated for a vector $\mathbf{c}$ such that $d^* = \text{cost}(Y, \{\mathbf{c}\}) = \min_{\mathbf{c}'} \text{cost}(Y, \{\mathbf{c}'\})$. But the steps of the same proof work in our case as well.

where P is any subset of Q_i, and Q_i is the set of coordinates on which $\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(r)}$ agree. Let us denote as Q the intersection of all Q_i from which the positions not satisfying $\mathcal{R}$ are removed. That is,

$$Q = \left\{ j \in \bigcap_{i \in [k]} Q_i \colon (\mathbf{x}_1^{(1)}[j], \mathbf{x}_2^{(1)}[j], \ldots, \mathbf{x}_k^{(1)}[j]) \in R_j \right\}.$$

Because of (11.3), there is an approximate solution where the coordinates $j \in Q$ are identified using $\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_k^{(1)}$. Let $\overline{Q} = [d] \setminus Q$. Now the idea is to solve the problem restricted to $\overline{Q}$ separately, and then complement the solution on Q by the values of $\mathbf{x}_i^{(1)}$. We prove that for the 'subproblem' restricted on $\overline{Q}$, the optimum value is *large*. Towards that we first prove the following lemma.

Lemma 11.10. *Let $J = (k, X = X_1^i \uplus \ldots X_k^i, \mathcal{R})$ be an instance of* Binary Constrained Partition Center. *Let $(\mathbf{c}_1^*, \ldots, \mathbf{c}_k^*)$ be an optimal solution for J, and $r \geq 2$ be an integer. Then, there exist $\{\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_1^{(r)}\} \subset X_1, \ldots, \{\mathbf{x}_k^{(1)}, \ldots, \mathbf{x}_k^{(r)}\} \subset X_k$ with the following properties. For each $i \in [k]$, let Q_i be the set of coordinates on which $\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(r)}$ agree, $Q = \left\{ j \in \bigcap_{i \in [k]} Q_i \colon (\mathbf{x}_1^{(1)}[j], \mathbf{x}_2^{(1)}[j], \ldots, \mathbf{x}_k^{(1)}[j]) \in R_j \right\}$, and $\overline{Q} = [d] \setminus Q$.*

- *For any $i \in [k]$ and $\mathbf{x} \in X_i$, $d_H^Q(\mathbf{x}, \mathbf{x}_i^{(1)}) - d_H^Q(\mathbf{x}, \mathbf{c}_i^*) \leq \frac{1}{r-1}\mathsf{OPT}(J)$, and*
- *$|\overline{Q}| \leq rk \cdot \mathsf{OPT}(J)$.*

Proof. Fix an integer $i \in [k]$. By substituting $Y = X_i$ and $\mathbf{c}^* = \mathbf{c}_i^*$ in Lemma 11.8, we get $\{\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(r)}\} \subset X_i$ such that for any $\mathbf{x} \in X_i$, $d_H^Q(\mathbf{x}, \mathbf{x}_i^{(1)}) - d_H^Q(\mathbf{x}, \mathbf{c}_i^*) \leq \frac{1}{r-1}\mathsf{OPT}(J)$. That is, we have proved the first condition in the lemma. The second condition is proved in the following claim.

Claim 11.11. $|\overline{Q}| \leq rk \cdot \mathsf{OPT}(J)$.

Proof. We claim that for each position $j \in \overline{Q}$ there exist $i \in [k]$ and $s \in [r]$ such that $\mathbf{x}_i^{(s)}[j] \neq \mathbf{c}_i^*[j]$. There are two kinds of positions in $\overline{Q}$. First, positions, where for some $i \in [k]$ vectors $\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(r)}$ do not agree, in this case certainly one of them does not agree with the corresponding position in $\mathbf{c}_i^*$. Second, positions j where for any $i \in [k]$, $\mathbf{x}_i^{(1)}[j] = \mathbf{x}_i^{(2)}[j] = \cdots = \mathbf{x}_i^{(r)}[j]$, but $(\mathbf{x}_1^{(1)}[j], \ldots, \mathbf{x}_k^{(1)}[j]) \notin R_j$. Then, there exists $i \in [k]$ such that $\mathbf{x}_i^{(1)}[j] \neq \mathbf{c}_i^*[j]$ because $(\mathbf{c}_i^*[j], \ldots, \mathbf{c}_k^*[j]) \in R_j$.

Now, for any $i \in [k]$ and $s \in [r]$, $\mathbf{x}_i^{(s)}$ contributes at most $\mathsf{OPT}(J)$ positions to $\overline{Q}$, since $d_H(\mathbf{x}_i^{(s)}, \mathbf{c}_i^*) \leq \mathsf{OPT}(J)$. Thus, in total there are at most $rk \cdot \mathsf{OPT}(J)$ positions in $\overline{Q}$. □

This completes the proof of the lemma. □

As mentioned earlier, we fix the entries of our solution in positions j of Q with values in $\mathbf{x}_1^{(1)}[j], \ldots, \mathbf{x}_k^{(1)}[j]$. Towards finding the entries of our solution in positions of $\overline{Q}$, we define the following problem and solve it.

BINARY CONSTRAINED PARTITION CENTER*

Input:	A positive integer k, a set $X \subseteq \{0,1\}^d$ of n vectors partitioned into $X_1 \uplus \ldots \uplus X_k$, a tuple of k-ary relations $\mathcal{R} = (R_1, \ldots, R_d)$, and for all $\mathbf{x} \in X$, $d_{\mathbf{x}} \in \mathbb{N}$
Task:	Among all tuples $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ of vectors from $\{0,1\}^d$ satisfying $\mathcal{R}$, find a tuple C that minimizes the integer D such that for all $i \in [k]$ and $\mathbf{x} \in X_i$, $d_H(\mathbf{x}, \mathbf{c}_i) \leq D - d_{\mathbf{x}}$.

Lemma 11.12. *Let $J' = (k, X = X_1 \uplus \ldots X_k, \mathcal{R}, (d_{\mathbf{x}})_{\mathbf{x}\in X})$ be an instance of* BINARY CONSTRAINED PARTITION CENTER**, $\mathsf{OPT}(J') \geq \frac{d}{c}$ for some integer c, and $0 < \delta < 1/c$. Then, there is an algorithm which runs in time $n^{\mathcal{O}(c^2k/\delta^2)}d^{\mathcal{O}(1)}$, and outputs a solution C of J', of cost at most $(1+\delta)\mathsf{OPT}(J')$ with probability at least $1 - n^{-2}$.*

Before proving Lemma 11.12, we explain how all these results are put together to form a proof of Lemma 11.5. We restate Lemma 11.5 for the convenience of the reader.

Lemma 11.5. *There is an algorithm for* BINARY CONSTRAINED PARTITION CENTER *that given an instance $J = (k, X = X_1^i \uplus \ldots \uplus X_k^i, \mathcal{R})$ and $0 < \varepsilon < 1/2$, runs in time $n^{\mathcal{O}((k/\varepsilon)^4)}d^{\mathcal{O}(1)}$, and outputs a solution of cost at most $(1+\varepsilon)\mathsf{OPT}(J)$ with probability at least $1 - n^{-2}$.*

Proof. Let $J = (k, X = X_1^i \uplus \ldots X_k^i, \mathcal{R})$ be the input instance of BINARY CONSTRAINED PARTITION CENTER, and $0 < \varepsilon < \frac{1}{2}$ be the error parameter. Let $(\mathbf{c}_1^*, \ldots, \mathbf{c}_k^*)$ be an optimal solution for J. Let $r \geq 2$ be an integer which we fix later. First, for each $i \in [k]$ we obtain r vectors $\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(r)} \in X_i$ which satisfy the conditions of Lemma 11.10. Their existence is guaranteed by Lemma 11.10, and we guess them in time $n^{\mathcal{O}(rk)}$ over all $i \in [k]$. For each $i \in [k]$, let Q_i be the set of coordinates on which $\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(r)}$ agree, $Q = \left\{ j \in \bigcap_{i\in[k]} Q_i \colon (\mathbf{x}_1^{(1)}[j], \mathbf{x}_2^{(1)}[j], \ldots, \mathbf{x}_k^{(1)}[j]) \in R_j \right\}$, and $\overline{Q} = [d] \setminus Q$. Next, we construct a solution $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ as follows. For each $i \in [k]$ and $j \in Q$, we set $\mathbf{c}_i[j] = \mathbf{x}_i^{(1)}[j]$.

Towards finding the entries of vectors $\mathbf{c}_1, \ldots, \mathbf{c}_k$ at the coordinates in $\overline{Q}$, we use Lemma 11.12. Let J' be the instance of BINARY CONSTRAINED PARTITION CENTER*, which is a natural restriction of J to $\overline{Q}$. That is, $J' = (k, X' = X_1' \uplus \ldots X_k', \mathcal{R}|_{\overline{Q}}, (d_{\mathbf{x}|_{\overline{Q}}})_{\mathbf{x}\in X'})$, where for each $i \in [k]$, $X_i' = \{\mathbf{x}|_{\overline{Q}} \colon \mathbf{x} \in X_i\}$ and for each $\mathbf{x} \in X_i$, $d_{\mathbf{x}|_{\overline{Q}}} = d_H^Q(\mathbf{x}, \mathbf{x}_i^{(1)})$. By the second condition in Lemma 11.10, we have that $|\overline{Q}| \leq rk \cdot \mathsf{OPT}(J)$.

Claim 11.13. $\mathsf{OPT}(J) \leq \mathsf{OPT}(J') \leq \left(1 + \frac{1}{r-1}\right)\mathsf{OPT}(J)$.

Proof. First, we prove that $\mathsf{OPT}(J) \leq \mathsf{OPT}(J')$. Towards that we show that we can transform a solution $C' = (\mathbf{c}'_1, \cdots, \mathbf{c}'_k)$ of J' with the objective value D to a solution C of J with the same objective value. For each $i \in [k]$, consider $\widehat{\mathbf{c}}_i$ which is equal to $\mathbf{x}_i^{(1)}$ restricted to Q, and to $\mathbf{c}'_i$ restricted to $\overline{Q}$, and the solution $\widehat{C} = (\widehat{\mathbf{c}}_1, \cdots, \widehat{\mathbf{c}}_k)$. Clearly, $\widehat{C}$ satisfies $\mathcal{R}$ since on $\overline{Q}$ it is guaranteed by C' being a solution to J', and on Q by construction of Q. The objective value of C is

$$\begin{aligned}
\max_{i\in[k],\mathbf{x}\in X_i} d_H(\mathbf{x}, \mathbf{c}_i) &= \max_{i\in[k],\mathbf{x}\in X_i} \left(d_H^{\overline{Q}}(\mathbf{x}, \mathbf{c}_i) + d_H^{Q}(\mathbf{x}, \mathbf{c}_i)\right) \\
&= \max_{i\in[k],\mathbf{x}\in X_i} \left(d_H(\mathbf{x}|_{\overline{Q}}, \mathbf{c}'_i) + d_H^{Q}(\mathbf{x}, \mathbf{x}_i^{(1)})\right) \\
&= \max_{i\in[k],\mathbf{x}\in X_i} \left(d_H(\mathbf{x}|_{\overline{Q}}, \mathbf{c}'_i) + d_{\mathbf{x}|_{\overline{Q}}}\right) = D.
\end{aligned}$$

Thus, $\mathsf{OPT}(J) \leq \mathsf{OPT}(J')$.

Next, we prove that $\mathsf{OPT}(J') \leq \left(1 + \frac{1}{r-1}\right)\mathsf{OPT}(J)$. Recall that $(\mathbf{c}_1^*, \ldots, \mathbf{c}_k^*)$ is an optimal solution for J. Then, $(\mathbf{e}_1^*, \ldots, \mathbf{e}_k^*)$, where each $\mathbf{e}_i^*$ is the restriction of $\mathbf{c}_i^*$ on $\overline{Q}$, is a solution for J'. For each $i \in [k]$ and $\mathbf{x} \in X_i$,

$$\begin{aligned}
d_H(\mathbf{x}|_{\overline{Q}}, \mathbf{e}_i^*) + d_{\mathbf{x}|_{\overline{Q}}} &= d_H^{\overline{Q}}(\mathbf{x}, \mathbf{c}_i^*) + d_H^{Q}(\mathbf{x}, \mathbf{x}_i^{(1)}) \\
&\leq d_H^{\overline{Q}}(\mathbf{x}, \mathbf{c}_i^*) + d_H^{Q}(\mathbf{x}, \mathbf{c}_i^*) + \frac{1}{r-1}\mathsf{OPT}(J) \qquad \text{(By Lemma 11.10)} \\
&\leq d_H(\mathbf{x}, \mathbf{c}_i^*) + \frac{1}{r-1}\mathsf{OPT}(J) \\
&\leq \left(1 + \frac{1}{r-1}\right)\mathsf{OPT}(J)
\end{aligned}$$

This completes the proof of the claim. □

Since $|\overline{Q}| \leq rk \cdot \mathsf{OPT}(J)$ and by Claim 11.13, we have that $\mathsf{OPT}(J') \geq \frac{|\overline{Q}|}{rk} = \frac{|\overline{Q}|}{c}$, where $c = rk$. Let $0 < \delta < \frac{1}{c}$ be a number which we fix later.

Now we apply Lemma 11.12 on the input J' and δ, and let $C' = (\mathbf{c}'_1, \ldots, \mathbf{c}'_k)$ be the solution for J' obtained. We know that the cost d' of $\mathbf{c}'$ is at most $(1+\delta)\mathsf{OPT}(J')$ with probability at least $1 - n^{-2}$. For the rest of the proof we assume that the cost $d' \leq (1+\delta)\mathsf{OPT}(J')$. Recall that we have partially computed the entries of the solution $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ for the instance J. That is, for each $j \in Q$ and $i \in [k]$, we have already set the value of $\mathbf{c}_i[j]$. Notice that $C' \subseteq \{0,1\}^{|\overline{Q}|}$. Since J' is obtained from J by restricting to $\overline{Q}$, there is a natural bijection f from $\overline{Q}$ to $[|\overline{Q}|]$ such that for each $\mathbf{x} \in X$ and $j \in \overline{Q}$, $\mathbf{x}[j] = \mathbf{y}[f(j)]$, where $\mathbf{y} = \mathbf{x}|_{\overline{Q}}$. Now for each $i \in [k]$ and $j \in \overline{Q}$, we set $\mathbf{c}_i[j] = \mathbf{c}'_i[f(j)]$.

In Claim 11.13, we have proven that the solution C of J obtained in this way has cost at most d'. By Lemma 11.12, we know that $d' \leq (1+\delta)\mathsf{OPT}(J')$. By Claim 11.13, $\mathsf{OPT}(J') \leq (1+\frac{1}{r-1})\mathsf{OPT}(J)$. Thus, we have that the cost of the solution C of J is at most $(1+\delta)(1+\frac{1}{r-1})\mathsf{OPT}(J)$. Now we fix $r = (1+\frac{4}{\varepsilon})$ and $\delta = \frac{\varepsilon}{(2\varepsilon+8)k}$. Then the cost of C is at most $(1+\varepsilon)\mathsf{OPT}(J)$.

Running time analysis. The number of choices for $\{\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_1^{(r)}\} \subset X_1$, $\ldots$, $\{\mathbf{x}_k^{(1)}, \ldots, \mathbf{x}_k^{(r)}\} \subset X_k$ is at most $n^{\mathcal{O}(rk)} = n^{\mathcal{O}(k/\varepsilon)}$. For each such choice, we run the algorithm of Lemma 11.12 which takes time at most $n^{\mathcal{O}(c^2k/\delta^2)}d^{\mathcal{O}(1)} = n^{\mathcal{O}((k/\varepsilon)^4)}d^{\mathcal{O}(1)}$. Thus, the total running time is $n^{\mathcal{O}((k/\varepsilon)^4)}d^{\mathcal{O}(1)}$. □

Now the only piece left is the proof of Lemma 11.12. We use the following tail inequality (a variation of Chernoff bound) in the proof of Lemma 11.12.

Proposition 11.14 (Lemma 1.2 [144])**.** *Let $X_1, \ldots, X_n$ be n independent* 0-1 *random variables, $X = \sum_{i=1}^n X_i$, and $0 < \varepsilon \leq 1$. Then,* $\Pr[X > \mathrm{E}[X] + \varepsilon n] \leq e^{-\frac{1}{3}n\varepsilon^2}$.

Finally, we prove Lemma 11.12.

Proof of Lemma 11.12. First, assume that $d < 9c^2 \log n/\delta^2$. If this is the case, we enumerate all possible solutions for J' and output the best solution. The number of solutions is at most $2^{kd} = n^{\mathcal{O}(c^2k/\delta^2)}$. Thus, in this case the algorithm is exact and deterministic, and the running time bound holds. For the rest of the proof we assume that $d \geq 9c^2 \log n/\delta^2$.

Binary Constrained Partition Center* can be formulated as a 0-1 optimization problem as explained below. For each $j \in [d]$ and tuple $t \in R_j$, we use a 0-1 variable $y_{j,t}$ to indicate whether the j^{th} entries of a solution form a tuple $t \in R_j$ or not. For any $i \in [k]$, $\mathbf{x} \in X_i$, $j \in [d]$ and $t \in R_j$, denote $\chi_i(\mathbf{x}[j], t) = 0$ if $\mathbf{x}[j] = t[i]$ and $\chi_i(\mathbf{x}[j], t) = 1$ if $\mathbf{x}[j] \neq t[i]$. Now Binary Constrained Partition Center* can be defined as the following 0-1 optimization problem.

$$
\begin{aligned}
&\min D \\
&\text{subject to} \\
&\sum_{t \in R_j} y_{j,t} = 1, && \text{for all } j \in [d]; \\
&\sum_{j \in [m]} \sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot y_{j,t} \leq D - d_{\mathbf{x}}, && \text{for all } i \in [k] \text{ and } \mathbf{x} \in X_i \\
&y_{j,t} \in \{0, 1\}, && \text{for all } j \in [d] \text{ and } t \in R_j.
\end{aligned}
\tag{11.4}
$$

Any solution $y_{j,t}$ ($j \in [d]$ and $t \in R_j$) to (11.4) corresponds to the solution $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ where for all $j \in [d]$ and $t \in R_j$ such that $y_{j,t} = 1$, we have $(\mathbf{c}_1[j], \ldots, \mathbf{c}_k[j]) = t$.

Now, we solve the above optimization problem using linear programming relaxation and obtain a fractional solution $y^\star_{j,t}$ ($j \in [d]$ and $t \in R_j$) with cost D'. Clearly, $D' \leq d_{opt} = \mathsf{OPT}(J')$. Now, for each $j \in [d]$, independently with probability $y^\star_{j,t}$, we set $y'_{j,t} = 1$ and $y'_{j,t'} = 0$, for any $t' \in R_j \setminus \{t\}$. Then $y'_{j,t}$ ($j \in [d]$ and $t \in R_j$) form a solution to (11.4). Next we construct the solution $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ to Binary Constrained Partition Center*, corresponding to $y'_{j,t}$ ($j \in [d]$ and $t \in R_j$). That is, for all $j \in [d]$ and $t \in R_j$ such that $y_{j,t} = 1$, we have $(\mathbf{c}_1[j], \ldots, \mathbf{c}_k[j]) = t$.

For the running time analysis, notice that solving the linear program and performing the random rounding takes polynomial time in the size of the problem (11.4). And the size of (11.4) is polynomial in the size of J', so the running time bound is satisfied. It remains to show that the constructed solution has cost at most $(1+\delta)\mathsf{OPT}(J')$ with probability at least $1 - n^{-2}$.

For any $j \in [d]$, the above random rounding procedure ensures that there is exactly one tuple $t \in R_j$ such that $y'_{j,t} = 1$. This implies that for any $j \in [d]$, $i \in [k]$ and $\mathbf{x} \in X_i$, $\sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot y'_{j,t}$ is a 0-1 random variable. Since for each $j \in [d]$ the rounding procedure is independent, we have that for any $i \in [k]$ and $\mathbf{x} \in X_i$ the random variables $(\sum_{t \in R_1} \chi_i(\mathbf{x}[1], t) \cdot y'_{1,t}), \ldots, (\sum_{t \in R_d} \chi_i(\mathbf{x}[d], t) \cdot y'_{j,t})$ are independent. Hence, for any $i \in [k]$ and $\mathbf{x} \in X_i$, the Hamming distance between $\mathbf{x}$ and $\mathbf{c}_i$, $d_H(\mathbf{x}, \mathbf{c}_i) = \sum_{j \in [d]} \sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot y'_{j,t}$, is the sum of d independent 0-1 random variables. For each $i \in [k]$ and $\mathbf{x} \in X_i$, we upper bound the expected value of $d_H(\mathbf{x}, \mathbf{c}_i)$ as follows.

$$\begin{aligned}
E[d_H(\mathbf{x}, \mathbf{c}_i)] &= E\left[\sum_{j \in [d]} \sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot y'_{j,t}\right] \\
&= \sum_{j \in [d]} \sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot E[y'_{j,t}] \\
&= \sum_{j \in [d]} \sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot y^\star_{j,t} \\
&\leq D' - d_{\mathbf{x}} \qquad \text{(By the constraints of (11.4))}
\end{aligned}$$

Fix $\varepsilon = \frac{\delta}{c}$. Then, by Proposition 11.14, for all $i \in [k]$, and $\mathbf{x} \in X_i$,

$$\Pr[d_H(\mathbf{x}, \mathbf{c}_i) > D' - d_{\mathbf{x}} + \varepsilon d] \leq e^{-\frac{1}{3} d \varepsilon^2}.$$

Therefore, by the union bound,

$$\Pr[\text{There exist } i \in [k] \text{ and } \mathbf{x} \in X_i \text{ such that } d_H(\mathbf{x}, \mathbf{c}_i) > D' - d_{\mathbf{x}} + \varepsilon d] \leq n \cdot e^{-\frac{1}{3} d \varepsilon^2} \tag{11.5}$$

We remind that $d \geq 9c^2 \log n/\delta^2 = 9 \log n/\varepsilon^2$ and so $n \cdot e^{-\frac{1}{3} d \varepsilon^2} \leq n^{-2}$. Thus, by (11.5),

$$\Pr[\text{There exist } i \in [k] \text{ and } \mathbf{x} \in X_i \text{ such that } d_H(\mathbf{x}, \mathbf{c}_i) > D' - d_{\mathbf{x}} + \varepsilon d] \leq n^{-2}. \tag{11.6}$$

Since $D' \leq \mathsf{OPT}(J')$ and $\mathsf{OPT}(J') \geq d/c$, $D' + \varepsilon d \leq (1 + c\varepsilon)\mathsf{OPT}(J')$. Then, the probability that there exist $i \in [k]$ and $\mathbf{x} \in X_i$ such that $d_H(\mathbf{x}, \mathbf{c}_i) > (1+c\varepsilon)\mathsf{OPT}(J') - d_{\mathbf{x}}$ is at most n^{-2} by (11.6). Since $c\varepsilon = \delta$, the proof is complete. □

11.4 Applications

In this section we explain the impact of Theorem 11.3 about Binary Constrained k-Center to other problems around low-rank matrix approximation. We would like to mention that Binary Constrained k-Center is very similar to the Binary Constrained Clustering problem from [88]. In Binary Constrained k-Center we want to minimize the maximum distance of a vector from the input set of vectors to the closest center, whereas in Binary Constrained Clustering the sum of distances is minimized. While these problems are different, the reduction we explain here, except a few details, are identical to the ones described in [88]. For reader's convenience, we give one reduction (Lemma 11.15) in full details and skip all other reductions, which are similar.

In the following lemma we show that L_1-Rank-r Approximation over $\mathsf{GF}(2)$ is a special case of Binary Constrained k-Center.

Lemma 11.15. *There is an algorithm that given an instance $(\mathbf{A}, r)$ of L_1-Rank-r Approximation over $\mathsf{GF}(2)$, where $\mathbf{A}$ is an $n \times d$-matrix and r is an integer, runs in time $\mathcal{O}(n + d + 2^{2r})$, and outputs an instance $J = (X, k = 2^r, \mathcal{R})$ of Binary Constrained k-Center with the following property. Given any α-approximate solution C to J, an α-approximate solution $\mathbf{L}$ to $(\mathbf{A}, r)$ can be constructed in time $\mathcal{O}(rnd)$ and vice versa.*

Proof. Notice that if $\mathsf{GF}(2)\text{−rank}(\mathbf{L}) \leq r$, then $\mathbf{L}$ has at most 2^r distinct rows, because each row is a linear combination of at most r vectors of a basis of the row space of $\mathbf{L}$. Moreover, L_1-Rank-r Approximation over $\mathsf{GF}(2)$ can also be stated as follows: find vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r \in \{0,1\}^d$ such that $\max_{i \in [n]} d_H(\mathbf{a}_i, S)$ is minimum, where $\mathbf{a}_1, \ldots, \mathbf{a}_n$ are the rows of $\mathbf{A}$ and

$$S = \{\mathbf{s} \in \{0,1\}^d \colon \mathbf{s} \text{ is a linear combination of } \mathbf{s}_1, \ldots, \mathbf{s}_r \text{ over } \mathsf{GF}(2)\}.$$

To encode an instance of L_1-Rank-r Approximation over $\mathsf{GF}(2)$ as an instance of Binary Constrained k-Center, we construct the following relation R. Set $k = 2^r$. Let $\Lambda = (\lambda_1, \ldots, \lambda_k)$ be the k-tuple composed of all distinct vectors in $\{0,1\}^r$. Thus, each element $\lambda_i \in \Lambda$ is a binary r-vector. We define $R = \{(x^\intercal \lambda_1, \ldots, x^\intercal \lambda_k) \mid x \in \{0,1\}^r\}$. Thus, R consists of $k = 2^r$ k-tuples and every k-tuple in R is a row of the matrix $\Lambda^\intercal \cdot \Lambda$. Now we define X to be the set of rows of $\mathbf{A}$ and for each $i \in [d]$, $R_i = R$. Our algorithm outputs the instance $J = (X, k, \mathcal{R} = (R_1, \ldots, R_d))$.

To show that the instance $(\mathbf{A}, r)$ of L_1-Rank-r Approximation over $\mathsf{GF}(2)$ is equivalent to the constructed instance J, assume first that the vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r \in$

$\{0,1\}^d$ compose an (approximate) solution of L_1-Rank-r Approximation over $\mathsf{GF}(2)$. For every $i \in [k]$ define the vector

$$\mathbf{c}_i = \lambda_i[1]\mathbf{s}_1 \oplus \cdots \oplus \lambda_i[r]\mathbf{s}_r,$$

where $\lambda_i^\intercal = (\lambda_i[1], \ldots, \lambda_i[r])$, $\oplus$ denotes the sum over $\mathsf{GF}(2)$, and define the tuple $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$. That is, C contains all linear combinations of $\mathbf{s}_1, \ldots, \mathbf{s}_r$. For every $i \in [k]$ and $j \in [d]$, we have that $\mathbf{c}_i[j] = (\mathbf{s}_1[j], \ldots, \mathbf{s}_r[j])\lambda_i$. Therefore, $(\mathbf{c}_1[j], \ldots, \mathbf{c}_k[j]) \in R$ for all $j \in [d]$. Thus, C is a solution to J of cost $\max_{\mathbf{x}\in X} d_H(\mathbf{x}, C)$.

For the opposite direction, assume that $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ is an (approximate) solution to J. We construct the vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r$ as follows. Let $j \in [d]$. We have that $(\mathbf{c}_1[j], \ldots, \mathbf{c}_k[j]) \in R$. Therefore, there is $\mathbf{x} \in \{0,1\}^r$ such that $(\mathbf{c}_1[j], \ldots, \mathbf{c}_k[j]) = (\mathbf{x}^\intercal\lambda_1, \ldots, \mathbf{x}^\intercal\lambda_k)$. We set $\mathbf{s}_i[j] = \mathbf{x}[i]$ for $i \in [r]$. Observe that vectors in C are linear combinations of the vectors $\mathbf{s}_1, \ldots, \mathbf{s}_r$. This immediately implies that for any α-approximate solution C of J an α-approximate solution $\mathbf{L}$ of $(\mathbf{A}, r)$ can be constructed in time $\mathcal{O}(rnd)$. □

Thus, Theorem 11.1 follows from Theorem 11.3 and Lemma 11.15.

Low Boolean-Rank Approximation. Let $\mathbf{A}$ be a binary $n \times d$ matrix. Now we consider the elements of $\mathbf{A}$ to be *Boolean* variables. The *Boolean rank* of $\mathbf{A}$ is the minimum r such that $\mathbf{A} = \mathbf{U} \wedge \mathbf{V}$ for a Boolean $n \times r$ matrix $\mathbf{U}$ and a Boolean $r \times d$ matrix $\mathbf{V}$, where the product is Boolean, that is, the logical $\wedge$ plays the role of multiplication and $\vee$ the role of sum. Here $0 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 1 = 1$, $0 \vee 0 = 0$, $0 \vee 1 = 1$, and $1 \vee 1 = 1$. Thus the matrix product is over the Boolean semi-ring $(0, 1, \wedge, \vee)$. This can be equivalently expressed as the normal matrix product with addition defined as $1 + 1 = 1$. Binary matrices equipped with such algebra are called *Boolean matrices*.

In Boolean L_1-Rank-r Approximation, we are given an $n \times d$ binary data matrix $\mathbf{A}$ and a positive integer r, and we seek a binary matrix $\mathbf{L}$ optimizing

$$\begin{aligned} &\text{minimize } \|\mathbf{A} - \mathbf{L}\|_1 \\ &\text{subject to } \operatorname{rank}(\mathbf{L}) \leq r. \end{aligned}$$

Here, by the rank of binary matrix $\mathbf{L}$ we mean its Boolean rank, and norm $\|\cdot\|_1$ is the *column sum norm*. Similar to Lemma 11.15, one can prove that Boolean L_1-Rank-r Approximation is a special case of Binary Constrained k-Center, where $k = 2^r$. Thus, we get the following corollary from Theorem 11.3.

Corollary 11.16. *There is an algorithm for* Boolean L_1-Rank-r Approximation *that given an instance $I = (\mathbf{A}, r)$ and $0 < \varepsilon < 1$, runs in time $n^{\mathcal{O}(2^{4r}/\varepsilon^4)}d^{\mathcal{O}(1)}$, and outputs a $(1+\varepsilon)$-approximate solution with probability at least $1 - 2n^{-2}$.*

Projective k-center. The Binary Projective k-Center problem is a variation of the Binary k-Center problem, where the centers of clusters are linear subspaces of bounded dimension r. (For $r = 1$ this is Binary k-Center and for $k = 1$ this is L_1-Rank-r Approximation over $\mathsf{GF}(2)$.) Formally, in Binary Projective k-Center we are given a set $X \subseteq \{0,1\}^d$ of n vectors and positive integers k and r. The objective is to find a family of r-dimensional linear subspaces $C = \{C_1, \ldots, C_k\}$ over $\mathsf{GF}(2)$ minimizing $\max_{\mathbf{x} \in X} d_H(\mathbf{x}, \bigcup_{i=1}^{k} C)$.

To see that Binary Projective k-Center is a special case of Binary Constrained k-Center, we observe that the condition that C_i is an r-dimensional subspace over $\mathsf{GF}(2)$ can be encoded (as in Lemma 11.15) by 2^r constraints. This observation leads to the following lemma.

Lemma 11.17. *There is an algorithm that given an instance (X, r, k) of* Binary Projective k-Center*, runs in time $\mathcal{O}(n+d+2^{\mathcal{O}(rk)})$, and outputs an instance $J = (X, k' = 2^{kr}, \mathcal{R})$ of* Binary Constrained k-Center *with the following property. Given any α-approximate solution C to J, an α-approximate solution C' to (X, r, k) can be constructed in time $\mathcal{O}(rknd)$ and vice versa.*

Combining Theorem 11.3 and Lemma 11.17 together, we get the following corollary.

Corollary 11.18. *There is an algorithm for* Binary Projective k-Center *that given an instance $I = (X, r, k)$ and $0 < \varepsilon < 1$, where $X \subseteq \{0,1\}^d$ is a set of n vectors and $r, k \in \mathbb{N}$, runs in time $n^{\mathcal{O}(2^{4kr}/\varepsilon^4)} d^{\mathcal{O}(1)}$, and outputs a $(1+\varepsilon)$-approximate solution with probability at least $1 - 2n^{-2}$.*

11.5 Conclusion

In this chapter we gave a randomized PTAS for the Binary Constrained k-Center problem. This yields the first approximation scheme for L_1-Rank-r Approximation over $\mathsf{GF}(2)$ and its Boolean variant. This result leaves several interesting open problems. The running time of our $(1+\varepsilon)$-approximation algorithm is $n^{\mathcal{O}(2^{4r} \cdot \varepsilon^{-4})} d^{\mathcal{O}(1)}$. How far is this running time from being optimal? A simple adaptation of the result of Cygan et al. [71] for Closest String, yields that already for $r = 1$, an $(1+\varepsilon)$-approximation in time $n^{\mathcal{O}(1)} \cdot f(\varepsilon)$, for any computable function f, would imply that $\mathsf{FPT} = \mathsf{W}[1]$. Also the existence of a PTAS for $r = 1$ with running time $f(\varepsilon) n^{o(1/\varepsilon)}$, for any computable function f, would contradict the Exponential Time Hypothesis [71]. But these results do not exclude the opportunity of having an algorithm of running time $f(r, \varepsilon) \cdot (nd)^{\mathsf{poly}(1/\varepsilon)}$ for some function f. Even the existence of an algorithm for L_1-Rank-r Approximation over $\mathsf{GF}(2)$ of running time $n^{\mathsf{poly}(r,\varepsilon)} d^{\mathcal{O}(1)}$ is an interesting open question.

Part V

Future Directions

Future Directions

We conclude with listing key open problems that follow from the results presented in this thesis. A number of open questions and concluding remarks are stated in the respective sections at the end of each chapter. However, here we would like to spotlight the most important directions for further research.

Parameterized complexity of k-Means The most intriguing question about parameterized complexity of k-Means concerns the parameterization by $d+k$. Recall that an $n^{\mathcal{O}(dk)}$-time algorithm by Inaba et al. [118] is known from the 1990s, however, so far nothing would contradict an FPT algorithm with respect to this parameter. On the other hand, the W[1]-hardness parameterized by k for constant d, shown by Cohen-Addad et al. [63] in the discrete case[2], strongly suggests that the continuous case might very well have the same behavior. Also, in Chapter 5, we have shown quite strong W[1]-hardness results for several variants of L_p-k-Clustering: for the parameter $d + D$ in the case $p = 0$, the parameter D in the case $p = \infty$, and the W[1]-hardness of L_p-Cluster Selection for the parameterization by $t + d$ in the case $p = 1$, and $t + D$ in the case $1 < p < \infty$. Recall that d is the dimension, k is the number of clusters, D is the optimal cost of clustering, and in the case of L_p-Cluster Selection t denotes the size of the target cluster; for the formal definition of L_p-Cluster Selection we refer to Chapter 5. While in general the parameters D, d and k can compare arbitrarily, intuitively it seems that the parameter D is more or less the "strongest" out of these three. From this discussion, we get naturally the following.

Open Problem 1. *Is k-Means W[1]-hard parameterized by $d+k$? Can this at least be said about some other variant of L_p-k-Clustering?*

Another natural open problem concerns FPT algorithms parameterized by D for versions of L_p-k-Clustering. In Chapter 5, we encountered a curious distinction: while for $p \in \{0, \infty\}$ we showed W[1]-hardness, and for $p = 1$ an FPT algorithm, for

[2]Recall that in the discrete version of a clustering problem, the set of possible center locations is given in the input.

the most interesting case $p = 2$ we have neither. We know only W[1]-hardness of L_2-Cluster Selection, meaning that our approach for an FPT algorithm is not applicable in this case. Thus we arrive to the second question of interest.

Open Problem 2. *Is there an* FPT *algorithm for* k-Means *parameterized by* D*, where input is in* $\mathbb{Z}^d$*?*

Constrained clustering problems In Chapter 7, for a number of constrained clustering problems we have shown the existence of a small-sized coreset. Namely, for any kind of constraints that can be defined in terms of partitioning the input into ℓ colors and measuring the cardinality of each color in each cluster, e.g. capacity or fairness constraints, there is an $\ell \cdot \mathsf{poly}(k, \log n, \varepsilon)$-sized $(1+\varepsilon)$-coreset for both k-Median in general discrete metric spaces and k-Means in $\mathbb{R}^d$. While this bound has a very modest dependence on the size of the input and approaches the best known coresets even in the vanilla setting, it is not "tight" in two aspects. First, while in general metric spaces it is known that the size $\Omega((k \log n)/\varepsilon)$ is required [16], for Euclidean k-Means in the vanilla setting there is a better upper bound of $\mathsf{poly}(k/\varepsilon)$ [84], that is, the size of the coreset is completely independent of the input size. The analogue of this would be interesting even for the simplest kind of constraints that can be defined in terms of only one color, like capacitated or lower-bounded clustering. Second, since there is no colors in the vanilla setting, there is also no dependence on ℓ for vanilla clustering coresets. While the size of the coreset depends linearly on ℓ in all coreset constructions for constrained clustering, to the best of our knowledge there is no known lower bound requiring such a dependence. And if we would aim not for a coreset, but just a $(1+\varepsilon)$-approximate algorithm, for k-Means in $\mathbb{R}^d$ it is known that any kind of constraints admits an efficient FPT-time algorithm that does not depend exponentially on ℓ, as long as the assignment problem can be solved in the same time [27]. We summarize the above in the following open problems.

Open Problem 3. *For* (α, β)-Fair Clustering*, is there a non-trivial coreset construction that achieves a size bound that is independent of the number of colors* ℓ*?*

Moreover, this question is interesting for any variant of fairness constraints, and in fact for any version of color-constrained clustering.

Open Problem 4. *For capacitated* k-Means *problem in the Euclidean space, is there a coreset of size independent of* n*?*

We have chosen capacitated clustering as a working example for Open Problem 4, however this kind of result would be interesting for any non-trivial clustering constraints.

Additionally, in Chapter 7 we provided a $(3+\varepsilon)$ approximation algorithm for color-constrained k-Median in general discrete metric spaces in time $f(k, \ell, \varepsilon) \cdot n^{\mathcal{O}(1)}$

for a certain function f. To the best of our knowledge, there are no known algorithms with a similar running time that improve this approximation factor. On the other hand, it is known that for the vanilla version, such an FPT-time algorithm allows to achieve an approximation factor of $(1+2/e+\varepsilon)$, and this is tight [64]. Thus, even for the simplest kind of constraints, such as capacitated clustering, beating the factor of 3 remains an interesting question.

Open Problem 5. *For capacitated k-Median problem in the general metric setting and any $\varepsilon > 0$, is there a $(3-\varepsilon)$-approximation algorithm with running time $f(k) \cdot n^{\mathcal{O}(1)}$, for some function f?*

Again, such results are not known for any kind of constraints, so progress here would be interesting for any constrained clustering problem.

Clustering affine subspaces In Chapter 8, we presented a $(1+\varepsilon)$-approximation algorithm for k-Means clustering of Δ-points in $\mathbb{R}^d$ with running time $f(k,\Delta,\varepsilon) \cdot (nd)^{\mathcal{O}(1)}$. Recall that Δ-points are vectors in $\mathbb{R}^d$ that can have at most Δ coordinates replaced with the unknown "?" value. Another way to look at this problem, is that the objects to cluster are axis-parallel affine subspaces in $\mathbb{R}^d$ of dimension at most Δ. A natural and intriguing question is whether it is possible to extend this result to clustering of *arbitrary* low-dimensional affine subspaces in the Euclidean space.

Open Problem 6. *Given a set of Δ-dimensional affine subspaces in $\mathbb{R}^d$, is it possible to find a $(1+\varepsilon)$-approximate k-Means clustering of these subspaces in time $f(k,\Delta,\varepsilon) \cdot (nd)^{\mathcal{O}(1)}$? The clustering objective here is the sum of squared distances from each subspace to the respective center, where the center is a point in $\mathbb{R}^d$ and the distance is the Euclidean distance from the center to the closest point on the subspace.*

Efficient dimensionality reduction for PCA with Outliers Recall that in Chapters 9 and 10 we presented an exact $n^{\mathcal{O}(rd)}$-time algorithm and a $(1+\varepsilon)$-approximate $n^{\mathcal{O}(r \log r \varepsilon^{-2})}$-time algorithm for PCA with Outliers. Since we also know that there is no $n^{o(d)}$-time algorithm with any approximation factor, these algorithms are practically tight. However, in our reduction the number of outliers is of order n, so better algorithms in the case $k = o(n)$ are not excluded. Moreover, for the α-heavy assumption on the outliers, we give a $2^{\mathsf{poly}(r,\log k,\log\log n)} \cdot (nd)^{\mathcal{O}(1)}$-time algorithm, which is, in particular, FPT-time when parameterized by $k+r$. So a natural question is, could the same be achieved for other assumptions on the outliers, or for a general-case $(1+\varepsilon)$-approximation algorithm with a suitable dependence on ε? Such an algorithm would be interesting even if the dependence of the running time on k is worse than ours, which is exponential in $\mathsf{poly}(\log k)$. Our algorithm is based on the existence of an efficient dimensionality reduction procedure for the problem, therefore one potential approach to addressing this question may lie in developing a custom dimensionality reduction routine that works well with the PCA with Outliers objective.

Open Problem 7. *Is there a* $(1+\varepsilon)$*-approximation algorithm for* PCA WITH OUTLIERS *with running time* $f(k,r,\varepsilon)\cdot(nd)^{\mathcal{O}(1)}$*, for some function* f*? Could it be possible with a certain reasonable assumption on the outliers?*

Bibliography

[1] Marcel R. Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. Streamkm++ a clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)*, 17:2–1, 2012.

[2] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635, 2004.

[3] Sara Ahmadian, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian. Clustering without over-representation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 267–275, 2019.

[4] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and Euclidean k-median by primal-dual algorithms. In *Proceedings of the 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–72. IEEE Computer Society, 2017.

[5] Sara Ahmadian and Chaitanya Swamy. Improved approximation guarantees for lower-bounded facility location. In *International Workshop on Approximation and Online Algorithms*, pages 257–271. Springer, 2012.

[6] Ravindra K. Ahuja, James B. Orlin, Clifford Stein, and Robert E. Tarjan. Improved algorithms for bipartite network flow, 1994.

[7] Salem Alelyani, Jiliang Tang, and Huan Liu. Feature selection for clustering: A review. In Charu C. Aggarwal and Chandan K. Reddy, editors, *Data Clustering: Algorithms and Applications*, pages 30–373. CRC Press, 2013.

[8] Paul D. Allison. *Missing data*, volume 136. Sage publications, 2001.

[9] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, May 2009.

[10] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[11] Bill Andreopoulos. Clustering categorical data. In Charu C. Aggarwal and Chandan K. Reddy, editors, *Data Clustering: Algorithms and Applications*, pages 277–304. CRC Press, 2013.

[12] D. Angluin and L.G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. *J. Computer and System Sciences*, 18(2):155 – 193, 1979.

[13] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. *ProPublica, May 23, 2016*.

[14] Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of Euclidean k-means. In Lars Arge and János Pach, editors, *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, volume 34 of *LIPIcs*, pages 754–767. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[15] Arturs Backurs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner. Scalable fair clustering. In *International Conference on Machine Learning*, pages 405–413, 2019.

[16] Daniel Baker, Vladimir Braverman, Lingxiao Huang, Shaofeng H-C Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for clustering in graphs of bounded treewidth. *arXiv preprint arXiv:1907.04733*, 2019.

[17] Frank Ban, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P. Woodruff. A PTAS for ℓ_p-low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 747–766. SIAM, 2019.

[18] Eduard Bartl, Radim Belohlávek, and Jan Konecny. Optimal decompositions of matrices with grades into binary and graded matrices. *Annals of Mathematics and Artificial Intelligence*, 59(2):151–167, Jun 2010.

[19] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[20] Luca Becchetti, Marc Bury, Vincent Cohen-Addad, Fabrizio Grandoni, and Chris Schwiegelshohn. Oblivious dimension reduction for k-Means: beyond subspaces and the Johnson-Lindenstrauss lemma. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 1039–1050, 2019.

[21] Radim Belohlávek and Vilém Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Computer and System Sciences*, 76(1):3–20, 2010.

[22] Jon Louis Bentley and James B. Saxe. Decomposable searching problems i: Static-to-dynamic transformation. *J. algorithms*, 1(4):301–358, 1980.

[23] Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. In *Advances in Neural Information Processing Systems*, pages 4954–4965, 2019.

[24] Ioana O. Bercea, Martin Groß, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R. Schmidt, and Melanie Schmidt. On the cost of essentially fair clusterings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[25] Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM review*, 37(4):573–595, 1995.

[26] Aditya Bhaskara and Srivatsan Kumar. Low rank approximation in the presence of outliers. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 116 of *LIPIcs*, pages 4:1–4:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[27] Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Faster Algorithms for the Constrained k-means Problem. *Theory of Computing Systems*, 62(1):93–115, 2018.

[28] Arnab Bhattacharyya, Édouard Bonnet, László Egri, Suprovat Ghoshal, Karthik C. S., Bingkai Lin, Pasin Manurangsi, and Dániel Marx. Parameterized intractability of even set and shortest vector problem. *CoRR*, abs/1909.01986, 2019.

[29] Avrim Blum, John Hopcroft, and Ravi Kannan. *Foundations of Data Science*. June 2017.

[30] Matteo Böhm, Adriano Fazzone, Stefano Leonardi, and Chris Schwiegelshohn. Fair clustering with multiple colors. *arXiv preprint arXiv:2002.07892*, 2020.

[31] Christina Boucher, Christine Lo, and Daniel Lokshantov. Consensus Patterns (probably) has no EPTAS. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015*, pages 239–250, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[32] Christina Boucher, Christine Lo, and Daniel Lokshtanov. Outlier detection for DNA fragment assembly. *CoRR*, abs/1111.0376, 2011.

[33] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal column-based matrix reconstruction. *Computing Research Repository - CORR*, 43, 03 2011.

[34] Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. Unsupervised feature selection for the k-means clustering problem. In *Advances in Neural Information Processing Systems 22 (NIPS)*, pages 153–161. Curran Associates, Inc., 2009.

[35] Christos Boutsidis, Anastasios Zouzias, Michael W. Mahoney, and Petros Drineas. Randomized dimensionality reduction for k-means clustering. *IEEE Trans. Information Theory*, 61(2):1045–1062, 2015.

[36] Thierry Bouwmans, Necdet Serhat Aybat, and El-hadi Zahzah. *Handbook of robust low-rank and sparse matrix decomposition: Applications in image and video processing.* Chapman and Hall/CRC, 2016.

[37] Vladimir Braverman, Dan Feldman, and Harry Lang. New frameworks for offline and streaming coreset constructions. *arXiv preprint arXiv:1612.00889*, 2016.

[38] Karl Bringmann, Pavel Kolev, and David P. Woodruff. Approximation algorithms for ℓ_0-low rank approximation. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pages 6651–6662, 2017.

[39] Jaroslaw Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 722–736. SIAM, 2015.

[40] Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2):23:1–23:31, 2017.

[41] Jaroslaw Byrka, Bartosz Rybicki, and Sumedha Uniyal. An approximation algorithm for uniform capacitated k-median problem with 1+\epsilon capacity

violation. In Quentin Louveaux and Martin Skutella, editors, *Integer Programming and Combinatorial Optimization - 18th International Conference, IPCO 2016, Liège, Belgium, June 1-3, 2016, Proceedings*, volume 9682 of *Lecture Notes in Computer Science*, pages 262–274. Springer, 2016.

[42] Toon Calders and Sicco Verwer. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21(2):277–292, 2010.

[43] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, 2011.

[44] Fabrizio Caruso. The SARAG library: Some algorithms in real algebraic geometry. In Andrés Iglesias and Nobuki Takayama, editors, *Mathematical Software - ICMS 2006*, pages 122–131, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[45] Deeparnab Chakrabarty and Chaitanya Swamy. Facility location with client latencies: Lp-based techniques for minimum-latency problems. *Mathematics of Operations Research*, 41(3):865–883, 2016.

[46] L. Sunil Chandran, Davis Issac, and Andreas Karrenbauer. On the parameterized complexity of biclique cover and partition. In *Proceedings of the 11th International Symposium on Parameterized and Exact Computation (IPEC)*, volume 63 of *LIPIcs*, pages 11:1–11:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[47] Venkat Chandrasekaran, Sujay Sanghavi, Pablo A. Parrilo, and Alan S. Willsky. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization*, 21(2):572–596, 2011.

[48] Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *J. Computer and System Sciences*, 65(1):129–149, 2002.

[49] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 642–651. ACM/SIAM, 2001.

[50] Danny Z. Chen, Jian Li, Hongyu Liang, and Haitao Wang. Matroid and knapsack center problems. *Algorithmica*, 75(1):27–52, 2016.

[51] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *J. Computer and System Sciences*, 72(8):1346–1367, 2006.

[52] Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 826–835. SIAM, 2008.

[53] Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

[54] Xingyu Chen, Brandon Fain, Liang Lyu, and Kamesh Munagala. Proportionally fair clustering. In *International Conference on Machine Learning*, pages 1032–1041, 2019.

[55] Yudong Chen, Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust matrix completion and corrupted columns. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 873–880, 2011.

[56] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. In *Advances in Neural Information Processing Systems*, pages 5029–5037, 2017.

[57] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

[58] Julia Chuzhoy and Yuval Rabani. Approximating k-median with non-uniform capacities. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 952–958. SIAM, 2005.

[59] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90. ACM, 2013.

[60] Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the 47tg annual ACM symposium on Theory of Computing (STOC)*, pages 163–172. ACM, 2015.

[61] Michael B. Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, page 1758–1777, USA, 2017. SIAM.

[62] Vincent Cohen-Addad. Approximation schemes for capacitated clustering in doubling metrics. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2241–2259. SIAM, 2020.

[63] Vincent Cohen-Addad, Arnaud de Mesmay, Eva Rotenberg, and Alan Roytman. The bane of low-dimensionality clustering. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 441–456. SIAM, 2018.

[64] Vincent Cohen-Addad, Anupam Gupta, Amit Kumar, Euiwoong Lee, and Jason Li. Tight FPT approximations for k-median and k-means. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 42:1–42:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[65] Vincent Cohen-Addad and Karthik C. S. Inapproximability of clustering in lp metrics. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 519–539. IEEE Computer Society, 2019.

[66] Vincent Cohen-Addad and Jason Li. On the fixed-parameter tractability of capacitated clustering. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 41:1–41:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[67] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, page 151–158, New York, NY, USA, 1971. Association for Computing Machinery.

[68] Christophe Crespelle, Pål Grønås Drange, Fedor V. Fomin, and Petr A. Golovach. A survey of parameterized algorithms and the complexity of edge modification, 2020.

[69] Cynthia S. Crowson, Elizabeth J. Atkinson, and Terry M. Therneau. Assessing calibration of prognostic risk scores. *Statistical methods in medical research*, 25(4):1692–1706, 2016.

[70] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[71] Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Lower Bounds for Approximation Schemes for Closest String. In *Proceedings of the 15th Scandinavian Symposium and Workshops on Algorithm*

Theory (SWAT), volume 53 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:10, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[72] Chen Dan, Kristoffer Arnsfelt Hansen, He Jiang, Liwei Wang, and Yuchen Zhou. On low rank approximation of binary matrices. *CoRR*, abs/1511.01699, 2015.

[73] Amit Datta, Michael Carl Tschantz, and Anupam Datta. Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. *Proceedings on privacy enhancing technologies*, 2015(1):92–112, 2015.

[74] H. Gökalp Demirci and Shi Li. Constant approximation for capacitated k-median with (1+epsilon)-capacity violation. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 73:1–73:14, 2016.

[75] Hu Ding and Jinhui Xu. A unified framework for clustering constrained data without locality property. *Algorithmica*, 82(4):808–852, 2020.

[76] Irit Dinur. Mildly exponential reduction from gap 3sat to polynomial-gap label-cover. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 23, 2016.

[77] Jerry Dischler. Putting machine learning into the hands of every advertiser. *Google Blog*, 10:2018, 2018.

[78] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[79] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[80] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[81] Eduard Eiben, Robert Ganian, Iyad Kanj, Sebastian Ordyniak, and Stefan Szeider. On clustering incomplete data. *CoRR*, abs/1911.01465, 2019.

[82] Uriel Feige. NP-hardness of hypercube 2-segmentation. *CoRR*, abs/1411.0821, 2014.

[83] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578, 2011.

[84] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1434–1453. SIAM, 2013.

[85] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2015.

[86] Qilong Feng, Jiaxin Hu, Neng Huang, and Jianxin Wang. Improved PTAS for the constrained k-means problem. *Journal of Combinatorial Optimization*, 37(4):1091–1110, 2019.

[87] Qilong Feng, Zhen Zhang, Ziyun Huang, Jinhui Xu, and Jianxin Wang. Improved algorithms for clustering with outliers. In *30th International Symposium on Algorithms and Computation (ISAAC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[88] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Approximation schemes for low-rank binary matrix approximation problems. *ACM Trans. Algorithms*, 16(1):12:1–12:39, 2020.

[89] Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, and Saket Saurabh. Covering vectors by spaces: Regular matroids. *SIAM J. Discret. Math.*, 32(4):2512–2565, 2018.

[90] Fedor V. Fomin, Petr A. Golovach, and Fahad Panolan. Parameterized low-rank binary matrix approximation. *Data Min. Knowl. Discov.*, 34(2):478–532, 2020.

[91] Fedor V. Fomin, Daniel Lokshtanov, Syed Mohammad Meesum, Saket Saurabh, and Meirav Zehavi. Matrix rigidity from the viewpoint of parameterized complexity. *SIAM J. Discrete Math.*, 32(2):966–985, 2018.

[92] Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 209–217, 2005.

[93] András Frank and Éva Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, March 1987.

[94] Zachary Friggstad, Kamyar Khodamoradi, Mohsen Rezapour, and Mohammad R. Salavatipour. Approximation schemes for clustering with outliers. *ACM Trans. Algorithms*, 15(2), February 2019.

[95] Yinghua Fu, Nianping Jiang, and Hong Sun. Binary matrix factorization and consensus algorithms. In *Proceedings of the International Conference on Electrical and Control Engineering (ICECE)*, pages 4563–4567. IEEE, 2010.

[96] Jie Gao, Michael Langberg, and Leonard J. Schulman. Analysis of incomplete data and an intrinsic-dimension Helly theorem. *Discret. Comput. Geom.*, 40(4):537–560, 2008.

[97] Jie Gao, Michael Langberg, and Leonard J. Schulman. Clustering lines in high-dimensional space: Classification of incomplete data. *ACM Trans. Algorithms*, 7(1):8:1–8:26, 2010.

[98] Howard N. Garb. Race bias, social class bias, and gender bias in clinical judgment. *Clinical Psychology: Science and Practice*, 4(2):99–120, 1997.

[99] Leszek Gasieniec, Jesper Jansson, and Andrzej Lingas. Approximation algorithms for hamming clustering problems. *J. Discrete Algorithms*, 2(2):289–301, 2004.

[100] Nicolas Gillis and Stephen A. Vavasis. On the complexity of robust PCA and ℓ_1-norm low-rank matrix approximation. *CoRR*, abs/1509.09236, 2015.

[101] Elena L. Glassman, Rishabh Singh, and Robert C. Miller. Feature engineering for clustering student solutions. In *Proceedings of the first ACM conference on Learning@ scale conference*, pages 171–172, 2014.

[102] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[103] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *ACM Journal of Experimental Algorithmics*, 13, 2008.

[104] David A. Gregory, Norman J. Pullman, Kathryn F. Jones, and J. Richard Lundgren. Biclique coverings of regular bigraphs and minimum semiring ranks of regular matrices. *J. Combinatorial Theory Ser. B*, 51(1):73–89, 1991.

[105] D. Grigoriev. Using the notions of separability and independence for proving the lower bounds on the circuit complexity. *Journal of Soviet Math.*, 14(5):1450–1456, 1980.

[106] Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228 – 248, 1999.

[107] Harold W. Gutch, Peter Gruber, Arie Yeredor, and Fabian J. Theis. ICA over finite fields - separability and algorithms. *Signal Processing*, 92(8):1796–1808, 2012.

[108] David Lee Hanson and Farroll Tim Wright. A bound on tail probabilities for quadratic forms in independent random variables. *The Annals of Mathematical Statistics*, 42(3):1079–1083, 1971.

[109] Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

[110] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 291–300. ACM, 2004.

[111] Moritz Hardt and Ankur Moitra. Algorithms and hardness for robust subspace recovery. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT)*, volume 30 of *JMLR Proceedings*, pages 354–375. JMLR.org, 2013.

[112] David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and computation*, 100(1):78–150, 1992.

[113] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[114] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[115] Lingxiao Huang, Shaofeng Jiang, and Nisheeth Vishnoi. Coresets for clustering with fairness constraints. In *Advances in Neural Information Processing Systems*, pages 7589–7600, 2019.

[116] Lingxiao Huang and Nisheeth K. Vishnoi. Coresets for clustering in euclidean spaces: Importance sampling is nearly optimal. *arXiv preprint arXiv:2004.06263*, 2020.

[117] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity. *J. Computer and System Sciences*, 63(4):512–530, 2001.

[118] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In *Proceedings of the 10th annual symposium on Computational Geometry*, pages 332–339. ACM, 1994.

[119] Piotr Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 428–434, 1999.

[120] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 604–613. ACM, 1998.

[121] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[122] Peng Jiang and Michael T. Heath. Mining discrete patterns via binary matrix factorization. In *ICDM Workshops*, pages 1129–1136. IEEE Computer Society, 2013.

[123] Peng Jiang, Jiming Peng, Michael Heath, and Rui Yang. *A Clustering Approach to Constrained Binary Matrix Factorization*, pages 281–303. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[124] Sheng-Yi Jiang, Qi Zheng, and Qian-Sheng Zhang. Clustering-based feature selection. *Acta Electronica Sinica*, 36(12):157–160, 2008.

[125] Yishan Jiao, Jingyi Xu, and Ming Li. On the k-closest substring and k-consensus pattern problems. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern (CPM)*, volume 3109 of *Lecture Notes in Comput. Sci.*, pages 130–144. Springer, 2004.

[126] Daniel M. Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *J. ACM*, 61(1):1–23, 2014.

[127] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, August 1987.

[128] Amir E. Khandani, Adlar J. Kim, and Andrew W. Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010.

[129] Samir Khuller, Robert Pless, and Yoram J. Sussmann. Fault tolerant k-center problems. *Theoretical Computer Science*, 242(1-2):237–245, 2000.

[130] YongSeog Kim, W. Nick Street, and Filippo Menczer. Evolutionary model selection in unsupervised learning. *Intell. Data Anal.*, 6(6):531–556, 2002.

[131] Tamás Király, Lap Chi Lau, and Mohit Singh. Degree bounded matroids and submodular flows. *Combinatorica*, 32(6):703–720, 2012.

[132] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[133] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[134] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. Fair k-center clustering for data summarization. In *36th International Conference on Machine Learning, ICML 2019*, pages 5984–6003. International Machine Learning Society (IMLS), 2019.

[135] Mehmet Koyutürk and Ananth Grama. Proximus: A framework for analyzing very high dimensional discrete-attributed datasets. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 147–156, New York, NY, USA, 2003. ACM.

[136] Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. Constant approximation for k-median and k-means with outliers via iterative rounding. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 646–659. ACM, 2018.

[137] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5:1–5:32, 2010.

[138] Ravi Kumar, Rina Panigrahy, Ali Rahimi, and David P. Woodruff. Faster algorithms for binary matrix factorization. In *Proceedings of the 36th International Conference on Machine Learning, (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 3551–3559. PMLR, 2019.

[139] Michael Langberg and Leonard J Schulman. Universal ε-approximators for integrals. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 598–607. SIAM, 2010.

[140] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. *ProPublica (2016)*, 9(1), 2016.

[141] Euiwoong Lee and Leonard J. Schulman. Clustering affine subspaces: Hardness and algorithms. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 810–827. SIAM, 2013.

[142] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, November 1983.

[143] Jian Li, Ke Yi, and Qin Zhang. Clustering with diversity. In *International Colloquium on Automata, Languages, and Programming*, pages 188–200. Springer, 2010.

[144] Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, 2002.

[145] Shi Li. On uniform capacitated k-median beyond the natural LP relaxation. *ACM Trans. Algorithms*, 13(2):22:1–22:18, 2017.

[146] Tao Li. A general model for clustering binary data. In Robert Grossman, Roberto J. Bayardo, and Kristin P. Bennett, editors, *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 188–197. ACM, 2005.

[147] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982.

[148] Satyanarayana V. Lokam. Complexity lower bounds using linear algebra. *Found. Trends Theor. Comput. Sci.*, 4:1–155, January 2009.

[149] Haibing Lu, Jaideep Vaidya, Vijayalakshmi Atluri, and Yuan Hong. Constraint-aware role mining via extended boolean matrix decomposition. *IEEE Trans. Dependable Sec. Comput.*, 9(5):655–669, 2012.

[150] Bin Ma and Xiaoming Sun. More efficient algorithms for closest string and substring problems. *SIAM J. Computing*, 39(4):1432–1443, 2009.

[151] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is NP-hard. In *Proceedings of the 3rd International Workshop on Algorithms and Computation (WALCOM)*, Lecture Notes in Comput. Sci., pages 274–285. Springer, 2009.

[152] Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[153] Konstantin Makarychev, Yury Makarychev, and Ilya Razenshteyn. Performance of johnson-lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 1027–1038, New York, NY, USA, 2019. Association for Computing Machinery.

[154] Rashmi Malhotra and Davinder K Malhotra. Evaluating consumer loans using neural networks. *Omega*, 31(2):83–96, 2003.

[155] Yair Marom and Dan Feldman. k-means clustering of lines for big data. In *Proceedings of the 33rd Annual Conference on Advances in Neural Information Processing Systems (NeurIPS)*, pages 12797–12806, 2019.

[156] Dániel Marx. Closest substring problems with small distances. *SIAM J. Comput.*, 38(4):1382–1410, 2008.

[157] N. Megiddo and K. Supowit. On the complexity of some common geometric location problems. *SIAM J. Computing*, 13(1):182–196, 1984.

[158] Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE Trans. Knowl. Data Eng.*, 20(10):1348–1362, 2008.

[159] Pauli Miettinen and Jilles Vreeken. Model order selection for boolean matrix factorization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 51–59. ACM, 2011.

[160] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford Ser. (2)*, 11:50–59, 1960.

[161] Barsha Mitra, Shamik Sural, Jaideep Vaidya, and Vijayalakshmi Atluri. A survey of role mining. *ACM Comput. Surv.*, 48(4):50:1–50:37, February 2016.

[162] Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 182–191. IEEE, 1995.

[163] Björn Ommer and Jitendra Malik. Multi-scale object detection by clustering lines. In *Proceedings of the 12th IEEE International Conference on Computer Vision (ICCV)*, pages 484–491. IEEE, 2009.

[164] James Orlin. Contentment in graph theory: covering graphs with cliques. *Nederl. Akad. Wetensch. Proc. Ser. A* **80**=*Indag. Math.*, 39(5):406–424, 1977.

[165] Rafail Ostrovsky and Yuval Rabani. Polynomial-time approximation schemes for geometric min-sum median clustering. *J. ACM*, 49(2):139–156, 2002.

[166] Amichai Painsky, Saharon Rosset, and Meir Feder. Generalized independent component analysis over finite alphabets. *IEEE Trans. Information Theory*, 62(2):1038–1053, 2016.

[167] Karl Pearson. LIII. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[168] Tomer Perets. *Clustering of lines*. Open University of Israel Ra'anana, Israel, 2011.

[169] Claudia Perlich, Brian Dalessandro, Troy Raeder, Ori Stitelman, and Foster Provost. Machine learning for targeted display advertising: Transfer learning in action. *Machine learning*, 95(1):103–127, 2014.

[170] A. A. Razborov. On rigid matrices. *Manuscript in russian*, 1989.

[171] Clemens Rösner and Melanie Schmidt. Privacy preserving clustering with constraints. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[172] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 143–152, 2006.

[173] Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Fair coresets and streaming algorithms for fair k-means. In *International Workshop on Approximation and Online Algorithms*, pages 232–251. Springer, 2019.

[174] Bao-Hong Shen, Shuiwang Ji, and Jieping Ye. Mining discrete patterns via binary matrix factorization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 757–766, New York, NY, USA, 2009. ACM.

[175] Jonathon Shlens. A tutorial on principal component analysis. *CoRR*, abs/1404.1100, 2014.

[176] Kirill Simonov, Fedor Fomin, Petr Golovach, and Fahad Panolan. Refined complexity of PCA with outliers. In *In Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 5818–5826, 2019.

[177] Christian Sohler and David P. Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 802–813. IEEE, 2018.

[178] Zhao Song, David P. Woodruff, and Peilin Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 688–701. ACM, 2017.

[179] Zoya Svitkina. Lower-bounded facility location. *ACM Transactions on Algorithms (TALG)*, 6(4):1–16, 2010.

[180] Jaideep Vaidya. Boolean matrix decomposition problem: Theory, variations and applications to data engineering. In *Proceedings of the 28th IEEE International Conference on Data Engineering (ICDE)*, pages 1222–1224. IEEE Computer Society, 2012.

[181] Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM Symposium on Access Control Models and (SACMAT)*, pages 175–184, 2007.

[182] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Mathematical Foundations of Computer Science (MFCS)*, volume 53 of *Lecture Notes in Comput. Sci.*, pages 162–176. Springer, 1977.

[183] Namrata Vaswani and Praneeth Narayanamurthy. Static and dynamic robust PCA and matrix completion: A review. *Proceedings of the IEEE*, 106(8):1359–1379, 2018.

[184] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[185] René Vidal, Yi Ma, and S. Shankar Sastry. *Generalized Principal Component Analysis*, volume 40 of *Interdisciplinary applied mathematics*. Springer, 2016.

[186] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 4th edition, 2015.

[187] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, USA, 1st edition, 2011.

[188] David P. Woodruff. *Sketching as a Tool for Numerical Linear Algebra*, volume 10 of *Foundations and Trends in Theoretical Computer Science*. Now Publishers Inc., 2014.

[189] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[190] John Wright, Arvind Ganesh, Shankar R. Rao, YiGang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Proceedings of 23rd Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2080–2088. Curran Associates, Inc., 2009.

[191] Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust PCA via outlier pursuit. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2496–2504. Curran Associates, Inc., 2010.

[192] Arie Yeredor. Independent component analysis over Galois fields of prime order. *IEEE Trans. Information Theory*, 57(8):5342–5359, 2011.

[193] Zhongyuan Zhang, Tao Li, Chris H. Q. Ding, and Xiang-Sun Zhang. Binary matrix factorization with applications. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 391–400. IEEE Computer Society, 2007.

Graphic design: Communication Division, UiB / Print: Skipnes Kommunikasjon AS

uib.no

ISBN: 9788230866511 (print)
9788230858189 (PDF)