CORESETS FOR CLUSTERING: FOUNDATIONS AND CHALLENGES

by
Xuan Wu

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
February, 2022

Abstract

Clustering is a fundamental task in machine learning and data analysis. The main challenge for clustering in big data sets is that classical clustering algorithms often do not scale well. Coresets are data reduction techniques that turn big data into a tiny proxy. Prior research has shown that coresets can provide a scalable solution to clustering problems and imply streaming and distributed algorithms. In this work, we aim to solve a fundamental question and two modern challenges in coresets for clustering.

Beyond Euclidean Space: Coresets for Clustering in Euclidean space have been well studied and coresets of constant size are known to exist. While very few results are known beyond Euclidean space. It becomes a fundamental problem that what kind of metric space admits constant-sized coresets for clustering. We focus on graph metrics which is a common ambient space for clustering. We provide positive results that assert constant-sized coresets exist in various families of graph metrics including graphs of bounded treewidth, planar graphs and the more general excluded-minor graphs.

Missing Value: Missing value is a common phenomenon in real data sets. Clustering under the existence of missing values is a very challenging task. In this work, we construct the first coresets for clustering with multiple missing values. Previously, such coresets were only known to exist when each data point has at most one missing value [1]. We further design a near-linear time algorithm to construct our coresets. This algorithm implies the first near-linear time approximation scheme for k-Means

clustering with missing values and improves a recent result by [2].

Simultaneous Coresets: Most classical coresets are limited to a specific clustering objective. When there are multiple potential objectives, a stronger notion of "simultaneous coresets" is needed. Simultaneous coresets provide the approximations for a family of objectives and can serve as a more flexible data reduction tool. In this work, we design the first simultaneous coresets for a large clustering family which includes both k-Median and k-Center.

Thesis Readers

Dr. Vladimir Braverman (Advisor)
Associate Professor
Department of Computer Science
Johns Hopkins University

Dr. Michael Dinitz
Associate Professor
Department of Computer Science
Johns Hopkins University

Dr. Xin Li
Associate Professor
Department of Computer Science
Johns Hopkins University

Dedicated to my dear parents.

I can not reach here,

without their unconditional love and support

Acknowledgements

First, I would like to thank my advisor, Vladimir Braverman. Vova has given me full academic freedom and always appreciates my results in research. It was an honor to work with him.

I thank Shaofeng Jiang and Robert Krauthgamer who were working closely with me in several projects. Their expertise is essential for me to find exciting results and write my papers well.

Last but not least, I want to thank Jian Li, who brought me into the world of theoretical computer science.

Contents

Abstract ii

Dedication iv

Acknowledgements v

Contents vi

List of Figures ix

Chapter 1 Introduction 1

1.1 Overview of the Thesis 2

1.1.1 Coresets for Clustering in Graph Metrics 3

1.1.2 Coresets for Ordered Weighted Clustering 4

1.1.3 Coresets for Clustering with Missing Values 4

1.1.4 The Lower Bound 5

1.2 Preliminaries 5

Chapter 2 Coresets for Clustering in Bounded Treewidth Graph . . 6

2.1 Introduction 6

2.1.1 Our Results 7

2.1.2 Technical Contributions 8

2.1.3 Additional Related Work 10

2.2 Preliminaries 10

2.3 Coresets for k-Median in Graph Metrics . . . 11
2.3.1 Bounding the Shattering Dimension . . . 15
2.3.2 Proof of the Structural Lemma . . . 18
2.3.3 Complexity of Min-linear Functions . . . 23

Chapter 3 Coresets for Clustering in Excluded-Minor Graphs . . . 25

3.1 Introduction . . . 25
3.1.1 Our Results . . . 25
3.1.2 Technical Contributions . . . 26
3.1.3 Additional Related Work . . . 30
3.2 Preliminaries . . . 31
3.3 Framework . . . 34
3.3.1 Iterative Size Reduction . . . 35
3.3.2 Importance Sampling . . . 37
3.3.3 Coresets via Terminal Embedding . . . 39
3.4 Coresets . . . 45
3.4.1 Excluded-minor Graphs . . . 45
3.4.2 Proof of Lemma 3.4.1 . . . 49
3.4.2.1 From Planar to Minor-excluded Graphs . . . 61
3.4.3 High-Dimensional Euclidean Spaces . . . 67

Chapter 4 Coresets for Ordered Weighted Clustering . . . 69

4.1 Introduction . . . 69
4.1.1 Our Contribution . . . 71
4.1.2 Overview of Techniques . . . 72
4.1.3 Additional Related Work . . . 73
4.2 Preliminaries . . . 73

4.3 The Basic Case: p-Centrum for $k = d = 1$ (one facility in one-dimensional data) . . . 74
4.3.1 Proofs of Technical Lemmas . . . 80
4.4 Simultaneous Coreset for Ordered k-Median . . . 82
4.4.1 Coreset for p-Centrum on Lines in $\mathbb{R}^d$ . . . 83
4.4.2 Coreset for p-Centrum in $\mathbb{R}^d$ . . . 87
4.4.3 Simultaneous Coreset for Ordered k-Median in $\mathbb{R}^d$ . . . 90

Chapter 5 Coreset for Clustering with Missing Values . . . 93
5.1 Our Results . . . 94
5.1.1 Technical Overview . . . 96
5.1.2 Additional Related Work . . . 97
5.2 Preliminaries . . . 98
5.3 Coresets . . . 98
5.3.1 Proof of Lemma 5.3.3: Shattering Dimension of $\mathbb{R}^d_?$ . . . 101
5.3.2 Proof of Lemma 5.3.4: Estimating Sensitivity Efficiently . . . 102
5.3.3 Proof of Lemma 5.3.6: Dynamic $O(1)$-Coresets for k-Center Clustering . . . 105

Chapter 6 The Lower Bound . . . 114
6.1 Coresets for Clustering in Graphs of Bounded Treewidth . . . 114
6.2 Coresets for Ordered Weighted Clustering . . . 118
6.3 Coresets for Clustering with Missing Values . . . 121
6.4 Coresets for Clustering in Euclidean Space . . . 122

Conclusions . . . 126

References . . . 127

List of Figures

Figure 4-1 Coreset construction for p-Centrum with $k = 1$ facilities in dimension $d = 1$. The left figure depicts the partition of the data into $X = (L \cup R) \cup Q$, where $P = L \cup R$ contains the p furthest points from an optimal center y^*. The right figure shows the different manners of splitting L and Q into intervals. 76

Chapter 1

Introduction

(k, z)-Clustering In this thesis, our main focus is the *metric* (k, z)-Clustering problem and its variants. The input of *metric* (k, z)-Clustering is a metric space $M = (V, d)$ and an n-point data set $X \subseteq V$, and the goal is to find a set $C \subseteq V$ of k points, called *center set*, that minimizes the objective function

$$\mathrm{cost}(X, C) := \sum_{x \in X} d^z(x, C),$$

where $d(x, C) := \min\{d(x, c) : c \in C\}$. The (k, z)-Clustering generalizes the well-known k-Median and k-Means clustering problems. Such clustering problems are essential tools in data analysis and are used in many application domains, such as genetics, information retrieval, and pattern recognition. However, finding an optimal clustering is a nontrivial task, and even in settings where polynomial-time algorithms are known, it is often challenging in practice because data sets are huge, and potentially distributed or arriving over time. To this end, a powerful data-reduction technique, called *coresets*, is of key importance.

Roughly speaking, a coreset is a compact summary of the data points by weighted points, that approximates the clustering objective for every possible choice of the center set. Formally, an ϵ-coreset for k-Median is a subset $D \subseteq V$ with weight $w : D \to \mathbb{R}_+$, such that for every k-subset $C \subseteq V$,

$$\sum_{x \in D} w(x) \cdot d(x, C) \in (1 \pm \epsilon) \cdot \mathrm{cost}(X, C).$$

This notion, sometimes called a strong coreset, was proposed in [3], following a weaker notion of [4]. Small-size coresets often translate to faster algorithms, more efficient storage/communication of data, and streaming/distributed algorithms via the merge-and-reduce framework, see e.g. [3, 5–8] and recent surveys [9–11].

Coresets for k-Median were studied extensively in Euclidean spaces, i.e., when $V = \mathbb{R}^{\mathsf{d}}$ and $d(x, y) = \|x - y\|_2$. The size of the first ϵ-coreset for k-Median, when they were first proposed [3], was $O(k(\frac{1}{\epsilon})^{\mathsf{d}} \cdot \log n)$, and it was improved to $O(k(\frac{1}{\epsilon})^{\mathsf{d}})$, which is independent of n, in [12]. Feldman and Langberg [13] drastically improved the dependence on the dimension d, from exponential to linear, achieving an ϵ-coreset of size $O(\frac{k}{\epsilon^2} \cdot \mathsf{d})$, and this bound was generalized to doubling metrics [7]. Recently, coresets for Euclidean (k, z)-Clustering of size *independent* of d and polynomial in $\frac{k}{\epsilon}$ were devised by [14, 15]. However, beyond the traditional Euclidean setting, there were not too many known results prior to this work.

1.1 Overview of the Thesis

We divide this thesis into 3 parts: (1) In the first part, we consider a fundamental question in the area (Challenge 1.1.1) that aims to characterize the conditions of the ambient metric space such that clustering in this metric space admits constant size coresets. We present our work to this question in Chapter 2 and Chapter 3. (2) In the second part, we consider non-traditional settings that include multiple objectives and missing values in clustering. In Chapter 4, we consider the setting that there exists multiple clustering objectives (e.g. we need to care about both k-Median and k-Center) and we show how to design simultaneous coresets for those objectives. In Chapter 5, we design the first coresets for clustering when there are multiple missing coordinates. (3) In Chapter 6, the third and last part, we present lower bounds of coresets for problems we consider in the first and second parts.

The main chapters are all based on Xuan Wu's publications. Chapter 2 is based on the theoretical part of [16]. Chapter 3 is based on [17]. Chapter 4 is based on [18]. Chapter 5 is based on [19]. The lower bound part, Chapter 6 is based on various lower bounds results in the above papers.

1.1.1 Coresets for Clustering in Graph Metrics

Many modern coreset constructions stem from a fundamental framework proposed by Feldman and Langberg [13], extending the importance sampling approach of Langberg and Schulman [20]. In this framework [13], the size of an ϵ-coreset for k-MEDIAN is bounded by $O(\text{poly}(k/\epsilon) \cdot \text{sdim})$, where sdim is the shattering (or VC) dimension of the family of distance functions. For a general metric space (V, d), a direct application of [13] results in a coreset of size $O_{k,\epsilon}(\log |V|)$, which is tight in the sense that in some instances, every coreset must have size $\Omega(\log |V|)$ (see Chapter 6). Therefore, to obtain coresets of size independent of the data set size $|X|$, we have to restrict our attention to specific metric spaces, which raises the following fundamental question.

Challenge 1.1.1. *Identify conditions on a data set X from metric space (V, d) that guarantee the existence (and efficient construction) of an ϵ-coreset for k-MEDIAN of size $O_{\epsilon,k}(1)$?*

This question has seen major advances recently. Coresets of size independent of X (and V) were obtained, including efficient algorithms, for several important special cases: high-dimensional Euclidean spaces [14, 21] (i.e., independently of the Euclidean dimension), and metrics with bounded doubling dimension [7].

We make significant progress to the above challenge, by designing new coresets for k-MEDIAN in three very different types of metric spaces. (i) In Chapter 2, we give the first $O_{\epsilon,k}(1)$-size coreset for bounded treewidth graphs; (ii) In Chapter 3, we give the first $O_{\epsilon,k}(1)$-size coreset for excluded-minor; and (iii) Based on the new

techniques we deleveop in Chapter 3, we give a simplified state-of-the-art coreset for high-dimensional Euclidean spaces (i.e., coreset-size independent of the Euclidean dimension with guarantees comparable to [21] but simpler analysis).

Finally, in Chapter 6, we provide lower bounds for coresets in bounded treewidth graphs and general graphs to complete our results.

1.1.2 Coresets for Ordered Weighted Clustering

An immediate challenge for using clustering algorithms is to choose a proper objective function. Popular choices of clustering objectives include k-Median and k-Center which respectively capture the average performance and the worst-case performance of the centers. Standard coresets have limited usefulness when there are multiple potential objectives. The simultaneous coresets, initially introduced by [22], aim to have the approximation hold for all the objectives simultaneously.

Challenge 1.1.2. *Can one construct simultaneous coresets for both k-Center and k-Median?*

In Chapter 4, we answer the above question affirmatively by designing simultaneous coresets for a more general class of clustering objectives Ordered k-Median which is a natural generalization of k-Center and k-Median. Moreover, in Chapter 6, we provide a lower bound for such coresets to complete our results.

1.1.3 Coresets for Clustering with Missing Values

Missing value is a common phenomenon. Perform clustering tasks under the existence of missing value is a challenging task. Only very recently, [2] designs the first polynomial time approximation scheme (PTAS) for k-Means with missing values. However, the algorithm is complicated and its running time is quadratic.

Challenge 1.1.3. *Are there small coresets for k-MEANS and the general (k, z)-CLUSTERING with missing values? Moreover, can one construct such coresets in near-linear time and thus provide the first scalable solution for these problems?*

We provide a positive answer to the above question in Chapter 5 and prove a lower bound of such coresets in Chapter 6 to complete our results.

1.1.4 The Lower Bound

While we put most our effort in proving upper bound, namely, constructing coresets, lower bound is a topic we can never ignore.

In Chapter 6, we provide lower bounds for the above three questions to complete our corresponding upper bounds. Moreover, we show the first lower bounds of coresets for clustering in general metric space and Euclidean space.

1.2 Preliminaries

We consider a general k-clustering problem called (k, z)-clustering, which asks to minimize the following objective function. This objective function (and problem) is also called k-MEDIAN when $z = 1$ and k-MEANS when $z = 2$.

Definition 1.2.1 ((k, z)-CLUSTERING)**.** For data set $X \subset \mathbb{R}^d$ and a center set $C \subset \mathbb{R}^d$ containing k (usual) points, let

$$\mathrm{cost}_z(X, C) := \sum_{x \in X} \mathrm{dist}^z(x, C).$$

Definition 1.2.2 (ϵ-Coreset for (k, z)-CLUSTERING)**.** For data set $X \subset \mathbb{R}^d$, we say a weighted set $S \subseteq X$ with weight function $w : S \to \mathbb{R}_+$ is an ϵ-coreset for (k, z)-CLUSTERING, if

$$\forall C \subset \mathbb{R}^d, |C| = k, \qquad \sum_{x \in S} w(x) \cdot \mathrm{dist}^z(x, C) \in (1 \pm \epsilon) \cdot \mathrm{cost}_z(X, C).$$

Chapter 2

Coresets for Clustering in Bounded Treewidth Graph

2.1 Introduction

In this Chapter, we initiate the study of coresets for clustering in *graph metrics*, i.e., the shortest-path metrics of graphs. As usual in these contexts, the focus is on edge-weighted graphs $G = (V, E)$ with a restricted topology, and in our case bounded treewidth. For ease of presentation, we present our coresets result for k-Median while our result can easily be extended to the general (k, z)-Clustering.

Clustering in Graph Metrics While clustering in Euclidean spaces is very common and well studied, clustering in graph metrics is also of great importance and has many applications. For instance, clustering is widely used for community detection in social networks [23], and is an important technique for the visualization of graph data [24]. Moreover, k-clustering on graph metrics is one of the central tasks in data mining of spatial (e.g., road) networks [25, 26], and it has been applied in various data analysis methods [27, 28], and many other applications can be found in a survey [29].

Despite the importance of graph k-Median, coresets for this problem were not studied before, and to the best of our knowledge, the only known constructions applicable to graph metrics are coresets for general n-point metrics $M = (V, d)$ with

$X = V$ [13, 30], that have size $\operatorname{poly}\log n$. In contrast, as mentioned above, coresets for Euclidean spaces usually have size independent of $n = |V|$ and sometimes even independent of the dimension d. Moreover, this generic construction assumes efficient access to the distance function, which is expensive in graphs and requires to compute all-pairs shortest paths.

To fill this gap, we study coresets for k-Median on the shortest-path metric of an edge-weighted graph G. As a baseline, we confirm that the $O(\log n)$ factor in coreset size is really necessary for general graphs, which motivates us to explore whether structured graphs admit smaller coresets. We achieve this by designing coresets whose size are independent of n when G has a bounded *treewidth* (see Definition 2.2.1), which is a special yet common graph family. Moreover, our algorithm for constructing the coresets runs in *near-linear time* (for every graph regardless of treewidth).

Indeed, treewidth is a well-studied parameter that measures how close a graph is to a tree [31, 32], and intuitively it guarantees a (small) vertex separator in every subgraph. Several important graph families have bounded treewidth: trees have treewidth at most 1, series-parallel graphs have treewidth at most 2, and k-outerplanar graphs, which are an important special case of planar graphs, have treewidth $O(k)$. In practice, treewidth is a good complexity measure for many types of graph data. A recent experimental study showed that real data sets in various domains including road networks of the US power grid networks and social networks such as an ego-network of Facebook, have small to moderate treewidth [33].

2.1.1 Our Results

Our main result is a near-linear time construction of a coreset for k-Median whose size depends linearly on the treewidth of G and is completely independent of $|X|$ (the size of the data set). This significantly improves the generic $O(\frac{k}{\epsilon^2} \cdot \log n)$ size bound from [13] whenever the graph has small treewidth.

Theorem 2.1.1 (Fast Coresets for Graph k-MEDIAN; see Theorem 2.3.1)**.** *For every edge-weighted graph $G = (V, E)$, $0 < \epsilon < 1$, and integer $k \geq 1$, k-MEDIAN of every data set $X \subseteq V$ (with respect to the shortest-path metric of G) admits an ϵ-coreset of size $\tilde{O}(\frac{k^2}{\epsilon^2}) \cdot \mathrm{tw}(G)$.*[1] *Furthermore, the coreset can be computed in time $\tilde{O}(|E|)$ with high probability.*

2.1.2 Technical Contributions

Our coreset construction employs the importance sampling framework proposed by Feldman and Langberg [13], although implemented differently as explained in Remark 2.3.1. A key observation of the framework is that it suffices to give a *uniform* upper bound on the *shattering dimension* (see Definition 2.2.2), denoted $\mathrm{sdim}_v(M)$, of the metric $M = (V, d)$ weighted by any point weight $v : V \to \mathbb{R}_+$. Our main technical contribution is a (uniform) shattering-dimension bound that is *linear* in the treewidth, and this implies the size bound of our coreset.

Theorem 2.1.2 (Shattering Dimension Bound; see Theorem 2.3.5)**.** *For every edge-weighted graph $G = (V, E)$ and every point weight function $v : V \to \mathbb{R}_+$, the shortest-path metric M of G satisfies $\mathrm{sdim}_v(M) \leq O(\mathrm{tw}(G))$.*

The shattering dimension of many important spaces was studied, including for Euclidean spaces [13] and for doubling spaces [7]. For graphs, the shattering dimension of an K_r-minor free graph (which includes bounded-treewidth graphs) is known to be $O(r)$ [34] for *unit weight* $v \equiv 1$, see Section 2.2 for details. However, a general point weight $v : V \to \mathbb{R}_+$ introduces a significant technical challenge which is illustrated below.

In our context, the shattering dimension is defined with respect to the set system of all *v-weighted* metric balls, where every such ball has a center $x \in V$ and a radius

[1] Throughout, we use $\tilde{O}(f)$ to denote $O(f \cdot \mathrm{polylog}(f))$.

$r \geq 0$, and is defined by

$$B_v(x, r) := \{y \in V : v(y) \cdot d(x, y) \leq r\}. \tag{2.1}$$

Roughly speaking, a bounded shattering dimension means that for every subset $H \subseteq V$, the number of ways this H is intersected by v-weighted metric balls is at most $\text{poly}(|H|)$. The main technical difficulty is that an arbitrary weight v can completely break the "continuity" of the space, which can be illustrated even in one-dimensional line $V = \mathbb{R}$ (and analogously in a simple path graph on $V = \{0, 1, \ldots, n\}$), where under unit weight $v \equiv 1$, every ball is a *contiguous* interval, but under a general weight v an arbitrary subset of points could form a ball; indeed, for a center $x = 0$ and radius $r = 1$, every point $y \geq 1$ can be made inside or outside of the ball $B_v(x, r)$ by setting $v(y) = \frac{1}{2y}$ or $v(y) = \frac{2}{y}$.

Our main technical contribution is to analyze the shattering dimension with general weight functions, which we outline now briefly (see Section 2.3 for a more formal overview). We start by showing a slightly modified balanced-separator theorem for bounded-treewidth graphs (Lemma 2.3.7), through which the problem of bounding the shattering dimension is reduced to bounding the "complexity" of shortest paths that cross one of a few vertex separators, each of size $O(\text{tw}(G))$. An important observation is that, if $S \subset V$ is a vertex separator and $x, y \in V$ belong to different components after removing S, then every path connecting x to y must *cross* S, and hence

$$d(x, y) = \min\{d(x, s_i) + d(s_i, y) : s_i \in S\}.$$

If we fix $x \in V$ and consider all $y \in V$, then we can think of each $d(s_i, y)$ as a real variable $z_i \in \mathbb{R}$, so instead of varying over all $y \in V$, which depends on the graph structure, we can vary over $|S|$ real variables, and each $d(x, \cdot)$ is the minimum of $|S|$ linear (actually affine) functions, or in short a min-linear function. Finally, we consider different $x \in V$ with the same separator S, and hence the same $|S|$ real variables, and we bound the "complexity" of these min-linear functions by relating it to the

arrangement number of hyperplanes, which is a well-studied concept in computational geometry. We believe our techniques may be useful for more general graph families, such as minor-free graphs.

2.1.3 Additional Related Work

Approximation algorithms have been extensively studied for k-Median in graph metrics, and here we only mention a small selection of results. In general graphs (which is equivalent to general metrics), it is NP-hard to approximate k-Median within $1 + \frac{2}{e}$ factor [35], and the state-of-art is a 2.675-approximation [36]. For planar graphs and more generally graphs excluding a fixed minor, a PTAS for k-Median was obtained in [37] based on local search, and it has been improved to be FPT (i.e. the running time is of the form $f(k, \epsilon) \cdot n^{O(1)}$) recently [38]. For general graphs, [39] proposed an $O(1)$-approximation that runs in near-linear time.

2.2 Preliminaries

Definition 2.2.1 (Tree Decomposition and Treewidth)**.** A tree decomposition of a graph $G = (V, E)$ is a tree $\mathcal{T}$ with node set $\mathcal{V}$, such that each node in $\mathcal{V}$, called a *bag*, is a subset of V, and the following conditions hold:

1. $\bigcup_{S \in \mathcal{V}} S = V$.
2. $\forall u \in V$, the nodes of $\mathcal{T}$ that contain u form a connected component in $\mathcal{T}$.
3. $\forall (u, w) \in E$, $\exists S \in \mathcal{V}$, such that $\{u, w\} \subseteq S$.

The treewidth of a graph G, denoted $\mathrm{tw}(G)$, is the smallest integer t, such that there exists a tree decomposition with maximum bag size $t + 1$.

A *nice* tree decomposition is a tree decomposition such that each bag has a degree

at most 3.[2] It is well known that there exists a nice tree decomposition of G with maximum bag size $O(\mathrm{tw}(G))$ [32].

Shattering Dimension As mentioned in Section 2.1, our coreset construction employs the Feldman-Langberg framework [13]. A key notion in the Feldman-Langberg framework is the *shattering dimension* of a metric space with respect to a point weight function.

Definition 2.2.2 (Shattering Dimension)**.** Given a point weight function $v : V \to \mathbb{R}_+$, the shattering dimension of $M = (V, d)$ with respect to v, denoted as $\mathrm{sdim}_v(M)$, is the smallest integer t, such that for every $H \subseteq V$ with $|H| \geq 2$, it holds that

$$|\{H \cap B_v(x, r) : x \in V, r \geq 0\}| \leq |H|^t.$$

Observe that the left-hand side counts the number of ways that H is intersected by all weighted balls, which were defined in (2.1). We remark that our notion shattering dimension is tightly related to the well-known *VC-dimension* (see for example [40]). In particular, let $\mathcal{B}_v := \{B_v(x, r) : x \in V, r \geq 0\}$ be the collection of all v-weighted balls, then the VC-dimension of the set system $(V, \mathcal{B}_v)$ is within a logarithmic factor to the $\mathrm{sdim}_v(M)$. It was shown in [34] that the VC-dimension of a K_r-minor free graph with unit weights $v \equiv 1$ is at most $O(r)$, which immediately implies an $O(r)$ bound also for the shattering dimension (under unit weight $v \equiv 1$).

2.3 Coresets for k-Median in Graph Metrics

In this section, we present a near-linear time construction for ϵ-coreset for k-Median in graph metrics, whose size is linear in the treewidth. This is formally stated in the following theorem.

[2]Usually, nice tree decompositions are defined to have additional guarantees, but we only need the bounded degree.

Theorem 2.3.1 (Coreset for Graph k-MEDIAN)**.** *For every edge-weighted graph $G = (V, E)$, $0 < \epsilon, \delta < 1$, and integer $k \geq 1$, k-MEDIAN of every data set $X \subseteq V$ (with respect to the shortest path metric of G) admits an ϵ-coreset of size $\tilde{O}\left(\frac{k^2}{\epsilon^2} \cdot (\mathrm{tw}(G) + \log(1/\delta))\right)$. Furthermore, it can be computed in time $\tilde{O}(|E|)$ with success probability $1 - \delta$.*

Our construction is based on the Feldman-Langberg framework [13], in which the coreset is constructed using *importance sampling*. While this framework is quite general, their implementation is tailored to Euclidean spaces and is less suitable for graphs metrics. In addition, their algorithm runs in $\tilde{O}(kn)$ time assuming access to pairwise distances, which is efficient in Euclidean spaces but rather expensive in graphs.

We give an efficient implementation of the Feldman-Langberg framework in graphs, and also provide an alternative analysis that is not Euclidean-specific. A similar strategy was previously employed for constructing coresets in doubling spaces [7], but that implementation is not applicable here because of the same efficiency issue (i.e., it requires oracle access to distances). We present our implementation and analysis of the framework below, and then put it all together to prove Theorem 2.3.1.

Importance Sampling Recall that at a high level, the importance sampling method consists of two steps.

1. For each data point $x \in X$, compute an importance $\sigma_x \in \mathbb{R}_+$.
2. Form a coreset by drawing N (to be determined later) independent samples from X, where each sample picks every $x \in X$ with probability proportional to σ_x, i.e., $p_x := \frac{\sigma_x}{\sum_{x' \in X} \sigma_{x'}}$, and assigns it weight $\frac{1}{p_x}$.

To implement the algorithm, we need to define σ_x and N. Following the Feldman-

Langberg framework, each importance σ_x is an *upper bound* on the *sensitivity*

$$\sigma_x^\star := \max_{C \subseteq V, |C|=k} \frac{d(x,C)}{\mathrm{cost}(X,C)},$$

which was introduced in [20] and represents the maximum possible contribution of x to the objective over all center sets C.

Let the *total importance* be $\sigma_X := \sum_{x \in X} \sigma_x$. Our key tool is the following bound on coreset size N in terms of σ_X and a uniform upper bound on $\mathrm{sdim}_v(M)$.

Lemma 2.3.2 ([41])**.** *Define the maximum shattering dimension as* $\mathrm{sdim}_{\max} := \max_{v:V\to\mathbb{R}_+} \mathrm{sdim}_v(M)$*. Then for*

$$N = \tilde{O}\Big(\frac{\sigma_X}{\epsilon^2}\Big(k \cdot \mathrm{sdim}_{\max} + \log\frac{1}{\delta}\Big)\Big),$$

the importance sampling procedure returns an ϵ-coreset with probability at least $1-\delta$.

Computing σ_x An efficient algorithm to compute σ_x was presented in [42], assuming that an $O(1)$-approximation to k-Median is given. Furthermore, an $O(k)$ bound on the total importance σ_X was shown.

Lemma 2.3.3 ([42])**.** *Suppose $C^\star$ is a ρ-approximate solution to the k-Median instance. Let $\sigma_x := \rho \cdot \Big(\frac{d(x,C^\star)}{\mathrm{cost}(X,C^\star)} + \frac{1}{|C^\star(x)|}\Big)$, where $C^\star(x) \subseteq X$ is the cluster of $C^\star$ that contains x. Then $\sigma_X = O(\rho k)$ and*

$$\forall x \in X, \quad \sigma_x \geq \Omega(\sigma_x^\star).$$

Thus, to construct the coreset in near-linear time, we need to compute an $O(1)$-approximation $C^\star$ fast, for which we use the following result of [39].

Lemma 2.3.4. *There is an algorithm that, given as input a weighted undirected graph $G=(V,E)$ and data set $X \subseteq V$, computes an $O(1)$-approximate solution for graph k-Median in time $\tilde{O}(|E|)$ with probability $1-o(1)$.*

Finally, we need a uniform shattering-dimension bound (with respect to treewidth). Such a bound, stated next, is our main technical contribution and its proof is presented in Section 2.3.1.

Theorem 2.3.5 (Shattering Dimension)**.** *For every edge-weighted graph $G = (V, E)$ and every point weight function $v : V \to \mathbb{R}_+$, the shortest-path metric M of G satisfies* $\mathrm{sdim}_v(M) \leq O(\mathrm{tw}(G))$.

Remark 2.3.1. Our implementation of the framework of [13] differs in several respects. First, their shattering dimension is defined with respect to *hyperbolic* balls instead of usual metric balls (as the underlying set system). Second, the choice of σ_x and the sampling bound are different. While they achieve an improved coreset size (linear in k), their analysis relies on Euclidean-specific properties and does not apply in graph metrics.

Putting It Together We are now in position to conclude our main result.

Proof of Theorem 2.3.1. Construct a coreset by the importance sampling procedure, where the importance σ_x is computed using Lemma 2.3.4. Then we can apply Lemma 2.3.3 with $\rho = O(1)$ to bound the total importance $\sigma_X = O(k)$. Combining this and the shattering dimension from Theorem 2.3.5, we can apply Lemma 2.3.2 with coreset size

$$N = \tilde{O}\Big(\frac{k^2}{\epsilon^2}\Big(\mathrm{tw}(G) + \log\frac{1}{\delta}\Big)\Big).$$

The running time is dominated by computing the importance σ_x for all $x \in X$, which we claim can be computed in time $\tilde{O}(|E|)$ by using Lemmas 2.3.3 and 2.3.4. Indeed, first compute $C^\star$ in time $\tilde{O}(|E|)$ using Lemma 2.3.4, then compute the clustering of X with respect to $C^\star$ and the associated distances $\{d(x, C^\star) : x \in X\}$ using a *single* Dijkstra execution in time $\tilde{O}(|E|)$ time (see Observation 1 of [39]). Finally, use this information to compute σ_x for all $x \in X$, and sample according to it, in total time $\tilde{O}(|X|)$. □

2.3.1 Bounding the Shattering Dimension

We give a technical overview before presenting the detailed proof of Theorem 2.3.5. Recall that the unit-weight case of shattering dimension was already proved in [34], and our focus is when $v : V \to \mathbb{R}_+$ is a general weight function.

The proof starts with a slightly modified balanced-separator theorem for bounded treewidth graphs (Lemma 2.3.7), through which the problem of bounding the shattering dimension is reduced to bounding the complexity of *bag-crossing* shortest paths for every bag.

A well-known fact is that every bag $\{s_i, \ldots, s_m\} \subseteq V$ in the tree decomposition is a vertex cut of size $m = O(\mathrm{tw}(G))$, and this leads to an important observation: if x and y belong to different components after removing this bag, then every path connecting x with y *crosses* the bag, and hence

$$d(x, y) = \min\{d(x, s_i) + d(s_i, y) : i \in [m]\}.$$

Now suppose we fix $x \in V$ and let y vary over V; then we can write $d(x, \cdot)$ as a min-linear function (which means the minimum of m linear functions) $f_x : \mathbb{R}^m \to \mathbb{R}_+$, whose variables are $z_i = d(s_i, y)$ for $i \in [m]$; notice that the terms $d(x, s_i)$ are constant with respect to y.

This alternative view of distances enables us to bound the complexity of shortest-paths, because the functions $\{f_x\}_x$ all have common variables $\{z_i = d(s_i, y)\}_{i\in[m]}$ in real domain (instead of variables in V), and more importantly, the domain of these functions has low dimension $m = O(\mathrm{tw}(G))$. Furthermore, the min-linear description also handles weights because $v(x) \cdot f_x$ is min-linear too. Finally, in a technical lemma (Lemma 2.3.8), we relate the complexity of a collection of min-linear functions of low dimension to the *arrangement number* of hyperplanes, which is a well-studied quantity in computational geometry.

Theorem 2.3.6 (Restatement of Theorem 2.3.5)**.** *For every edge-weighted graph*

$G = (V, E)$ and every point weight function $v : V \to \mathbb{R}_+$, the shortest-path metric M of G satisfies $\mathrm{sdim}_v(M) \leq O(\mathrm{tw}(G))$.

Proof. Fix a point weight $v : V \to \mathbb{R}_+$. We bound the shattering dimension by verifying the definition (see Definition 3.2.1). Fix a subset of points $H \subseteq V$ with $|H| \geq 2$. By Definition 3.2.1, we need to show

$$|\{H \cap B_v(x, r) : x \in V, r \geq 0\}| \leq |H|^{O(\mathrm{tw}(G))}.$$

We interpret this as a counting problem, in which we count the number of distinct subsets $H \cap B_v(x, r)$ over two variables x and r. To make the counting easier, our first step is to "remove" the variable r, so that we could deal with the center x only.

Relating to Permutations For $x \in V$, let π_x be the permutation of H such that points $y \in H$ are ordered by $d(x, y) \cdot v(y)$ (in non-increasing order) and ties are broken consistently. Since $H \cap B_v(x, r)$ corresponds to a prefix of π_x, and every π_x has at most $|H|$ prefixes, we have

$$|\{H \cap B_v(x, r) : x \in V, r \geq 0\}| \leq |H| \cdot |\{\pi_x : x \in V\}|.$$

Hence it suffices to show

$$|\{\pi_x : x \in V\}| \leq |H|^{O(\mathrm{tw}(G))}, . \tag{2.2}$$

Next, we divide the graph (not necessarily a partition) into $\mathrm{poly}(|H|)$ parts using the following structural lemma of bounded treewidth graphs, so that each part is "simply structured". We prove the following lemma in Section 2.3.2.

Lemma 2.3.7 (Structural Lemma)**.** *Given graph $G(V, E)$, and $H \subseteq V$, there exists a collection $\mathcal{S} \subseteq 2^V$ of subsets of V, such that the following holds.*

1. $\bigcup_{A \in \mathcal{S}} A = V$.
2. $|\mathcal{S}| \leq \mathrm{poly}(|H|)$.

3. *For each $A \in \mathcal{S}$, either $|A| \leq O(\mathrm{tw}(G))$, or i) $|A \cap H| \leq O(\mathrm{tw}(G))$ and ii) there exists $P \subseteq V$ with $|P| \leq O(\mathrm{tw}(G))$ such that there is no edge in E between A and $V \setminus (A \cup P)$.*

Let $\mathcal{S}$ be the collection of subsets asserted by Lemma 2.3.7. Since $\bigcup_{A \in \mathcal{S}} A = V$ and $|\mathcal{S}| \leq \mathrm{poly}(|H|)$, it suffices to count the number of permutations for each part. Formally, it suffices to show that

$$\forall A \in \mathcal{S}, \qquad |\{\pi_x : x \in A\}| \leq |H|^{O(\mathrm{tw}(G))}. \tag{2.3}$$

Counting Permutations for Each $A \in \mathcal{S}$ The easy case is when $|A| \leq O(\mathrm{tw}(G))$:

$$|\{\pi_x : x \in A\}| \leq |A| \leq O(\mathrm{tw}(G)) \leq |H|^{O(\mathrm{tw}(G))}.$$

Then we focus on proving Inequality (2.3) for the other case, where i) $|A \cap H| \leq O(\mathrm{tw}(G))$ and ii) there exists $P \subseteq V$ with $|P| \leq O(\mathrm{tw}(G))$ such that there is no edge between A and $V \setminus (A \cup P)$, by item 3 of Lemma 2.3.7.

Now fix such an A. Let $H_A := H \cap A$, and let $Q := P \cup H_A$. Write $Q = \{q_1, \ldots, q_m\}$.

Since there is no edge between A and $V \setminus (A \cup P)$, for $x \in A$ and $y \in H$, we know that

$$d(x, y) = \min_{q_i \in Q} \{d(x, q_i) + d(q_i, y)\}.$$

Alternative Representation of $d(x, y)$ We write $d(x, y)$ in an alternative way. If we fix $y \in H$ and vary x, then $d(x, y)$ may be represented as a min-linear function in variables $z_i := d(x, q_i)$. Specifically, for $y \in H$, define $f_y : \mathbb{R}^m \to \mathbb{R}_+$ as

$$f_y(z_1, \ldots, z_m) := \min_{i \in [m]} \{z_i + d(y, q_i)\}.$$

Note that $d(y, q_i)$ is constant in f_y. By definition, $f_y(d(x, q_1), \ldots, d(x, q_m)) = d(x, y)$.

We also rewrite π_x under this new representation of distances. For $a \in \mathbb{R}^m$, define τ_a as a permutation of H that is ordered by $v(y) \cdot f_y(a)$, in the same rule as in π_x (i.e.

non-decreasing order and ties are broken consistently as in π_x). Then we have

$$\pi_x = \tau_{(d(x,q_1),\ldots,d(x,q_m))},$$

which implies

$$|\{\pi_x : x \in A\}| \leq |\{\tau_a : a \in \mathbb{R}^m\}|.$$

Thus, it remains to analyze $|\{\tau_a : a \in \mathbb{R}^m\}|$. We bound this quantity via the following technical lemma, which describes the complexity of a collection of min-linear functions with bounded dimension. Its proof appears in Section 2.3.3.

Lemma 2.3.8 (Complexity of Min-Linear Functions)**.** *Suppose $f_1, \ldots, f_s$ are s functions such that for every $i \in [s]$,*

- *$f_i : \mathbb{R}^l \to \mathbb{R}$, and*
- *$f_i(x) = \min_{j \in [l]}\{g_{ij}(x)\}$ where each $g_{ij} : \mathbb{R}^l \to \mathbb{R}$ is a linear function.*

For $x \in \mathbb{R}^l$, let σ_x be the permutation of $[s]$ such that $i \in [s]$ is ordered by $f_i(x)$ (in non-increasing order), and ties are broken consistently. Then $|\{\sigma_x : x \in \mathbb{R}^l\}| \leq O(sl)^{O(l)}$.

Applying Lemma 2.3.8 with $s = |H|$, $l = m$ and the collection of min-linear functions $\{v(y) \cdot f_y : y \in H\}$, we conclude that

$$|\{\tau_a : a \in \mathbb{R}^m\}| \leq |H|^{O(m)} \leq |H|^{O(\mathrm{tw}(G))}$$

where the last inequality follows from $|H_A| \leq O(\mathrm{tw}(G))$ and $|P| \leq O(\mathrm{tw}(G))$ and $m = |Q| = |H_A \cup P| \leq |H_A| + |P| \leq O(\mathrm{tw}(G))$. □

2.3.2 Proof of the Structural Lemma

Lemma 2.3.9 (Restatement of Lemma 2.3.7)**.** *Given graph $G(V, E)$, and $H \subseteq V$, there exists a collection $\mathcal{S} \subseteq 2^V$ of subsets of V, such that the following holds.*

1. $\bigcup_{A\in\mathcal{S}} A = V$.

2. $|\mathcal{S}| \leq \text{poly}(|H|)$.

3. *For each* $A \in \mathcal{S}$*, either* $|A| \leq O(\text{tw}(G))$*, or i)* $|A \cap H| \leq O(\text{tw}(G))$ *and ii) there exists* $P \subseteq V$ *with* $|P| \leq O(\text{tw}(G))$ *such that there is no edge in* E *between* A *and* $V \setminus (A \cup P)$.

Proof. Let $\mathcal{T}$ be a nice tree decomposition of $G(V, E)$ with maximum bag size $O(\text{tw}(G))$ (see Section 2.2). For a subtree $\mathcal{T}^\star$ of $\mathcal{T}$, let $V(\mathcal{T}^\star)$ be the union of points in all bags of $\mathcal{T}^\star$. For a subset of bags $\mathcal{B}$ of $\mathcal{T}$,

- Define $\mathcal{T}_{\setminus\mathcal{B}}$ as the set of subtrees of $\mathcal{T}$ resulted by removing all bags in $\mathcal{B}$ from $\mathcal{T}$;
- for $\mathcal{T}^\star \in \mathcal{T}_{\setminus\mathcal{B}}$, define $\partial_{\mathcal{B}}(\mathcal{T}^\star) \subseteq \mathcal{B}$ as the subset of bags in $\mathcal{B}$ via which $\mathcal{T}^\star$ connects to bags outside of $\mathcal{T}^\star$;
- for $\mathcal{T}^\star \in \mathcal{T}_{\setminus\mathcal{B}}$, define $V_{\setminus\mathcal{B}}(\mathcal{T}^\star) := V(\mathcal{T}^\star) \setminus \bigcup_{S\in\partial_{\mathcal{B}}(\mathcal{T}^\star)} S$.

Given a nice tree decomposition $\mathcal{T}$, we have the following theorem for constructing balanced separators, which will be useful for defining $\mathcal{S}$.

Theorem 2.3.10 (Balanced Separator of A Tree Decomposition [31])**.** *Suppose* $\mathcal{T}^\star$ *is a subtree of* $\mathcal{T}$ *and* $w : V \to \{0, 1\}$ *is a point weight function. There exists a bag* S *in* $\mathcal{T}^\star$*, such that any subtree* $\mathcal{T}' \in \mathcal{T}^\star_{\setminus S}$ *satisfies* $w(V_{\setminus S}(\mathcal{T}')) \leq \frac{2}{3} w(V(\mathcal{T}^\star))$.

The first step is to construct a subset of bags that satisfy the following nice structural properties.

Lemma 2.3.11. *There exists a subset of bags* $\mathcal{B} = \mathcal{B}_H$ *of* $\mathcal{T}$*, such that the following holds.*

1. $|\mathcal{B}| = \text{poly}(|H|)$.

2. *For every* $S \in \mathcal{B}$*,* $|S| = O(\text{tw}(G))$.

Algorithm 1 Balanced-Decomp($\mathcal{T}^\star, w$)

Require: subtree $\mathcal{T}^\star$ of $\mathcal{T}$, point weight $w : V \to \{0, 1\}$
Ensure: set of bags $\mathcal{B}$
1: **if** $w(V(\mathcal{T}^\star)) \leq O(\mathrm{tw}(G))$ **then**
2: return $\mathcal{B} \leftarrow \emptyset$
3: **else**
4: apply Theorem 2.3.10 on $(\mathcal{T}^\star, w)$, let S be the asserted bag, and write $\mathcal{T}^\star_{\setminus S} \leftarrow \{\mathcal{T}_1^\star, \ldots, \mathcal{T}_l^\star\}$
5: define point weight $w' : V \to \{0, 1\}$, such that $w'(u) = 0$ if $u \in S$ and $w'(u) = w(u)$ otherwise
6: for $i \in [l]$, let $\mathcal{B}'_i \leftarrow$ Balanced-Decomp$(\mathcal{T}_i^\star, w')$
7: return $\mathcal{B} \leftarrow (\bigcup_{i\in[l]} \mathcal{B}'_i) \cup S$
8: **end if**

3. *For every* $\mathcal{T}^\star \in \mathcal{T}_{\setminus\mathcal{B}}$, $|\partial_{\mathcal{B}}(\mathcal{T}^\star)| \leq 2$.

4. *For every* $\mathcal{T}^\star \in \mathcal{T}_{\setminus\mathcal{B}}$, $|V_{\setminus\mathcal{B}}(\mathcal{T}^\star) \cap H| = O(\mathrm{tw}(G))$.

Proof. The proof strategy is to start with a set of bags $\mathcal{B}_1$ such that items 1, 2 and 4 hold. Then for each $\mathcal{T}^\star \in \mathcal{T}_{\setminus\mathcal{B}}$, we further "divide" it by a few more bags, and the newly added bags $\mathcal{B}_2$ combined with $\mathcal{B}_1$ would satisfy all items.

To construct $\mathcal{B}_1$, we apply Theorem 2.3.10 which constructs balanced separators. The first step of our argument is no different from constructing a balanced separator decomposition, expect that we need explicitly that each separator is a bag of $\mathcal{T}$. We describe our balanced separator decomposition in Algorithm 1 which makes use of Theorem 2.3.10.

Define $w : V \to \{0, 1\}$ as $w(u) = 1$ if $u \in H$ and $w(u) = 0$ otherwise. Call Algorithm 1 with $(\mathcal{T}, w)$, and denote the resulted bags as $\mathcal{B}_1$, i.e. $\mathcal{B}_1 :=$ Balanced-Decomp$(\mathcal{T}, w)$.

Analyzing $\mathcal{B}_1$ We show $\mathcal{B}_1$ satisfies Items 1, 2 and 4.

- Item 2 is immediate since $\mathcal{B}_1$ is a set of bags, and the width of the tree decomposition is $O(\mathrm{tw}(G))$.

- Since $\mathcal{T}$ is a nice tree decomposition, each node of it has degree at most 3. so

Algorithm 2 Boundary-Reduction($\mathcal{T}, \mathcal{B}_1$)

Require: tree decomposition $\mathcal{T}$, subset of bags $\mathcal{B}_1$
Ensure: set of bags $\mathcal{B}_2$
1: initialize $\mathcal{B}_2 \leftarrow \emptyset$
2: **for** $\mathcal{T}^\star \in \mathcal{T}_{\setminus \mathcal{B}_1}$ **do**
3: **if** $|\partial_{\mathcal{B}_1}(\mathcal{T}^\star)| > 2$ **then**
4: let $\widehat{\mathcal{T}^\star}$ be the subtree of $\mathcal{T}$ formed by including bags in $\partial_{\mathcal{B}_1}(\mathcal{T}^\star)$ and their connecting edges to $\mathcal{T}^\star$
5: let $\widetilde{\mathcal{T}^\star}$ be the minimal subtree of $\widehat{\mathcal{T}^\star}$ that contains all bags in $\partial_{\mathcal{B}_1}(\mathcal{T}^\star)$
6: let $\mathcal{B}^\star$ be the set of bags in $\widetilde{\mathcal{T}^\star}$ with degree at least 3
7: update $\mathcal{B}_2 \leftarrow \mathcal{B}_2 \cup \mathcal{B}^\star$
8: **end if**
9: **end for**
10: return $\mathcal{B}_2$

each recursive invocation of Algorithm 1 creates at most 3 new subtrees, and each subtree has its weight decreased by $\frac{1}{3}$ (by Theorem 2.3.10). Moreover, the initial weight is $|H|$, and the recursive calls terminate when the weight is $O(\mathrm{tw}(G))$ (see Line 2), we conclude $|\mathcal{B}_1| = O(3^{\log_{\frac{3}{2}}(|H|)}) = O(|H|^{2.71}) = \mathrm{poly}(|H|)$, which is item 1.

- Because of the observation in the comment of Line 5, $w(V(T^\star))$ in Line 2 is exactly $V_{\setminus \mathcal{B}_1}(\mathcal{T}^\star) \cap H$, which implies item 4.

We further modify $\mathcal{B}_1$ so that item 3 is satisfied. The modification procedure is listed in Algorithm 2. Roughly speaking, we check each subtree $\mathcal{T}^\star \in \mathcal{T}_{\setminus \mathcal{B}_1}$, and if it violates item 3, we add more bags *inside* $\mathcal{T}^\star$, i.e. $\mathcal{B}^\star$ in line 6, so that $|\partial_{\mathcal{B}_1 \cup \mathcal{B}^\star}(\mathcal{T}^\star)| \leq 2$. This modification may be viewed as a refinement for the decomposition defined in Algorithm 1.

Call Algorithm 2 with $(\mathcal{T}, \mathcal{B}_1)$, and let $\mathcal{B}_2 :=$ Boundary-Reduction$(\mathcal{T}, \mathcal{B}_1)$. We formally analyze $\mathcal{B} := \mathcal{B}_1 \cup \mathcal{B}_2$ as follows.

Analyzing $\mathcal{B} := \mathcal{B}_1 \cup \mathcal{B}_2$ Item 2 follows immediately since both $\mathcal{B}_1$ and $\mathcal{B}_2$ are sets of bags. Now consider an iteration of Algorithm 2 on $\mathcal{T}^\star \in \mathcal{T}_{\setminus \mathcal{B}_1}$. By the definition of $\mathcal{B}^\star$, we know that $\widetilde{\mathcal{T}^\star}_{\setminus (\mathcal{B}_1 \cup \mathcal{B}^\star)}$ contains paths only (each having two boundary bags).

Hence each subtree $\mathcal{T}' \in \widehat{T^\star}_{\setminus(\mathcal{B}_1 \cup \mathcal{B}^\star)}$ satisfies $|\partial_{\mathcal{B}_1 \cup \mathcal{B}^\star}(\mathcal{T}')| \leq 2$. Since Algorithm 2 runs in a tree-by-tree basis, we conclude item 3.

Still consider one iteration of Algorithm 2. By using item 2 of the definition of the tree decomposition, we have that for every $\mathcal{T}' \in \widehat{\mathcal{T}^\star}_{\setminus(\mathcal{B}_1 \cup \mathcal{B}^\star)}$, $V_{\setminus(\mathcal{B}_1 \cup \mathcal{B}^\star)}(\mathcal{T}') \subseteq V_{\setminus \mathcal{B}_1}(\mathcal{T}^\star)$. Combining this with the fact that $\mathcal{B}_1$ satisfies item 4 (as shown above), we conclude that $\mathcal{B}$ also satisfies item 4. Finally, by the fact that the number of nodes of degree at least 3 is at most the number of leaves, we conclude that $|\mathcal{B}^\star| \leq |\partial_{\mathcal{B}_1}(\mathcal{T}^\star)|$. Then

$$|\mathcal{B}_2| \leq \sum_{\mathcal{T}^\star \in \mathcal{T}_{\setminus \mathcal{B}_1}} |\mathcal{B}^\star| \leq \sum_{\mathcal{T}^\star \in \mathcal{T}_{\setminus \mathcal{B}_1}} |\partial_{\mathcal{B}_1}(\mathcal{T}^\star)| \leq O(1) \cdot |\mathcal{B}_1|,$$

where the last inequality is by the degree constraint of the nice tree decomposition. Therefore, $|\mathcal{B}| \leq |\mathcal{B}_1| + |\mathcal{B}_2| \leq \text{poly}(|H|)$, which concludes item 1. This finishes the proof of Lemma 2.3.11. □

Suppose $\mathcal{B}$ is the set asserted by Lemma 2.3.11. Let $\mathcal{S} := \mathcal{B} \cup \{V_{\setminus \mathcal{B}}(\mathcal{T}^\star) : \mathcal{T}^\star \in \mathcal{T}_{\setminus \mathcal{B}}\}$. It is immediate that $\bigcup_{A \in \mathcal{S}} A = V$. By item 1 of Lemma 2.3.11 and the degree constraint of the nice tree decomposition $\mathcal{T}$, $|\mathcal{T}_{\setminus \mathcal{B}}| = \text{poly}(|H|)$. Hence, $|\mathcal{S}| \leq \text{poly}(|H|)$.

By item 2 of Lemma 2.3.11, we know for every $A \in \mathcal{B}$, $|A| \leq O(\text{tw}(G))$. Now consider $\mathcal{T}^\star \in \mathcal{T}_{\setminus \mathcal{B}}$, and let $A := V_{\setminus \mathcal{B}}(\mathcal{T}^\star) \in \mathcal{S}$. By item 4 of Lemma 2.3.11, $|A \cap H| \leq O(\text{tw}(G))$. Therefore, we only need to show there exists $P \subseteq V$ such that $|P| \leq O(\text{tw}(G))$ and there is no edge between A and $V \setminus (A \cup P)$. We have the following fact for a tree decomposition.

Fact 2.3.12 (A Bag is A Vertex Cut)**.** *Suppose $\mathcal{T}^\star$ is a subtree of the tree decomposition $\mathcal{T}$, and S is a bag in $\mathcal{T}^\star$. Then*

- *For $1 \leq i < j \leq l$, $V_{\setminus S}(\mathcal{T}_i^\star) \cap V_{\setminus S}(\mathcal{T}_j^\star) = \emptyset$.*
- *There is no edge between $V_{\setminus S}(\mathcal{T}_i^\star)$ and $V_{\setminus S}(\mathcal{T}_j^\star)$ for $1 \leq i < j \leq l$. In other words, S is a vertex cut for $V_{\setminus S}(\mathcal{T}_i^\star)$ for all $i \in [l]$.*

Define $P := \bigcup_{S \in \partial_{\mathcal{B}}(\mathcal{T}^\star)} S$. By Fact 2.3.12 and item 3 of Lemma 2.3.11, we know that $|P| \leq O(\mathrm{tw}(G))$, and there is no edge between A and $V \setminus (A \cup P)$. This finishes the proof of Lemma 2.3.7. □

2.3.3 Complexity of Min-linear Functions

Lemma 2.3.13 (Restatement of Lemma 2.3.8)**.** *Suppose we have s functions $f_1, \ldots, f_s$ such that for every $i \in [s]$,*

- *$f_i : \mathbb{R}^l \to \mathbb{R}$, and*
- *$f_i(x) = \min_{j \in [l]}\{g_{ij}(x)\}$ where $g_{ij} : \mathbb{R}^l \to \mathbb{R}$ are linear functions.*

For $x \in \mathbb{R}^l$, let σ_x be the permutation of $[s]$ such that $i \in [s]$ is ordered by $f_i(x)$ (in non-increasing order), and ties are broken consistently. Then $|\{\sigma_x : x \in \mathbb{R}^l\}| \leq O(sl)^{O(l)}$.

Proof. The proof strategy is to relate the number of permutations to the arrangement number of hyperplanes. The main tool that we use is the upper bound of the number of arrangements of hyperplanes. Specifically, as stated in Theorem 2.2 of [43], p hyperplanes of dimension d can partition $\mathbb{R}^d$ into $O(p)^d$ regions. At a high level, we start with "removing" the min in f_i's, by partitioning $\mathbb{R}^l$ into *linear* regions in which $f_i(x)$'s are simply linear functions. We bound the number of linear regions using the arrangement bound. Since $f_i(x)$'s are linear functions in each linear region, we may interpret them as l-dimensional hyperplanes. Then, we bound the number of σ_x's that are formed by s hyperplanes of dimension l using the arrangement bound again. The lemma is thus concluded by combining the two parts. We implement the two steps as follows.

We call $R \subseteq \mathbb{R}^l$ a linear region, if R is a maximal region satisfying that for all $i \in [s]$, there exists $j_i \in [l]$ such that $f_i(x) = g_{ij_i}(x)$ holds for all $x \in R$. Observe that for each $i \in [s]$ and $j \in [l]$, the set of $x \in \mathbb{R}^l$ such that $f_i(x) = g_{ij}(x)$ may be

represented by the intersection of at most l halfspaces of dimension l. (For example, when $j = 1$, the set is determined by $g_{i1}(x) \leq g_{i2}(x)$ and $g_{i1}(x) \leq g_{i3}(x)$ and ... and $g_{i1}(x) \leq g_{il}(x)$.) Hence, the boundaries of linear regions must be formed by those intersections. Therefore, the number of linear regions is upper bounded by $O(sl)^l$ using the arrangement number bound.

Suppose $R \subseteq \mathbb{R}^l$ is a linear region. Then for any $i \in [s]$, $f_i(x)$ $(x \in R)$ may be interpreted as a l-dimensional hyperplane $\mathcal{P}_i$. Hence, any maximal subset $S \subseteq R$ such that $\forall x, y \in S$, $\sigma_x = \sigma_y$, is a (convex) region whose boundaries are formed by the intersection of (any two of) the hyperplanes $\mathcal{P}_i$'s (noting that the intersection is of dimension at most l). We call such S's invariant regions. Apply the arrangement number bound again, we can upper bound the number of invariant regions in a linear region by $O(s)^{O(l)}$.

Note that invariant regions subdivide linear regions and each invariant region introduces exactly one permutation σ_x. Therefore, we can upper bound the distinct number of permutations by the total number of invariant regions, i.e., $\left|\left\{\sigma_x : x \in \mathbb{R}^l\right\}\right| \leq O(sl)^l \cdot O(s)^{O(l)} \leq O(sl)^{O(l)}$. This concludes the lemma. □

Chapter 3

Coresets for Clustering in Excluded-Minor Graphs

3.1 Introduction

In this Chapter, we continue to answer the question proposed by Challenge 1.1.1. Specifically, we construct new coresets for clustering in excluded-minor graphs and in Euclidean spaces. As in Chapter 2, we present our results for k-Median while our results can easily be extended to (k, z)-Clustering.

3.1.1 Our Results

Our coreset constructions are based on the well-known importance sampling framework of [13], but with subtle deviations that introduce significant advantages. Our first technical idea is to relax the goal of computing the final coreset in one shot: we present a general reduction that turns an algorithm that computes a coreset of size $O(\text{poly}(k/\epsilon) \log \|X\|_0)$ into an algorithm that computes a coreset of size $O(\text{poly}(k/\epsilon))$. The reduction is very simple and efficient, by straightforward iterations. Thus, it suffices to construct a coreset of size $O(\text{poly}(k/\epsilon) \log \|X\|_0)$. We construct this using the importance sampling framework [13], but applied in a subtly different way, called terminal embedding, in which distances are slightly distorted, trading accuracy for (hopefully) a small shattering dimension. It still remains to bound the shattering

dimension, but we are now much better equipped — we can distort the distances (design a new embedding or employ a known one), and we are content with dimension bound $O_{k,\epsilon}(\log \|X\|_0)$, instead of the usual $O_{k,\epsilon}(1)$.

Coresets for Excluded-minor Graphs A *minor* of graph G is a graph H obtained from G by a sequence of edge deletions, vertex deletions or edge contractions. We are interested in graphs G that exclude a fixed graph H as a minor, i.e., they do not contain H as a minor. Excluded-minor graphs have found numerous applications in theoretical computer science and beyond and they include, for example, planar graphs and bounded-treewidth graphs. Besides its practical importance, k-Median in planar graphs received significant attention in approximation algorithms research [38, 39, 44]. Our framework yields the first ϵ-coreset of size $O_{k,\epsilon}(1)$ for k-Median in excluded-minor graphs. We stress that our technical approach is significantly different from Chapter 2, in the sense that we introduce a novel iterative construction and a relaxed terminal embedding of excluded-minor graph metrics (see Section 3.1.2), and overall bypass bounding the shattering dimension by $O(1)$ (which is the technical core in Chapter 2).

Coresets for High-dimensional Euclidean Space A simple application of our new framework yields a near-linear time construction of coreset of size $\text{poly}(k/\epsilon)$ in Euclidean space, which too is independent of the dimension. Compared to the state of the art [21], our result achieves essentially the same size bound, while greatly simplifying the analysis.

3.1.2 Technical Contributions

Iterative Size Reduction This technique is based on an idea so simple that it may seem too naive: Basic coreset constructions have size $O_{k,\epsilon}(\log n)$, so why not apply it repeatedly, to obtain a coreset of size $O_{k,\epsilon}(\log\log n)$, then $O_{k,\epsilon}(\log\log\log n)$ and so on? One specific example is the size bound $O(\epsilon^{-2}k\log n)$ for a general n-point metric

space [13], where this does not work because $n = |V|$ is actually the size of the *ambient* space, irrespective of the *data* set X. Another example is the size bound $O(\epsilon^{-m} k \log n)$ for Euclidean space $\mathbb{R}^m$ [3], where this does not work because $n = \|X\|_1$ is the total weight of the data points X, which coresets do not reduce (to the contrast, they maintain it). These examples suggest that one should avoid two pitfalls: dependence on V and dependence on the total weight.

We indeed make this approach work by requiring an algorithm $\mathcal{A}$ that constructs a coreset of size $O(\log \|X\|_0)$, which is *data-dependent* (recall that $\|X\|_0$ is the number of *distinct* elements in a weighted set X). Specifically, we show in Theorem 3.3.1 that, given an algorithm $\mathcal{A}$ that constructs an ϵ'-coreset of size $O(\text{poly}(k/\epsilon') \log \|X\|_0)$ for every ϵ' and $X \subseteq V$, one can obtain an ϵ-coreset of size $\text{poly}(k/\epsilon)$ by simply applying $\mathcal{A}$ iteratively. It follows by setting ϵ' carefully, so that it increases quickly and eventually $\epsilon' = O(\epsilon)$. See Section 3.3.1 for details.

Not surprisingly, the general idea of applying the sketching/coreset algorithm iteratively was also used in other related contexts (e.g. [45–47]). Moreover, a related two-step iterative construction was applied in a recent coreset result [21]. Nevertheless, the exact implementation of iterative size reduction in coresets is unique in the literature. As can be seen from our results, this reduction fundamentally helps to achieve new or simplified coresets of size *independent* of data set. We expect the iterative size reduction to be of independent interest to future research.

Terminal Embeddings To employ the iterative size reduction, we need to construct coresets of size $\text{poly}(k/\epsilon) \cdot \log \|X\|_0$. Unfortunately, a direct application of [13] yields a bound that depends on the number of vertices $|V|$, irrespective of X. To bypass this limitation, the framework of [13] is augmented (in fact, we use a refined framework proposed in [41]), to support controlled modifications to the distances $d(\cdot, \cdot)$. As explained more formally in Section 3.3.2, one represents these modifications using a set of functions $\mathcal{F} = \{f_x : V \to \mathbb{R}_+ \mid x \in X\}$, that corresponds to the modified

distances from each x, i.e., $f_x(\cdot) \leftrightarrow d(x, \cdot)$. Many previous papers [13, 20, 41, 48] work directly with the distances and use the function set $\mathcal{F} = \{f_x(\cdot) = d(x, \cdot) \mid x \in X\}$, or a more sophisticated but still direct variant of hyperbolic balls (where each f_x is an affine transformation of $d(x, \cdot)$). A key difference is that we use a "proxy" function set $\mathcal{F}$, where each $f_x(\cdot) \approx d(x, \cdot)$. This introduces a tradeoff between the approximation error (called distortion) and the shattering dimension of $\mathcal{F}$ (which controls the number of samples), and overall results in a smaller coreset. Such tradeoff was first used in [7] to obtain small coresets for doubling spaces, and was recently used in [21] to reduce the coreset size for Euclidean spaces. This proxy function set may be alternatively viewed as a *terminal embedding* on X, in which both the distortion of distances (between X and all of V) and the shattering dimension are controlled.

Such terminal embedding (Section 3.3.3) maintains $(1+\epsilon)$-multiplicative distortion of the distances. When this embedding achieves dimension bound $O(\text{poly}(k/\epsilon) \log \|X\|_0)$, we combine it with the aforementioned iterative size reduction, to further reduce the size to be independent of X. It remains to actually design embeddings of this type, which we achieve (as explained further below), for excluded-minor graphs and for Euclidean spaces, and thus we overall obtain $O_{\epsilon,k}(1)$-size coresets in both settings.

Terminal Embedding for Euclidean Spaces Our terminal embedding for Euclidean spaces is surprisingly simple, and is a great showcase for our new framework. In a classical result [13], it has been shown that $\text{sdim}_{\max}(\mathcal{F}) = O(m)$ for Euclidean distance in $\mathbb{R}^m$ without distortion. On the other hand, we notice a terminal embedding version of Johnson-Lindenstrauss Lemma was discovered recently [49]. Our terminal embedding bound (Section 3.4.16) follows by directly combining these two results, see Section 3.4.3 for details.

We note that without our iterative size reduction technique, plugging in the recent terminal Johnson-Lindenstrauss Lemma [49] into classical importance sampling frameworks, such as [13, 41] does not yield any interesting coreset. Furthermore, the

new terminal Johnson-Lindenstrauss Lemma was recently used in [21] to design coresets for high-dimensional Euclidean spaces. Their size bounds are essentially the same as ours, however they go through a complicated analysis to directly show a shattering dimension bound $\text{poly}(k/\epsilon)$. This complication is not necessary in our method, because by our iterative size reduction it suffices to show a very loose $O_{k,\epsilon}(\log \|X\|_0)$ dimension bound, and this follows immediately from the Johnson-Lindenstrauss result.

Terminal Embedding for Excluded-minor Graphs The technical core of the terminal embedding for excluded-minor graphs is how to bound the shattering dimension. In our proof, we reduce the problem of bounding the shattering dimension into finding a representation of the distance functions on $X \times V$ as a set of *min-linear* functions. Specifically, we need to find for each x a min-linear function $g_x : \mathbb{R}^s \to \mathbb{R}$ of the form $g_x(t) = \min_{1 \le i \le s}\{a_i t_i + b_i\}$, where $s = O(\log \|X\|_0)$, such that $\forall c \in V$, there is $t \in \mathbb{R}^s$ with $d(x, c) = g_x(t)$.

The central challenge is how to relate the graph structure to the structure of shortest paths $d(x, c)$. To demonstrate how we relate them, we start with discussing the simple special case of bounded treewidth graphs. For bounded treewidth graphs, the vertex separator theorem is applied to find a subset $P \subseteq V$, through which the shortest path $x \rightsquigarrow y$ has to pass. This translates into the following

$$d(x, c) = \min_{p \in P}\{d(x, p) + d(p, c)\},$$

and for each $x \in X$, we can use this to define the desired min-linear function $g_x(d(p_1, c), \ldots, d(p_m, c)) = d(x, c)$, where we write $P = \{p_1, \ldots, p_m\}$.

However, excluded-minor graphs do not have small vertex separator, and we use the shortest-path separator [50, 51] instead. Now assume for simplicity that the shortest paths $x \rightsquigarrow c$ all pass through a fixed shortest path l. Because l itself is a shortest path, we know

$$\forall x \in X, c \in V, \quad d(x, c) = \min_{u_1, u_2 \in l}\{d(x, u_1) + d(u_1, u_2) + d(u_2, c)\}.$$

Since l can have many (i.e. $\omega(\log \|X\|_0)$) points, we need to discretize l by designating $\text{poly}(\epsilon^{-1})$ *portals* P_x^l on l for each $x \in X$ (and similarly P_c^l for $c \in V$). This only introduces $(1+\epsilon)$ distortion to the distance, which we can afford.

Then we create $d'_x : l \to \mathbb{R}_+$ to approximate $d(x,u)$'s, using distances from x to the portals P_x^l (and similarly for $d(c,u)$). Specifically, for the sake of presentation, assume $P_x^l = \{p_1, p_2, p_3\}$ $(p_1 \le p_2 \le p_3)$, interpret l as interval $[0,1)$, then for $u \in [0, p_1)$, define $d'_x(u) = d(x,0)$, for $u \in [p_1, p_2)$, define $d'_x(u) = d(x,p_1)$, and so forth. Hence, each $d'_x(\cdot)$ is a piece-wise linear function of $O(|P_x^l|)$ pieces (again, similarly for $d'_c(\cdot)$), and this enables us to write

$$d(x,c) \approx d'(x,c) := \min_{u_1,u_2 \in P_x^l \cup P_c^l} \{d'_x(u_1) + d(u_1,u_2) + d'_c(u_2)\}.$$

Therefore, it suffices to find a min-linear representation for $d'(x,\cdot)$ for $x \in X$. However, the piece-wise linear structure of d'_x creates extra difficulty to define min-linear representations. To see this, still assume $P_x^l = \{p_1, p_2, p_3\}$. Then to determine $d'_x(u)$ for $u \in P_x^l \cup P_c^l$, we not only need to know $d(x,p_i)$ for $p_i \in P_x^l$, but also need to know which sub-interval $[p_i, p_{i+1})$ that u belongs to. (That is, if $u \in [p_1, p_2)$, then $d'_x(u) = d(x,p_1)$.) Hence, in addition to using distances $\{c\} \times P_c^l$ as variables of g_x, the relative ordering between points in $P_x^l \cup P_c^l$ is also necessary to evaluate $d'(x,c)$.

Because $c \in V$ can be arbitrary, we cannot simply "remember" the ordering in g_x. Hence, we "guess" this ordering, and for each fixed ordering we can write g_x as a min-linear function of few variables. Luckily, we can afford the "guess" since $|P_x^l \cup P_c^l| = \text{poly}(\epsilon^{-1})$ which is independent of X. A more detailed overview can be found in Section 3.4.1.

3.1.3 Additional Related Work

Tightly related to coresets and terminal embedding, dimensionality reduction has also been studied for clustering in Euclidean spaces. Compared with coresets which reduce

the data set size while keeping the dimension, dimensionality reduction aims to find a low-dimensional representation of data points (but not necessarily reduce the number of data points). As a staring point, a trivial application of Johnson-Lindenstrauss Lemma [52] yields a dimension bound $O(\epsilon^{-2}\log n)$ for (k,z)-Clustering. For k-Means with $1+\epsilon$ approximation ratio, [53] showed an $O(k/\epsilon^2)$ dimension bound for data-oblivious dimension reduction and an $O(k/\epsilon)$ bound for the data-dependent setting. Moreover, the same work [53] also obtained a data-oblivious $O(\epsilon^{-2}\log k)$ dimension bound for k-Means with approximation ratio $9+\epsilon$. Very recently, [54] obtained an $\tilde{O}(\epsilon^{-6}(\log k+\log\log n))$ dimension bound for k-Means and [55] obtained an $O(\epsilon^{-2}\log\frac{k}{\epsilon})$ bound for (k,z)-Clustering. Both of them used data-oblivious methods and have approximation ratio $1+\epsilon$. Dimensionality reduction techniques are also used for constructing dimension-free coresets in Euclidean spaces [14, 21, 41, 54].

3.2 Preliminaries

Notations Let $V^k := \{C \subseteq V : |C| \le k\}$ denote the collection of all subsets of V of size at most k. [1] For integer $n, i > 0$, let $\log^{(i)} n$ denote the i-th iterated logarithm of n, i.e. $\log^{(1)} n := \log n$ and $\log^{(i)} n := \log(\log^{(i-1)} n)$ $(i \ge 2)$. Define $\log^\star n$ as the number of times the logarithm is iteratively applied before the result is at most 1, i.e. $\log^\star n := 0$ if $n \le 1$ and $\log^\star n = 1 + \log^\star(\log n)$ if $n > 1$. For a weighted set S, denote the weight function as $w_S : S \to \mathbb{R}_+$. Let $\mathrm{OPT}_z(X)$ be the optimal objective value for (k,z)-Clustering on X, and we call a subset $C \subseteq V$ an (α,β)-approximate solution for (k,z)-Clustering on X if $|C| = \alpha k$ and $\mathrm{cost}_z(X,C) := \sum_{x\in X} w_X(x)\cdot(d(x,C))^z \le \beta\cdot\mathrm{OPT}_z(X)$.

Functional Representation of Distances We consider sets of functions $\mathcal{F}$ from V to $\mathbb{R}_+$. Specifically, we consider function sets $\mathcal{F} = \{f_x : V \to \mathbb{R}_+ \mid x \in X\}$ that is

[1] Strictly speaking, V^k is the collection of all ordered k-tuples of V, but here we use it to denote the subsets. Note that tuples may contain repeated elements so the subsets in V^k are of size at most k.

indexed by the weighted data set $X \subseteq V$, and intuitively $f_x(\cdot)$ is used to measure the distance from $x \in X$ to a point in V. Because we interpret f_x's as distances, for a subset $C \subseteq V$, we define $f_x(C) := \min_{c \in C} f_x(C)$, and define the clustering objective accordingly as

$$\mathrm{cost}_z(\mathcal{F}, C) := \sum_{f_x \in \mathcal{F}} w_{\mathcal{F}}(f_x) \cdot (f_x(C))^z.$$

In fact, in our applications, we will use $f_x(y)$ as a "close" approximation to d. We note that this functional representation is natural for k-Clustering, since the objective function only uses distances from X to every k-subset of V only. Furthermore, we do not require the triangle inequality to hold for such functional representations.

Shattering Dimension For $c \in V, r \geq 0$, define $B_{\mathcal{F}}(c, r) := \{f \in \mathcal{F} : f(c) \leq r\}$. We emphasize that c is from the ambient space V in addition to the data set X. Intuitively, $B_{\mathcal{F}}(c, r)$ is the ball centered at c with radius r when the f functions are used to measure distances. For example, consider $X = V$ and let $f_x(\cdot) := d(x, \cdot)$ for $x \in V$. Then $B_{\mathcal{F}}(c, r) = \{f_x \in \mathcal{F} : d(c, x) \leq r\}$, which corresponds to the metric ball centered at c with radius r.

We introduce the notion of shattering dimension in Definition 3.2.1. In fact, the shattering dimension may be defined with respect to any set system [56], but we do not need this generality here and thus we consider only the shattering dimension of the "metric balls" system.

Definition 3.2.1 (Shattering Dimension [56])**.** Suppose $\mathcal{F}$ is a set of functions from V to $\mathbb{R}_+$. The shattering dimension of $\mathcal{F}$, denoted as $\mathrm{sdim}(\mathcal{F})$, is the smallest integer t, such that for every $\mathcal{H} \subseteq \mathcal{F}$ with $|\mathcal{H}| \geq 2$,

$$\forall \mathcal{H} \subseteq \mathcal{F}, |\mathcal{H}| \geq 2, \quad |\{B_{\mathcal{H}}(c, r) : c \in V, r \geq 0\}| \leq |\mathcal{H}|^t. \tag{3.1}$$

The shattering dimension is tightly related to the well-known VC-dimension [57], and they are equal to each other up to a logarithmic factor [56, Corollary 5.12,

Lemma 5.14]. In our application, we usually do not use $\mathrm{sdim}(\mathcal{F})$ directly. Instead, given a point weight $v : X \to \mathbb{R}_+$, we define $\mathcal{F}_v := \{f_x \cdot v(x) \mid x \in X\}$, and then consider the maximum of $\mathrm{sdim}(\mathcal{F}_v)$ over all possible v, defined as $\mathrm{sdim}_{\max}(\mathcal{F}) := \max_{v:X\to\mathbb{R}_+} \mathrm{sdim}(\mathcal{F}_v)$.

3.3 Framework

We present our general framework for constructing coresets. Our first new idea is a generic reduction, called iterative size reduction, through which it suffices to find a coreset of size $O(\log \|X\|_0)$ only in order to get a coreset of size independent of X. This general reduction greatly simplifies the coreset construction, and in particular, as we will see, "old" techniques such as importance sampling gains new power and becomes useful for new settings such as excluded-minor graphs.

Roughly speaking, the iterative size reduction turns a coreset construction algorithm $\mathcal{A}(X, \epsilon)$ with size $O(\text{poly}(\epsilon^{-1}k) \cdot \log \|X\|_0)$ into a construction $\mathcal{A}'(X, \epsilon)$ with size $\text{poly}(\epsilon^{-1}k)$. To define $\mathcal{A}'$, we simply iteratively apply $\mathcal{A}$, i.e. $X_i := \mathcal{A}(X_{i-1}, \epsilon_i)$, and terminate when $\|X_i\|_0$ does not decrease. However, if $\mathcal{A}$ is applied for t times in total, the error of the resulted coreset is accumulated as $\sum_{i=1}^{t} \epsilon_t$. Hence, to make the error bounded, we make sure $\epsilon_i \geq 2\epsilon_{i-1}$ and $\epsilon_t = O(\epsilon)$, so $\sum_{i=1}^{t} \epsilon_i = O(\epsilon)$. Moreover, our choice of ϵ_i also guarantees that $\|X_i\|_0$ is roughly $\text{poly}(\epsilon^{-1}k \cdot \log^{(i)} \|X\|_0)$. Since $\log^{(i)} \|X\|_0$ decreases very fast with respect to i, $\|X_i\|_0$ becomes $\text{poly}(\epsilon^{-1}k)$ in about $t = \log^{\star} \|X\|_0$ iterations. The detailed algorithm $\mathcal{A}'$ can be found in Algorithm 3, and we present the formal analysis in Theorem 3.3.1.

To construct the actual coresets which is to be used with the reduction, we adapt the importance sampling method that was proposed by Feldman and Langberg [13]. In previous works, the size of the coresets from importance sampling is related to the shattering dimension of metric balls system (i.e. in our language, it is the shattering dimension of $\mathcal{F} = \{d(x, \cdot) \mid x \in X\}$.) Instead of considering the metric balls only, we give a generalized analysis where we consider a general set of "distance functions" $\mathcal{F}$ that has some error but is still "close" to d. The advantage of doing so is that we could trade the accuracy with the shattering dimension, which in turn reduces the size of the coreset.

We particularly examine the following set of functions $\mathcal{F} = \{f_x : V \to \mathbb{R}_+ \mid x \in X\}$ where $\mathcal{F}$ introduces a multiplicative $(1+\epsilon)$ error to d, i.e. $\forall x \in X, c \in V$, $d(x,c) \leq f_x(c) \leq (1+\epsilon) \cdot d(x,c)$. Such a small distortion is already very helpful to obtain an $O(\log \|X\|_0)$ shattering dimension for minor-free graphs and Euclidean spaces.

In this section, we will discuss how $\mathcal{F}$ implies efficient coresets, and the dimension bounds will be analyzed in Section 3.4 where we also present the coreset results.

3.3.1 Iterative Size Reduction

Theorem 3.3.1 (Iterative Size Reduction)**.** *Let $\rho \geq 1$ be a constant and let $\mathcal{M}$ be a family of metric spaces. Assume $\mathcal{A}(X,k,z,\epsilon,\delta,M)$ is a randomized algorithm that constructs an ϵ-coreset of size $\epsilon^{-\rho}s(k)\log\delta^{-1}\log\|X\|_0$ for (k,z)-Clustering on every weighted set $X \subseteq V$ and metric space $M(V,d) \in \mathcal{M}$, for every $z \geq 1, 0 < \epsilon, \delta < \frac{1}{4}$, running in time $T(\|X\|_0,k,z,\epsilon,\delta,M)$ with success probability $1-\delta$. Then algorithm $\mathcal{A}'(X,k,z,\epsilon,\delta,M)$, stated in Algorithm 3, computes an ϵ-coreset of size $\tilde{O}(\epsilon^{-\rho}s(k)\log\delta^{-1})$ for (k,z)-Clustering on every weighted set $X \subseteq V$ and metric space $M(V,d) \in \mathcal{M}$, for every $z \geq 1$, $0 < \epsilon, \delta < \frac{1}{4}$, in time*

$$O(T(\|X\|_0,k,z,O(\epsilon/(\log\|X\|_0)^{\frac{1}{\rho}}),O(\delta/\|X\|_0),M)\cdot\log^\star\|X\|_0),$$

and with success probability $1-\delta$.

Proof. For the sake of presentation, let $n := \|X\|_0$, $s := s(k)$, and $\Gamma := s\epsilon^{-\rho}\log\delta^{-1}$. We start with proving in the following that X_t is an $O(\epsilon)$-coreset of X with size $\max\{160000\Gamma^4, 20\Gamma\rho^3 2^{3\rho+3}\}$ with probability $1-O(\delta)$.

Let $a_i := \|X_i\|_0$. Then by definition of X_i,

$$\begin{aligned} a_i &= s\epsilon_i^{-\rho}\log a_{i-1}\log\delta_i^{-1} \\ &= s\epsilon_i^{-\rho}\log a_{i-1}(\log a_{i-1}+\log\delta^{-1}) \\ &\leq s\epsilon_i^{-\rho}\log\delta^{-1}(\log a_{i-1})^2 \end{aligned} \tag{3.2}$$

Algorithm 3 Iterative size reduction $\mathcal{A}'(X, k, z, \epsilon, \delta, M)$

Require: algorithm $\mathcal{A}(X, k, z, \epsilon, \delta, M)$ that computes an ϵ-coreset for (k, z)-Clustering on X with size $\epsilon^{-\rho} s(k) \log \delta^{-1} \log \|X\|_0$ and success probability $1 - \delta$.

1: let $X_0 := X$, and let t be the largest integer such that $\log^{(t-1)} \|X\|_0 \geq \max\{20\epsilon^{-\rho} s(k) \log \delta^{-1}, \rho 2^{\rho+1}\}$
2: **for** $i = 1, \cdots, t$ **do**
3: let $\epsilon_i := \epsilon / (\log^{(i)} \|X\|_0)^{\frac{1}{\rho}}$, $\delta_i := \delta / \|X_{i-1}\|_0$
4: let $X_i := \mathcal{A}(X_{i-1}, k, z, \epsilon_i, \delta_i, M)$
5: **end for**
6: $X_{t+1} := \mathcal{A}(X_t, k, z, \epsilon, \delta, M)$
7: return X_{t+1}

where the inequality is by $\log a_{i-1} + \log \delta^{-1} \leq \log a_{i-1} \cdot \log \delta^{-1}$, which is equivalent to $(\log a_{i-1} - 1)(\log \delta^{-1} - 1) \geq 1$ and the latter is true because $a_{i-1} \geq \epsilon^{-\rho} \geq \epsilon^{-1} \geq 4$ and $\delta < \frac{1}{4}$.

Next we use induction to prove that $a_i \leq 20\Gamma \log \delta^{-1} (\log^{(i)} n)^3$ for all $i = 1, \ldots, t$. This is true for the base case when $i = 1$, since $a_1 \leq s\epsilon_1^{-\rho} \log \delta^{-1} (\log n)^2 \leq \Gamma (\log n)^3 \leq 20\Gamma (\log n)^3$. Then consider the inductive case $i \geq 2$ and assume the hypothesis is true for $i - 1$. We have

$$
\begin{aligned}
a_i &\leq s\epsilon_i^{-\rho} \log \delta^{-1} (\log a_{i-1})^2 && \text{by (3.2)} \\
&= \Gamma \log^{(i)} n \cdot (\log a_{i-1})^2 && \text{by definition of } \epsilon_i \\
&\leq \Gamma \log^{(i)} n \cdot (\log(20\Gamma (\log^{(i-1)} n)^3))^2 && \text{by induction hypothesis} \\
&= \Gamma \log^{(i)} n \cdot (\log(20\Gamma) + 3 \log^{(i)} n)^2 \\
&\leq \Gamma \log^{(i)} n \cdot (2(\log(20\Gamma))^2 + 18(\log^{(i)} n)^2) && \text{by } (a+b)^2 \leq 2a^2 + 2b^2 \\
&\leq 20\Gamma (\log^{(i)} n)^3,
\end{aligned}
$$

where the last inequality follows from the fact that $\log(20\Gamma) \leq \log(\log^{(i-1)} n) = \log^{(i)} n$, by $i \leq t$ and the definition of t. Hence we conclude $a_i \leq 20\Gamma (\log^{(i)} n)^3$. This in particular implies that $a_t \leq 20\Gamma (\log^{(t)} n)^3$, and by definition of t, we have

$\log^{(t)} n < \max\{20\Gamma, \rho 2^{\rho+1}\}$. Hence,

$$a_t \leq \max\{160000\Gamma^4, 20\Gamma\rho^3 2^{3\rho+3}\}.$$

By the guarantee of $\mathcal{A}$, we know that X_t is a $\Pi_{i=1}^{t}(1+\epsilon_i)$-coreset for X. Note that $a \geq 2^\rho \log a$ for every $a \geq \rho 2^{\rho+1}$, so we have $\epsilon_{i+1} \geq 2\epsilon_i$ for $i \leq t$, which implies that $\sum_{i=1}^{t} \epsilon_i \leq 2\epsilon_t$. Hence we conclude that

$$\Pi_{i=1}^{t}(1+\epsilon_i) \leq \exp\left(\sum_{i=1}^{t}\epsilon_i\right) \leq \exp(2\epsilon_t) \leq \exp\left(\frac{2\epsilon}{(\log^{(t)} n)^{\frac{1}{\rho}}}\right) \leq \exp(2\epsilon) \leq 1 + 10\epsilon,$$

where the second last inequality follows from $\log^{(t)} n = \log(\log^{(t-1)} n) \geq \log(\rho 2^{\rho+1}) \geq 1$ for $\rho \geq 1$, and the last inequality follows by the fact that $\exp(2\epsilon) \leq 1 + 10\epsilon$ for $\epsilon \in (0, 1)$. For the failure probability, we observe that $a_{i-1} \geq \epsilon_{i-1}^{-\rho} \geq \log^{(i-1)} n$, hence $\delta_i = \frac{\delta}{a_{i-1}} \leq \frac{\delta}{\log^{(i-1)} n}$, and the total failure probability is

$$\sum_{i=1}^{t} \delta_i \leq \delta\left(\frac{1}{n} + \frac{1}{\log n} + \cdots + \frac{1}{\log^{(t-1)} n}\right) \leq O(\delta),$$

where again we have used $\log^{(t-1)} n \geq \rho 2^{\rho+1} \geq 4$, by definition of t and $\rho \geq 1$.

Therefore, X_t is an $O(\epsilon)$-coreset of X with size $\max\{160000\Gamma^4, 20\Gamma\rho^3 2^{3\rho+3}\}$ with probability $1 - O(\delta)$. Finally, in the end of algorithm $\mathcal{A}'$, we apply $\mathcal{A}$ again on X_t with parameter ϵ and δ to obtain an $O(\epsilon)$-coreset of X with size

$$s\epsilon^{-\rho} \log \delta^{-1} \log(\max\{160000\Gamma^4, 20\Gamma\rho^3 2^{3\rho+3}\}) = \tilde{O}(s\epsilon^{-\rho} \log \delta^{-1})$$

with probability $1 - O(\delta)$.

To see the running time, we note that $t = O(\log^\star n)$, and we run $\mathcal{A}$ for $t+1$ times. Moreover, since $\epsilon_i \geq \epsilon_1$ and $\delta_i \geq \delta_1$, the running time of each call of $\mathcal{A}$ is at most $T(\|X\|_0, k, z, \epsilon_1, \delta_1, M)$. This completes the proof of Theorem 3.3.1. □

3.3.2 Importance Sampling

We proceed to design the algorithm $\mathcal{A}$ required by Theorem 3.3.1. It is based on the importance sampling algorithm introduced by [13, 20], and at a high level consists of two steps:

1. Computing probabilities: for each $x \in X$, compute $p_x \geq 0$ such that $\sum_{x \in X} p_x = 1$.

2. Sampling: draw N (to be determined later) independent samples from X, each drawn from the distribution $(p_x : x \in X)$, and assign each sample x a weight $\frac{w_X(x)}{p_x \cdot N}$ to form a coreset D.

The key observation in the analysis of this algorithm is that the sample size N, which is also the coreset size $\|D\|_0$, is related to the shattering dimension (see Definition 3.2.1) of a suitably defined set of functions [13, Theorem 4.1]. The analysis in [13] has been subsequently improved [41, 48], and we make use of [41, Theorem 31], restated as follows.

Lemma 3.3.2 (Analysis of Importance Sampling [41]). *Fix $z \geq 1$, $0 < \epsilon < \frac{1}{2}$, an integer $k \geq 1$ and a metric space (V, d). Let $X \subseteq V$ have weights $w_X : V \to \mathbb{R}_+$ and let $\mathcal{F} := \{f_x : V \to \mathbb{R}_+ \mid x \in X\}$ be a corresponding set of functions with weights $w_{\mathcal{F}}(f_x) = w_X(x)$. Suppose $\{\sigma_x\}_{x \in X}$ satisfies*

$$\forall x \in X, \quad \sigma_x \geq \sigma_x^{\mathcal{F}} := \max_{C \in V^k} \frac{w_X(x) \cdot (f_x(C))^z}{\mathrm{cost}_z(\mathcal{F}, C)},$$

and set a suitable

$$N = O(\epsilon^{-2} \sigma_X (k \cdot \mathrm{sdim}_{\max}(\mathcal{F}) \cdot \log(\mathrm{sdim}_{\max}(\mathcal{F})) \cdot \log \sigma_X + \log \tfrac{1}{\delta})),$$

where $\sigma_X := \sum_{x \in X} \sigma_x$ and

$$\mathrm{sdim}_{\max}(\mathcal{F}) := \max_{v : X \to \mathbb{R}_+} \mathrm{sdim}\,(\mathcal{F}_v)\,, \qquad \mathcal{F}_v := \{f_x \cdot v(x) \mid x \in X\}.$$

Then the weighted set D of size $\|D\|_0 = N$ returned by the above importance sampling algorithm satisfies, with high probability $1 - \delta$,

$$\forall C \in V^k, \quad \sum_{x \in D} w_D(x) \cdot (f_x(C))^z \in (1 \pm \epsilon) \cdot \mathrm{cost}_z(\mathcal{F}, C).$$

Remark 3.3.1. We should explain how [41, Theorem 31] implies Lemma 3.3.2. First of all, the bound in [41] is with respect to VC-dimension, and we transfer to shattering dimension by losing a logarithmic factor (see Section 5.2 for the relation between VC-dimension and shattering dimension). Another main difference is that the functions therein are actually not from V to $\mathbb{R}_+$. For $\mathcal{F} = \{f_x : V \to \mathbb{R}_+ \mid x \in X\}$, they consider $\mathcal{F}^k := \{f_x(C) = \min_{c \in C}\{f_x(c)\} \mid x \in X\}$, and their bound on the sample size is

$$N = \tilde{O}(\epsilon^{-2}\sigma_X(\mathrm{sdim}_{\max}(\mathcal{F}^k) \cdot \log \sigma_X + \log \tfrac{1}{\delta})).$$

The notion of balls and shattering dimension they use (for $\mathcal{F}^k$) is the natural extension of our Definition 3.2.1 (from functions on V to functions on V^k), where a ball around $C \in V^k$ is $B_{\mathcal{F}}(C, r) = \{f_x \in \mathcal{F} : f_x(C) \le r\}$, and (3.1) is replaced by

$$\left|\{B_{\mathcal{H}}(C, r) : C \in V^k, r \ge 0\}\right| \le |\mathcal{H}|^t.$$

Our Lemma 3.3.2 follows from [41, Theorem 31] by using the fact $\mathrm{sdim}(\mathcal{F}^k) \le k \cdot \mathrm{sdim}(\mathcal{F})$ from [13, Lemma 6.5].

Terminal Embedding. As mentioned in Section 3.1, $\mathcal{F}$ in Lemma 3.3.2 corresponds to the distance function d, i.e., $f_x(\cdot) = d(x, \cdot)$, and Lemma 3.3.2 is usually applied directly to the distances, i.e., on a function set $\mathcal{F} = \{f_x(\cdot) = d(x, \cdot) \mid x \in X\}$. In our applications, we instead use Lemma 3.3.2 with a "proxy" function set $\mathcal{F}$ that is viewed as a *terminal embedding* on X, in which both the distortion of distances (between X and all of V) and the shattering dimension are controlled.

In what follows, we discuss how terminal embedding is used to construct coresets.

3.3.3 Coresets via Terminal Embedding

Recall that our terminal embedding distorts distances between V and X multiplicatively, i.e.,

$$\forall x \in X, c \in V, \qquad d(x, c) \le f_x(c) \le (1 + \epsilon)\, d(x, c). \tag{3.3}$$

This natural guarantee works very well for (k, z)-Clustering in general. In particular, using such $\mathcal{F}$ in Lemma 3.3.2, our importance sampling algorithm will produce (with high probability) an $O(z\epsilon)$-coreset for (k, z)-Clustering.

Sensitivity Estimation. To compute a coreset using Lemma 3.3.2 we need to define, for every $x \in X$,

$$\sigma_x \geq \sigma_x^{\mathcal{F}} = \max_{C \in V^k} \frac{w_X(x) \cdot (f_x(C))^z}{\mathrm{cost}_z(\mathcal{F}, C)}.$$

The quantity $\sigma_x^{\mathcal{F}}$, usually called the *sensitivity* of point $x \in X$ with respect to $\mathcal{F}$ [13, 20]; essentially measures the maximal contribution of x to the clustering objective over all possible centers $C \subseteq V$. Since $f_x(y)$ approximates $d(x, y)$ by (3.3), it actually suffices to estimate the sensitivity with respect to d instead of $\mathcal{F}$, given by

$$\sigma_x^\star := \max_{C \in V^k} \frac{w_X(x) \cdot (d(x, C))^z}{\mathrm{cost}_z(X, C)}. \tag{3.4}$$

Even though computing $\sigma_x^\star$ exactly seems computationally difficult, we shown next (in Lemma 3.3.3) that a good estimate can be efficiently computed given an $(O(1), O(1))$-approximate clustering. A weaker version of this lemma was presented in [42] for the case where X has unit weights, and we extend it to X with general weights. We will need the following notation. Given a subset $C \subseteq V$, denote the nearest neighbor of $x \in X$, i.e., the point in C closest to x with ties broken arbitrarily, by $\mathrm{NN}_C(x) := \arg\min\{d(x, y) : y \in C\}$. The tie-breaking guarantees that every x has a unique nearest neighbor, and thus $\mathrm{NN}_C(.)$ partitions X into $|C|$ subsets. The *cluster of x under C* is then defined as $C(x) := \{x' \in X : \mathrm{NN}_C(x') = \mathrm{NN}_C(x)\}$.

Lemma 3.3.3. *Fix $z \geq 1$, an integer $k \geq 1$, and a weighted set X. Given $C^{\mathrm{apx}} \in V^k$ that is an (α, β)-approximate solution for (k, z)-Clustering on X, define for every $x \in X$,*

$$\sigma_x^{\mathrm{apx}} := w_X(x) \cdot \left(\frac{(d(x, C^{\mathrm{apx}}))^z}{\mathrm{cost}_z(X, C^{\mathrm{apx}})} + \frac{1}{w_X(C^{\mathrm{apx}}(x))} \right).$$

Then $\sigma_x^{\mathrm{apx}} \geq \Omega(\sigma_x^\star / (\beta 2^{2z}))$ for all $x \in X$, and $\sigma_X^{\mathrm{apx}} := \sum_{x \in X} \sigma_x^{\mathrm{apx}} \leq 1 + \alpha k$.

Before proving this lemma, we record the following approximate triangle inequality for distances raised to power $z \geq 1$.

Claim 3.3.4. *For all $x, x', y \in V$ we have $d^z(x,y) \leq 2^{z-1} \cdot [d^z(x,x') + d^z(x',y)]$.*

Proof of Claim 3.3.4. We first use the triangle inequality,

$$d^z(x,y) \leq [d(x,x') + d(x',y)]^z$$

and since $a \mapsto a^z$ is convex (recall $z \geq 1$), all $a, b \geq 0$ satisfy $(\frac{a+b}{2})^z \leq \frac{a^z+b^z}{2}$, hence

$$\leq 2^{z-1}[d^z(x,x') + d^z(x',y)].$$

The claim follows. □

Proof of Lemma 3.3.3. Given C^*, we shorten the notation by setting $\mu := \mathrm{NN}_{C^{\mathrm{apx}}}$, and let X^{apx} be the weighted set obtained by mapping all points of X by μ. Formally, $X^{\mathrm{apx}} := \{\mu(x) : x \in X\}$ where every $y \in X^{\mathrm{apx}}$ has weight $w_{X^{\mathrm{apx}}}(y) := \sum_{x \in X : \mu(x) = y} w_X(x)$. Then obviously

$$\forall x \in X, \quad w_X(C^{\mathrm{apx}}(x)) = \sum_{x' \in C^{\mathrm{apx}}(x)} w_X(x') = w_{X^{\mathrm{apx}}}(\mu(x)).$$

Upper bound on σ_X^{apx}. Using the above,

$$\sigma_X^{\mathrm{apx}} = \sum_{x \in X} \sigma_x^{\mathrm{apx}} = \sum_{x \in X} w_X(x) \cdot \left(\frac{d^z(x, \mu(x))}{\mathrm{cost}_z(X, C^{\mathrm{apx}})} + \frac{1}{w_{X^{\mathrm{apx}}}(\mu(x))} \right),$$

and we can bound

$$\sum_{x \in X} w_X(x) \cdot \frac{1}{w_{X^{\mathrm{apx}}}(\mu(x))} = \sum_{y \in X^{\mathrm{apx}}} w_{X^{\mathrm{apx}}}(y) \cdot \frac{1}{w_{X^{\mathrm{apx}}}(y)} \leq \|C^{\mathrm{apx}}\|_0 \leq \alpha k,$$

and we conclude that $\sigma_X^{\mathrm{apx}} \leq 1 + \alpha k$, as required.

Lower bound on σ_x^{apx} (relative to $\sigma_x^\star$). Aiming to prove this as an upper bound on $\sigma_x^\star$, consider for now a fixed $C \in V^k$. We first establish the following inequality,

that relates the cost of X^{apx} to that of X.

$$\begin{aligned}
\text{cost}_z(X^{\text{apx}}, C) &= \sum_{y \in X^{\text{apx}}} w_{X^{\text{apx}}}(y) \cdot d^z(y, C) \\
&= \sum_{x \in X} w_X(x) \cdot d^z(\mu(x), C) \\
&\leq 2^{z-1} \sum_{x \in X} w_X(x) \cdot [d^z(\mu(x), x) + d^z(x, C)] && \text{by Claim 3.3.4} \\
&= 2^{z-1} \cdot [\text{cost}_z(X, C^{\text{apx}}) + \text{cost}_z(X, C)] && \text{as } C^{\text{apx}} \text{ is } (\alpha, \beta)\text{-approximation} \\
&\leq 2^{z-1}(\beta + 1) \cdot \text{cost}_z(X, C). && (3.5)
\end{aligned}$$

Now aiming at an upper bound on $\sigma_x^\star$, observe that

$$\frac{d^z(X, C)}{\text{cost}_z(X, C)} \leq 2^{z-1} \cdot \left[\frac{d^z(x, \mu(x)) + d^z(\mu(x), C)}{\text{cost}_z(X, C)} \right] \qquad \text{by Claim 3.3.4} \qquad (3.6)$$

and let us bound each term separately. For the first term, since C^{apx} is an (α, β)-approximation,

$$\frac{d^z(x, \mu(x))}{\text{cost}_z(X, C)} \leq \beta \cdot \frac{d^z(x, \mu(x))}{\text{cost}_z(X, C^{\text{apx}})}.$$

The second term is

$$\begin{aligned}
\frac{d^z(\mu(x), C)}{\text{cost}_z(X, C)} &\leq (\beta + 1)2^{z-1} \cdot \frac{d^z(\mu(x), C)}{\text{cost}_z(X^{\text{apx}}, C)} && \text{by (3.5)} \\
&= (\beta + 1)2^{z-1} \cdot \frac{d^z(\mu(x), C)}{\sum_{y \in X^{\text{apx}}} w_{X^{\text{apx}}}(y) \cdot d^z(y, C)} \\
&\leq (\beta + 1)2^{z-1} \cdot \frac{1}{w_{X^{\text{apx}}}(\mu(x))}.
\end{aligned}$$

Plugging these two bounds into (3.6), we obtain

$$\frac{d^z(x, C)}{\text{cost}_z(X, C)} \leq (\beta + 1)2^{2z-2} \cdot \left[\frac{d^z(x, \mu(x))}{\text{cost}_z(X, C^{\text{apx}})} + \frac{1}{w_{X^{\text{apx}}}(\mu(x))} \right] = (\beta + 1)2^{2z-2} \cdot \frac{\sigma_x^{\text{apx}}}{w_X(x)}.$$

Using the definition in (3.4), we conclude that $(\beta+1)2^{2z-2} \cdot \sigma_x^{\text{apx}} \geq \sigma_x^\star$, which completes the proof of Lemma 3.3.3. □

Conclusion. Our importance sampling algorithm for this type of terminal embedding is listed in Algorithm 4. By a direct combination of Lemma 3.3.2 and Lemma 3.3.3, we conclude that the algorithm yields a coreset, which is stated formally in Lemma 3.3.5.

Algorithm 4 Coresets for (k, z)-Clustering for $\mathcal{F}$ with multiplicative distortion

1: compute an $(O(1), O(1))$-approximate solution C^{apx} for (k, z)-Clustering on X
2: for each $x \in X$, let $\sigma_x := w_X(x) \cdot \left(\frac{(d(x,C^{\text{apx}}))^z}{\text{cost}_z(X,C^{\text{apx}})} + \frac{1}{w_X(C^{\text{apx}}(x))} \right)$ ▷ as in Lemma 3.3.3
3: for each $x \in X$, let $p_x := \frac{\sigma_x}{\sum_{y \in X} \sigma_y}$
4: draw $N := O\left(\epsilon^{-2} 2^{2z} k \cdot \left(zk \log k \cdot \text{sdim}_{\max}(\mathcal{F}) + \log \frac{1}{\delta} \right)\right)$ independent samples from X, each from the distribution $(p_x : x \in X)$ ▷ $\text{sdim}_{\max}$ as in Lemma 3.3.2
5: let D be the set of samples, and assign each $x \in D$ a weight $w_D(x) := \frac{w_X(x)}{p_x N}$
6: return the weighted set D

Lemma 3.3.5. *Fix $0 < \epsilon, \delta < \frac{1}{2}$, $z \geq 1$, an integer $k \geq 1$, and a metric space $M(V, d)$. Given a weighted set $X \subseteq V$ and respective $\mathcal{F} = \{f_x : V \to \mathbb{R}_+ \mid x \in X\}$ such that*

$$\forall x \in X, c \in V, \qquad d(x, c) \leq f_x(c) \leq (1 + \epsilon) \cdot d(x, c),$$

Algorithm 4 computes a weighted set $D \subseteq X$ of size

$$\|D\|_0 = O\left(\epsilon^{-2} 2^{2z} k \left(zk \log k \cdot \text{sdim}_{\max}(\mathcal{F}) + \log \frac{1}{\delta} \right)\right),$$

that with high probability $1 - \delta$ is an ϵ-coreset for (k, z)-Clustering on X.

The running time of Algorithm 4 is dominated by the sensitivity estimation, especially line 1 which computes an $(O(1), O(1))$-approximate solution. In Lemma 3.3.6 we present efficient implementations of the algorithm, both in metric settings and in graph settings.

Lemma 3.3.6. *Algorithm 4 can be implemented in time $\tilde{O}(k\|X\|_0)$ if it is given oracle access to the distance d, and it can be implemented in time $\tilde{O}(|E|)$ if the input is an edge-weighted graph $G = (V, E)$ and M is its shortest-path metric.*

Proof. The running time is dominated by Step 1 which requires an $(O(1), O(1))$-approximation in both settings. For the metric setting where oracle access to d is given, [58] gave an $\tilde{O}(k\|X\|_0)$ algorithm for both k-Median ($z = 1$) and k-Means ($z = 2$), and it has been observed to work for general z in a recent work [21].

For the graph setting, Thorup [39, Theorem 20] gave an $(2, 12+o(1))$-approximation for graph k-MEDIAN in time $\tilde{O}(|E|)$, such that the input points are *unweighted*. Even though not stated in his result, we observe that his approach may be easily modified to handle weighted inputs as well, and we briefly mention the major changes.

- Thorup's first step [39, Algorithm D] is to compute an $(\tilde{O}(\log |V|), O(1))$-approximation F by successive uniform independent sampling. This can be naturally modified to sampling proportional to the weights of the input points.
- Then, the idea is to use the Jain-Vazirani algorithm [59] on the bipartite graph $F \times X$. To make sure the running time is $\tilde{O}(|V|)$, the edges of $F \times X$ sub-sampled by picking, for each $x \in X$, only $\tilde{O}(1)$ neighbors in F. This sampling is oblivious to weights, and hence still goes through. Let the sampled subgraph be G'.
- Finally, the Jain-Vazirani algorithm is applied on G' to obtain the final $(2, 12 + o(1))$-approximation. However, we still need to modify Jain-Vazirani to work with weighted inputs. Roughly, Jain-Vazirani algorithm is a primal-dual method, so the weights are easily incorporated to the linear program, and the primal-dual algorithm is naturally modified so that dual variables are increased at a rate that is proportional to their weight in the linear program.

After obtaining C^{apx}, the remaining steps of Algorithm 4 trivially runs in time $\tilde{O}(k\|X\|_0)$ when oracle access to d is given. However, for the graph setting, the trivial implementation of Step 2 which requires to compute $\text{cost}_1(X, C^{\text{apx}})$ needs to run $\tilde{O}(k)$ single-source-shortest-paths from points in C^{apx}, and this leads to a running time $\tilde{O}(k|V|)$. In fact, as observed in [39, Observation 1], only one single-source-shortest-path needs to be computed, by running Dijkstra's algorithm on a virtual point x_0 which connects to each point in C^{apx} to x_0 with 0 weight.

This completes the proof of Lemma 3.3.6. □

3.4 Coresets

We now apply the framework developed in Section 3.3 to design coresets of size independent of X for two different settings, including excluded-minor graphs (in Section 3.4.1), and high-dimensional Euclidean spaces (in Section 3.4.3). Our workhorse will be Lemma 3.3.5 which effectively translates a terminal embedding $\mathcal{F}$ with low distortion on $X \times V$ and low shattering dimension $\mathrm{sdim}_{\max}$ into an efficient algorithm to construct a coreset whose size is linear in $\mathrm{sdim}_{\max}(\mathcal{F})$.

We therefore turn our attention to designing such terminal embeddings. For excluded-minor graphs, we design a terminal embedding $\mathcal{F}$ with multiplicative distortion $1+\epsilon$ of the distances, and dimension $\mathrm{sdim}_{\max}(\mathcal{F}) = O(\mathrm{poly}(k/\epsilon) \cdot \log \|X\|_0)$. For Euclidean spaces, we employ a known terminal embedding with similar guarantees. In both settings, even though the shattering dimension depends on $\|X\|_0$, it still implies coresets of size independent of X by our iterative size reduction (Theorem 3.3.1). We thus obtain the first coreset (of size independent of X and V) for excluded-minor graphs (Corollary 3.4.2), and a simpler state-of-the-art coreset for Euclidean spaces (Corollary 3.4.18).

3.4.1 Excluded-minor Graphs

Our terminal embedding for excluded-minor graphs is stated in the next lemma. Previously, the shattering dimension of the shortest-path metric of graphs excluding a fixed graph H_0 as a minor was studied only for unit point weight, for which Bousquet and Thomassé [34] proved that $\mathcal{F} = \{d(x,\cdot) \mid x \in X\}$ has shattering dimension $\mathrm{sdim}(\mathcal{F}) = O(|H_0|)$. For arbitrary point weight, i.e., $\mathrm{sdim}_{\max}(\mathcal{F})$, it is still open to get a bound that depends only on $|H_0|$, although the special case of bounded treewidth graph resolved in Chapter 2. Note that both of these results use no distortion of the distances, i.e., they bound $\mathcal{F} = \{d(x,\cdot) \mid x \in X\}$. Our terminal embedding

handles the most general setting of excluded-minor graphs and arbitrary point weight, although it bypasses the open question by allowing a small distortion and dependence on X.

Lemma 3.4.1 (Terminal Embedding for Excluded-minor Graphs)**.** *For every edge-weighted graph $G = (V, E)$ that excludes some fixed minor and whose shortest-path metric is denoted as $M = (V, d)$, and for every weighted set $X \subseteq V$, there exists a set of functions $\mathcal{F} := \{f_x : V \to \mathbb{R}_+ \mid x \in X\}$ such that*

$$\forall x \in X, c \in V, \qquad d(x, c) \le f_x(c) \le (1 + \epsilon) \cdot d(x, c),$$

and $\mathrm{sdim}_{\max}(\mathcal{F}) = \tilde{O}(\epsilon^{-2}) \cdot \log \|X\|_0$.

Let us present now an overview of the proof of Lemma 3.4.1, deferring the full details to Section 3.4.2. Our starting point is the following approach developed in Chapter 2 for bounded-treewidth graphs. (The main purpose is to explain how vertex separators are used as portals to bound the shattering dimension, but unfortunately additional technical details are needed.) The first step in this approach reduces the task of bounding the shattering dimension to counting how many distinct permutations of X one can obtain by ordering the points of X according to their distance from a point c, when ranging over all $c \in V$. An additional argument uses the bounded treewidth to reduce the range of c from all of V to a subset $\hat{V} \subset V$, that is separated from X by a vertex-cut $P \subset V$ of size $|\hat{P}| = O(1)$. This means that every path, including the shortest-path, between every $x \in X$ and every $c \in \hat{V}$ must pass through $\hat{P}$, therefore

$$d(x, c) = \min\{d(x, p) + d(p, c) : p \in \hat{P}\},$$

and the possible orderings of X are completely determined by these values. The key idea now is to replace the hard-to-control range of $c \in \hat{V}$ with a richer but easier range of $|\hat{P}| = O(1)$ real variables. Indeed, each $d(x, \cdot)$ is captured by a *min-linear*

function, which means a function of the form $\min_i a_i y_i + b_i$ with real variables $\{y_i\}$ that represent $\{d(p,c)\}_{p\in\hat{P}}$ and fixed coefficients $\{a_i, b_i\}$. Therefore, each $d(x,\cdot)$ is captured by a min-linear function $g_x : \mathbb{R}^{|\hat{P}|} \to \mathbb{R}_+$, and these functions are all defined on the same $|\hat{P}| = O(1)$ real variables. In this representation, it is easy to handle the point weight $v : X \to \mathbb{R}_+$ (to scale all distances from x), because each resulting function $v(x) \cdot g_x$ is still min-linear. Finally, the number of orderings of the set $\{g_x\}_{x\in X}$ of min-linear functions, is counted using the arrangement number for hyperplanes, which is a well-studied quantity in computational geometry.

To extend this approach to excluded-minor graphs (or even planar graphs), which do not admit small vertex separators, we have to replace vertex separators with shortest-path separators [50, 51]. In particular, we use these separator theorem to partition the whole graph into a few parts, such that each part is separated from the graph by only a few shortest paths, see Lemma 3.4.4 for planar graphs (which is a variant of a result known from [60]) and Lemma 3.4.12 for excluded-minor graphs. However, the immediate obstacle is that while these separators consist of a few paths, their total size is unbounded (with respect to X), which breaks the above approach because each min-linear function has too many variables. A standard technique to address this size issue is to discretize the path separator into *portals*, and reroute through them a shortest-path from each $x \in X$ to each $c \in V$. This step distorts the distances, and to keep the distortion bounded multiplicatively by $1 + \epsilon$, one usually finds inside each separating shortest-path l, a set of portals $P_l \subset l$ whose spacing is at most $\epsilon \cdot d(x,c)$. However, $d(x,c)$ could be very small compared to the entire path l, hence we cannot control the number of portals (even for one path l).

Vertex-dependent Portals In fact, all we need is to represent the relative ordering of $\{d(x,\cdot) : x \in X\}$ using a set of *min-linear functions* over a few real variables, and these variables do not have to be the distance to *fixed portals* on the separating shortest paths. (Recall this description is eventually used by the

arrangement number of hyperplanes to count orderings of X.) To achieve this, we first define *vertex-dependent* portals P_c^l with respect to a separating shortest path l *and* a vertex $c \in V$ (notice this includes also P_x^l for $x \in X$). and then a shortest path from $x \in X$ to $c \in V$ passing through l is rerouted through portals $P_x^l \cup P_c^l$, as follows. First, since l is itself a shortest path, $d(x,c) = \min_{u_1,u_2 \in l}\{d(x,u_1) + d(u_1,u_2) + d(u_2,c)\}$. Observe that $d(u_1,u_2)$ is already linear, because one real variable can "capture" a location in l, hence we only need to approximate $d(x,u_1)$ and $d(c,u_2)$. To do so, we approximate the distances from c to every vertex on the path l, i.e., $\{d(c,u)\}_{u \in l}$, using only the distances from c to its portal set P_c^l, i.e., $\{d(c,p)\}_{p \in P_c^l}$. Moreover, between successive portals this approximate distance is a linear function, and it actually suffices to use $|P_c^l| = \text{poly}(1/\epsilon)$ portals, which means that $d(c,u)$ can be represented as a *piece-wise linear* function in $\text{poly}(1/\epsilon)$ real variables.

Note that the above approach ends up with the minimum of piece-wise linear (rather than linear) functions, which creates extra difficulty. In particular, we care about the relative ordering of $\{d(x,\cdot) : x \in X\}$ over all $c \in V$, and to evaluate $d(x,c)$ we need the pieces that c and x generate, i.e., information about $P_c^l \cup P_x^l$. Since the number of $c \in V$ is unbounded, we need to "guess" the structure of P_c^l, specifically the ordering between the portals in P_c^l and those in P_x^l. Fortunately, since every $|P_c^l| \leq \text{poly}(1/\epsilon)$, such a "guess" is still affordable, and this would prove Lemma 3.4.1.

Corollary 3.4.2 (Coresets for Excluded-Minor Graphs)**.** *For every edge-weighted graph $G = (V,E)$ that excludes a fixed minor, every $0 < \epsilon, \delta < 1/2$ and integer $k \geq 1$, k-Median of every weighted set $X \subseteq V$ (with respect to the shortest path metric of G) admits an ϵ-coreset of size $\tilde{O}(\epsilon^{-4}k^2 \log \frac{1}{\delta})$. Furthermore, such a coreset can be computed in time $\tilde{O}(|E|)$ with success probability $1 - \delta$.*

Proof. By combining Lemma 3.3.5, Lemma 3.3.6 with our terminal embedding from Lemma 3.4.1, we obtain an efficient algorithm for constructing a coreset of size

$\tilde{O}(\epsilon^{-4}k^2 \log \|X\|_0)$. This size can be reduced to the claimed size (and running time) using the iterative size reduction of Theorem 3.3.1. □

Remark 3.4.1. This result partly extends to (k, z)-Clustering for all $z \geq 1$. The importance sampling algorithm and its analysis are immediate, and in particular imply the existence of a coreset of size $\tilde{O}(\epsilon^{-4}k^2 \log \frac{1}{\delta})$. However we rely on known algorithm for $z = 1$ in the step of computing an approximate clustering (needed to compute sampling probabilities).

3.4.2 Proof of Lemma 3.4.1

For the sake of presentation, we start with proving the planar case, since this already requires most of our new technical ideas. The statement of terminal embedding for planar graphs is as follows, and how the proof can be modified to work for the minor-excluded case is discussed in Section 3.4.2.1.

Lemma 3.4.3 (Terminal Embedding for Planar Graphs)**.** *For every edge-weighted planar graph $G = (V, E)$ whose shortest path metric is denoted as $M = (V, d)$ and every weighted set $X \subseteq V$, there exists a set of functions $\mathcal{F} = \mathcal{F}_X := \{f_x : V \to \mathbb{R}_+ \mid x \in X\}$ such that for every $x \in X$, and $c \in V$, $f_x(c) \in (1 \pm \epsilon) \cdot d(x, c)$, and $\mathrm{sdim}_{\max}(\mathcal{F}) = \widetilde{O}(\epsilon^{-2}) \log \|X\|_0$.*

By definition, $\mathrm{sdim}_{\max}(\mathcal{F}) = \max_{v:X\to\mathbb{R}_+}(\mathcal{F}_v)$, so it suffices to bound $\mathrm{sdim}(\mathcal{F}_v)$ for every v. Also, by the definition of sdim, it suffices to prove for every $\mathcal{H} \subseteq \mathcal{F}_v$ with $|\mathcal{H}| \geq 2$,

$$|\{B_{\mathcal{H}}(c, r) : c \in V, r \geq 0\}| \leq \mathrm{poly}(\|X\|_0) \cdot |\mathcal{H}|^{\tilde{O}(\epsilon^{-2}) \log \|X\|_0}.$$

Hence, we fix some $v : X \to \mathbb{R}_+$ and $\mathcal{H} \subseteq \mathcal{F}_v$ with $|\mathcal{H}| \geq 2$ throughout the proof.

General Reduction: Counting Relative Orderings For $\mathcal{H} \subseteq \mathcal{F}$ and $c \in V$, let $\sigma_c^{\mathcal{H}}$ be the permutation of $\mathcal{H}$ ordered by $v(x) \cdot f_x(c)$ in non-decreasing order and ties

are broken arbitrarily. Then for a fixed $c \in V$ and very $r \geq 0$, the subset $B_{\mathcal{H}}(c, r) \subseteq \mathcal{H}$ is exactly the subset defined by some prefix of $\sigma_c^{\mathcal{H}}$. Hence,

$$|\{B_{\mathcal{H}}(c, r) : c \in V, r \geq 0\}| \leq |\mathcal{H}| \cdot \left|\{\sigma_c^{\mathcal{H}} : c \in V\}\right|.$$

Therefore, it suffices to show

$$\left|\{\sigma_c^{\mathcal{H}} : c \in V\}\right| \leq \mathrm{poly}(\|X\|_0) \cdot |\mathcal{H}|^{\tilde{O}(\epsilon^{-2}) \log \|X\|_0}.$$

Hence, this reduces the task of bounding of shattering dimension to counting the number of relative orderings of $\{v(x) \cdot f_x(c) \mid x \in X\}$.

Next, we use the following structural lemma for planar graphs to break the graph into few parts of simple structure, so we can bound the number of permutations for c coming from each part. We note that a variant of this lemma has been proved in [60], where the key idea is to use the *interdigitating* trees. For completeness, we give a full proof of this lemma in the following.

Lemma 3.4.4 (Structural Property of Planar Graphs, see also [60])**.** *For every edge-weighted planar graph $G = (V, E)$ and subset $S \subseteq V$, V can be broken into parts $\Pi := \{V_i\}_i$ with $|\Pi| = \mathrm{poly}(|S|)$ and $\bigcup_i V_i = V$, such that for every $V_i \in \Pi$,*

1. *$|S \cap V_i| = O(1)$,*
2. *there exists a collection of shortest paths $\mathcal{P}_i$ in G with $|\mathcal{P}_i| = O(1)$ and removing the vertices of all paths in $\mathcal{P}_i$ disconnects V_i from $V \setminus V_i$ (points in V_i are possibly removed).*

Furthermore, such Π and the corresponding shortest paths $\mathcal{P}_i$ for $V_i \in \Pi$ can be computed in $\tilde{O}(|V|)$ time[2]*.*

The proof of Lemma 3.4.4 is based on the following property of general trees. We note that the special case when $R = T$ was proved in [60, Lemma 3.1] and our proof is based on it. Nonetheless, we also provide the proof for completeness.

[2]This lemma is used only in the analysis in this section.

Lemma 3.4.5. *Let T be a tree of degree at most 3 and let R be a subset of nodes in T. There is a partition of the nodes of T with $\mathrm{poly}(|R|)$ parts, such that each part is a subtree of T that contains $O(1)$ nodes of R and has at most 4 boundary edges*[3] *connecting to the rest of T. Such partition can be computed in time $\tilde{O}(|T|)$, where $|T|$ is the number of nodes in T.*

Proof. We give an algorithm to recursively partition T in a top-down manner. The recursive algorithm takes a subtree T' as input, and if $|T' \cap R| \geq 4$, it chooses an edge e from T' and run recursively on the two subtrees T'_1 and T'_2 that are formed by removing e from T'. Otherwise, the algorithm simply declares the subtree T' a desired part and terminate, if $|T' \cap R| < 4$. Next, we describe how e is picked provided that $|T' \cap R| \geq 4$.

If T' has at most 3 boundary edges, we pick an edge $e \in T'$ such that each of the two subtrees T'_1, T'_2 formed by removing e satisfies $\frac{1}{3}|T' \cap R| \leq |T'_j \cap R| \leq \frac{2}{3}|T' \cap R|$, for $j = 1, 2$. By a standard application of the balanced separator theorem (see e.g. Lemma 1.3.1 of [61]), such edge always exists and can be found in time $O(|T'|)$.

Now, suppose T' has exactly 4 boundary edges. Then we choose an edge $e \in T'$, such that each of the two subtrees T'_1 and T'_2 formed by removing e has at most 3 boundary edges. Such e must exist because the maximum degree is at most 3, and such e may be found in time $O(|T'|)$ as well. To see this, suppose the four endpoints (in T') of the four boundary edges are a, b, c, d. It is possible that they are not distinct, but they can have a multiplicity of at most 2 because otherwise the degree bound 3 is violated. If any point has a multiplicity 2, say a and b, then it has to be a leaf node in T' (again, because of the degree constraint), and we can pick the unique tree edge in T' connecting a as our e. Now we assume the four points are distinct, and consider the unique paths P_1, P_2 that connect a, b and c, d respectively. If P_1 and P_2 intersect, then the intersection must contain an edge as otherwise the intersections are at nodes

[3] Here a boundary edge is an edge that has exactly one endpoint in the subtree.

only which means each of them have degree at least 4, a contradiction. Hence, we pick the intersecting edge as our e. Finally, if P_1 and P_2 are disjoint, we consider the unique path P_3 that connects a and c, and we pick edge $e := e'$ in P_3 that is outside both P_1 and P_2 to separate a and b from c and d.

We note that there are no further cases regarding the number of boundary edges of T', since in the case of 4 boundaries edges, both T'_1 and T'_2 have at most 3 boundary edges and it reduces to the first case.

It remains to analyze the size of the partition. By the property of balanced separator, we know that such recursive partition has $O(\log |R|)$ depth. Hence the total number of subtrees is $2^{O(\log |R|)} = \text{poly}(|R|)$. Finally, we note that in each level of depth, we scan the whole tree once, so the running time is upper bounded By $O(\log |R|) \cdot |T| = \tilde{O}(|T|)$. □

Proof of Lemma 3.4.4. We assume G is triangulated, since otherwise we can triangulate G and assign weight $+\infty$ to the new edges so that the shortest paths are the same as before. Let T be a shortest path tree of G from an arbitrary root vertex. Let $G^\star$ be the planar dual of G. Let $T^\star$ be the set of edges e of $G^\star$ such that the corresponding edge of e in G is not in T. Indeed, T and $T^\star$ are sometimes called *interdigitating* trees, and it is well known that $T^\star$ is a spanning tree of $G^\star$ (see e.g. [61]).

Choose $R^\star$ to be the set of faces that contain at least one point from S. We apply Lemma 3.4.5 on $R = R^\star$ and $T = T^\star$ to obtain $\Pi^\star$, the collection of resulted subtrees of $T^\star$. Then $|\Pi^\star| = \text{poly}(|S|)$, and each part $C^\star$ in $\Pi^\star$ is a subset of faces in G such that only $O(1)$ of these faces contain some point in S on their boundaries. For a part $C^\star$ in $\Pi^\star$, let $V(C^\star)$ be the set of vertices in G that are contained in the faces in $C^\star$. Recall that G is triangulated, so each face can only contain $O(1)$ vertices from S on its boundary. Therefore, for each part $C^\star$ in $\Pi^\star$, $|C^\star \cap S| = O(1)$.

Still by Lemma 3.4.5, each part $C^\star$ in $\Pi^\star$ corresponds to a subtree in $T^\star$, and

it has at most 4 boundary edges connecting to the rest of $T^\star$. By the well-known property of planar duality (see e.g. [61]), each $C^\star$ is bounded by the fundamental cycles in T of the boundary edges. We observe that the vertices of a fundamental cycle lie on 2 shortest paths in G via the least common ancestor in T (recalling that T is the shortest path tree). So by removing at most 8 shortest paths in G, $V(C^\star)$ is disconnected from $V \setminus V(C^\star)$ for every $C^\star \in \Pi^\star$.

Therefore, we can choose $\Pi := \{V(C^\star) : C^\star \in \Pi^\star\}$. For the running time, we note that both the triangulation and the algorithm in Lemma 3.4.5 run in $\tilde{O}(|V|)$ time. This completes the proof. □

Applying Lemma 3.4.4 with $S = X$ (noting that S is an unweighted set), we obtain $\Pi = \{V_i\}_i$ with $|\Pi| = \text{poly}(\|X\|_0)$, such that each part $V_i \in \Pi$ is separated by $O(1)$ shortest paths $\mathcal{P}_i$. Then

$$\left|\{\sigma_c^{\mathcal{H}} : c \in V\}\right| \leq \sum_{V_i \in \Pi} \left|\{\sigma_c^{\mathcal{H}} : c \in V_i\}\right|.$$

Hence it suffices to show for every $V_i \in \Pi$, it holds that

$$\left|\{\sigma_c^{\mathcal{H}} : c \in V_i\}\right| \leq |\mathcal{H}|^{\tilde{O}(\epsilon^{-2}) \log \|X\|_0}. \tag{3.7}$$

Since $\bigcup_i V_i = V$, it suffices to define functions $f_x(\cdot)$ for $c \in V_i$ for every i independently. Therefore, we fix $V_i \in \Pi$ throughout the proof. In the following, our proof proceeds in three parts. The first defines functions $f_x(\cdot)$ on V_i, the second analyzes the distortion of f_x's, and the final part analyzes the shattering dimension.

Part I: Definition of f_x on V_i By Lemma 3.4.4 we know $|V_i \cap X| = O(1)$. Hence, the "simple" case is when $x \in V_i \cap T$, for which we define $f_x(\cdot) := d(x, \cdot)$.

Otherwise, $x \in X \setminus V_i$. Write $\mathcal{P}_i := \{P_j\}_j$. Since P_j's are shortest paths in G, and removing $\mathcal{P}_i$ from G disconnects V_i from $V \setminus V_i$, we have the following fact.

Fact 3.4.6. *For $c \in V_i$ and $x \in X \setminus V_i$, there exists $P_j \in \mathcal{P}_i$ and $c', x' \in P_j$, such that $d(c, x) = d(c, c') + d(c', x') + d(x', x)$.*

Let $d_j(c, x)$ be the length of the shortest path from c to x that uses *at least one* point in P_j. For each $P_j \in \mathcal{P}_i$, we will define $f_x^j : V_i \to \mathbb{R}_+$, such that $f_x^j(c)$ is within $(1 \pm \epsilon) \cdot d_j(c, x)$, and let

$$f_x(c) := \min_{P_j \in \mathcal{P}_i} f_x^j(c), \qquad \forall c \in V_i.$$

Hence, by Fact 3.4.6, the guarantee that $f_x^j(c) \in (1 \pm \epsilon) \cdot d_j(c, x)$ implies $f_x(c) \in (1 \pm \epsilon) \cdot d(x, c)$, as desired. Hence we focus on defining f_x^j in the following.

Defining $f_x^j : V_i \to \mathbb{R}_+$ Suppose we fix some $P_j \in \mathcal{P}_i$, and we will define $f_x^j(c)$, for $c \in V_i$. By Fact 3.4.6 and the optimality of shortest paths, we have

$$d_j(x, c) = \min_{c', x' \in P_j} \{d(c, c') + d(c', x') + d(x', x)\}.$$

For every $y \in V$, we will define $l_y^j : P_j \to \mathbb{R}_+$ such that $l_y^j(y') \in (1 \pm \epsilon) \cdot d(y, y')$ for every $y' \in P_j$. Then, we let

$$f_x^j(c) := \min_{c', x' \in P_j} \{l_c^j(c') + d(c', x') + l_x^j(x')\},$$

and this would imply $f_x^j(c) \in (1 \pm \epsilon) \cdot d_j(x, c)$. So it remains to define $l_y^j : P_j \to \mathbb{R}_+$ for every $y \in V$.

Defining $l_y^j : P_j \to \mathbb{R}_+$ Fix $y \in V$ and we will define $l_y^j(y')$ for every $y' \in P_j$. Pick $h_y \in P_j$ that satisfies $d(y, h_y) = d(y, P_j)$. Since P_j is a shortest path, we interpret P_j as a segment in the real line. In particular, we let the two end points of P_j be 0 and 1, and P_j is a (discrete) subset of $[0, 1]$.

Define $a, b \in P_j$ such that $a \le h_y \le b$ are the two *furthest* points on the two sides of h on P_j that satisfy $d(h_y, a) \le \frac{d(y,h_y)}{\epsilon}$ and $d(h_y, b) \le \frac{d(y,h_y)}{\epsilon}$. Then construct a sequence of points $a = q_1 \le q_2 \ldots$ in the following way. For $t = 1, 2, \ldots$, if there exists $u \in (q_t, 1] \cap P_j$ such that $d(q_t, u) > \epsilon \cdot d(y, h_y)$, then let q_{t+1} be the smallest such u; if such u does not exist, then let $q_{t+1} := b$ and terminate. Essentially, this breaks P_j into segments of length $\epsilon \cdot d(y, h_y)$, except that the last one that ends with b may be shorter. Denote this sequence as $Q_y := (q_1 = a, \ldots, q_m = b)$.

Claim 3.4.7. *For every $y \in V$, $|Q_y| = O(\epsilon^{-2})$.*

Proof. By the definition of Q_y, for $1 \le t \le m-2$, $d(q_t, q_{t+1}) > \epsilon \cdot d(y, h_y)$. On the other hand, by the definition of a and b, $d(q_1, q_m) = d(a,b) \le O(\frac{d(y,h_y)}{\epsilon})$. Therefore, $|Q_y| \le O(\epsilon^{-2})$, as desired. □

Definition of f_x on V_i: Recap Define

$$l_y^j(y') := \begin{cases} d(h_y, y') & \text{if } y' < a = q_1 \text{ or } y' > b = q_m \\ d(y, q_t) & \text{if } q_t \le y' < q_{t+1}, 1 \le t < m \\ d(y, q_m) & \text{if } y' = b = q_m \end{cases} \tag{3.8}$$

where $h_y \in P_j$, $Q_y = \{q_t\}_t \subset P_j$. To recap,

- if $x \in X \cap V_i$, then $f_x(c) := d(x,c)$;
- otherwise $x \in X \setminus V_i$, $f_x(c) := \min_{P_j \in \mathcal{P}_i} f_x^j(c)$, where

$$f_x^j(c) := \min_{c', x' \in P_j} \{l_c^j(c') + d(c', x') + l_x^j(x')\}. \tag{3.9}$$

Finally,

$$f_x(c) := \min_{P_j \in \mathcal{P}_i} f_x^j(c), \qquad \forall c \in V_i. \tag{3.10}$$

Part II: Distortion Analysis The distortion of l's is analyzed in the following Lemma 3.4.8, and the distortion for f_x follows immediately from the above definitions.

Lemma 3.4.8. *For every $P_j \in \mathcal{P}_i$, $y \in V$, $y' \in P_j$, $l_y^j(y') \in (1 \pm \epsilon) \cdot d(y, y')$.*

Proof. If $y' = q_m = b$, by definition $l_y^j(y') = d(y, q_m) = d(y, y')$. Then consider the case when $y' < a = q_1$ or $y' > b = q_m$.

$$\begin{aligned} l_y^j(y') &= d(h_y, y') \\ &\in d(y', y) \pm d(y, h_y) \\ &\in d(y', y) \pm \epsilon \cdot d(y', h_y), \end{aligned}$$

where the last inequality follows from $d(y', h_y) > \frac{d(y,h_y)}{\epsilon}$. This implies $d(y, y') \in (1 \pm \epsilon) \cdot l_y^j(y')$.

Otherwise, $q_t \le y' < q_{t+1}$ for some $1 \le t < m$. By the definition of q_t's and the definition of h_y,

$$\begin{aligned} d(y, y') &\in d(y, q_t) \pm d(q_t, y') \\ &\in d(y, q_t) \pm \epsilon \cdot d(y, h_y) \\ &\in d(y, q_t) \pm \epsilon \cdot d(y, y') \\ &\in l_y^j(y') \pm \epsilon \cdot d(y, y'), \end{aligned}$$

which implies $l_y^j(y') \in (1 \pm \epsilon) \cdot d(y, y')$. This finishes the proof of Lemma 3.4.8. □

Part III: Shattering Dimension Analysis Recall that we fixed $v : X \to \mathbb{R}_+$ and $\mathcal{H} \subseteq \mathcal{F}_v$ with $|\mathcal{H}| \ge 2$. Now we show

$$\left|\{\sigma_c^{\mathcal{H}} : c \in V_i\}\right| \le |\mathcal{H}|^{\tilde{O}(\epsilon^{-2}) \log \|X\|_0}. \tag{3.11}$$

Let $H := \{x : v(x) \cdot f_x \in \mathcal{H}\}$, so $|H| = |\mathcal{H}|$. Recall that $|V_i \cap X| = O(1)$ by Lemma 3.4.4, so $|V_i \cap H| = O(1)$. Hence, if we could show

$$\left|\{\sigma_c^{\mathcal{H}} : c \in V_i\}\right| \le N(|H|)$$

for $\mathcal{H}$ such that $H \cap V_i = \emptyset$, then for general $\mathcal{H}$,

$$\left|\{\sigma_c^{\mathcal{H}} : c \in V_i\}\right| \le N(|H| - |V_i \cap H|) \cdot |H|^{O(|V_i \cap H|)} \le N(|H|) \cdot |H|^{O(1)}.$$

Therefore, it suffices to show (3.11) under the assumption that $H \cap V_i = \emptyset$.

In the following, we will further break V_i into $|H|^{\tilde{O}(\epsilon^{-2})}$ parts, such that for each part V', f_x on V' may be alternatively represented as a *min-linear* function.

Lemma 3.4.9. *Let $u = |\mathcal{P}_i|$. There exists a partition Γ of V_i, such that the following holds.*

1. $|\Gamma| \le |H|^{\tilde{O}(\epsilon^{-2})\cdot u}$.

2. $\forall V' \in \Gamma$, $\forall x \in H$, *there exists* $g_x : \mathbb{R}^s \to \mathbb{R}_+$ *where* $s = O(\epsilon^{-2})$, *such that* g_x *is a minimum of* $O(\epsilon^{-4}u)$ *linear functions on* $\mathbb{R}^s$, *and for every* $c \in V'$, *there exists* $y \in \mathbb{R}^s$ *that satisfies* $f_x(c) = g_x(y)$.

Proof. Before we actually prove the lemma, we need to examine $f_x^j(c)$ and l_y^j more closely. Suppose some $P_j \in \mathcal{P}_i$ is fixed. Recall that for $y \in V, y' \in P_j$ (defined in (3.8)),

$$l_y^j(y') := \begin{cases} d(h_y, y') & \text{if } y' < a = q_1 \text{ or } y' > b = q_m \\ d(y, q_t) & \text{if } q_t \le y' < q_{t+1}, 1 \le t < m \\ d(y, q_m) & \text{if } y' = b = q_m \end{cases}$$

where $h_y \in P_j$, $Q_y = \{q_t\}_t \subset P_j$. Hence, for every y, l_y^j is a *piece-wise linear* function with $O(|Q_y|) = O(\epsilon^{-2})$ (by Claim 3.4.7) pieces, where the transition points of l_y^j are $Q_y \cup \{0, 1\}$ (noting that $d(h_y, y')$ is linear since $h_y, y' \in P_j$).

Using that l's are piece-wise linear, we know for $c \in V_i, x \in X \setminus V_i$,

$$\begin{aligned} f_x^j(c) &= \min_{c', x' \in P_j} \{l_c^j(c') + d(c', x') + l_x^j(x')\} && \text{defined in (3.9)} \\ &= \min_{c', x' \in Q_c \cup Q_x \cup \{0,1\}} \{l_c^j(c') + d(c', x') + l_x^j(x')\}. && \text{as } l\text{'s are piece-wise linear} \end{aligned}$$

Hence, to evaluate $f_x^j(c)$ we only need to evaluate $l_c^j(c')$ and $l_x^j(x')$ at $c', x' \in Q_c \cup Q_x \cup \{0, 1\}$, and in particular we need to find the piece in l_c^j and l_x^j that every $c', x' \in Q_c \cup Q_x \cup \{0, 1\}$ belong to, and then evaluate a linear function. Precisely, the piece that every c', x' belongs to is determined by the relative ordering of points $Q_x \cup Q_c$ (recalling that they are from P_j). Thus, the pieces are not only determined by x, but also by c which is the variable, and this means without the information about the pieces, f_x cannot be represented as a min-linear function g_x. Therefore, the idea is to find a partition Γ of V_i, such that for c in each part $V' \in \Gamma$, the relative ordering of Q_c with respect to $\{Q_x : x \in H\}$ is the same. We note that we need to consider the ordering of Q_c with respect to all Q_x's, because we care about the relative orderings of all f_x's.

Defining Γ For $1 \le j \le u$, $c \in V_i$, let τ_c^j be the ordering of Q_c with respect to $\bigcup_{y \in H} Q_y$ on P_j. Here, an ordering of Q_c with respect to $\left(\bigcup_{y \in H} Q_y\right)$ is defined by their ordering on P_j which is interpreted as the real line. In our definition of Γ, we will require each part $V' \in \Gamma$ to satisfy that $\forall c \in V'$, the tuple of orderings $(\tau_c^1, \ldots, \tau_c^u)$ remains the same. That is, V_i is partitioned according to the joint relative ordering τ_c^j's on all shortest paths $P_j \in \mathcal{P}_i$.

Formally, for $1 \le j \le u$, let $\Lambda^j := \{\tau_c^j : c \in V_i\}$ be the collection of distinct ordering τ_c^j on P_j over points $c \in V_i$. Define

$$\Lambda := \Lambda^1 \times \ldots \times \Lambda^u$$

as the tuples of τ_j's for $1 \le j \le u$ (here, the $\times$ operator is the Cartesian product). For $(\tau_1, \ldots, \tau_u) \in \Lambda$, define

$$V_i^{(\tau_1,\ldots,\tau_u)} := \{c \in V_i : (\tau_c^1 = \tau_1) \wedge \ldots \wedge (\tau_c^u = \tau_u)\}$$

as the subset of V_i such that the ordering τ_c^j for each $1 \le j \le u$ agrees with the given tuple. Finally, we define the partition as

$$\Gamma := \{V_i^{(\tau_1,\ldots,\tau_u)} : (\tau_1, \ldots, \tau_u) \in \Lambda\}.$$

Bounding $|\Gamma|$ By Claim 3.4.7, we know $|Q_y| = O(\epsilon^{-2})$ for every $y \in V$. Hence, $\left|\bigcup_{y \in H} Q_y\right| = O\left(\epsilon^{-2}|H|\right)$. Therefore, for every $j \in [u]$,

$$|\Lambda^j| \le \binom{O(\epsilon^{-2}|H|)}{O(\epsilon^{-2})} = O\left(\epsilon^{-1}|H|\right)^{O(\epsilon^{-2})}.$$

Therefore,

$$|\Gamma| \le \Pi_{1 \le j \le u}|\Lambda^j| \le O\left(\epsilon^{-1}|H|\right)^{O(\epsilon^{-2}u)} \le |H|^{\tilde{O}(\epsilon^{-2})\cdot u},$$

as desired.

Defining g_x By our definition of Γ, we need to define g_x for each $V' \in \Gamma$. Now, fix tuple $(\tau_1, \ldots, \tau_u) \in \Lambda$, so the part corresponds to this tuple is $V' = V_i^{(\tau_1,\ldots,\tau_u)}$, and

we will define g_x with respect to such V'. Similar to the definition of f_x's (see (3.10)), we define $g_x : \mathbb{R}^s \to \mathbb{R}_+$ to have the form

$$g_x(y) := \min_{P_j \in \mathcal{P}_i} g_x^j(y).$$

Then, for $1 \le j \le u$, $x \in H$, define $g_x^j : \mathbb{R}^s \to \mathbb{R}$ of $s := O(\epsilon^{-2})$ variables $(q_1, \ldots, q_m, d(c, q_1), \ldots, d(c, q_m), h_c)$ for $q_i \in Q_c$, such that

$$g_x^j(q_1, \ldots, q_m, d(c, q_1), \ldots, d(c, q_m), h_c) = \min_{c', x' \in Q_c \cup Q_x \cup \{0,1\}} \{l_c^j(c') + d(c', x') + l_x^j(x')\}.$$

We argue that for every $1 \le j \le u$, g_x^j may be viewed as a minimum of $O(\epsilon^{-4})$ linear functions whose variables are the same with that of g_x^j.

- Linearity. Suppose $c \in V'$, and fix $c', x' \in Q_c \cup Q_x \cup \{0, 1\}$. By the above discussions, $l_c^j(c')$ could take values only from $\{d(c, q_i) : q_i \in Q_c\} \cup \{d(h_c, c')\}$. Since $\forall q_i \in Q_c$, $d(c, q_i)$ is a variable of g_x^j, and $d(h_c, c') = |h_c - c'|$ is linear and that h_c is also a variable of g_x^j, we conclude that $l_c^j(c')$ may be written as a linear function of the same set of variables of g_x^j. By a similar argument, we have the same conclusion for l_x^j. Therefore, $l_c^j(c') + d(c', x') + l_x^j(x')$ may be written as a linear function of $(q_1, \ldots, q_m, d(c, q_1), \ldots, d(c, q_m), h_c)$.

- Number of linear functions. By Claim 3.4.7, we have

  $$\forall y \in V, \qquad |Q_y| = O(\epsilon^{-2}),$$

  hence $|Q_c \cup Q_x \cup \{0, 1\}| = O(\epsilon^{-2})$. Therefore, there are $O(\epsilon^{-4})$ pairs of $c', x' \in Q_c \cup Q_x \cup \{0, 1\}$.

Therefore, item 2 of Lemma 3.4.9 follows by combining this with the definition of g_x. We completed the proof of Lemma 3.4.9. □

Now suppose Γ is the one that is guaranteed by Lemma 3.4.9. Since

$$\left|\{\sigma_c^{\mathcal{H}} : c \in V_i\}\right| \le \sum_{V' \in \Gamma} \left|\{\sigma_c^{\mathcal{H}} : c \in V'\}\right|$$

and

$$|\Gamma| \leq |H|^{\tilde{O}(\epsilon^{-2})\cdot u} \leq |H|^{\tilde{O}(\epsilon^{-2})}, \tag{3.12}$$

where the last inequality is by Lemma 3.4.4 (recalling $u = |\mathcal{P}_i|$), it suffices to show for every $V' \in \Gamma$,

$$\left|\{\sigma_c^{\mathcal{H}} : c \in V'\}\right| \leq |H|^{\tilde{O}(\epsilon^{-2})\log\|X\|_0}. \tag{3.13}$$

Fix some $V' \in \Gamma$. By Lemma 3.4.9, for every $x \in H$ there exists a min-linear function $g_x : \mathbb{R}^s \to \mathbb{R}_+$ ($s = O(\epsilon^{-2})$)), such that for every $c \in V'$, there exists $y \in \mathbb{R}^s$ that satisfies $f_x(c) = g_x(y)$. For $y \in \mathbb{R}^s$ define π_y^H as a permutation of H that is ordered by $g_x(y)$ in non-increasing order and ties are broken in a way that is consistent with σ. Then

$$\left|\{\sigma_c^{\mathcal{H}_v} : c \in V'\}\right| \leq \left|\{\pi_y^H : y \in \mathbb{R}^s\}\right|. \tag{3.14}$$

To bound the number of permutations π_y^H, we restates the following lemma which relates the number of relative orderings of g_x's to the arrangement number in computational geometry.

Lemma 3.4.10 (Restatement of Lemma 2.3.8)**.** *Suppose there are m functions $g_1, \ldots, g_m$ from $\mathbb{R}^s$ to $\mathbb{R}$, such that $\forall i \in [m]$, g_i is of the form*

$$g_i(x) := \min_{j\in[t]}\{g_{ij}(x)\},$$

where g_{ij} is a linear function. For $x \in \mathbb{R}^s$, let π_x be the permutation of $[m]$ ordered by $g_i(x)$. Then,

$$|\{\pi_x : x \in \mathbb{R}^s\}| \leq (mt)^{O(s)}.$$

Applying Lemma 3.4.10 on g_x's for $x \in H$ with parameters $s = O(\epsilon^{-2})$, $t = O(\epsilon^{-4}u) = O\left(\epsilon^{-4}\log\|X\|_0\right)$ and $m = |H|$, we obtain

$$\left|\{\pi_y^H : y \in \mathbb{R}^s\}\right| \leq O\left(\epsilon^{-1}|H|\log\|X\|_0\right)^{O(\epsilon^{-2})} \leq |H|^{\tilde{O}(\epsilon^{-2})\cdot\log\|X\|_0}. \tag{3.15}$$

Thus, (3.13) is implied by combining (3.15) with (3.14). Finally, we complete the proof of Lemma 3.4.3 by combining the above three parts of the arguments.

3.4.2.1 From Planar to Minor-excluded Graphs

The strategy for proving the minor-excluded case is similar to the planar case. Hence, we focus on presenting the major steps and highlight the differences, while omitting repetitive arguments. The terminal embedding lemma that we need to prove is restated as follows.

Lemma 3.4.11 (Restatement of Lemma 3.4.1)**.** *For every edge-weighted graph $G = (V, E)$ whose shortest path metric is denoted as $M = (V, d)$, and every weighted set $X \subseteq V$, given that G excludes some fixed minor, there exists a set of functions $\mathcal{F} := \{f_x : V \to \mathbb{R}_+ \mid x \in X\}$ such that for every $x \in X$, and $c \in V$, $d(x, c) \le f_x(c) \le (1 + \epsilon) \cdot d(x, c)$, and $\mathrm{sdim}_{\max}(\mathcal{F}) = \tilde{O}(\epsilon^{-2}) \cdot \log \|X\|_0$.*

Similar to the planar case, we fix $v : X \to \mathbb{R}_+$ and $\mathcal{H} \subseteq \mathcal{F}_v$ with $|\mathcal{H}| \ge 2$ throughout the proof. Then σ_c^H is defined the same as before, and it suffices to prove

$$|\{\sigma_c^{\mathcal{H}} : c \in V\}| \le \mathrm{poly}(\|X\|_0) \cdot |\mathcal{H}|^{\tilde{O}(\epsilon^{-2}) \log \|X\|_0}.$$

Next, we used a structural lemma to break V into several parts where each part is separated by a few shortest paths. In the planar case, we showed in Lemma 3.4.4 that the number of parts is $O(\|X\|_0)$, and only $O(1)$ separating shortest paths in G are necessary. However, the proof of Lemma 3.4.4 heavily relies on planarity, and for minor-excluded graphs, we only manage to prove the following weaker guarantee.

Lemma 3.4.12 (Structural Property of Minor-excluded Graphs)**.** *Given edge-weighted graph $G = (V, E)$ that excludes a fixed minor, and a subset $S \subseteq V$, there is a collection $\Pi := \{V_i\}_i$ of V with $|\Pi| = \mathrm{poly}(|S|)$ and $\bigcup_i V_i = V$ such that for every $V_i \in \Pi$ the following holds.*

1. $|S \cap V_i| = O(1)$.

2. *There exists an integer* t_i *and* t_i *groups of paths* $\mathcal{P}_1^i, \ldots, \mathcal{P}_{t_i}^i$ *in* G*, such that*

   (a) $|\bigcup_{j=1}^{t_i} \mathcal{P}_j^i| = O(\log |S|)$

   (b) *removing the vertices of all paths in* $\bigcup_{j=1}^{t_i} \mathcal{P}_j^i$ *disconnects* V_i *from* $V \setminus V_i$ *in* G *(possibly removing points in* V_i*)*

   (c) *for* $1 \leq j \leq t_i$*, let* G_j^i *be the sub-graph of* G *formed by removing all paths in* $\mathcal{P}_1^i, \ldots, \mathcal{P}_{j-1}^i$ *(define* $G_1^i = G$*), then every path in* $\mathcal{P}_j^i$ *is a shortest path in* G_j^i*.*

The lemma follows from a recursive application of the balanced shortest path separator theorem in [51, Theorem 1], stated as follows.

Lemma 3.4.13 (Balanced Shortest Path Separator [51])**.** *Given edge-weighted graph* $G = (V, E)$ *that excludes a fixed minor with non-negative vertex weight*[4]*, there is a set of vertices* $S \subseteq V$*, such that*

1. $S = P_1 \cup P_2 \cup \ldots$ *where* P_i *is a set of shortest paths in the graph formed by removing* $\bigcup_{j<i} P_j$

2. $\sum_i |P_i| = O(1)$*, where the hidden constant depends on the size of the excluded minor*

3. *the weight of every component in the graph formed by removing* S *from* G *is at most half the weight of* V*.*

Proof of Lemma 3.4.12. Without loss of generality, we assume G is a connected graph. We will apply Lemma 3.4.13 on G recursively to define the partition Π and the groups of shortest paths $\{\mathcal{P}_j^i\}_j$ associated with the parts. The detailed procedure,

[4] [51, Theorem 1] only states the special case with unit vertex weight, while the general weighted version was discussed in a note of the same paper.

called DEF-Π, is defined in Algorithm 5. We assume there is a global Γ initialized as $\Gamma = \emptyset$ which is constructed throughout the execution of the recursive algorithm. The execution of the algorithm starts with DEF-Π(G, $\emptyset$, S).

Roughly, the procedure DEF-Π takes a sub-graph G', a set $\mathsf{sep} = \{\mathcal{P}_j\}_j$ of groups of paths and S as input, such that G' corresponds to a component in a graph formed by removing all paths in sep from G. The procedure execute on such G' and find shortest paths in G' using Lemma 3.4.13. The found shortest paths are segmented (with respect to S) and added to the collection Π. Then the found shortest paths are removed from G' to form a new graph G''. Components in G'' that contain less than 2 points in S are made new parts in Π, and the procedure DEF-Π is invoked recursively on other components in G''.

Algorithm 5 Procedure DEF-Π($G' = (V', E')$, sep, S)

apply Lemma 3.4.13 on graph G' with vertex weight 1 if $x \in V' \cap S$ and 0 otherwise, and let $\mathcal{P}$ be the set of shortest paths in G' guaranteed by the lemma.
for $P \in \mathcal{P}$ **do**
    interpret P as interval $[0, 1]$, list $S \cap P = \{x_1, \ldots, x_m\}$ and $0 \le x_1 \le \ldots \le x_m \le 1$
    segment P into sub-paths $\mathcal{P}' = \{[0, x_1], [x_1, x_2], \ldots, [x_m, 1]\}$
    for $P' \in \mathcal{P}'$ **do**
        include P' in Π, and define the set of associated groups of shortest paths as $\mathsf{sep} \cup \{P'\}$
    end for
end for
let G'' be the graph formed by removing all paths in $\mathcal{P}$, and let $\mathcal{C} = \{C_i\}_i$ be its components
include the union of all components with no intersection with S as a single part in Π, and define the set of associated groups of paths as $\mathsf{sep} \cup \mathcal{P}$
for $C_i \in \mathcal{C}$ **do**
    if $|C_i \cap S| = 1$ **then**
        include C_i as a new part in Π, and define the set of associated groups of paths as $\mathsf{sep} \cup \mathcal{P}$
    else if $|C_i \cap S| \ge 2$ **then**
        call DEF-Π($G''[C_i]$, $\mathsf{sep} \cup \{\mathcal{P}\}$, S) ▷ $G''[C_i]$ is the induced sub-graph of G'' on vertex set C_i
    end if
end for

By construction and Lemma 3.4.13, it is immediate that $\bigcup_{V_i \in \Pi} V_i = V$, and item 2.(b), 2.(c) also follows easily. To see item 1, we observe that we have two types of V_i's in Π. One is from the shortest paths $\mathcal{P}$ (Line 6), and because of the segmentation, the intersection with S is at most 2. The other type is the components in G'' whose intersection with S is by definition at most 1 (Line 10, 13). Therefore, it remains to upper bound $|\Pi|$, and show item 2.(a) which requires a bound of $|\bigcup_{j=1}^{t_i} \mathcal{P}_j^i| = O(\log |S|)$ for all $V_i \in \Pi$.

First, we observe that at any execution of Gen-Π, it is always the case that $0 \leq |\mathsf{sep}| \leq O(\log |S|)$, because Lemma 3.4.13 guarantees the weight of every component in G'' is halved. This also implies that the total number of executions of GEN-Π is $\text{poly}(|S|)$. Therefore, $\forall V_i \in \Pi$, $|\bigcup_{j=1}^{t_i} \mathcal{P}_j^i| \leq O(\log |S|)$, which proves item 2.(a).

Bounding $|\Pi|$ Observe that there are three places where we include a part V_i in Π, and we let Π_1 be the subset of those included at Line 6, Π_2 be those included at Line 10, and Π_3 be those included at Line 13. Then $|\Pi| \leq |\Pi_1| + |\Pi_2| + |\Pi_3|$.

If $V_i \in \Pi_1$, then V_i is a sub-path of some $P \in \mathcal{P}$, where $\mathcal{P}$ is defined at Line 1. We observe that the number of all $V_i \in \Pi_1$ such that $V_i \cap S \neq \emptyset$, i.e. $|\{V_i \in \Pi_1 : V_i \cap S \neq \emptyset\}|$, is at most $O(|S|)$. This is because we remove paths $P \in \mathcal{P}$ in every recursion, which means any point in S can only participate in at most one such P during the whole execution, and hence any point in S can intersect at most two sub-paths $V_i \in \Pi_1$ such that $V_i \cap S \neq \emptyset$ (because $|V_i \cap S| \leq 2$ by the segmentation at Line 4). On the other hand, if $V_i \in \Pi_1$ and $V_i \cap S = \emptyset$, then no segmentation was performed and $V_i = P$ for P at Line 2. Therefore, the number of such V_i's is bounded by the total number of execution of DEF-Π multiplied by the size of $\mathcal{P}$ at Line 2, which is at most $\text{poly}(|S|)$. Therefore, we conclude that $|\Pi_1| = \text{poly}(|S|)$.

Finally, since every $V_i \in \Pi_3$ satisfies $|V_i \cap S| = 1$ (at Line 12 and 13), and we observe that subsets in Π_3 are disjoint, so we immediately have $|\Pi_3| = O(|S|)$. For Π_2, we note that only one $V_i \in \Pi_2$ could be included in each execution of DEF-Π, so

$|\Pi_2| = \text{poly}(|S|)$.

We conclude the proof of Lemma 3.4.12 by combining all the above discussions. □

As before, we still apply the Lemma 3.4.12 with $S = X$ (which is unweighted set) to obtain $\Gamma = \{V_i\}_i$ with $|\Pi| = O(\text{poly}(\|X\|_0))$, and it suffices to prove for each $V_i \in \Pi$

$$|\{\sigma_c^{\mathcal{H}} : c \in V_i\}| \leq |\mathcal{H}|^{\tilde{O}(\epsilon^{-2}) \log \|X\|_0}.$$

To proceed, we fix V_i and define functions $f_x(\cdot)$ for $c \in V_i$. However, compared with Lemma 3.4.4, the separating shortest paths in Lemma 3.4.12 are not from the original graph G, but is inside some sub-graph generated by removing various other separating shortest paths. Also, the number of shortest paths in the separator is increased from $O(1)$ to $O(\log \|X\|_0)$.

Hence, we need to define f_x's with respect to the new structure of the separating shortest paths. Suppose $\{\mathcal{P}_1^i, \ldots, \mathcal{P}_{t_i}^i\}$ is the t_i groups of paths guaranteed by Lemma 3.4.12. Also as in the lemma, suppose G_j^i is the sub-graph of G formed by removing all paths in $\mathcal{P}_1^i, \ldots, \mathcal{P}_{j-1}^i$ (define $G_1^i = G$). For $1 \leq j \leq t_i$, $P \in \mathcal{P}_j^i$ and $x, y \in V$, let $d_j^P(x, y)$ denote the length of the shortest path from x to y using edges in G_j^i and uses at least one point of P. Then, analogue to Fact 3.4.6, we have the following lemma.

Lemma 3.4.14. *For $c \in V_i$ and $x \in V \setminus V_i$, there exists $1 \leq j \leq t_i$, $P \in \mathcal{P}_j^i$ and $c', x' \in P$, such that $d(c, x) = d_j^P(c, c') + d_j^P(c', x') + d_j^P(x', x)$.*

Proof. First, we observe that the shortest path $c \rightsquigarrow x$ has to intersect (at a vertex of) at least one path contained in $\{\mathcal{P}_j^i\}_j$, because removing $\bigcup_{j=1}^{t_i} \mathcal{P}_j$ disconnects V_i from $V \setminus V_i$. Suppose j_0 is the smallest j such that $c \rightsquigarrow x$ intersects a shortest path in $\mathcal{P}_j^i$, and let $P \in \mathcal{P}_{j_0}^i$ be any intersected path in $\mathcal{P}_{j_0}^i$.

Then, this implies that (the edge set of) $c \rightsquigarrow x$ is totally contained in sub-graph $G_{j_0}^i$, since $G_{j_0}^i$ is formed by removing only groups $\mathcal{P}_j^s$ with $j < j_0$ which do not

intersect $c \rightsquigarrow x$. Hence, we have $d(c,x) = d_{G^i_{j_0}}(c,x)$, where $d_{G^i_{j_0}}$ is the shortest path metric in sub-graph $G^i_{j_0}$. By Lemma 3.4.12, P is a shortest path in $G^i_{j_0}$, so $c \rightsquigarrow x$ has to cross P at most once, which implies there exists $c', x' \in P$, such that $d(x,c) = d^P_j(c,c') + d^P_j(c',x') + d^P_j(x',x)$, as desired. □

Using Lemma 3.4.14 and by the optimality of the shortest path, we conclude that

$$\forall c \in V_i, x \in X, \quad d(c,x) = \min_{1 \le j \le t_i} \min_{P \in \mathcal{P}^i_j} \min_{c', x' \in P} \{d^P_j(c,c') + d^P_j(c',x') + d^P_j(x',x)\}.$$

Then, for each $1 \le j \le t_i$, path $P \in \mathcal{P}^i_j$, we use the same way as in the planar case to define the approximate distance function l to approximate $d^P_j(y,y')$ for $y \in V$ and $y' \in P$. The f_x is then defined similarly, and the distortion follows by a very similar argument as in Lemma 3.4.8.

The analysis of shattering dimension is also largely the same as before, except that the definition of u in the statement of Lemma 3.4.9 is slightly changed because of the new structural lemma. The new statement is presented as follows, and the proof of it is essentially as before.

Lemma 3.4.15. *Let $u = |\bigcup_{j=1}^{t_i} \mathcal{P}^i_j|$. There exists a partition Γ of V_i, such that the following holds.*

1. $|\Gamma| \le |H|^{\tilde{O}(\epsilon^{-2}) \cdot u}$.
2. $\forall V' \in \Gamma$, $\forall x \in H$, *there exists* $g_x : \mathbb{R}^s \to \mathbb{R}_+$ *where* $s = O(\epsilon^{-2})$, *such that* g_x *is a minimum of* $O(\epsilon^{-4}u)$ *linear functions on* $\mathbb{R}^s$, *and for every* $c \in V'$, *there exists* $y \in \mathbb{R}^s$ *that satisfies* $f_x(c) = g_x(y)$.

We apply the lemma with the new bound of $u = |\bigcup_{j=1}^{t_i} \mathcal{P}^i_j| = O(\log \|X\|_0)$ (by Lemma 3.4.12), and the bound in (3.13) is increased to

$$|\Gamma| \le |H|^{\tilde{O}(\epsilon^{-2}) \cdot u} \le |H|^{\tilde{O}(\epsilon^{-2}) \log \|X\|_0}.$$

Finally, to complete the proof of Lemma 3.4.1, we again use Lemma 3.4.10 on each $V' \in \Gamma$ to conclude the desired shattering dimension bound.

3.4.3 High-Dimensional Euclidean Spaces

We present a terminal embedding for Euclidean spaces, with a guarantee that is similar to that of excluded-minor graphs. For these results, the ambient metric space (V, d) of all possible centers is replaced by a Euclidean space.[5]

Lemma 3.4.16. *For every $\epsilon \in (0, 1/2)$ and finite weighted set $X \subset \mathbb{R}^m$, there exists $\mathcal{F} = \{f_x : \mathbb{R}^m \to \mathbb{R}_+ \mid x \in X\}$ such that*

$$\forall x \in X, c \in \mathbb{R}^m, \qquad \|x - c\|_2 \le f_x(c) \le (1+\epsilon)\|x - c\|_2,$$

and $\mathrm{sdim}_{\max}(\mathcal{F}) = O(\epsilon^{-2} \log \|X\|_0)$.

Proof. The lemma follows immediately from the following terminal version of the Johnson-Lindenstrauss Lemma [52], proved recently by Narayanan and Nelson [49].

Theorem 3.4.17 (Terminal Johnson-Lindenstrauss Lemma [49])**.** *For every $\epsilon \in (0, 1/2)$ and finite $S \subset \mathbb{R}^m$, there is an embedding $g : S \to \mathbb{R}^t$ for $t = O(\epsilon^{-2} \log |S|)$, such that*

$$\forall x \in S, y \in \mathbb{R}^m, \quad \|x - y\|_2 \le \|g(x) - g(y)\|_2 \le (1+\epsilon)\|x - y\|_2.$$

Given $X \subset \mathbb{R}^m$, apply Theorem 3.4.17 with $S = X$ (as an unweighted set), and define for every $x \in X$ the function $f_x(c) := \|g(x) - g(c)\|_2$. Then $\mathcal{F} = \{f_x \mid x \in X\}$ clearly satisfies the distortion bound. The dimension bound follows by plugging $t = O(\epsilon^{-2} \log \|X\|_0)$ into the bound $\mathrm{sdim}_{\max}(\mathcal{F}) = O(t)$ known from [13, Lemma 16.3].[6] □

Corollary 3.4.18 (Coresets for Euclidean Spaces)**.** *For every $0 < \epsilon, \delta < 1/2$, $z \ge 1$, and integers $k, m \ge 1$, Euclidean (k, z)-Clustering of every weighted set $X \subset \mathbb{R}^m$*

[5] It is easily verified that as long as X is finite, our entire framework from Section 3.3 extends to $V = \mathbb{R}^m$ with ℓ_2 norm. For example, all maximums (e.g., in Lemma 3.3.2) are well-defined by using compactness arguments on a bounding box.

[6] The following is proved in [13, Lemma 16.3]. For every $S \subset \mathbb{R}^t$, the function set $\mathcal{H} := \{h_x \mid x \in S\}$ given by $h_x(y) = \|x - y\|_2$, has shattering dimension $\mathrm{sdim}_{\max}(\mathcal{H}) = O(t)$.

admits an ϵ-coreset of size $\tilde{O}(\epsilon^{-4}2^{2z}k^2\log\frac{1}{\delta})$. Furthermore, such a coreset can be computed[7] *in time $\tilde{O}(k\|X\|_0 m)$ with success probability $1-\delta$.*

Proof. By combining Lemma 3.3.5, Lemma 3.3.6 with our terminal embedding from Lemma 3.4.16, we obtain an efficient algorithm for constructing a coreset of size $\tilde{O}(\epsilon^{-4}2^{2z}k^2\log\|X\|_0)$. This size can be reduced to the claimed size (and running time) using the iterative size reduction of Theorem 3.3.1. □

Remark 3.4.2 (Comparison to [21]). For (k,z)-Clustering in Euclidean spaces, our algorithms can also compute an ϵ-coreset of size $\tilde{O}(\epsilon^{-O(z)}k)$, which offers a different parameters tradeoff than Corollary 3.4.18. This alternative bound is obtained by simply replacing the application of Lemma 3.3.2 (which is actually from [41]) with [21, Lemma 3.1] (which itself is a result from [13], extended to weighted inputs).

Our two coreset size bounds are identical to the state-of-the-art bounds proved by Huang and Vishnoi [21] (in the asymptotic sense). Their analysis is different, and bounds $\mathrm{sdim}_{\max}$ independently of X using a dimensionality-reduction argument for clustering objectives. In contrast, we require only a loose bound $\mathrm{sdim}_{\max}(\mathcal{F}) = O(\mathrm{poly}(\epsilon^{-1})\cdot\log\|X\|_0)$, which follows immediately from [49], and the coreset size is then reduced iteratively using Theorem 3.3.1, which simplifies the analysis greatly.

[7] We assume that evaluating $\|x-y\|_2$ for $x,y\in\mathbb{R}^m$ takes time $O(m)$.

Chapter 4

Coresets for Ordered Weighted Clustering

4.1 Introduction

In this Chapter, we study coresets for a class of clustering problems, called ordered weighted clustering, which generalizes the classical k-Center and k-Median problems. In these clustering problems, the objective function is computed by ordering the n data points by their distance to their closest center, then taking a weighted sum of these distances, using predefined weights $v_1 \geq \cdots \geq v_n \geq 0$. These clustering problems can interpolate between k-Center (the special case where $v_1 = 1$ is the only non-zero weight) and k-Median (unit weights $v_i = 1$ for all i), and therefore offer flexibility in prioritizing points with large service cost, which may be important for applications like Pareto (multi-objective) optimization and fair clustering. In general, fairness in machine learning is seeing a surge in interest, and is well-known to have many facets. In the context of clustering, previous work such as the fairlets approach of [62], has addressed protected classes, which must be identified in advance. In contrast, ordered weighted clustering addresses fairness towards remote points (which can be underprivileged communities), without specifying them in advance. This is starkly different from many application domains, where remote points are considered as outliers (to be ignored) or anomalies (to be detected), see e.g., the well-known

survey by [63].

Formally, we study two clustering problems in Euclidean space $\mathbb{R}^d$. In both of them, the input is n data points $X \subset \mathbb{R}^d$ (and $k \in [n]$), and the goal is to find k centers $C \subset \mathbb{R}^d$ that minimize a certain objective $\text{cost}(X, C)$. In Ordered k-Median, there is a predefined non-decreasing weight vector $v \in \mathbb{R}^n_+$, and the data points $X = \{x_1, \ldots, x_n\}$ are ordered by their distance to the centers, i.e., $d(x_1, C) \geq \cdots \geq d(x_n, C)$, to define the objective

$$\text{cost}_v(X, C) := \sum_{i=1}^{n} v_i \cdot d(x_i, C), \tag{4.1}$$

where throughout $d(\cdot, \cdot)$ refers to ℓ_2 distance, extended to sets by the usual convention $\text{dist}(x, C) := \min_{c \in C} \text{dist}(x, c)$. This objective follows the Ordered Weighted Averaging (OWA) paradigm of [64], in which data points are weighted according to a predefined weight vector, but in order of their contribution to the objective. The p-Centrum problem is the special case where the first p weights equal 1 and the rest are 0, denoting its objective function by $\text{cost}_p(X, C)$. Observe that this problem already includes both k-Center (as $p = 1$) and k-Median (as $p = n$).

Recall that for a set X, a weighted set X' is an ϵ-coreset of X for clustering objective $\text{cost}(\cdot, \cdot)$ if it approximates the objective within factor $1 \pm \epsilon$, i.e.,

$$\forall C \subset \mathbb{R}^d, |C| = k, \quad \text{cost}(X', C) \in (1 \pm \epsilon)\,\text{cost}(X, C),$$

and the *size* of X' is the number of *distinct* points in it.[1]

The above coreset definition readily applies to ordered weighted clustering. However, a standard coreset is constructed for a specific clustering objective, i.e., a single weight vector $v \in \mathbb{R}^n_+$, which might limit its usefulness. The notion of a *simultaneous* coreset, introduced recently by [22], requires that all clustering objectives are preserved, i.e., the $(1 + \epsilon)$-approximation holds for all weight vectors in addition to all centers. This

[1] A common alternative definition is that X' is as a set with weights $w : X' \to \mathbb{R}_+$, which represent multiplicities, and then size is the number of non-zero weights. This would be more general if weights are allowed to be fractional, but then one has to extend the definition of $\text{cost}(\cdot, \cdot)$ accordingly.

"simultaneous" feature is valuable in data analysis, since the desired weight vector might be application and/or data dependent, and thus not known when the data reduction is applied. Moreover, since ordered weighted clustering includes classical clustering, e.g., k-MEDIAN and k-CENTER as special cases, all these different analyses may be performed on a single simultaneous coreset.

4.1.1 Our Contribution

Our main result is (informally) stated as follows. To simplify some expressions, we use $O_{\epsilon,d}(\cdot)$ to suppress factors depending only on ϵ and d. The precise dependence appears in the technical sections.

Theorem 4.1.1 (informal version of Theorem 4.4.6)**.** *There exists an algorithm that, given an n-point data set $X \subset \mathbb{R}^d$ and $k \in [n]$, computes a simultaneous ϵ-coreset of size $O_{\epsilon,d}(k^2 \log^2 n)$ for ORDERED k-MEDIAN.*

Our main result is built on top of a coreset result for p-CENTRUM (the special case of ORDERED k-MEDIAN in which the weight vector is 1 in the first p components and 0 in the rest). For this special case, we have an improved size bound, that avoids the $O(\log^2 n)$ factor, stated as follows. Note that this coreset is for a single value of p (and not simultaneous).

Theorem 4.1.2 (informal version of Theorem 4.4.4)**.** *There exists an algorithm that, given an n-point data set $X \subset \mathbb{R}^d$ and $k, p \in [n]$, computes an ϵ-coreset of size $O_{\epsilon,d}(k^2)$ for p-CENTRUM.*

The size bounds in the two theorems are nearly tight. The dependence on n in Theorem 4.1.1 is unavoidable, because we can show that the coreset size has to be $\Omega(\log n)$, even when $k = d = 1$. (See Theorem 6.2).

For both Theorem 4.1.1 and Theorem 4.1.2, the hidden dependence on ϵ and d is $(\frac{1}{\epsilon})^{d+O(1)}$. This factor matches known lower bounds [D. Feldman, private communica-

tion] and state-of-the-art constructions of coresets for k-Center (which is a special case of Ordered k-Median) [65].

A main novelty of our coreset is that it preserves the objective for all weights ($v \in \mathbb{R}^n_+$ in the objective function) simultaneously. It is usually easy to combine coresets for two data sets, but in general it is not possible to combine coresets for two different objectives. Moreover, even if we manage to combine coresets for two objectives, it is still nontrivial to achieve a small coreset size for infinitely many objectives (all possible weight vectors $v \in \mathbb{R}^n_+$). See the overview in Section 4.1.2 for more details on the new technical ideas needed to overcome these obstacles.

4.1.2 Overview of Techniques

We start with discussing Theorem 4.1.2 (which is a building block for Theorem 4.1.1). Its proof is inspired by [12], who constructed coresets for k-Median clustering in $\mathbb{R}^d$ by reducing the problem to its one-dimensional case. We can apply a similar reduction, but the one-dimensional case of p-Centrum is significantly different from k-Median. One fundamental difference is that the objective counts only the p largest distances, hence the subset of "contributing" points depends on the center. We deal with this issue by introducing a new bucketing scheme and a charging argument that relates the error to the p largest distances. See Section 4.3 for more details.

The technical difficulty in Theorem 4.1.1 is two-fold: how to combine coresets for two different weight vectors, and how to handle infinitely many weight vectors. The key observation is that every Ordered k-Median objective can be represented as a linear combination of p-Centrum objectives (see Lemma 4.4.7). Thus, it suffices to compute a simultaneous coreset for p-Centrum for all $p \in [n]$. We achieve this by "combining" the individual coresets for all $p \in [n]$, while crucially utilizing the special structure of our construction of a p-Centrum coreset, but unfortunately losing an $O(\log n)$ factor in the coreset size. In the end, we need to "combine" the n coresets

for all $p \in [n]$, but we can avoid losing an $O(n)$ factor by discretizing the values of p, so that only $O(\log n)$ coresets are combined, The result is a simultaneous coreset of size $O_{\epsilon,d}(\log^2 n)$, see Section 4.4 for more details.

4.1.3 Additional Related Work

Ordered k-Median and its special case p-Centrum generalize k-Center and are thus APX-hard even in $\mathbb{R}^2$ [66]. However, p-Centrum may be solved optimally in polynomial time for special cases such as lines and trees [67]. The first provable approximation algorithm for Ordered k-Median was proposed by [68], and they gave 2-approximation for trees and $O(\log n)$-approximation for general metrics. The approximation ratio for general metrics was drastically improved to 38 by [69], improved to $18+\epsilon$ by [70], and finally a $(5+\epsilon)$-approximation was obtained very recently by [71].

4.2 Preliminaries

Throughout this Chapter, we use capital letters other than I and J to denote finite subsets of $\mathbb{R}^d$. We recall some basic terminology from [12]. For a set $Y \subset \mathbb{R}$, define its *mean point* to be

$$\mu(Y) := \frac{1}{|Y|} \sum_{y \in Y} y, \tag{4.2}$$

and its *cumulative error* to be

$$\delta(Y) := \sum_{y \in Y} |y - \mu(Y)|. \tag{4.3}$$

Let $I(Y) := [\inf Y, \sup Y]$ denote the smallest closed interval containing Y. The following facts from [12] will be useful in our analysis.

Lemma 4.2.1. *For every $Y \subset \mathbb{R}$ and $z \in \mathbb{R}$, the following holds,*

- $\sum_{y \in Y} \Big| |z-y| - |z-\mu(Y)| \Big| \leq \delta(Y)$*; and*
- *if $z \notin I(Y)$ then $\sum_{y \in Y} |y-z| = |Y| \cdot |\mu(Y) - z|$.*

It will be technically more convenient to treat a coreset as a point set $X' \subset \mathbb{R}^d$ associated with integer weights $w : X' \to \mathbb{N}$, which is equivalent to a multiset (with weights representing multiplicity), and thus the notation of $\mathrm{cost}_v(X', C)$ in (4.1) is well-defined. (These weights w are unrelated to the predefined weights $\{v_i\}$.) While our algorithm always produces X' with integral weights w, our proof requires fractional weights during the analysis, and thus we extend (4.2) and (4.3) to a point set Y with weights $w : Y \to \mathbb{R}_+$ by defining $\mu_w(Y) := \frac{1}{\sum_{y \in Y} w(y)} \sum_{y \in Y} w(y) \cdot y, \delta_w(Y) := \sum_{y \in Y} w(y) \cdot |y - \mu(Y)|$.

We will use the fact that in one-dimensional Euclidean space, p-Centrum can be solved (exactly) in polynomial time by dynamic programming, as shown by [67].

Lemma 4.2.2 ([67])**.** *There is a polynomial-time algoritm that, given a set of one-dimensional points $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}$ and parameters $k, p \in [n]$, computes a set of k centers $C \subset \mathbb{R}^d$ that minimizes $\mathrm{cost}_p(X, C)$.*

4.3 The Basic Case: p-Centrum for $k = d = 1$ (one facility in one-dimensional data)

In this section, we illustrate our main ideas by constructing a coreset for p-Centrum in the special case of one facility in one-dimensional Euclidean space (i.e., $k = d = 1$). This is not a simultaneous coreset, but rather for a single p. The key steps of our construction described below will be repeated, with additional technical complications, also in the general case of p-Centrum, i.e., k facilities in dimension d.

We will need two technical lemmas, whose proofs appear in Section 4.3.1.

The first lemma bounds $\delta(Y)$ by the cost of connecting Y to an arbitrary point outside $I(Y)$ (which in turn is part of the objective in certain circumstances).

Lemma 4.3.1. *Let $Y \subset \mathbb{R}$ be a set with (possibly fractional) weights $w : Y \to \mathbb{R}_+$. Then for every $z \in \mathbb{R}$ such that $z \notin I(Y)$ or z is an endpoint of $I(Y)$, $\delta_w(Y) \leq$*

$2\sum_{y\in Y} w(y)\cdot|y-z|$.

Recall that $k=1$, hence the cost in an instance of p-Centrum is the sum of the p largest distances to the center. In the analysis of our coreset it will be useful to replace some points of the input set X with another set Y. The second lemma will be used to bound the resulting increase in the cost; it considers two sequences, denoted X and Y, of the connection costs, and bounds the difference between the sum of the p largest values in X and that in Y by a combination of ℓ_∞ and ℓ_1 norms.

Lemma 4.3.2. *Let $X=(x_1,\ldots,x_n)$ and $Y=(y_1,\ldots,y_n)$ be two sequences of real numbers. Then for all $S\subseteq[n]$, $|\operatorname{top}_p(X)-\operatorname{top}_p(Y)|\le p\max_{i\in S}|x_i-y_i|+\sum_{i\in[n]\setminus S}|x_i-y_i|$, where $\operatorname{top}_p(Z)$ is the sum of the p largest numbers in Z.*

Outline of the Coreset Construction In the context of a one-dimensional point set $X\subset\mathbb{R}$, the term *interval* will mean a subset of X that spans a contiguous subsequence under a fixed ordering of the points, i.e., a subset $\{x_i,\ldots,x_j\}$ when the points in X are ordered as $x_1\le\ldots\le x_n$. Informally, our coreset construction works as follows. First, use Lemma 4.2.2 to find an optimal center y^*, its corresponding optimal cost OPT, and a subset $P\subset X$ of size $|P|=p$ that contributes to the optimal cost. Then partition the data into three intervals, namely $X=L\cup R\cup Q$, as follows. Points from P that are $\le y^*$ are placed in L, points from P that are $>y^*$ are placed in R, and all other points are placed in $Q=X\setminus P$. Now split L, Q and R into sub-intervals, in a greedy manner that we describe below, and represent the data points in each sub-interval by adding to the coreset a single point, whose weight is equal to the number of data points it replaces. See Figure 4-1 for illustration.

To split L into sub-intervals, scan its points from the smallest to the largest and pack them into the same sub-interval J as long as their cumulative error $\delta(J)$ is below a threshold set to $\Theta(\varepsilon\cdot\mathrm{OPT})$. This ensures, by Lemma 4.3.1, a lower bound on their total connection cost to the optimal center y^*, which we use to upper bound the

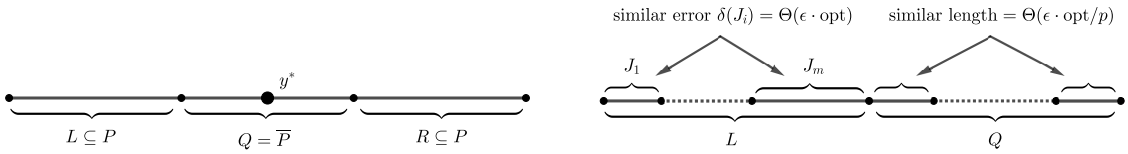

Figure 4-1. Coreset construction for p-CENTRUM with $k = 1$ facilities in dimension $d = 1$. The left figure depicts the partition of the data into $X = (L \cup R) \cup Q$, where $P = L \cup R$ contains the p furthest points from an optimal center y^*. The right figure shows the different manners of splitting L and Q into intervals.

---

number of such intervals (which immediately affects the size of the coreset) by $O\left(\frac{1}{\epsilon}\right)$. The split of R is done similarly. To split $Q = X \setminus P$, observe that the distance from every $q \in Q$ to the center y^* is less than $\frac{\text{OPT}}{p}$, hence the diameter of Q is less than $\frac{2\text{OPT}}{p}$, and Q can be partitioned into $O(\frac{1}{\varepsilon})$ sub-intervals of length $O(\frac{\varepsilon \text{OPT}}{p})$. Observe that the construction for Q differs from that of L and R.

Let D denote the coreset resulting from the above construction. To prove that the resulting coreset has the desired error bound for every potential center $y \in \mathbb{R}$, we define an intermediate set Z that contains a mix of points from X and D. We stress that Z depends on the potential center $y \in \mathbb{R}$, which is possible because Z is used only in the analysis. The desired error bound follows by bounding both $|\operatorname{cost}(Z, y) - \operatorname{cost}(X, y)|$ and $|\operatorname{cost}(Z, y) - \operatorname{cost}(D, y)|$, (here we use Lemma 4.3.2), and by the triangle inequality.

Detailed Construction and Coreset Size

We now give a formal description of our coreset construction. Let $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}$ be the input data set, and recall that $\operatorname{cost}_p(X, y)$ for a point $y \in \mathbb{R}$ is the sum of the p largest numbers in $\{|x_1 - y|, \ldots, |x_n - y|\}$. Denote the optimal center by $y^* := \operatorname{argmin}_{y \in \mathbb{R}} \operatorname{cost}_p(X, y)$, and the corresponding optimal cost by $\text{OPT} := \operatorname{cost}_p(X, y^*)$. By Lemma 4.2.2, y^* and OPT can be computed in polynomial time. Next, sort X by distances to y^*. For simplicity, we shall assume the above notation for X is already in this sorted order, i.e., $|x_1 - y^*| \geq \cdots \geq |x_n - y^*|$. Thus, $\operatorname{cost}_p(X, y^*) = \sum_{i=1}^{p} |x_i - y^*|$. Let $P := \{x_1, \ldots, x_p\}$, $L := \{x_i \leq y^* : x_i \in P\}$, $R := \{x_i > y^* : x_i \in P\}$ and

$Q := X \setminus P$. By definition, X is partitioned into L, Q and R, which form three intervals located from left to right. We now wish to split L, Q and R into sub-intervals, and then we will add to D the mean of the points in each sub-interval, with weight equal to the number of such points.

Split L into sub-intervals from left to right greedily, such that the cumulative error of each interval J does not exceed $\frac{2\varepsilon \cdot \mathrm{OPT}}{21}$, and each sub-intervals is maximal, i.e., the next point cannot be added to it. Split R into sub-intervals similarly but from right to left. We need to bound the number of sub-intervals produced in this procedure. For sake of analysis, we consider an alternative split of L that is fractional, i.e., allows assigning a point fractionally to multiple sub-intervals, say 1/3 to the sub-interval to its left and 2/3 to the sub-interval to its right. The advantage of this fractional split is that all but the last sub-interval have cumulative error *exactly* $\frac{2\varepsilon \cdot \mathrm{OPT}}{21}$.

We show in Lemma 4.3.3 that the number of sub-intervals produced in the original integral split is at most twice that of the fractional split, and thus it would suffice to bound the latter by $O(\frac{1}{\varepsilon})$.

Lemma 4.3.3. *The number of sub-intervals in the integral split is at most twice than that of the fractional split.*

Proof. It suffices to show that for every t, the first $2t$ sub-intervals produced by the integral partitioning contain at least as many points as the first t sub-intervals produced by the fractional partitioning. For this, the key observation is that in a fractional split, only the two endpoints of a sub-interval may be fractional, because the cumulative error of a singleton set is 0.

We prove the above by induction on t. The base case $t = 1$ follows from these observations, since the fractional sub-interval may be broken into two integral sub-intervals, each with cumulative error at most $\frac{2\varepsilon \cdot \mathrm{OPT}}{21}$. Suppose the claim holds for $t-1$ and let us show that it holds for t. Since the cumulative error is monotone in adding

new points, we may assume the first $t-1$ sub-intervals from fractional split contain as many points as the first $2(t-1)$ sub-intervals from integral split. Now similarly to the base case, the t-th fractional interval may be broken into two integral sub-intervals, and this proves the inductive step. □

To see that the number of sub-intervals produced by a fractional partitioning of L is $O(\frac{1}{\varepsilon})$, we use Lemma 4.3.1. Suppose there are m such sub-intervals $J_1, ..., J_m$. We can assume that the first $m-4$ of them do not contain y^* and have cumulative error at least $\frac{2\varepsilon \cdot \mathrm{OPT}}{21}$, because at most two sub-intervals can contain y^*, and at most one sub-interval from each of L and R may have cumulative error less than $\frac{2\varepsilon \cdot \mathrm{OPT}}{21}$. By Lemma 4.3.1 and the fact that y^* is not in the first $i \leq m-4$ sub-intervals, $\mathrm{OPT} \geq \sum_{i=1}^{m-4} \sum_{x:x\in J_i} |x-y^*| \geq \frac{1}{2}\sum_{i=1}^{m-4} \delta_w(J_i) = (m-4)\frac{\varepsilon \cdot \mathrm{OPT}}{21}$. Thus $m = O(\frac{1}{\varepsilon})$, and by Lemma 4.3.3 a similar bound holds also for the number of sub-intervals in the integral split of L and of R. Now split Q greedily into maximal sub-intervals of length $\leq \frac{\varepsilon \cdot \mathrm{OPT}}{3p}$. Since $\max_{q\in Q} |q-y^*| \leq |x_p - y^*| \leq \frac{\mathrm{OPT}}{p}$, the length of $I(Q)$ is $\leq \frac{2\mathrm{OPT}}{p}$, and we conclude that Q is split into at most $\frac{3}{\varepsilon}+1$ sub-intervals.

Finally, construct the coreset D from the sub-intervals, by adding to D the mean of each sub-interval in D, with weight that is the number of points in this sub-interval. Since the total number of sub-intervals is $O(\frac{1}{\epsilon})$, the size of the coreset D is also bounded by $O(\frac{1}{\epsilon})$.

Coreset Accuracy To prove that D is an ϵ-coreset for X, fix a potential center $y \in \mathbb{R}$ and let us prove that $|\operatorname{cost}_p(D,y) - \operatorname{cost}_p(X,y)| \leq \varepsilon \cdot \mathrm{OPT}$, where we interpret D as a multi-set. Let $P_1 \subseteq X$ denote the set of p points in X that are farthest from y. Now define an auxiliary set $Z := \{z_1, \ldots, z_n\}$, as follows. For each $i \in [n]$, let $X_i \subset X$ be the sub-interval containing x_i in the construction of the coreset (recall it uses the optimal center y^* and *not* y), and let $\pi(x_i) = \mu(X_i)$ be its representative in the coreset D. Now if (a) $i \leq p$; (b) $y \notin X_i$; and (c) $P_1 \cap X_i$ is either empty or all of X_i; then let

$z_i := \pi(x_i)$. Otherwise, let $z_i := x_i$.

We now aim to bound $|\operatorname{cost}_p(Z, y) - \operatorname{cost}_p(D, y)|$ using Lemma 4.3.2 with $S = \{p+1, ..., n\}$. Consider first some $i \in S$ (i.e., $i > p$). Then

$$\begin{aligned} |d(z_i, y) - d(\pi(x_i), y)| &\leq |z_i - \pi(x_i)| \\ &= |x_i - \pi(x_i)| && (4.4) \\ &\leq \frac{\varepsilon \cdot \mathrm{OPT}}{3p}. && (4.5) \end{aligned}$$

Consider next $i \notin S$ (i.e., $i \leq p$). We can have $z_i \neq \pi(x_i)$ only if $y \in X_i$ or if $P_1 \cap X_i$ is neither empty nor all of X_i. This can happen for at most 7 distinct sub-intervals X_i, because the former case can happen for at most 3 sub-intervals X_i (by a simple case analysis of how many sub-intervals might have an endpoint at y, e.g., two from L, or one from each of L, R, Q) and because P_1 is contained in 2 intervals (to the left and right of y), and each of them can intersect at most 2 distinct sub-intervals X_i without containing all of X_i. We obtain

$$\begin{aligned} &\sum_{i=1}^{p} |d(z_i, y) - d(\pi(x_i), y)| \\ &= \sum_{i \in [p]: z_i \neq \pi(x_i)} |d(x_i, y) - d(\pi(x_i), y)| && (4.6) \\ &\leq \sum_{X_i : i \in [p], (y \in X_i) \vee (P_1 \cap X_i \neq \emptyset, X_i)} \delta(X_i) && (4.7) \\ &\leq 7 \cdot \frac{2\varepsilon \cdot \mathrm{OPT}}{21} && (4.8) \\ &= \frac{2\varepsilon \cdot \mathrm{OPT}}{3}, && (4.9) \end{aligned}$$

where (4.7) is by Lemma 4.2.1, and (4.9) is by the fact that these X_i are from L or R (recall $i \leq p$) and thus have a bounded cumulative error. Applying Lemma 4.3.2 to our $S = \{p+1, ..., n\}$ together with (4.5) and (4.9), we obtain $|\operatorname{cost}_p(Z, y) - \operatorname{cost}_p(D, y)| \leq p \cdot \frac{\varepsilon \cdot \mathrm{OPT}}{3p} + \frac{2\varepsilon \cdot \mathrm{OPT}}{3} = \varepsilon \cdot \mathrm{OPT}$.

Lastly, we need to prove that $\operatorname{cost}_p(Z, y) = \operatorname{cost}_p(X, y)$. We think of Z as if it is obtained from X by replacing each x_i with its corresponding $z_i = \pi(x_i) = \mu(X_i)$. We can of course restrict attention to indices where $z_i \neq x_i$, which happens only if all

three requirements (a)-(c) hold. Moreover, whenever this happens for point x_i, it must happen also for all points in the same sub-interval X_i, i.e., every $x_j \in X_i$ is replaced by $z_j = \pi(x_j) = \mu(X_i)$. By requirement (c), X_i is either disjoint from P_1 or contained in P_1. In the former case, points $x_j \in X_i$ do not contribute to $\text{cost}_P(X, y)$ because they are not among the p farthest points, and then replacing all $x_j \in X_i$ with $z_j = \mu(X_i)$ would maintain this, i.e., the corresponding points z_j do not contribute to $\text{cost}_p(Z, y)$. In the latter case, the points in X_i contribute to $\text{cost}_P(X, y)$ because they are among the p farthest points, and replacing every $x_j \in X_i$ with $z_j = \mu(X_i)$ would maintain this, i.e., the corresponding points z_j contribute to $\text{cost}_p(Z, y)$. Moreover, their total contribution is the same because using requirement (b) that $y \notin X_i$, we can write their total contribution as $\sum_{x_j \in X_i} d(x_j, y) = |X_i| \cdot d(\mu(X_i), y) = \sum_{x_j \in X_i} d(\pi(x_j), y)$.

4.3.1 Proofs of Technical Lemmas

Lemma 4.3.4 (restatement of Lemma 4.3.1)**.** *Let $Y \subset \mathbb{R}$ be a set with (possibly fractional) weights $w : Y \to \mathbb{R}_+$. Then for every $z \in \mathbb{R}$ such that $z \notin I(Y)$ or z is an endpoint of $I(Y)$,*

$$\delta_w(Y) \leq 2 \sum_{y \in Y} w(y) \cdot |y - z|.$$

Proof. Assume w.l.o.g. that z is to the left of $I(Y)$, i.e., $z \leq \inf_{y \in Y} y$. Partition Y into $Y_L = \{y \in Y : y \leq \mu_w(Y)\}$ and $Y_R = Y \setminus Y_L$. Define $w_L := \sum_{x \in Y_L} w(x)$ and $w_R := \sum_{x \in Y_R} w(x)$. Since

$$(w_L + w_R) \cdot \mu_w(Y) = \sum_{y \in Y} w(y) \cdot y = \sum_{y \in Y_L} w(y) \cdot y + \sum_{y \in Y_R} w(y) \cdot y,$$

we have that

$$\sum_{y \in Y_L} w(y)(\mu_w(Y) - y) = \sum_{y \in Y_R} w(y)(y - \mu_w(Y)).$$

For every $y \in Y_L$ we actually have $z \le y \le \mu_w(Y)$, and we conclude that

$$\begin{aligned}
2\sum_{y\in Y} w(y)(y-z) &= 2(w_L + w_R)\cdot(\mu_w(Y) - z) \\
&\ge 2w_L \cdot (\mu_w(Y) - z) \\
&\ge 2\sum_{y\in Y_L} w(y)(\mu_w(Y) - y) \\
&= \sum_{y\in Y_L} w(y)(\mu_w(Y) - y) + \sum_{y\in Y_R} w(y)(y - \mu_w(Y)) \\
&= \delta_w(Y).
\end{aligned}$$

□

Lemma 4.3.5 (restatement of Lemma 4.3.2)**.** *Let $X = (x_1, \ldots, x_n)$ and $Y = (y_1, \ldots, y_n)$ be two sequences of real numbers. Then for all $S \subseteq [n]$,*

$$|\operatorname{top}_p(X) - \operatorname{top}_p(Y)| \le p \max_{i\in S} |x_i - y_i| + \sum_{i\in[n]\setminus S} |x_i - y_i|,$$

where $\operatorname{top}_p(Z)$ is the sum of the p largest numbers in Z.

Proof. For all $T \subseteq [n], |T| = p$,

$$\begin{aligned}
\left|\sum_{i\in T}(x_i - y_i)\right| &\le \sum_{i\in T\cap S} |x_i - y_i| + \left|\sum_{i\in T\setminus S}(x_i - y_i)\right| \\
&\le p \max_{i\in S} |x_i - y_i| + \sum_{i\in[n]\setminus S} |x_i - y_i|.
\end{aligned}$$

Now let $X_1 \subseteq [n]$ be the set of indices of the p largest numbers in X, then by the above inequality

$$\begin{aligned}
\operatorname{top}_p(X) = \sum_{i\in X_1} x_i &\le \sum_{i\in X_1} y_i + \left|\sum_{i\in X_1}(x_i - y_i)\right| \\
&\le \operatorname{top}_p(Y) + p \max_{i\in S} |x_i - y_i| + \sum_{i\in[n]\setminus S} |x_i - y_i|.
\end{aligned}$$

By symmetry, the same upper bound holds also for $\operatorname{top}_p(Y) - \operatorname{top}_p(X)$, and the lemma follows. □

4.4 Simultaneous Coreset for Ordered k-Median

In this section we give the construction of a simultaneous coreset for Ordered k-Median on data set $X \subset \mathbb{R}^d$ (Theorem 4.4.6), which in turn is based on a coreset for p-Centrum (Theorem 4.4.4). In both constructions, we reduce the general instance in $\mathbb{R}^d$ to an instance X' that lies on a small number of lines in $\mathbb{R}^d$. The reduction is inspired by a projection procedure of [12], that goes as follows. We start with an initial centers set C, and then for each center $c \in C$, we shoot $O(\frac{1}{\epsilon})^d$ lines from center c to different directions, and every point in X is projected to its closest line. The projection cost is bounded because the number of lines shot from each center is large enough to accurately discretize all possible directions. The details appear in Section 4.4.2.

For the projected instance X', we construct a coreset for each line in X' using ideas similar to the case $d = k = 1$, which was explained in Section 4.3. However, the error of the coreset cannot be bounded line by line, and instead, we need to address the cost globally for all lines altogether, see Lemma 4.4.1 for the formal analysis. Finally, to construct a coreset for p-Centrum in $\mathbb{R}^d$, the initial centers set C for the projection procedure is picked using some polynomial-time $O(1)$-approximation algorithm, such as by [71]. A coreset of size $O_{\epsilon,d}(k^2)$ is obtained by combining the projection procedure with Lemma 4.4.1.

To deal with the infinitely many potential weights in the simultaneous coreset for Ordered k-Median, the key observation is that it suffices to construct a simultaneous coreset for p-Centrum for $O(\frac{\log n}{\epsilon})$ different value of p, and then "combine" the corresponding p-Centrum coresets. An important structural property of the p-Centrum coreset is that it is formed by mean points of some sub-intervals. This enables us to "combine" coresets for p-Centrum by "intersecting" all their sub-intervals into even smaller intervals. However, this idea works only when the sub-

intervals are defined on the same set of lines, which were generated by the projection procedure. To resolve this issue, we set the centers set C in the projection procedure to be the union of all centers needed for p-Centrum in all the $O(\log n)$ values of p. Since the combination of the coresets for p-Centrum yields even smaller sub-intervals, the error analysis for the individual coreset for p-Centrum still carries on. The size of the simultaneous coreset is $O(\log^2 n)$-factor larger than that for (a single) p-Centrum, because we combine $O(\log n)$ coresets for p-Centrum, and we use $O(\log n)$ times more centers in the projection procedure. The detailed analysis appears in Section 4.4.3.

4.4.1 Coreset for p-Centrum on Lines in $\mathbb{R}^d$

Below, we prove the key lemma that bounds the error of the coreset for p-Centrum for a data set that may be represented by lines. The proof uses the idea introduced for the $k = d = 1$ case in Section 4.3. In particular, we define an intermediate (point) set Z to help compare the costs between the coreset and the true objective. The key difference from Section 4.3 in defining Z is that the potential centers might not be on the lines, so extra care should be taken. Moreover, we use a global cost argument to deal with multiple lines in X. We also introduce parameters s and t_l in the lemma. These parameters are to be determined with respect to the initial center set C in the projection procedure, and eventually we want $(s + \sum_{l\in\mathcal{L}} t_l)$ to be $O(\mathrm{OPT}_p)$ where OPT_p is the optimal for p-Centrum. We introduce these parameters to have flexibility in picking s and t_l, which we will need later when we construct a simultaneous coreset that uses a more elaborate set of initial centers C.

Lemma 4.4.1. *Suppose $k \in \mathbb{Z}_+$, $\epsilon \in (0, 1)$, $X \subset \mathbb{R}^d$ is a data set, and $\mathcal{L}$ is a collection of lines in $\mathbb{R}^d$. Furthermore,*

- *X is partitioned into $\{X_l \mid l \in \mathcal{L}\}$, where $X_l \subseteq l$ for $l \in \mathcal{L}$, and*
- *$\forall l \in \mathcal{L}$, X_l is partitioned into a set of disjoint sub-intervals $\mathcal{Y}_l$, such that $\forall Y \in \mathcal{Y}_l$, either $\mathrm{len}(I(Y)) \le O(\frac{\epsilon}{p} \cdot s)$ or $\delta(Y) \le O(\frac{\epsilon}{k} \cdot t_l)$ for some $s, t_l > 0$.*

Then for all sets $C \subset \mathbb{R}^d$ of k centers, the weighted set $D := \{\mu(Y) \mid Y \in \mathcal{Y}_l, l \in \mathcal{L}\}$ with weight $|Y|$ for element $\mu(Y)$ satisfies $|\operatorname{cost}_p(D, C) - \operatorname{cost}_p(X, C)| \leq O(\epsilon) \cdot (s + \sum_{l \in \mathcal{L}} t_l)$.

Proof. Suppose $X = \{x_1, \ldots, x_n\}$. The proof idea is similar to the $d = k = 1$ case as in Section 4.3. In particular, we construct an auxiliary set of points $Z := \{z_1, \ldots, z_n\}$, and the error bound is implied by bounding both $|\operatorname{cost}_p(Z, C) - \operatorname{cost}_p(D, C)|$ and $|\operatorname{cost}_p(Z, C) - \operatorname{cost}_p(X, C)|$ for all k-subset $C \subset \mathbb{R}^d$.

Notations For $x_i \in X$, let $Y_i \in \mathcal{Y}_l$ denote the unique sub-interval that contains x_i, where l is the line that Y_i belongs to, and let $\pi(x_i)$ denote the unique coreset point in Y_i (which is $\mu(Y_i)$). Define $M := \{x_i \mid \operatorname{len}(Y_i) \leq O(\frac{\epsilon}{p} \cdot s)\}$ and $N := X \setminus M$. We analyze the error for any given $C = \{c_1, \ldots, c_k\}$ and let $\mathcal{C}_i = \{x \in X : \arg\min_{j \in [k]} d(x, c_j) = c_i\}$ (ties are broken arbitrarily) be the cluster induced by C. If $x \in \mathcal{C}_i$, we say x is served by c_i. Let $P_1 \subset X$ denote the set of p farthest points to C. Define c_{il} to be the projection of c_i onto line l.

Defining Z We define z_i to be either x_i or $\pi(x_i)$ as follows. For $x_i \in M$, let $z_i = x_i$. For $x_i \in N$, if

a) $I(Y_i)$ does not contain any c_{jl} for $j \in [n], l \in \mathcal{L}$, and

b) all points in Y_i are served by a unique center, and

c) Y_i is either contained in P_1 or does not intersect P_1,

then we define $z_i := \pi(x_i)$ otherwise $z_i := x_i$.

Let error $:= \epsilon \cdot (s + \sum_{l \in \mathcal{L}} t_l)$. It suffices to show $|\operatorname{cost}_p(D, C) - \operatorname{cost}_p(Z, C)| \leq O(\text{error})$ and $|\operatorname{cost}_p(Z, C) - \operatorname{cost}_p(X, C)| \leq O(\text{error})$.

Part I: $|\operatorname{cost}_p(D, C) - \operatorname{cost}_p(Z, C)| \leq O(\text{error})$ To prove $|\operatorname{cost}_p(D, C) - \operatorname{cost}_p(Z, C)| \leq$

$O(\text{error})$, we apply Lemma 4.3.2 with $S = M$, so

$$\begin{aligned}
&|\text{cost}_p(D, C) - \text{cost}_p(Z, C)| \\
&\leq p \cdot \max_{x_i \in M} |d(\pi(x_i), C) - d(z_i, C)| \\
&\quad + \sum_{x_i \in N} |d(\pi(x_i), C) - d(z_i, C)|. \qquad (4.10)
\end{aligned}$$

Observe that $x_i \in M$ implies $|d(\pi(x_i), x_i)| \leq \text{len}(I(Y_i)) = O(\frac{\epsilon}{p} \cdot s)$. Therefore,

$$p \cdot \max_{x_i \in M} |d(\pi(x_i), C) - d(z_i, C)| \leq p \cdot O(\frac{\epsilon}{p} \cdot s) \cdot p = O(\epsilon \cdot s). \qquad (4.11)$$

Then we bound the second term $\sum_{x_i \in N} |d(\pi(x_i), C) - d(z_i, C)|$. We observe that in each line l, there are only $O(k)$ distinct sub-intervals $Y_i \in \mathcal{Y}_l$ induced by $x_i \in N$ such that $z_i = x_i$. Actually, for each line l, there are at most k sub-intervals $Y \in \mathcal{Y}_l$ such that $I(Y)$ contains some c_{il} for $i \in [n]$, and there are at most $2k$ sub-intervals whose points are served by at least 2 centers, and there are at most $4k$ intervals that intersect P_1 but are not fully contained in P_1. Hence,

$$\begin{aligned}
\sum_{x_i \in N} |d(\pi(x_i), C) - d(z_i, C)| &\leq \sum_{x_i \in N : z_i = x_i} d(\pi(x_i), x_i) \\
&\leq \sum_{Y_i : x_i \in N, z_i = x_i} \delta(Y_i) \\
&\leq \sum_{l \in \mathcal{L}} O(k) \cdot \frac{\epsilon}{k} \cdot t_l \qquad (4.12) \\
&= O(\epsilon) \cdot \sum_{l \in \mathcal{L}} t_l. \qquad (4.13)
\end{aligned}$$

Combining Inequality 4.11 and 4.13 with 4.10, we conclude that $|\text{cost}_p(D, C) - \text{cost}_p(Z, C)| \leq O(\text{error})$.

Part II: $|\text{cost}_p(Z, C) - \text{cost}_p(X, C)| \leq O(\text{error})$ Let $\mathcal{Y}' \subseteq \bigcup_{l \in \mathcal{L}} \mathcal{Y}_l$ be the set of sub-intervals Y_i such that $x_i \in N$ and a) - c) hold (i.e. $z_i = \pi(x_i)$). Note that by construction, the only difference between Z and X is due to replacing points in sub-intervals $Y \in \mathcal{Y}'$ with $|Y|$ copies of $\mu(Y)$, thus it suffices to analyze this replacement error.

Let $P_Z \subseteq Z$ be (multi)-set of the p-furthest points of Z from C. We start with showing that, the coreset point $\mu(Y)$ of $Y \in \mathcal{Y}'$, is fully contained in P_Z or does not intersect P_Z. Consider some $Y \in \mathcal{Y}'$ and assume points in Y are all served by c_j. Denote the endpoints of interval $I(Y)$ as a and b. Let $l \in \mathcal{L}$ be the line that contains Y. Since $I(Y)$ does not contain c_{jl}, then either $\angle abc_j > \frac{\pi}{2}$ or $\angle bac_j > \frac{\pi}{2}$. W.l.o.g., we assume that $\angle abc_j > \frac{\pi}{2}$. By Observation 4.4.2, we know that if $Y \cap P_1 = \emptyset$, then $\mu(Y) \notin P_Z$; on the other hand, if $Y \subseteq P_1$, then $\mu(Y) \in P_Z$.

Observation 4.4.2. *Let ΔABC denote a triangle where $\angle ABC > \frac{\pi}{2}$. Let E be a point on the edge BC then $|AC| \geq |AE| \geq |AB|$.*

Hence, if $Y \in \mathcal{Y}'$ has empty intersection with P_1, the $|Y|$ copies of $\mu(Y)$ in Z does not contribute to either $\text{cost}_p(Z, C)$ or $\text{cost}_p(X, C)$. Thus, it remains to bound the error for $Y \in \mathcal{Y}'$ such that $Y \subseteq P_1$. In the 1-dimensional case as in Section 4.3, replacing Y with the mean $\mu(Y)$ does not incur any error, as the center is at the same line with the interval $I(Y)$. However, this replacement might incur error in the d-dimensional case since the center might be *outside* the line that contains the sub-interval Y. Luckily, this error has been analyzed in [12, Lemma 2.8], and we adapt their argument in Lemma 4.4.3 (shown below). By Lemma 4.4.3, $\forall c_j \in C, l \in \mathcal{L}$, the total replacement error for all sub-intervals $Y \in \mathcal{Y}' \cap \mathcal{Y}_l$ such that i) $Y \subseteq P_1$ and ii) all points in Y are served by c_j, is at most $O(\frac{\epsilon}{k} \cdot t_l)$. Therefore,

$$\begin{aligned} |\,\text{cost}_p(Z, C) - \text{cost}_p(X, C)| &\leq \sum_{l \in \mathcal{L}} \sum_{c_j \in C} O(\frac{\epsilon}{k} \cdot t_l) \\ &= O(\epsilon) \cdot \sum_{l \in \mathcal{L}} t_l \\ &\leq O(\text{error}). \end{aligned}$$

This finishes the proof of Lemma 4.4.1. □

Lemma 4.4.3. *Let $l \subset \mathbb{R}^d$ be a line and $c \in \mathbb{R}^d$ be a point. Define c_l be the projection of c on l. Assume that $X_1, \ldots, X_m \subset l$ are finite sets of points in l such*

that $I(X_1), \ldots, I(X_m)$ *are disjoint,* $\delta(X_i) \le r$ *and* $\forall i \in [m]$, $c_l \notin I(X_i)$. *Then*

$$\left| \sum_{i=1}^{m} \sum_{x \in X_i} d(x, c) - \sum_{i=1}^{m} |X_i| \cdot d(\mu(X_i), c) \right| \le O(r).$$

Proof. W.l.o.g. we assume $\forall i \in [m]$, X_i is not a singleton, since $\sum_{x \in X_i} d(x, c) = |X_i| \cdot d(\mu(X_i), c)$ for singleton X_i. Let $\mathrm{err}_i := \sum_{x \in X_i} d(x, c) - |X_i| \cdot d(\mu(X_i), c)$. In [12, Lemma 2.5], it was shown that $\mathrm{err}_i \ge 0$ for all $i \in [m]$ (using that $c_l \notin I(X_i)$). Furthermore, it follows from the argument of [12, Lemma 2.8] that, if each X_i is modified into a weighted set X_i' with *real* weight $w_i : X_i' \to \mathbb{R}_+$, such that $I(X_i) = I(X_i')$ and each X_i' has the same cumulative error $\delta_{w_i}(X_i') = r$, then

- $\forall i \in [m]$, $\mathrm{err}_i' := \sum_{x \in X_i'} w(x) \cdot d(x, c) - \left(\sum_{x \in X_i'} w_i(x) \right) \cdot d(\mu_{w_i}(X_i'), c) \ge 0$, and
- $\sum_{i \in [m]} \mathrm{err}_i' = \sum_{i \in [m]} \sum_{x \in X_i'} w(x) \cdot d(x, c) - \sum_{i \in [m]} \left(\left(\sum_{x \in X_i'} w_i(x) \right) \cdot d(\mu_{w_i}(X_i'), c) \right) \le O(r)$.

Hence, it suffices to show that it is possible to modify each X_i into a real weighted set X_i' with $I(X_i') = I(X_i)$, such that $\delta(X_i') = r$ and $\mathrm{err}_i' \ge \mathrm{err}_i$ for all $i \in [m]$.

For each $i \in [m]$, find two points $a \ne b \in I(X_i)$ such that $\mu(X_i)$ is the midpoint of a and b. Such a and b must exist since we assume X_i is not a singleton. We form X_i' by adding points a and b with the same (real-valued) weight into X_i, such that $\delta_{w_i}(X_i') = r$, then $\mathrm{err}_i' \ge \mathrm{err}_i$ follows from the geometric fact that $d(c, a) + d(c, b) \ge 2d(c, \mu(X_i))$. This concludes Lemma 4.4.3. □

4.4.2 Coreset for p-Centrum in $\mathbb{R}^d$

We now prove the theorem about a coreset for p-Centrum. As discussed above, we use a projection procedure inspired by [12] to reduce to line cases, and then apply Lemma 4.4.1 to get the coreset.

Theorem 4.4.4. *Given* $k \in \mathbb{Z}_+$, $\epsilon \in (0, 1)$, *an* n*-point data set* $X \subset \mathbb{R}^d$, *and* $p \in [n]$, *there exists an* ϵ*-coreset* $D \subset \mathbb{R}^d$ *of size* $O(\frac{k^2}{\epsilon^{d+1}})$ *for* p*-Centrum. Moreover, it can be*

computed in polynomial time.

We start with a description of how we reduce to the line case, which will be used again in the simultaneous coreset.

Reducing to Lines: Projection Procedure Consider an m-point set $C := \{c_1, \ldots, c_m\} \subset \mathbb{R}$ which we call projection centers. We will define a new data set X' by projecting points in X to some lines defined with respect to C. The lines are defined as follows. For each $c_i \in C$, construct an ϵ-net N_i for the unit sphere centered at c_i, and for $u \in N_i$, define l_{iu} as the line that passes through c_i and u. Let $\mathcal{L} := \{l_{iu} \mid i \in [m], u \in N_i\}$ be the set of projection lines. Then X' is defined by projecting each data point $x \in X$ to the nearest line in $\mathcal{L}$. Since N_i's are ϵ-nets on unit spheres in $\mathbb{R}^d$, we have $|\mathcal{L}| \leq O(\frac{1}{\epsilon})^d \cdot |C|$. The cost of this projection is analyzed below in Lemma 4.4.5.

Lemma 4.4.5 (projection cost)**.** *For all $C' \subset \mathbb{R}^d$ and $p \in [n]$, $|\operatorname{cost}_p(X', C') - \operatorname{cost}_p(X, C')| \leq O(\epsilon) \cdot \operatorname{cost}_p(X, C)$.*

Proof. For $x \in X$, denote the projection of x by $\sigma(x) \in X'$, and for $y \in X'$ let $\sigma^{-1}(y) \in X$ be any $x \in X$ such that $\sigma(x) = y$. Observe that $\forall x \in X$, $d(x, C') - d(\sigma(x), C') \leq d(x, \sigma(x)) \leq O(\epsilon) \cdot d(x, C)$, where the first inequality is by triangle inequality, and the last inequality is by the definition of $\sigma(x)$.

Let $A \subseteq X'$ be the p-furthest points from C' in X'. Then

$$\begin{aligned}\operatorname{cost}_p(X', C') = \sum_{x \in A} d(x, C') &\leq \sum_{x \in A} d(\sigma^{-1}(x), C') + O(\epsilon) \cdot \sum_{x \in A} d(\sigma^{-1}(x), C) \\ &\leq \operatorname{cost}_p(X, C') + O(\epsilon) \cdot \operatorname{cost}_p(X, C).\end{aligned}$$

Similarly, let $B \subseteq X$ be the p-furthest points from C' in X. Then

$$\begin{aligned}\operatorname{cost}_p(X, C') = \sum_{x \in B} d(x, C') &\leq \sum_{x \in B} d(\sigma(x), C') + O(\epsilon) \cdot \sum_{x \in B} d(x, C) \\ &\leq \operatorname{cost}_p(X', C') + O(\epsilon) \cdot \operatorname{cost}_p(X, C).\end{aligned}$$

This finishes the proof. □

We remark that both the projection center and the candidate center C' in Lemma 4.4.5 are not necessarily k-subsets. This property is not useful for the coreset for p-Centrum, but it is crucially used in the simultaneous coreset in Section 4.4.3.

Below we give the proof of Theorem 4.4.4.

Proof of Theorem 4.4.4. For the purpose of Theorem 4.4.4, we pick C to be an $O(1)$-approximation to the optimal centers for the k-facility p-Centrum on X, i.e., $\text{cost}_p(X, C) \leq O(1) \cdot \text{OPT}_p$, where OPT_p is the optimal value for the p-Centrum. Such C may be found in polynomial time by applying known approximation algorithms, say by [71]. As analyzed in Lemma 4.4.5, for such choice of C, the error incurred in X' because of the projections is bounded by $O(\epsilon) \cdot \text{OPT}_p$.

We will apply Lemma 4.4.1 on X'. The line partitioning $\{X'_l \mid l \in \mathcal{L}\}$ of X' that we use in Lemma 4.4.1 is naturally induced by the line set $\mathcal{L}$ resulted from the projection procedure. Then, for each $l \in \mathcal{L}$, we define the disjoint sub-intervals $\mathcal{Y}_l$ as follows. Let $S := X' \cap l$, let $S_1 \subseteq S$ be the subset of the p-furthest points from C, and let $S_2 := S \setminus S_1$. We then break S_1 and S_2 into sub-intervals, using similar method as in Section 4.3. Let $\text{APX} := \text{cost}_p(X', C)$, and let APX_l be the contribution of S in APX. Break S_1 into sub-intervals according to cumulative error δ with threshold $O(\frac{\epsilon \cdot \text{APX}_l}{k})$, similar with how we deal with L and R in Section 4.3. Break S_2 into maximal sub-intervals of length $\Theta(\frac{\epsilon \cdot \text{APX}}{p})$, similar with Q in Section 4.3. Again, similar with the analysis in Section 4.3, the number of sub-intervals is at most $O(\frac{k}{\epsilon})$ for each $l \in \mathcal{L}$.

Finally, we apply Lemma 4.4.1 with $t_l := \text{APX}_l$ and $s := \text{APX}$, and it yields a multi-set D such that $|\text{cost}_p(D, C') - \text{cost}_p(X, C')| \leq O(\epsilon) \cdot (\text{APX} + \sum_{l \in \mathcal{L}} \text{APX}_l) = O(\epsilon) \cdot \text{APX} = O(\epsilon) \cdot \text{OPT}_p$. On the other hand, the size of D is upper bounded by $|\mathcal{L}| \cdot \frac{k}{\epsilon} \leq O(\frac{k}{\epsilon^{d+1}}) \cdot |C| \leq O(\frac{k^2}{\epsilon^{d+1}})$. This concludes Theorem 4.4.4. □

4.4.3 Simultaneous Coreset for Ordered k-Median in $\mathbb{R}^d$

In this section we prove our main theorem that is stated below as Theorem 4.4.6. As discussed before, we first show it suffices to give simultaneous coreset for p-Centrum for $O(\log n)$ values of p. Then we show how to combine these coresets to obtain a simultaneous coreset.

Theorem 4.4.6. *Given $k \in \mathbb{Z}_+$, $\epsilon \in (0,1)$ and an n-point data set $X \subset \mathbb{R}^d$, there exists a simultaneous ϵ-coreset of size $O(\frac{k^2 \log^2 n}{\epsilon^d})$ for Ordered k-Median. Moreover, it can be computed in polynomial time.*

We start with the following lemma, which reduces simultaneous coresets for Ordered k-Median to simultaneous coresets for p-Centrum.

Lemma 4.4.7. *Suppose $k \in \mathbb{Z}_+$, $\varepsilon \in (0,1)$, $X \subset \mathbb{R}^d$ and D is a simultaneous ϵ-coreset for the k-facility p-Centrum problem for all $p \in [n]$. Then D is a simultaneous ϵ-coreset for Ordered k-Median.*

Proof. Suppose $X = \{x_1, \ldots, x_n\}$. We need to show for any center C and any weight v, $\mathrm{cost}_v(D, C) \in (1 \pm \epsilon) \cdot \mathrm{cost}_v(X, C)$. Fix a center C and some weight v. We assume w.l.o.g. $d(x_1, C) \geq \ldots \geq d(x_n, C)$. By definition we have $\mathrm{cost}_v(X, C) = \sum_{i=1}^n v_i \cdot d(x_i, C)$ and $\mathrm{cost}_p(X, C) = \sum_{i=1}^p d(x_i, C)$ for any p. Since D is an ε-coreset of X for p-Centrum on every $p \in [n]$, $\mathrm{cost}_p(D, C) \in (1 \pm \varepsilon)\,\mathrm{cost}_p(X, C)$. Let $v_{n+1} := 0$, and we have

$$\begin{aligned}
\mathrm{cost}_v(D, C) &= \sum_{p=1}^{n} (v_p - v_{p+1}) \cdot \mathrm{cost}_p(D, C) \\
&\in (1 \pm \epsilon) \cdot \sum_{p=1}^{n} (v_p - v_{p+1}) \cdot \mathrm{cost}_p(X, C) \\
&= (1 \pm \epsilon) \cdot \mathrm{cost}_v(X, C).
\end{aligned}$$

□

With the help of the following lemma, we only need to preserve the objective for p's taking powers of $(1+\epsilon)$. In other words, it suffices to construct simultaneous coresets to preserve the objective for only $O(\frac{\log n}{\epsilon})$ distinct values of p's.

Lemma 4.4.8. *Let $X, C \subset \mathbb{R}^d$ and $p_1, p_2 \in [n]$ such that $p_1 \leq p_2 \leq (1+\epsilon) \cdot p_1$. Then $\mathrm{cost}_{p_1}(X, C) \leq \mathrm{cost}_{p_2}(X, C) \leq (1+\epsilon) \cdot \mathrm{cost}_{p_1}(X, C)$.*

Proof. We assume w.l.o.g. $d(x_1, C) \geq \ldots \geq d(x_n, C)$. By definition,

$$\mathrm{cost}_{p_2}(X, C) = \mathrm{cost}_{p_1}(X, C) + \sum_{i=p_1}^{p_2} d(x_i, C) \geq \mathrm{cost}_{p_1}(X, C).$$

On the other hand,

$$\begin{aligned}
\mathrm{cost}_{p_2}(X, C) &= \mathrm{cost}_{p_1}(X, C) + \sum_{i=p_1+1}^{p_2} d(x_i, C) \\
&\leq \mathrm{cost}_{p_1}(X, C) + (p_2 - p_1) \cdot \frac{1}{p_1} \mathrm{cost}_{p_1}(X, C) \\
&\leq \mathrm{cost}_{p_1}(X, C) + \epsilon \cdot p_1 \cdot \frac{1}{p_1} \cdot \mathrm{cost}_{p_1}(X, C) \\
&= (1+\epsilon) \cdot \mathrm{cost}_{p_1}(X, C).
\end{aligned}$$

□

We are now ready to present the proof of Theorem 4.4.6.

Proof of Theorem 4.4.6. As mentioned above, by Lemma 4.4.8, it suffices to obtain an ϵ-coreset for $O(\frac{\log n}{\epsilon})$ values of p's. Denote the set of these values of p's as W.

We use a similar framework as in Theorem 4.4.4, and we start with a projection procedure. However, the projection centers are different from those in Theorem 4.4.4. For each $p \in W$, we compute an $O(1)$-approximate solution C_p for p-Centrum, which is a k-subset. Then, we define $C := \bigcup_{p \in P} C_p$ be the union of all these centers, so $|C| \leq O(\frac{k \cdot \log n}{\epsilon})$. Let X' be the projected data set. By Lemma 4.4.5, the projection cost is bounded by $O(\epsilon) \cdot \mathrm{cost}_p(X, C) \leq O(\epsilon) \cdot \mathrm{cost}_p(X, C_p) \leq O(\epsilon) \cdot \mathrm{OPT}_p$, for all $p \in W$.

Following the proof of Theorem 4.4.4, for each $p \in W$, we apply Lemma 4.4.1 on the projected set X' in exactly the same way, and denote the resulted coreset as D_p. By a similar analysis, for each p, the size of D_p is $O(\frac{k^2 \cdot \log n}{\epsilon^{d+1}})$.

Then we describe how to combine D_p's to obtain the simultaneous coreset. A crucial observation is that, the coresets D_p's are constructed by replacing sub-intervals with their mean points, and for all $p \in W$, the D_p's are built on the same set of lines. Therefore, we can combine the sub-intervals resulted from all D_p's. Specifically, combining two intervals $[a, b]$ and $[c, d]$ yields $[\min\{a, c\}, \max\{a, c\}]$, $[\max\{a, c\}, \min\{b, d\}]$, $[\min\{b, d\}, \max\{b, d\}]$. For any particular p, in the combined sub-intervals, the length upper bound and the δ upper bound required in Lemma 4.4.1 still hold. Hence the coreset D resulted from the combined sub-intervals is a simultaneous coreset for all $p \in W$. By Lemma 4.4.7 and Lemma 4.4.8, D is a simultaneous ϵ-coreset for Ordered k-Median.

The size of D is thus $O(\log n)$ times the coreset for a single p. Therefore, we conclude that the above construction gives a simultaneous ϵ-coreset with size $O(\frac{k^2 \log^2 n}{\epsilon^{d+1}})$, which completes the proof of Theorem 4.4.6. □

Chapter 5

Coreset for Clustering with Missing Values

In this Chapter, we consider coresets and approximation algorithms for k-clustering problems, particularly k-Means and more generally (k, z)-Clustering (see Definition 1.2.1), for points in $\mathbb{R}^d$ with *missing values (coordinates)*. The presence of missing values in data sets is a common phenomenon, and dealing with it is a fundamental challenge in data science. While data imputation is a very popular method for handling missing values, it often requires prior knowledge which might not be available, or statistical assumptions on the missing values that might be difficult to verify [72, 73]. In contrast, our worst-case approach does not requires any prior knowledge. Specifically, in our context of clustering, the distance $\mathrm{dist}(x, c)$ between a clustering center point c and a data point x is evaluated only on the available (i.e., non-missing) coordinates. Similar models that aim to minimize clustering costs using only the available coordinates have been proposed in previous work [74–77], and some other relevant works were discussed in a survey [78].

Clustering under this distance function, which is evaluated only on the available coordinates, is a formidable computational challenge, because distances do not satisfy the triangle inequality, and therefore many classical and effective clustering algorithms, such as k-Means++ [79], cannot be readily applied or even be defined properly.

Despite the algorithmic interest in clustering with missing values, the problem is still not well understood and only a few results are known. In a pioneering work, Gao, Langberg and Schulman [80] initiated the algorithmic study of the k-Center problem with missing values. They took a geometric perspective and interpreted the k-Center with missing values problem as an affine-subspace clustering problem, and followup work [81, 82] has subsequently improved and generalized their algorithm. Only very recently, approximation algorithms for objectives other than k-Center, particularly k-Means, were obtained for the limited case of at most one missing coordinate in each input point [1] or for constant number of missing coordinates [2].

We focus on designing coresets for clustering with missing values. Roughly speaking, an ϵ-coreset is a small proxy of the data set, such that the clustering objective is preserved within $(1 \pm \epsilon)$ factor for all center sets (see Definition 1.2.2 for formal definition). Efficient constructions of small ϵ-coresets usually lead to efficient approximations schemes, since the input size is reduced to that of the coreset, see e.g. [1, 7, 8]. Moreover, apart from speeding up approximation algorithms in the classical setting (offline computation), coresets can also be applied to design streaming [83–85], distributed [86–88], and dynamic algorithms [89, 90], which are effective methods/models for dealing with big data, and recently coresets were used even in neural networks [91].

5.1 Our Results

Coresets. Our main result, stated in Theorem 5.1.1, is a near-linear time construction of coresets for k-Means with missing values. Here, an ϵ-coreset for k-Means for a data set X in $\mathbb{R}^d$ with missing coordinates is a weighted subset $S \subseteq X$ with weights $w : S \to \mathbb{R}_+$, such that

$$\forall C \subset \mathbb{R}^d, |C| = k, \qquad \sum_{x \in S} w(x) \cdot \mathrm{dist}^2(x, C) \in (1 \pm \epsilon) \sum_{x \in X} \mathrm{dist}^2(x, C),$$

where $\mathrm{dist}(x,c) := \sqrt{\sum_{i:x_i \text{ not missing}} (x_i - c_i)^2}$, and $\mathrm{dist}(x,C) := \min_{c \in C} \mathrm{dist}(x,c)$; note that the center set C does not contain missing values. More generally, our coreset also works for (k,z)-Clustering, which includes k-Median.

Theorem 5.1.1 (Informal version of Theorem 5.3.1)**.** *There is an algorithm that, given $0 < \epsilon < 1/2$, integers $d, j, k \geq 1$, and a set $X \subset \mathbb{R}^d$ of n points each having at most j missing values, it constructs with constant probability an ϵ-coreset for k-Means on X of size $m = (jk)^{O(\min\{j,k\})} \cdot (\epsilon^{-1} d \log n)^2$, and runs in time $\tilde{O}\left((jk)^{O(\min\{j,k\})} \cdot nd + m\right)$.*

Our coreset size is only a low-degree polynomial of d, ϵ and $\log n$, and can thus deal with moderately-high dimension or large data set. The dependence on k (number of clusters) and j (maximum number of missing values per point) is also a low-degree polynomial as long as at least one of k and j is small. Furthermore, the space complexity of our construction algorithm is near-linear, and since our coreset is clearly mergeable, it is possible to apply the merge-and-reduce method [83] to convert our construction into a streaming algorithm of space $\mathrm{poly}\log n$. Prior to our result, the only known coreset construction for clustering with missing values is for the special case $j = 1$ [1] and has size $k^{O(k)} \cdot (\epsilon^{-2} d \log n)$. Since our coreset has size $\mathrm{poly}(k\epsilon^{-1} d \log n)$ when $j = 1$, it improves the dependence on k over that of [1] by a factor of $k^{O(k)}$.

Near-linear time PTAS for k-Means with missing values. Very recently, a PTAS for k-Means with missing values, was obtained by Eiben, Fomin, Golovach, Lochet, Panolan, and Simonov [2]. Its time bound is *quadratic*, namely $O(2^{\mathrm{poly}(jk/\epsilon)} \cdot n^2 d)$, and since our coreset can be constructed in near-linear time, we can speedup this PTAS to *near-linear* time by first constructing our coreset and then running this PTAS on the coreset.

Corollary 5.1.2 (Near-linear time PTAS for k-Means with missing values)**.** *There is an algorithm that, given $0 < \epsilon < 1/2$, integers $d, j, k \geq 1$, and a set $X \subset \mathbb{R}^d$ of n points each having at most j missing values, it finds with constant probability a*

$(1+\epsilon)$-*approximation for* k-MEANS *on* X*, and runs in time* $\tilde{O}\Big((jk)^{O(\min\{j,k\})} \cdot nd + 2^{\mathrm{poly}(jk/\epsilon)} \cdot d^{O(1)}\Big)$.

5.1.1 Technical Overview

Our coreset construction is based on the importance sampling framework introduced by Feldman and Langberg [13] and subsequently improved and generalized by [17, 41]. In the framework, one first computes an importance score σ_x for every data point $x \in X$, and then draws independent samples with probabilities proportional to these scores. When no values are missing, the importance scores can be computed easily, even for general metric spaces [17, 41, 42]. However, a significant challenge with missing values is that distances do not satisfy the triangle inequality, hence importance scores cannot be easily computed.

We overcome this hurdle using a method introduced by Varadarajan and Xiao [92] for projective clustering (where the triangle inequality similarly does not hold). They reduce the importance-score computation to the construction of a coreset for k-CENTER objective; this method is quite different from earlier approaches, e.g. [13, 17, 41, 42], and yields a coreset for k-MEANS whose size depends linearly on $\log n$ and of course on the size of the k-CENTER coreset. (Mathematically, this arises from the sum of all importance scores.) We make use of this reduction, and thus focus on constructing (efficiently) a small coreset for k-CENTER with missing values.

An immediate difficulty is how to deal with the missing values. We show that it is possible to find a collection of subsets of coordinates $\mathcal{I}$ (so each $I \in \mathcal{I}$ is a subset of $[d]$), such that if we construct k-CENTER coresets S_I on the data set "restricted" to each $I \in \mathcal{I}$, then the union of these S_I's is a k-CENTER coreset for the original data set with missing values. Crucially, we ensure that each "restricted" data set does not contain any missing value, so that it is possible to use a classical coreset construction for k-CENTER. Finally, we show in a technical lemma how to find a collection as

necessary of size $|\mathcal{I}| \leq (jk)^{O(\min\{j,k\})}$.

Since a "restricted" data set does not contain any missing values, we can use a classical k-Center coreset construction, and a standard construction has size $O(k\epsilon^{-d})$ [65], which is known to be tight. We bypass this ϵ^{-d} limitation by observing that actually $\tilde{O}(1)$-coreset for k-Center suffices, even though the final coreset error is ϵ. We observe that an $\tilde{O}(1)$-coreset can be constructed using a variant of Gonzalez's algorithm [93].

To implement Gonzalez's algorithm, a key step is to find the *furthest* neighbor of a given subset of at most $O(k)$ points, and a naive implementation of this runs in linear time, which overall yields a quadratic-time coreset construction, because the aforementioned reduction of [92] actually requires $\Theta(n/k)$ successive runs of Gonzalez's algorithm. To resolve this issue, we propose a fully-dynamic implementation of Gonzalez's algorithm so that a furthest-point query is answered in time $\text{poly}(k \log n)$, and the point-set is updated between successive runs instead of constructed from scratch. Our dynamic algorithm is based on a random-projection method that was proposed for furthest-point queries in the streaming setting [94]. Specifically, we project the (restricted) data set onto several random directions, and on each projected (one-dimensional) data set we apply a data structure for intervals.

5.1.2 Additional Related Work

Recently, attention was given to some non-traditional settings of coresets for clustering, including coresets for Gaussian mixture models (GMM) [95, 96]; coresets for logistic regressions [97]; and coresets for clustering under fairness constraints [98]. Also considered were settings that capture uncertainty, for example when each point is only known to lie in a line (i.e., clustering lines) [1], and when each point comes from a finite set (i.e., clustering point sets) [99].

5.2 Preliminaries

We represent a data point as a vector in $(\mathbb{R} \cup \{?\})^d$, and a coordinate takes "?" if and only if it is missing. Let $\mathbb{R}^d_?$ be a shorthand for $(\mathbb{R} \cup \{?\})^d$. Throughout, we consider a data set $X \subset \mathbb{R}^d_?$. The distance is evaluated only on the coordinates that are present in both x, y, i.e.,

$$\forall x, y \in \mathbb{R}^d_?, \qquad \mathrm{dist}(x, y) := \sqrt{\sum_{i: x_i, y_i \neq ?} (x_i - y_i)^2}.$$

For $x \in \mathbb{R}^d_?$, we denote the set of coordinates that are not missing by $I_x := \{i : x_i \neq ?\}$. For integer $m \geq 1$, let $[m] := \{1, \ldots, m\}$. For two points $p, q \in \mathbb{R}^d_?$ and an index set $I \subseteq I_p \cap I_q$, we define the *I-induced distance* to be $\mathrm{dist}_I(p, q) := \sqrt{\sum_{i \in I} (p_i - q_i)^2}$. A point $x \in \mathbb{R}^d_?$ is called a j-point if it has at most j missing coordinates, i.e., $|I_x| \geq d - j$.

5.3 Coresets

We prove our main theorem in this section.

Theorem 5.3.1. *There is an algorithm that, given as input a data set $X \subset \mathbb{R}^d_?$ of size $n = |X|$ consisting of j-points and parameters $k, z \geq 1$ and $0 < \epsilon < 1/2$, constructs with constant probability an ϵ-coreset of size $m = \tilde{O}\left(z^z \cdot \frac{(j+k)^{j+k+1}}{j^j k^{k-z-2}} \cdot \frac{(d \log n)^{\frac{z+2}{2}}}{\epsilon^2}\right)$ for (k, z)-Clustering of X, and runs in time $\tilde{O}\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-2}} \cdot nd + m\right)$.*

We remark that $\frac{(j+k)^{j+k}}{j^j k^k} = (jk)^{O(\min(j,k))}$. To see this, assume $j \geq k$ w.l.o.g., so $\frac{(j+k)^j}{j^j} = (1 + \frac{k}{j})^j \leq e^{\frac{k}{j} \cdot j} = e^k$ and $\frac{(j+k)^k}{k^k} \leq (j+k)^k$.

Theorem 5.3.1 is the main theorem of this Chapter, and we present the proof in this section. As mentioned in Section 5.1.1 and previous chapters, the coreset is constructed via importance sampling, by following three major steps.

1. For each data point $x \in X$, compute an importance score $\sigma_x \geq 0$.

2. Draw N (to be determined later) independent samples from X, such that $x \in X$ is sampled with probability $p_x \propto \sigma_x$.

3. Denote the sampled (multi) set as S, and for each $x \in S$ define its weight $w(x) := \frac{1}{p_x N}$. Report the weighted set S as the coreset.

The importance score σ_x is usually defined as (an approximation) of the *sensitivity* of x, denoted

$$\sigma_x^\star := \sup_{C \subset \mathbb{R}^d, |C|=k} \frac{\mathrm{dist}^z(x, C)}{\mathrm{cost}_z(X, C)}, \tag{5.1}$$

which measures the maximum possible relative contribution of x to the objective function.

Usually, there are two main challenges with this approach. First, the sensitivity (5.1) is not efficiently computable because it requires to optimize over all k-subsets $C \subset \mathbb{R}^d$. Second, one has to determine the number of samples N (essentially the coreset size) based on a probabilistic analysis of the event that S is a coreset. Prior work on coresets has studied these issues extensively and developed a general framework, and we shall use the variant stated in Theorem 5.3.2 below. This framework only needs an approximation to the sensitivities $\{\sigma_x^\star\}_{x \in X}$, more precisely it requires overestimates $\sigma_x \geq \sigma_x^\star$ whose sum $\sum_{x \in X} \sigma_x$ is bounded. Moreover, it relates the number of samples N to a quantity called the *weighted shattering dimension* $\mathrm{sdim}_{\max}$, which roughly speaking measures the complexity of a space (set of points) by the number of distinct ways that metric balls can intersect it. The definition below has an extra complication of a point weight v, which originates from the weight in the importance sampling procedure, and thus we need a uniform upper bound, denoted $\mathrm{sdim}_{\max}$, over all possible weights.[1]

[1] In principle, this uniform upper bound is not necessary, and an upper bound for weights corresponding to the importance score suffices, but a uniform upper bound turns out to be technically easier to deal with.

Definition 5.3.1 (Shattering dimension)**.** Given a *weight* function $v : \mathbb{R}^d_? \to \mathbb{R}_+$, let $\mathrm{sdim}_v(\mathbb{R}^d_?)$ be the smallest integer t such that

$$\forall H \subset \mathbb{R}^d_?, |H| \geq 2 \qquad \left|\left\{B^H_v(c,r) : c \in \mathbb{R}^d, r \geq 0\right\}\right| \leq |H|^t,$$

where $B^H_v(c,r) := \{x \in H : v(x) \cdot \mathrm{dist}(x,c) \leq r\}$. Moreover, let $\mathrm{sdim}_{\max}(\mathbb{R}^d_?) := \sup_{v:\mathbb{R}^d_? \to \mathbb{R}_+} \mathrm{sdim}_v(\mathbb{R}^d_?)$.

Strictly speaking, Theorem 5.3.2 has been proposed and proved only for metric spaces, but the proof is applicable also in our setting (where dist need not satisfy the triangle inequality), because it only concerns the *binary* relation between data points and center points (without an indirect use of a third point, e.g., by triangle inequality.)

Theorem 5.3.2 ([41][2])**.** *Let $X \subset \mathbb{R}^d_?$ be a data set, and let $k, z \geq 1$. Consider the importance sampling procedure with importance scores that satisfy $\sigma_x \geq \sigma^\star_x$ for all $x \in X$, and with a sufficiently large number of samples*

$$N = \tilde{O}\left(\epsilon^{-2} k z^z \,\mathrm{sdim}_{\max}(\mathbb{R}^d_?) \sum_{x \in X} \sigma_x\right).$$

Then with constant probability it reports an ϵ-coreset for (k,z)-Clustering.

Proof of Theorem 5.3.1. Because of Theorem 5.3.2, it suffices to bound $\mathrm{sdim}_{\max}(\mathbb{R}^d_?)$, and to provide an efficient algorithm to estimate σ_x whose sum is bounded. These two components are provided in Lemma 5.3.3 and Lemma 5.3.4 stated below (their proofs appear in Sections 5.3.1 and 5.3.2), Plugging these two lemmas into Theorem 5.3.2, the main theorem follows. □

Lemma 5.3.3 (Shattering dimension bound)**.** $\mathrm{sdim}_{\max}(\mathbb{R}^d_?) = O(d)$.

[2]Our theorem statement is based on [41, Theorem 31], adapted to our context. One difference is that their theorem is about VC-dimension, but it is also applicable for shattering dimension. Another difference is that we use a more direct terminology that is specialized to metric balls in $\mathbb{R}^d_?$ instead of a general range space.

Lemma 5.3.4. *There is an algorithm that, given a data set $X \subset \mathbb{R}^d_?$ of n j-points, for (k,z)-*CLUSTERING *computes importance scores $\{\sigma_x\}_{x\in X}$ such that with constant probability,*

- $\sigma_x \geq \sigma^\star_x$ *for all* $x \in X$*; and*
- $\sum_{x\in X} \sigma_x \leq O\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-z-1}} \cdot \sqrt{d^z \cdot \log^{z+2} n}\right)$,

and its running time is $\tilde{O}\left(\frac{(j+k)^{j+k+2}}{j^j k^{k-2}} \cdot nd\right)$.

5.3.1 Proof of Lemma 5.3.3: Shattering Dimension of $\mathbb{R}^d_?$

We now prove Lemma 5.3.3, which asserts that $\mathrm{sdim}_{\max}(\mathbb{R}^d_?) = O(d)$. We remark that the shattering dimension bound for $\mathbb{R}^d$ without missing values has been proved in [13, Lemma 16.1] and our proof is actually an extension of it.

Proof of Lemma 5.3.3. Let us verify Definition 5.3.1. Consider $H \subset \mathbb{R}^d_?$ and a weight function $v : \mathbb{R}^d_? \to \mathbb{R}_+$. Recall that given $c \in \mathbb{R}^d$ and $r \geq 0$, we have $B^H_v(c,r) = \{h \in H : v(h) \cdot \mathrm{dist}(h,c) \leq r\}$ and $\mathrm{dist}(h,c)^2 = \sum_{i\in I_h}(h_i - c_i)^2$ for $h \in H$. We need to show that

$$\left|\{B^H_v(c,r) : c \in \mathbb{R}^d, r \geq 0\}\right| \leq |H|^{O(d)}. \tag{5.2}$$

Observe that

$$\begin{aligned} h \in B^H_v(c,r) &\iff v(h) \cdot \mathrm{dist}(h,c) \leq r \\ &\iff -r^2 + \sum_{i\in I_h}(v^2(h)h_i^2 + v^2(h)c_i^2 - 2v^2(h)h_i c_i) \leq 0. \end{aligned}$$

Next, we write this inequality in an alternative way, that separates terms depending h from those depending on c and r, more precisely as an inner-product $\langle f(h), g(c,r)\rangle \leq 0$ for vectors $f(h), g(c,r) \in \mathbb{R}^{3d+1}$. Now consider $f : H \to \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}$ and

$g : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}$ such that $f(h) = (p, q, t, -1)$, where $p, q, t \in \mathbb{R}^d$ and for $i \in [d]$

$$p_i = \begin{cases} v^2(h) \cdot h_i^2 & \text{if } i \in I_h \\ 0 & \text{otherwise} \end{cases} \qquad q_i = \begin{cases} v^2(h) & \text{if } i \in I_h \\ 0 & \text{otherwise} \end{cases}$$

$$t_i = \begin{cases} -2v^2(h) \cdot h_i & \text{if } i \in I_h \\ 0 & \text{otherwise} \end{cases}$$

and $g(c, r) = (y, z, w, r^2)$, where $y, z, w \in \mathbb{R}^d$, $y_i = 1, z_i = c_i^2, w_i = c_i$ for $i \in [d]$. Then we have

$$h \in B_v^H(c, r) \iff \langle f(h), g(c, r) \rangle \leq 0.$$

For a vector $t \in \mathbb{R}^{3d+1}$, let $\text{proj}_-^H(t) := \{h \in H : \langle f(h), t \rangle \leq 0\}$ be the subset of H that has nonpositive inner-product with t (it can be viewed also as projection or a halfspace). Therefore, by (5.2), we have

$$\begin{aligned} \left|\{B_v^H(c, r) : c \in \mathbb{R}^d, r \geq 0\}\right| &= \left|\{\text{proj}_-^H(g(c, r)) : c \in \mathbb{R}^d, r \geq 0\}\right| \\ &\leq \left|\{\text{proj}_-^H(t) : t \in \mathbb{R}^{3d+1}\}\right|. \end{aligned}$$

We observe that

$$\left|\{\text{proj}_-^H(t) : t \in \mathbb{R}^{3d+1}\}\right| \leq |H|^{O(d)},$$

since this may be related to the shattering dimension of halfspaces in $\mathbb{R}^{3d+1}$, which is $O(d)$ and is a well-known fact in the PAC learning theory (cf. [100, Chapter 7.2]). This concludes the proof of Lemma 5.3.3. □

5.3.2 Proof of Lemma 5.3.4: Estimating Sensitivity Efficiently

We use a technique introduced by Varadarajan and Xiao [92] that reduces the sensitivity-estimation problem to the problem of constructing a coreset for k-Center clustering. This coreset concept is defined as follows.

Definition 5.3.2. An *α-coreset for k-Center* of a data set $X \subset \mathbb{R}^d_?$ is a subset $Y \subseteq X$ such that

$$\forall C \subset \mathbb{R}^d, |C| = k, \qquad \max_{x \in X} \operatorname{dist}(x, C) \leq \alpha \cdot \max_{y \in Y} \operatorname{dist}(y, C).$$

Note that the error parameter α represents a multiplicative factor, which is slightly different from that of ϵ in ϵ-coreset for (k, z)-Clustering, and roughly corresponds to $\alpha = 1 + \epsilon$. The reasoning is that $\max_{y \in Y} \operatorname{dist}(y, C)$ for $Y \subseteq X$ is always no more than $\max_{x \in X} \operatorname{dist}(x, C)$, and therefore we only need to measure the contraction-side error.

The reduction in Lemma 5.3.5 was presented in [92], and we restate its algorithmic steps in Algorithm 6. This needs access to some Algorithm $\mathcal{A}$ that constructs an α-coreset for k-Center on a point set $X \subset \mathbb{R}^d_?$. Each iteration i calls Algorithm $\mathcal{A}$ to construct a k-Center coreset for the current point set X (which is initially the entire data set), assign sensitivity estimates $O(\alpha^z / i)$ to every coreset point, and then remove these coreset points from X. These iterations are repeated until X is empty.

Algorithm 6 Sensitivity estimation from [92, Lemma 3.1] for data set $X \subset \mathbb{R}^d_?$

Require: algorithm $\mathcal{A}$ that constructs α-coreset for k-Center

```
i ← 1
while X ≠ ∅ do
    P ← A(X)
    for x ∈ P do
        σ_x ← O(α^z / i)
    end for
    X ← X \ P
    i ← i + 1
end while
```

Lemma 5.3.5 ([92, Lemma 3.1])**.** *Suppose algorithm $\mathcal{A}$ constructs an α-coreset of size $T = T(\alpha, d, j, k)$ for k-Center an input $X \subset \mathbb{R}^d_?$. Then Algorithm 6 (which makes calls to this Algorithm $\mathcal{A}$) computes sensitivities $\{\sigma_x\}$ for (k, z)-Clustering satisfying that $\sigma_x \geq \sigma^\star_x$ for all $x \in X$, and $\sum_{x \in X} \sigma_x \leq \alpha^z \cdot T \log |X|$.*

However, there are two outstanding technical challenges. First, there is no known construction of a small k-Center coreset for our clustering with missing values setting. Moreover, as can be seen from Algorithm 6, this reduction executes the k-Center coreset construction $\frac{|X|}{T}$ times (where T is the size of the coreset as in Lemma 5.3.5), and when using a naive implementation of the k-Center coreset construction, which naturally requires $\Omega(|X|)$ time, results overall in quadratic time, which is not very efficient.

First, to deal with question marks, we employ a certain family $\mathcal{I}$ of subset of coordinates (so each $I \in \mathcal{I}$ is a subset of $[d]$), and we *restrict* the data set X on each $I \in \mathcal{I}$. Each restricted data set (restricted on some I) may be viewed as a data set in $\mathbb{R}^I$, without any question marks. We show that the union of k-Center coresets on all restricted data sets with respect all to $I \in \mathcal{I}$, forms a valid k-Center coreset for X (which has question marks), provided that the family $\mathcal{I}$ has a certain combinatorial property. Naturally, the size of this coreset for X depends on an upper bound on $|\mathcal{I}|$.

Second, since the choice of family $\mathcal{I}$ is oblivious to the data set, it suffices to design an efficient algorithm for k-Center coreset for any restricted data set. We observe that the efficiency bottleneck in Algorithm 6 is the repeated invocation of Algorithm $\mathcal{A}$ to construct a coreset, even though its input changes only a little between consecutive invocations. Hence, we design a dynamic algorithm, that maintains a k-Center coreset on the restricted data sets under point updates. Our algorithm may be viewed as a variant of Gonzalez's algorithm [93], and we maintain it efficiently by a random projection idea that was used e.g. in [94]. In particular, we "project" the data points onto several one-dimensional lines in $\mathbb{R}^d$, and we maintain an interval data structure (that is based on balanced trees) to dynamically maintain the result of our variant of Gonzalez's algorithm. We summarize the dynamic algorithm in the following lemma.

Lemma 5.3.6. *There is a randomized dynamic algorithm with the following guarantees. The input is a dynamic set $X \subset \mathbb{R}^d_?$ of j-points, such that X undergoes*

q adaptive updates (point insertions and deletions) and the points ever added are fixed in advance (non-adaptively). The algorithm maintains per update in time $\tilde{O}\left(\frac{(j+k)^{j+k+1}}{j^j k^k} \cdot (j + k \log q)(d + k^2 \log q)\right)$, a subset $Y \subseteq X$ such that the size of Y is only $|Y| \leq O\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-1}} \cdot \log d\right)$ and with constant probability, Y is an $O(k\sqrt{d \log q})$-coreset for k-Center on X after every update.

The proof of the lemma can be found in Section 5.3.3, and here we proceed to the proof of Lemma 5.3.4.

Proof of Lemma 5.3.4. We plug in the dynamic algorithm in Lemma 5.3.6 as $\mathcal{A}$ in Lemma 5.3.5. Specifically, line 3 and 7 of Algorithm 6 are replaced by the corresponding query and update procedure. Since $|X| = n$, and each point is inserted and deleted for exactly once, algorithm 6 needs $q = O(n)$ insertions and deletions of points. Moreover, the set of points ever added is just X which is fixed. Thus, α is replaced by $O(k\sqrt{d \log n})$ and T is replaced by $O\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-1}} \cdot \log d\right)$. Therefore, for (k, z)-Clustering, this computes σ_x for $x \in X$ such that $\sigma_x \geq \sigma_x^\star$, and that

$$\sum_{x \in X} \sigma_x \leq \alpha^z \cdot T \cdot \log n = O\left(\frac{(j+k)^{j+k+1}}{j^j k^{k-z-1}} \cdot \sqrt{d^z \cdot \log^{z+2} n}\right).$$

The total running time is bounded by $\tilde{O}\left(\frac{(j+k)^{j+k+2}}{j^j k^{k-2}} \cdot nd\right)$ for implementing $O(n)$ updates. □

5.3.3 Proof of Lemma 5.3.6: Dynamic $O(1)$-Coresets for k-Center Clustering

As mentioned, the high level idea is to identify a collection $\mathcal{I}$ of subsets of coordinates (so each $I \in \mathcal{I}$ satisfies $I \subseteq [d]$), construct an α-coreset (a will be determined is the later context) Y_i for k-Center on the data set X with coordinates *restricted* on each $I_i \in \mathcal{I}$, and then the union $\bigcup_i Y_i$ would be the overall $\alpha\sqrt{d}$-coreset for k-Center on X. The exact definition of restricted data set goes as follows.

Definition 5.3.3. For a point $p \in \mathbb{R}^d_?$ and a subset $I \subseteq I_p$, define $p_{|I} \in \mathbb{R}^I$ in the obvious way, by selecting the coordinates $\{p_i\}_{i \in I}$. Define the *I-restricted data set* to be $X_{|I} := \{p_{|I} : p \in X, I \subseteq I_p\}$. Since each vector in $X_{|I}$ arises from a specific vector in X, a subset $Y \subseteq X_{|I}$ corresponds to a specific subset of X, and we shall denote this subset by Y^{-1}.

We observe that the metric space on the restricted data set becomes a usual metric space, i.e. it satisfies the triangle inequality, and can be realized as a point set in $\mathbb{R}^I$ which does not contain question marks. Therefore, this reduces our goal to constructing k-Center coresets for this usual data set. However, the size of the coreset yielded from this approach would depend on the size of the family $\mathcal{I}$. Hence, a key step is to identify a small set $\mathcal{I}$ such that the union of the coreset restricted on $\mathcal{I}$ is an accurate coreset. To this end, we consider the so-called (j, k, d)-family of coordinates as in Definition 5.3.4. This family itself is purely combinatorial, but we will show in Lemma 5.3.7 that such a family actually suffices for the accuracy of the coreset, and we show in Lemma 5.3.8 the existence of a small family.

Definition 5.3.4. A family of sets $\mathcal{I} \subset 2^{[d]}$ is called a (j, k, d)-family if for any $J, K \subset [d], J \cap K = \emptyset, |J| = j, |K| = k$, there exists an $I \in \mathcal{I}$ such that $I \cap J = \emptyset$ and $K \subset I$.

Lemma 5.3.7. *Suppose $\mathcal{I}$ is a (j, k, d)-family Let $X \subseteq \mathbb{R}^d_?$ be a set of j-points, and for every $I \in \mathcal{I}$, let Y_I be an α-coreset for k-Center on $X_{|I}$. Then $\cup_{I \in \mathcal{I}} Y_I^{-1}$ is an $\alpha\sqrt{d}$-coreset for k-Center on X.*

Proof. It suffices to show that for any center set $C = \{c^1, \ldots, c^k\} \subseteq \mathbb{R}^d$ with k points and $x \in X$, if $\mathrm{dist}(x, C) \geq r$ for some $r \geq 0$, then we can find a coreset point $y \in \cup_{I \in \mathcal{I}} Y_I^{-1}$ such that $\mathrm{dist}(y, C) \geq \frac{r}{\alpha\sqrt{d}}$.

For $i \in [k]$, let $t_i \in \arg\max_{t \in I_x} |x_t - c^i_t|$, i.e., t_i is the index of coordinate that contributes the most in distance $\mathrm{dist}(x, c^i)$, so $|x_{t_i} - c^i_{t_i}| \geq \frac{r}{\sqrt{d}}$. Let K be any k-subset

such that $K \subseteq I_x$ and $\{t_1, \ldots, t_k\} \subseteq K$. Since $\mathcal{I}$ is a (j,k,d)-family and $|I_x| \geq d - j$, by definition, there exists an $I \subseteq \mathcal{I}$ such that $K \subseteq I \subseteq I_x$. We note that

$$\mathrm{dist}(x_{|I}, C_{|I}) = \mathrm{dist}_I(x, C) = \min_{i \in [k]} \mathrm{dist}_I(x, c^i) \geq \min_{i \in [k]} \mathrm{dist}_K(x, c^i) \geq \min_{i \in [k]} |x_{t_i} - c^i_{t_i}| \geq \frac{r}{\sqrt{d}}.$$

Since $I \subseteq I_x$, we know that $x_{|I} \in X_{|I}$. As Y_I is an α-coreset for $X_{|I}$, we know that there exists $y \in Y_I^{-1}$ such that

$$\mathrm{dist}(y, C) \geq \mathrm{dist}_I(y, C) = \mathrm{dist}(y_{|I}, C_{|I}) \geq \frac{\mathrm{dist}(x_{|I}, C_{|I})}{\alpha} \geq \frac{r}{\alpha\sqrt{d}}.$$

□

Next, we show the existence of a small (j,k,d)-family. We remark that this combinatorial structure has been employed in designing fault-tolerant data structures and algorithms (cf. [101–103]). Similar bounds were obtained in their different contexts and languages, and here we provide a proof for completeness.

Lemma 5.3.8. *There is a (j,k,d)-family $\mathcal{I}$ of size $O\left(\frac{(j+k)^{j+k+1}}{j^j k^k} \log d\right)$. Moreover, there is a randomized algorithm that constructs $\mathcal{I}$ in time $O(d \cdot |\mathcal{I}|)$ with probability at least $1 - \frac{1}{d^{j+k}}$.*

Proof. Set $t = \frac{(j+k)^{j+k+1}}{j^j k^k} \cdot 2 \log d$. We add t random sets into $\mathcal{I}$ where each random set is generated by independently including each element of $[d]$ with probability $\frac{k}{j+k}$. For a set $J \subseteq [d], |J| = j$ and a set $K \subseteq [d], |K| = k$ such that $J \cap K = \emptyset$, the probability that a random set generated in the above way contains K but avoids J, is

$$\left(\frac{j}{j+k}\right)^j \cdot \left(\frac{k}{j+k}\right)^k.$$

Since there are at most d^{j+k} tuples of such J and K, by union bound and the choice of t, the probability that $\mathcal{I}$ is a (j,k,d)-family is at least

$$1 - d^{j+k}\left(1 - (\frac{j}{j+k})^j \cdot (\frac{k}{j+k})^k\right)^t \geq 1 - \frac{1}{d^{j+k}}$$

□

Gonzalez's algorithm yields k-Center coreset for restricted data set. Finally, the k-Center coreset for the restricted data set on each $I \in \mathcal{I}$ would be constructed using an approximate version of Gonzalez's algorithm [93]. We note that while Gonzalez's algorithm was originally designed as an approximation algorithm for k-Center, the approximate solution actually serves as a good coreset for k-Center (see Lemma 5.3.9). The assumption that the input forms a metric space is crucial in Lemma 5.3.9, and this is guaranteed since we run this variant of Gonzalez only on a restricted data set which satisfies the triangle inequality.

Lemma 5.3.9 (Approximate Gonzalez)**.** *Let (M, d) be a metric space. Let $A \subset M$ be a set of n points and consider the following variant of Gonzalez's greedy algorithm. Set $B = \{b_0\}$ for an arbitrary $b_0 \in A$. Repeat for k times, where each time we add a c-approximation of B's furthest point into B. Precisely, add $b_i \in A$ such that $c \cdot \mathrm{dist}(b_i, B) \geq \max_{a \in A} \mathrm{dist}(a, B)$ into B. Then B is a $(1+2c)$-coreset for k-Center on A.*

Proof. Fix a center set $C = \{c_1, \ldots, c_k\}$ with k points and let $r := \max_{b \in B} \mathrm{dist}(b, C)$. Then we have $\bigcup_{i=1}^{k} \mathrm{Ball}(c_i, r)$ covers B where $\mathrm{Ball}(x, r) = \{y : \mathrm{dist}(x, y) \leq r\}$ is the ball centered at x with radius r. It suffices to prove that $A \subseteq \bigcup_{i=1}^{k} \mathrm{Ball}(c_i, (2c+1)r)$.

Since k balls $B(c_1, r), \cdots, B(c_k, r)$ cover B and $|B| = k+1$, by pigeonhole principle, there exists $b_i, b_j \in B, i < j$ that are contained in a same ball $B(c_i, r)$. W.l.o.g., we assume $b_i, b_j \in B(c_1, r)$. Now fix $a \in A \setminus B$, since a has never been added into B, we have

$$\begin{aligned}
\mathrm{dist}(a, B) &\leq \mathrm{dist}(a, \{b_1, \ldots, b_{j-1}\}) \\
&\leq c \cdot \mathrm{dist}(b_j, \{b_1, \ldots, b_{j-1}\}) \\
&\leq c \cdot \mathrm{dist}(b_i, b_j) \\
&\leq c \cdot (\mathrm{dist}(b_i, c_1) + \mathrm{dist}(b_j, c_1)) \\
&\leq 2cr.
\end{aligned}$$

Thus $A \subseteq \bigcup_{i=1}^{k+1} \mathrm{Ball}(b_i, 2cr) \subseteq \bigcup_{i=1}^{k} \mathrm{Ball}(c_i, (2c+1)r)$. □

Dynamic implementation of Gonzalez's algorithm. To make this k-Center coreset construction dynamic, we adapt the random projection technique to Gonzalez's algorithm, so that it suffices to dynamically execute Gonzalez's algorithm on a set of one-dimensional lines in $\mathbb{R}^d$.

Random projection. We call a sample from the d-dimensional standard normal distribution $N(0, I_d)$ a d-dimensional *random vector* for simplicity. To implement (the variant of) Gonzalez's algorithm as in Lemma 5.3.9 in the dynamic setting, we project the point set to several random vectors and use one dimensional data structure to construct k-Center coreset in each of the one dimensional projected data set.

Note that the key step in Gonzalez's algorithm is the furthest neighbor search, and we would show that our projection method eventually yields an $O(k\sqrt{\log n})$-approximation of the furthest neighbor with high probability. The following two facts about normal distribution are crucial in our argument, and Lemma 5.3.12 is our main technical lemma.

Fact 5.3.10. *Let $u \in \mathbb{R}^d$ and let $v \sim N(0, I_d)$ be a random vector, then $\langle u, v/|u| \rangle \sim N(0,1)$.*

Fact 5.3.11. *Let $Z \sim N(0,1)$, then there exists some universal constant $c > 0$ such that $P[|Z| \leq \frac{1}{k}] \leq \frac{c}{k}$, and $P[|Z| \geq t] \leq e^{-c \cdot t^2}$ for any $t > 0$.*

Lemma 5.3.12. *Let $X \subset \mathbb{R}^d, |X| = n$ and $\mathcal{V}$ be a collection of $t = O(k \log n + \log \delta^{-1})$ random vectors in $\mathbb{R}^d$, then with probability $1 - \delta$, for every $C \subseteq X, |C| \leq k$ and every $x \in X$, there exists a vector $v \in \mathcal{V}$ such that (i) $|x \cdot v - c \cdot v| \geq \Omega(\frac{1}{k}) \cdot \|c - x\|_2$ for every $c \in C$ and (ii) $|a \cdot v - b \cdot v| \leq O(\sqrt{\log n}) \cdot \|a - b\|_2$ for every $a, b \in X$.*

Proof. Fix a subset $C \subseteq X, |C| \leq k$, a point x and a random vector v. For every $c \in C$, since $(c - x) \cdot v / \|c - x\|_2 \sim N(0,1)$, by Fact 5.3.11, the probability that $|c \cdot v - x \cdot v| \geq$

$\Omega(\frac{1}{k}) \cdot \|c-x\|_2$ is at least $1-\frac{1}{4k}$. For every $a, b \in X$, since $(a-b)\cdot v/\|a-b\|_2 \sim N(0,1)$, by Fact 5.3.11, the probability that $|a \cdot v - b \cdot v| \leq \sqrt{\log n}\|a-b\|_2$ is at most $\frac{1}{4n^2}$.

Since there are k choices of $c \in C$ and at most n^2 choices of $a, b \in X$, by union bound, with probability at least $1 - k \cdot \frac{1}{4k} - n^2 \cdot \frac{1}{4n^2} = \frac{1}{2}$, the following two events hold, (i) $|x\cdot v - c\cdot v| \geq \Omega(\frac{1}{k}) \cdot \|c-x\|_2$ for every $c \in C$ and (ii) $|a\cdot v - b\cdot v| \leq O(\sqrt{\log n}) \cdot \|a-b\|_2$ for every $a, b \in X$.

Now since $\mathcal{V}$ contains t random vectors, the probability that there exists one vector $v \in \mathcal{V}$ that satisfies (i) and (ii) is at least $1 - \frac{1}{2^t}$.

Finally, by union bound, since there are at most $(n+1)^{k+1}$ choices of $C \subseteq X, |C| = k$ and $x \in X$, the probability such that for every C and x, there exists $v \in \mathcal{V}$ such that (i) and (ii) happen is at least $1 - \frac{(n+1)^{k+1}}{2^t} \geq 1 - \delta$. □

In the next lemma, we present a dynamic algorithm that combines the random projection idea with a one-dimensional data structure. This combining with the (j, k, d)-family idea would immediately imply Lemma 5.3.6.

Lemma 5.3.13. *There is a dynamic algorithm that for every $P \subseteq \mathbb{R}^m$ subject to at most q adaptive point insertions and deletions where the set of points ever added is fixed in advance, maintains set $Q \subseteq P$ with $|Q| \leq k+1$ such that with probability at least $1-\delta$, Q is an $O(k\sqrt{\log q})$-coreset for k-Center on P after every update, in time $O\Big((k^2 \log q + m)(k \log q + \log \delta^{-1})\Big)$ per update.*

Proof of Lemma 5.3.6. The dynamic algorithm is initialized by using Lemma 5.3.8 to construct a (j, k, d)-family $\mathcal{I}$ of size $O\left(\frac{(j+k)^{j+k+1}}{j^j k^k} \log d\right)$, and then for each $I \in \mathcal{I}$, we create a data structure $\mathcal{D}_I$ (that maintains $X|_I$), using Lemma 5.3.13 with failure probability $\delta := \Theta\left(\frac{1}{|\mathcal{I}|}\right)$. Upon each update of X, we update each $\mathcal{D}_I$, and obtain the maintained coreset Y_I from $\mathcal{D}_I$.

Analysis. Since we pick $\delta = \Theta\left(\frac{1}{|\mathcal{I}|}\right)$ for all $\mathcal{D}_I$'s, with constant probability all data structures $\mathcal{D}_I$'s succeed simultaneously. The running time follows immediately from

Lemma 5.3.8 and Lemma 5.3.13. The coreset accuracy follows from Lemma 5.3.7 and Lemma 5.3.13 (noting that we need to suffer a $\sqrt{d}$ factor because of Lemma 5.3.7). □

Proof of Lemma 5.3.13. We assume there is a data structure $\mathcal{T}$ that maintains a set of real numbers and supports the following operations, all running in $O(\log n)$ time where n is the number of elements currently present in the structure.

- $\textsc{Remove}(x)$: Remove an element x from the structure.
- $\textsc{Add}(x)$: Add an element x to the structure.
- $\textsc{UpperBound}(x)$: Return the largest element that is at most x.
- $\textsc{LowerBound}(x)$: Return the smallest element that is at least x.

Note that such $\mathcal{T}$ may be implemented by using a standard balanced binary tree.

Furthest point query. We also need $\textsc{Furthest}(C)$ query, where $C \subset \mathbb{R}$ and it asks for an element x that has the largest distance to C (and it should return an arbitrary element if $C = \emptyset$). This $\textsc{Furthest}(C)$ can be implemented by using $O(|C|)$ many $\textsc{UpperBound}$ and $\textsc{LowerBound}$ operations, which then takes $O(|C| \log n)$ time in total. To see this, assume $C = \{c_1, \ldots, c_k\}$ where $c_1 \leq \ldots \leq c_k$ then the clusters partitoned by C is $(-\infty, \frac{1}{2}(c_1 + c_2)], (\frac{1}{2}(c_1 + c_2), \frac{1}{2}(c_2 + c_3)], \cdots, (\frac{1}{2}(c_{k+1} + c_k), +\infty)$ and we can find the potential furthest points in each cluster by querying the following,

$$\begin{aligned}
&\textsc{UpperBound}(-\infty), \textsc{LowerBound}\left(\frac{1}{2}(c_1 + c_2)\right), \\
&\textsc{UpperBound}\left(\frac{1}{2}(c_1 + c_2)\right), \textsc{LowerBound}\left(\frac{1}{2}(c_2 + c_3)\right) \\
&\ldots \\
&\textsc{UpperBound}\left(\frac{1}{2}(c_{k+1} + c_k)\right), \textsc{LowerBound}(+\infty)
\end{aligned}$$

and the furthest point to C among the above $2k = O(|C|)$ many points is what we seek for.

The dynamic algorithm is presented in Algorithm 7. The algorithm samples a set of independent random vectors $\mathcal{V}$ (in a data oblivious way), then creates an above-mentioned interval structure $\mathcal{T}_v$ for each $v \in \mathcal{V}$. When we insert/delete a point x, the update is performed on every $\mathcal{T}_v$ with the projection $\langle x, v\rangle$. The coreset for the current data set P can be computed on the fly by simulating the Gonzalez's algorithm. In particular, this is where the Furthest query is used, and we find an approximate furthest point in P by taking the furthest point in each $\mathcal{T}_v$, and select the one that is the relative furthest in P.

Algorithm 7 Dynamic Gonzalez's algorithm

procedure INIT ▷ initialize an empty structure
  $l \leftarrow O(k \log q + \log \delta^{-1})$, and draw l independent random vectors in $\mathbb{R}^m$, denotes as $\mathcal{V}$
  initialize $\mathcal{T}_v$ for each $v \in \mathcal{V}$
end procedure
procedure UPDATE(x)
  insert/delete $\langle x, v\rangle$ for each $v \in \mathcal{V}$
end procedure
procedure GET-CORESET(k)
  $Q \leftarrow \emptyset$
  for $i = 1, \ldots, k+1$ **do**
    for $v \in \mathcal{V}$, let $x_v \in P$ satisfy $\langle x_v, v\rangle = \mathcal{T}_v$.FURTHEST($\langle Q, v\rangle$)
      ▷ where $\langle Q, v\rangle := \{\langle x, v\rangle : x \in Q\}$
    $v^\star \leftarrow \arg\max_{v \in \mathcal{V}} \operatorname{dist}(x_v, Q)$
    $Q \leftarrow Q \cup \{x_{v^\star}\}$
  end for
  return Q
end procedure

Analysis. Let A be the set of points ever added, so $|A| \le q$. Recall that A is fixed in advance. By applying Lemma 5.3.12 in A, we know that with probability $1 - \delta$, the following event $\mathcal{E}$ happens. For every $C \subseteq A, |C| \le k$, every $x \in A$, there exists $v \in \mathcal{V}$, such that

(i) $|\langle c - x, v\rangle| \ge \Omega(\frac{1}{k}) \cdot \|x - c\|_2$ for every $c \in C$, and

(ii) $|\langle a - b, v\rangle| \le O(\sqrt{\log q}) \cdot \|a - b\|_2$ for every $a, b \in A$.

Now condition on $\mathcal{E}$. Suppose the current point set is P. Suppose we run the Get-Coreset subroutine and we query $\mathcal{T}_v$.Furthest($\langle Q, v\rangle$) for some v and Q. Suppose $x \in P \subseteq A$ is the current furthest point to Q. Because of $\mathcal{E}$, there exists a vector $v \in \mathcal{V}$ such that (i) and (ii) hold. By (i), we have that $\mathrm{dist}(\langle x, v\rangle, \langle Q, v\rangle) \geq \Omega(\frac{1}{k}) \cdot \mathrm{dist}(x, Q)$. By (ii), we know that for any $p \in P$ and $c \in Q$, $|\langle p - c, v\rangle| \leq O(\sqrt{\log q})\|p - c\|_2$, so $\mathrm{dist}(\langle p, v\rangle, \langle Q, v\rangle) \leq O(\sqrt{\log q}) \cdot \mathrm{dist}(p, Q)$. So if $\mathcal{T}_v$.Furthest($\langle Q, v\rangle$) returns an answer $\langle p, v\rangle$, we know that

$$\mathrm{dist}(p, Q) \geq \frac{\mathrm{dist}(\langle p, v\rangle, \langle Q, v\rangle)}{O(\sqrt{\log q})} \geq \frac{\mathrm{dist}(\langle x, v\rangle, \langle Q, v\rangle)}{O(\sqrt{\log q})} \geq \Omega\left(\frac{1}{k\sqrt{\log q}}\right) \cdot \mathrm{dist}(x, Q).$$

Thus, p is an $O(k\sqrt{\log q})$-approximation of the furthest point to Q. This combining with Lemma 5.3.9. implies the error bound.

Running time. For the running time, we note that for each update of P, we need to update $\mathcal{T}_v$ for each $v \in \mathcal{V}$ accordingly. Thus we need to pay $O(lm)$ time (recalling that $l = O(k \log q + \log \delta^{-1})$ was defined in Algorithm 7) to compute all the inner products and $O(l \log q)$ time to update all $\mathcal{T}_v$'s. The main loop in Get-Coreset requires $O(kl)$ many Furthest($\cdot$) queries and this runs in $O(k^2 l \log q)$ time in total. In conclusion, the running time of each update (and maintaining coreset) is bounded by

$$O\left((k^2 \log q + m) \cdot l\right) = O\Big((k^2 \log q + m)(k \log q + \log \delta^{-1})\Big).$$

□

Chapter 6

The Lower Bound

6.1 Coresets for Clustering in Graphs of Bounded Treewidth

We present an $\Omega(\frac{k}{\epsilon} \cdot \mathrm{tw}(G))$ lower bound for clustering in graphs, which matches the linear dependence on $\mathrm{tw}(G)$ of our coreset construction in Chapter 2. Previously, no lower bounds for k-Median were known. In fact, even the $O(\log n)$ factor for general metrics was not justified. Since our hard instance in Theorem 6.1.1 consists of $O(\frac{k}{\epsilon} \cdot 2^t)$ vertices, it readily implies for the first time that the $O(\log n)$ factor is optimal for general metrics (see Corollary 6.1.5).

Our lower bound is actually split into two theorems: one for the tree case ($\mathrm{tw}(G) = 1$) and one for the other cases ($\mathrm{tw}(G) \geq 2$). Ideally, we would use a unified argument, but unfortunately the general argument for $\mathrm{tw}(G) \geq 2$ does not apply in the special case $\mathrm{tw}(G) = 1$ because some quantity is not well defined, and we thus need to employ a somewhat different argument for the tree case.

Theorem 6.1.1 (Lower Bound for Graphs with Treewidth ≥ 2)**.** *For every $0 < \epsilon < 1$ and integers $t, k \geq 1$, there exists an unweighted graph $G = (V, E)$ with $\mathrm{tw}(G) \leq t + 1$, such that any ϵ-coreset for k-Median on data set $X = V$ has size $\Omega(\frac{k}{\epsilon} \cdot t)$.*

Proof. The vertex set of $G_{k,\epsilon}$ is defined as $L \cup R \cup \{u_0\}$. Let $m := \frac{k}{\epsilon}$. Both L

and R consist of m groups, i.e. $L := \bigcup_{i=1}^{m} L_i$ and $R := \bigcup_{i=1}^{m} R_i$. For $i \in [m]$, L_i consists of t elements, and R_i consists of 2^t elements. Let $L_i := \{l_j^{(i)} : j \in [t]\}$, and $R_i := \{r_J^{(i)} : J \subseteq [t]\}$. Since $t \geq 1$, L is non-empty. Define a special connection point u_0 to which all points of $L \cup R$ connect to (the specific way of connection is defined in the next paragraph).

The edge set is defined as follows. All edges are of weights 1. Connect all points in $L \cup R$ to u_0. For each $i \in [m]$, for each $l_j^{(i)} \in L_i$ and $r_J^{(i)} \in R_i$, if $j \in J$, add an edge $\{l_j^{(i)}, r_J^{(i)}\}$. Finally, let $T = \Omega(\frac{m}{k} \cdot 2^t)$, and make $T - 1$ copies of each point in L, which we call shadow vertices: for each $l_j^{(i)}$, create $T - 1$ vertices, and connect them to $l_j^{(i)}$ directly (so they form a star with center $l_j^{(i)}$).

Fact 6.1.2. *All distances in $G_{k,\epsilon}$ are 2, except that the distances between L_i and R_i ($i \in [m]$) are 1.*

For simplicity, we use G to represent $G_{k,\epsilon}$ in the following.

Treewidth Analysis: First, consider removing u_0 from G, and define the resultant graph as G'. Then $\mathrm{tw}(G) \leq \mathrm{tw}(G') + 1$. Observe that G' has m components: $\{L_i \cup R_i : i \in [m]\}$, so it suffices to bound the treewidth for each component. For each such component, since removing L_i makes all points in the component isolated, we conclude that the treewidth of the component is at most $|L_i| = t$. Therefore, we conclude that $\mathrm{tw}(G) \leq t + 1$.

Error Analysis: Suppose $D \subseteq V$ (with weight w) is an $O(\epsilon)$-coreset of size $o(\frac{k}{\epsilon} \cdot t)$. By manipulating the weight w, we assume w.l.o.g. that D does not contain the shadow vertices. Pick any k-subset $S \subseteq [m]$, such that for every $i \in S$, $|D \cap L_i| \leq \frac{t}{2}$ and $|D \cap R_i| \leq \frac{t}{2}$. Such S must exist, since otherwise there would be $m - k = \Omega(\frac{k}{\epsilon})$ number of i's, such that $|D \cap L_i| + |D \cap R_i| > \frac{t}{2}$, which contradicts $|D| = o(\frac{k}{\epsilon} \cdot t)$.

We would then pick two subsets $P_i, Q_i \subseteq [t]$ for each $i \in S$, which correspond to two points in R_i and encode two subsets of L_i, as in the following claim.

Claim 6.1.3. *For each $i \in S$, there exists $P_i, Q_i \subseteq [t]$, such that*

1. *If $l_j^{(i)} \in D \cap L_i$, then $j \in P_i$ and $j \in Q_i$.*
2. *If $r_J^{(i)} \in D \cap R_i$, then $J \neq P_i$ and $J \neq Q_i$.*
3. *$|P_i| \leq |D \cap L_i| + O(1)$, and $|Q_i| \geq t - O(1)$.*

Proof. Suppose $D \cap R_i = \{r_{J_1}^{(i)}, \ldots, r_{J_s}^{(i)}\}$, and let $\mathcal{J} = \{J_1, \ldots, J_s\}$. Find the minimum cardinality P_i such that item 1 and 2 holds: this is equivalent to find the smallest P', such that $(D \cap L_i) \cup P' \notin \mathcal{J}$. Such P' may be found in a greedy way: try out all 0-subsets, 1-subsets, ..., until $(D \cap L_i) \cap P' \notin \mathcal{J}$. Since $|D \cap R_i| \leq \frac{t}{2}$ and $|D \cap L_i| \leq \frac{t}{2}$, such greedy procedure must end after trying out $O(1)$-subsets and hence $|P_i| \leq |D \cap L_i| + O(1)$.

Let Q_i denote the set with the maximum cardinality such that item 1 and 2 holds. By a similar argument, we can prove that $|Q_i| \geq t - O(1)$. □

Based on this claim, we define $C_1 := \{r_{P_i}^{(i)} : i \in S\}$, and $C_2 := \{r_{Q_i}^{(i)} : i \in S\}$. Observe that the cost on both C_1 and C_2 are the same on the coreset D (by item 1). However, the objective on C_1 and C_2 differ by an $\Omega(\epsilon)$ factor (where we use item 2 and 3). To see it,

$$\begin{aligned}
\mathrm{cost}(V, C_1) =& \underbrace{2}_{u_0} + \underbrace{2(m-k) \cdot T \cdot t}_{\text{cost of } L \setminus L_i} + \underbrace{2(m \cdot 2^t - k)}_{\text{cost of } R} + \underbrace{T \cdot \sum_{i \in S} 2(t - |P_i|) + |P_i|}_{\text{cost of } L_i} \\
=& 2 + 2(m-k) \cdot T \cdot t + 2(m \cdot 2^t - k) + 2ktT - (2-1)T \cdot \sum_{i \in S} |P_i| \\
\geq& 2 + 2(m-k) \cdot T \cdot t + 2(m \cdot 2^t - k) + 2ktT - kT \cdot (\frac{t}{2} + O(1)).
\end{aligned}$$

Similarly,

$$\begin{aligned}
\mathrm{cost}(V, C_2) =& 2 + 2(m-k) \cdot T \cdot t + 2(m \cdot 2^t - k) + T \cdot \sum_{i \in S} 2(t - |Q_i|) + |Q_i| \\
\leq& 2 + 2(m-k) \cdot T \cdot t + 2(m \cdot 2^t - k) + 2ktT - kT \cdot (t - O(1))
\end{aligned}$$

So,

$$\begin{aligned}\frac{\text{cost}(V, C_1)}{\text{cost}(V, C_2)} &\geq 1 + \frac{kT \cdot (\frac{t}{2} - O(1))}{2 + 2(m-k) \cdot T \cdot t + 2(m \cdot 2^t - k) + 2ktT - kT \cdot (t - O(1))} \\ &\geq 1 + \Omega(\epsilon)\end{aligned}$$

where the last inequality is by $m = \frac{k}{\epsilon}$ and $T = \Omega(\frac{m}{k} \cdot 2^t)$. This contradicts the fact that D is an $O(\epsilon)$-coreset. □

Then we prove for the special case with treewidth 1, which is the tree case.

Theorem 6.1.4 (Lower Bound for Star Graphs)**.** *For every $0 < \epsilon < 1/3$ and integer $k \geq 1$, there exists an (unweighted) start graph $G = (V, E)$ with $|V| = O(\frac{k}{\epsilon})$ such that any ϵ-coreset for k-MEDIAN on data set $X = V$ has size $\Omega(\frac{k}{\epsilon})$.*

Proof. Denote the root node of the star graph $G = (V, E)$ by r and leaf nodes by $x_1, \ldots, x_n$ ($n \geq \frac{100k}{\epsilon}$). Suppose $D \subseteq V$ (with weight w) is an ϵ-coreset of size $o(\frac{k}{\epsilon})$. Let $W = \sum_{x \in D \setminus \{r\}} w(x)$. Consider a k-center set where all centers are on r. We have that

$$W = \sum_{x \in D} w(x) \cdot d(x, r) \geq (1 - \epsilon) \cdot \text{cost}(X, r) = (1 - \epsilon) \cdot n, \tag{6.1}$$

where the inequality is from the fact that D is an ϵ-coreset.

Next, we construct two center sets C_1 and C_2. Let $C_1 \subseteq V \setminus (D \cup \{r\})$ be a collection of k distinct leaf nodes that are not in D. Let C_2 be the collection of k nodes in $D \setminus \{r\}$ with largest weights. By construction, we have that

$$W' = \sum_{x \in C_2} w(x) \geq \frac{kW}{|D|} \geq 100\epsilon W \geq 50\epsilon n, \tag{6.2}$$

where the last inequality is by Inequality (6.1). Moreover, since D is an ϵ-coreset, we have that

$$2W + w(r) = \text{cost}(D, C_1) \leq (1 + \epsilon) \cdot \text{cost}(V, C_1) \leq 2 \cdot (2n + 1). \tag{6.3}$$

By symmetry, $\mathrm{cost}(V, C_1) = \mathrm{cost}(V, C_2)$. Then by the definition of coreset, we have

$$\frac{\mathrm{cost}(D, C_1)}{\mathrm{cost}(D, C_2)} = \frac{2W + w(r)}{2(W - W') + w(r)} \leq \frac{1+\epsilon}{1-\epsilon}.$$

However, by Inequalities (6.2) and (6.3), we have

$$\begin{aligned}
\frac{2W + w(r)}{2(W - W') + w(r)} &\geq \frac{2W + w(r)}{2W + w(r) - 50 \cdot 2\epsilon n} && \text{(Ineq. (6.2))} \\
&\geq \frac{2 \cdot (2n+1)}{2 \cdot (2n+1) - 100\epsilon n} && \text{(Ineq. (6.3))} \\
&> \frac{1+\epsilon}{1-\epsilon},
\end{aligned}$$

which is a contradiction. This completes the proof. □

Combining Theorems 6.1.4 and 6.1.1, we obtain a lower bound of $\Omega(\frac{k}{\epsilon} \cdot \mathrm{tw}(G))$ for the coreset size. Moreover, we observe that the hard instance in Theorem 6.1.1 has $O(2^t)$ nodes, which in fact implies an $\Omega(\log n)$ size lower bound for general graphs. We state this corollary as follows.

Corollary 6.1.5. *For every $0 < \epsilon < 1$ and integers $n, k \geq 1$, there exists an unweighted graph $G = (V, E)$ with $|V| = O(n)$ such that any ϵ-coreset for k-Median on data set $X = V$ has size $\Omega(\frac{k}{\epsilon} \cdot \log n)$.*

6.2 Coresets for Ordered Weighted Clustering

In this section we show that the size of a simultaneous coreset for Ordered k-Median, and in fact even for the special case p-Centrum, must grow with n, even for $k = d = 1$. More precisely, we show that it must depend at least logarithmically on n, and therefore our upper bound in Theorem 4.4.6 is nearly tight with respect to n.

Theorem 6.2.1. *For every (sufficiently large) integer n and every $n^{-1/3} < \epsilon < 1/2$, there exists an n-point set $X \subset \mathbb{R}$, such that any simultaneous ϵ-coreset of X for p-Centrum with $k = 1$ has size $\Omega(\epsilon^{-1/2} \log n)$.*

While a simultaneous coreset preserves the objective value for all possible centers (in addition to all $p \in [n]$), our proof shows that even *one specific* center already requires $\Omega(\log n)$ size. Our proof strategy is as follows. Suppose D is a simultaneous ϵ-coreset for Ordered k-Median on $X \subset \mathbb{R}$ with $k = 1$, and let $c \in \mathbb{R}$ be some center to be picked later. Since D is a simultaneous coreset for Ordered k-Median, it is in particular a coreset for p-Centrum problems for all $p \in [n]$. Let $W_X(p) := \mathrm{cost}_p(X, c)$ be the cost as a function of p, and let $W_D(p)$ be similarly for the coreset D, when we view X, D and the center c as fixed. It is easy to verify that $W_D(\cdot)$ is a piece-wise linear function with only $O(|D|)$ pieces. Now since D is a simultaneous ϵ-coreset, the function $W_D(\cdot)$ has to approximate $W_X(\cdot)$ in the entire range, and it would suffice to find an instance X and a center c for which $W_X(p)$ cannot be approximated well by a few linear pieces. (Note that this argument never examines the coreset D explicitly.) The detailed proof follows.

Proof. Throughout, let $F(x) := \sqrt{x}$. Now consider the point set $X := \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}$, defined by its prefix-sums $\sum_{j \in [i]} x_j = F(i)$ (for all $i \in [n]$). It is easy to see that $1 = x_1 > x_2 > \cdots > x_n > 0$. Fix center $c := 0$ and consider a simultaneous ϵ-coreset D of size $|D|$. Since D is a simultaneous coreset for Ordered k-Median, $W_D(p) \in (1 \pm \epsilon) \cdot W_X(p)$ for all $p \in [n]$, where by definition $W_X(p) = F(p)$.

We will need the following claim, which shows that each linear piece in $W_D(\cdot)$ (denote here by g) cannot be too "long", as otherwise the relative error exceeds ϵ. We shall use the notation $[a..b] = \{a, a+1, \ldots, b\}$ for two integers $a \leq b$.

Claim 6.2.2. *Let F be as above and let $g : \mathbb{R} \to \mathbb{R}$ be a linear function. Then for every two integers $a \geq 1$ and $b \geq (1 + \frac{1}{\epsilon})^2$ satisfying $\frac{b}{a} \geq (1 + 12\sqrt{\epsilon})^4$, there exists an integer $p \in [a..b]$ such that $g(p) \notin (1 \pm \epsilon) \cdot F(p)$.*

Proof. We may assume both $g(a) \in (1 \pm \epsilon) \cdot F(a)$ and $g(b) \in (1 \pm \epsilon) \cdot F(b)$, as otherwise the claim is already proved. Since g is linear, it is given by $g(x) = k(x - a) + g(a)$,

where $k := \frac{g(b)-g(a)}{b-a}$. Let $\widehat{p} := \lfloor\sqrt{ab}\rfloor$. Observe that $\widehat{p} \in [a..b]$, thus it suffices to prove that

$$\frac{g(\widehat{p})}{F(\widehat{p})} < 1 - \epsilon.$$

The intuition of picking $\widehat{p} = \lfloor\sqrt{ab}\rfloor$ is that $x = \sqrt{ab}$ maximizes $\frac{F(x)}{\widehat{g}(x)}$, where $\widehat{g}(x)$ is the linear function passing through $(a, F(a))$ and $(b, F(b))$, and we know that this $\widehat{g}(x)$ should be "close" to g as $g(a) \in (1 \pm \epsilon) \cdot F(a)$ and $g(b) \in (1 \pm \epsilon) \cdot F(b)$. (Notice that since F is concave, $\widehat{g}(x) \leq F(x)$ for all $x \in [a, b]$.)

To analyze this more formally,

$$\begin{aligned}
g(\widehat{p}) &= \frac{g(b) - g(a)}{b - a}(\lfloor\sqrt{ab}\rfloor - a) + g(a) \\
&= \frac{(\lfloor\sqrt{ab}\rfloor - a) \cdot g(b) + (b - \lfloor\sqrt{ab}\rfloor) \cdot g(a)}{b - a} \\
&\in (1 \pm \epsilon) \cdot \frac{(\lfloor\sqrt{ab}\rfloor - a)\sqrt{b} + (b - \lfloor\sqrt{ab}\rfloor)\sqrt{a}}{b - a} \\
&= (1 \pm \epsilon) \cdot \frac{\lfloor\sqrt{ab}\rfloor + \sqrt{ab}}{\sqrt{a} + \sqrt{b}}.
\end{aligned}$$

Therefore,

$$0 < \frac{g(\widehat{p})}{F(\widehat{p})} \leq (1 + \epsilon) \cdot \frac{\lfloor\sqrt{ab}\rfloor + \sqrt{ab}}{(\sqrt{a} + \sqrt{b}) \cdot \sqrt{\lfloor\sqrt{ab}\rfloor}}.$$

To simplify notation, let $t := \left(\frac{b}{a}\right)^{1/4}$ and $s := \left(\frac{\lfloor\sqrt{ab}\rfloor}{\sqrt{ab}}\right)^{1/2}$. By simple calculations, our assumptions $\frac{b}{a} \geq (1+12\sqrt{\epsilon})^4$ and $b \geq (1+\frac{1}{\epsilon})^2$ imply the following facts: $t+t^{-1} \geq 2+11\epsilon$ and $\frac{1}{1+\epsilon} \leq s \leq 1$. And now we have

$$\begin{aligned}
(1 + \epsilon) \cdot \frac{\lfloor\sqrt{ab}\rfloor + \sqrt{ab}}{(\sqrt{a} + \sqrt{b}) \cdot \sqrt{\lfloor\sqrt{ab}\rfloor}} &= (1 + \epsilon) \cdot \frac{s + s^{-1}}{t + t^{-1}} \\
&\leq \frac{(1 + \epsilon)(2 + \epsilon)}{2 + 11\epsilon} \\
&< 1 - \epsilon.
\end{aligned}$$

Altogether, we obtain $0 < \frac{g(\widehat{p})}{F(\widehat{p})} < 1 - \epsilon$, which completes the proof of Claim 6.2.2. □

We proceed with the proof of Theorem 6.2.1. Recall that $W_D(p)$ is the sum of the p largest distances from points in D to c, when multiplicities are taken into account. Thus, if the p-th largest distance for all $p \in [a..b]$ arise from the same point of D (with appropriate multiplicity), then $W_D(p) - W_D(p-1)$ is just that distance, regardless of p, which means that $W_D(p)$ is linear in this range. It follows that $W_D(p)$ is piece-wise linear with at most $|D|$ pieces. By Claim 6.2.2 and the error bound of the coreset, if a linear piece of $W_D(p)$ spans $p = [a..b]$ where $(1+\frac{1}{\epsilon})^2 \leq b \leq n$, then $b \leq (1+12\sqrt{\epsilon})^4 \cdot a \leq (1+O(\sqrt{\epsilon})) \cdot a$. Since all the linear pieces span together all of $[n]$, we conclude that $|D| = \Omega\left(\log_{1+O(\sqrt{\epsilon})}(\epsilon^2 n)\right) = \Omega\left(\frac{\log n}{\sqrt{\epsilon}}\right)$ and proves Theorem 6.2.1. □

6.3 Coresets for Clustering with Missing Values

Recall that in Chapter 5, we design a coreset of size $(jk)^{O(\min(j,k))}\,\mathrm{poly}(\epsilon^{-1}d\log n)$ for k-Means with missing values. In this Chapter, we prove the following lower bound to assert the necessity of the exponential dependence on $\min(j,k)$.

Theorem 6.3.1. *Consider the k-Means with missing values problem in $\mathbb{R}^d$ where each point can have at most j missing coordinates. Assume there is an algorithm that constructs an ϵ-coreset of size $f(j,k) \cdot \mathrm{poly}(\epsilon^{-1}d\log n)$, then $f(j,k)$ can not be as small as $2^{o(\min(j,k))}$.*

Proof. Consider the following n points instance with $j = k = \Theta(\log n)$, and $d = 2j$. For a subset I of $[d]$, we define a data point $p(I)$ such that $p(I)_i = 1$ if $i \in I$ and $p(I)_i =?$ otherwise. Then we let the data set $P = \{p(I) | I \subseteq [d], |I| = j\}$. We remark that we can make $|P| = \binom{d}{j} = n$ by choosing a proper $j = \Theta(\log n)$.

We prove that any 1/2-coreset of P should contain every point in P. Let D be such a coreset and assume $p(I) \notin D$, we choose the following $k = j$ centers. For every $i \in I$, we define a center $c^i \in \mathbb{R}^d$ such that the i-th coordinate of c^i is 0 and the other coordinates of c^i are 1. We observe that, for any $i \in I$, $\mathrm{dist}(p(I), c^i) = 1$.

Meanwhile for any other $p(I') \neq p(I)$, there must be a $i' \in I \setminus I'$ since $|I| = |I'|$, thus $\text{dist}(p(I'), c^{i'}) = 0$. This should imply that the cost on coreset is 0 while the cost on P is 1 which makes a contradiction.

Since $j = k = \Theta(\log n)$, $d = 2j$, we have $2^{o(\min(j,k))} \cdot \text{poly}(d \log n) = o(n)$. Thus $f(j,k)$ can not be as small as $2^{o(\min(j,k))}$. □

6.4 Coresets for Clustering in Euclidean Space

We prove the following lower bound for coresets in Euclidean space. While this lower bound still has a gap of at least $\frac{1}{\sqrt{\epsilon}}$ from the known upper bounds, it is in fact the first nontrivial lower bound for the Euclidean setting.

Theorem 6.4.1 (Lower Bound for 1D Lines)**.** *For every $0 < \epsilon < 1/24$ and integer $k \geq 1$, there exists a set of data points $V \subseteq \mathbb{R}$, such that every ϵ-coreset for k-MEDIAN of V has size $\Omega(\frac{k}{\sqrt{\epsilon}})$, even if the coreset may use any point in $\mathbb{R}$.*

We first introduce our technical Lemma 6.4.2, which shows a quadratic function cannot be approximated by an affine linear function in a short interval.

Lemma 6.4.2. *Let $\epsilon \in (0, 1/12)$ and $1/2 \leq p \leq q \leq 1$. Suppose $f(x) = ax^2 + b$ where $a > 0, b$, $g(x)$ is a non-negative linear function on $[p, q]$, and $g(x) \in (1 \pm \epsilon) f(x)$ for every $x \in [p, q]$, then $q \leq p + \sqrt{24(b/a + 1)\epsilon}$.*

Proof. Since $g(x)$ is non-negative and $g(x) \in (1 \pm \epsilon) f(x), \forall x \in [p, q]$, we have that $\int_p^q g(x)dx \in (1 \pm \epsilon) \int_p^q f(x)dx$. By computation, $\int_p^q f(x)dx = \int_p^q (ax^2 + b)dx = \frac{a(q^3 - p^3)}{3} + b(q - p)$. Since $g(x)$ is linear, $\int_p^q g(x)dx = (g(p) + g(q))(q - p)/2$. By the fact that $g(x) \in (1 \pm \epsilon) f(x) = (1 \pm \epsilon)(ax^2 + b)$, we have $\int_a^b g(x)dx \in (1 \pm \epsilon)(ap^2 + aq^2 + 2b)(q - p)/2$. But $\int_a^b g(x)dx \leq (1 + \epsilon) \int_a^b (ax^2 + b)dx = (1 + \epsilon)(a(q^3 - p^3)/3 + b(q - p))$, so we have that,

$$\frac{a(q^3-p^3)/3+b(q-p)}{(ap^2+aq^2+2b)(q-p)/2} \geq \frac{1-\epsilon}{1+\epsilon} \geq 1-2\epsilon.$$

So we have $p^2+q^2-\frac{2pq}{1-6\epsilon}-\frac{12b\epsilon}{(1-6\epsilon)a} \leq 0$. Since $\frac{1}{2} \leq p \leq q \leq 1$ and $0 < \epsilon < 1/12$, we have that

$$(q-p)^2 \leq \frac{12b\epsilon}{(1-6\epsilon)a} + \frac{12\epsilon pq}{1-6\epsilon} \leq 24(b/a+1)\epsilon$$

which implies $q \leq p + \sqrt{24(b/a+1)\epsilon}$. □

Now we are ready to prove Theorem 6.4.1.

Proof of Theorem 6.4.1. We first prove the basic case $k = 1$. Without loss of generality, we can assume $V = [-1, 1]$.[1] Let $f(x) := \text{cost}(V, \{x\}) = \int_V \|t - x\| dt$ denote the cost of connecting V to $x \in \mathbb{R}$ (the 1-MEDIAN value). Note that $f(x) = x^2 + 1$ on $[-1, 1]$.

Assume D is an ϵ-coreset of V for the 1-median problem. Recall that D is a weighted set and may contain the ambient points of the real line. Let $g(x) := \text{cost}(D, \{x\})$. Then $g(x)$ is a piecewise linear function and the transition from one affine linear function to another happens only when x crosses a coreset point. So the number of pieces is at most $|D| + 1$. We need to prove $g(x)$ has at least $\Omega(\frac{1}{\sqrt{\epsilon}})$ pieces.

Since D is an ϵ-coreset of V, we have that $g(x) \in (1 \pm \epsilon)f(x)$ for every $x \in [-1, 1]$. Assume $g(x)$ has m pieces in $[1/2, 1]$ and their connecting points are $x_0 = 1/2 < x_1 < ... < x_m = 1$. Then $g(x)$ is affine linear in $[x_{i-1}, x_i]$ but $g(x) \in (1 \pm \epsilon)f(x)$, so by Lemma 6.4.2, $x_i < x_{i-1} + \sqrt{24(1/1+1)\epsilon} < x_{i-1} + 7\sqrt{\epsilon}$. So $m \geq \frac{1/2}{7\sqrt{\epsilon}} = \frac{1}{14\sqrt{\epsilon}}$.

Now we consider the case of general k. We put k copies of $[-1, 1]$ in the real line. In particular, we let $V_i = [-1 + (i-1)t, 1 + (i-1)t]$ for a large enough positive number $t > \frac{39}{\sqrt{\epsilon}}$ and $V = \cup_{i=1}^k V_i$. Let $S_i = [1/2 + (i-1)t, 1 + (i-1)t]$ be a subset of V_i. We partition each S_i into $m = \Theta(1/\sqrt{\epsilon})$ intervals $S_{i1}, ..., S_{im}$, such that each of them has length $13\sqrt{\epsilon}$.

[1] For discretization, we can let $V = \{0, \pm\frac{1}{m}, ..., \pm 1\}$ for large enough m.

Now, for the sake of contradiction, we assume there is an ϵ-coreset D of V for the k-median problem, such that $|D| < mk/2$. By averaging, we know that there is a $j^* \in [k]$ such that $\cup_{i=1}^{k} S_{ij^*}$ contains at most $k/2$ points of D. Let $b = |D \cap (\cup_{i=1}^{k} S_{ij^*})|$ in the following, then $b < k/2$. So there are $k - b > k/2$ many $i \in [k]$ such that S_{ij^*} doesn't contain any coreset point. We assume $S_{0j^*} = [a_0, b_0]$ and let x be a variable in $[a_0, b_0]$. We construct the following set of centers $C_x = \{x_i : i \in [k]\}$ where if $S_{ij^*} \cap D \neq \emptyset$, $x_i = x + (i-1)t$, otherwise $x_i = (i-1)t$.

Let $f(x) = \text{cost}(V, C_x)$, then $f(x) = b + (k-b)(x^2+1) = (k-b)x^2 + k$. We note that $k \leq f(x) \leq 2k$ on $[a_0, b_0]$. Let $D_i = D \cap [(i-4/3)t, (i-2/3)t]$, $D' = \cup_{i\in[k]} D_i$ and $D'' = D \setminus D'$. We first claim that $|\text{cost}(D'', C_x) - \text{cost}(D'', C_{a_0})| \leq O(\sqrt{\epsilon}k/t)$. Actually, note that when $x \in [a_0, b_0]$, the connection cost of points in D'' is always $\Omega(t)$, along with the fact that when $x \in [a_0, b_0]$, $\text{cost}(D'', C_x) \leq \text{cost}(D, C_x) \leq (1+\epsilon)\text{cost}(V, C_x) \leq 3k$, we know that the total weight of D'' is at most $3k/t$. Now, since x changes by at most $13\sqrt{\epsilon}$, the connection cost of every point in D'' changes by at most $13\sqrt{\epsilon}$, so we have

$$|\text{cost}(D'', C_x) - \text{cost}(D'', C_{a_0})| \leq \frac{39\sqrt{\epsilon}k}{t}.$$

Let $g(x) = \text{cost}(D', C_x) + \text{cost}(D'', C_{a_0})$. Since $t > \frac{39}{\sqrt{\epsilon}}$ and $f(x) \geq k$ for every $x \in [a_0, b_0]$, we conclude that when $x \in [a_0, b_0]$,

$$|g(x) - \text{cost}(D, C_x)| = |\text{cost}(D'', C_{a_0}) - \text{cost}(D'', C_x)| \leq \frac{39\sqrt{\epsilon}k}{t} \leq \epsilon f(x).$$

But $\text{cost}(D, C_x) \in (1 \pm \epsilon) f(x)$ on $[a_0, b_0]$, so we have that $g(x) \in (1 \pm 2\epsilon) f(x)$ on $[a_0, b_0]$.

Now we show that $g(x)$ is an affine linear function in $[a_0, b_0]$. We note that $D' = \cup_{i\in[k]} D_i$ and $\text{cost}(D_i, C_x) = \text{cost}(D_i, \{x_i\})$ for any $x \in [a_0, b_0]$. If $x_i = x + (i-1)t$ then by construction, x_i never crosses any coreset point in D_i, so $\text{cost}(D_i, \{x_i\})$ remains affine linear. On the other hand, if $x_i = (i-1)t$, then $\text{cost}(D_i, \{x_i\})$ is a constant. So we know that $g(x) = \text{cost}(D', C_x) + \text{cost}(D'', C_{a_0}) = \sum_{i\in[k]} \text{cost}(D_i, \{x_i\}) + \text{cost}(D'', C_{a_0})$

is an affine linear function on $[a_0, b_0]$. But $g(x) \in (1 \pm 2\epsilon)f(x)$ on $[a_0, b_0]$, by Lemma 6.4.2, we know that the length of $[a_0, b_0]$ is at most $\sqrt{24(\frac{k}{k-b} + 1) \cdot 2\epsilon} \leq \sqrt{144\epsilon} = 12\sqrt{\epsilon}$, arriving at a contradiction. $\square$

Conclusions

Coresets are important data reduction tools for clustering problems. Coresets for clustering have received great attentions from big data algorithm and machine learning algorithm research community. In this dissertation, we make various contributions to this line of research.

We make contributions to a fundamental question on what kind of metric space admit constant-sized coresets, by constructing such coresets in bounded treewidth graph and the more general excluded-minor graph.

Moreover, we resolve new challenges appearing in modern application scenario of coresets. In particular, we have designed simultaneous coresets and coresets for clustering with missing values.

We also provide various lower bounds to support our upper bound results.

References

1. Marom, Y. & Feldman, D. *k-Means Clustering of Lines for Big Data* in *NeurIPS* (2019), 12797–12806.
2. Eiben, E. *et al. EPTAS for* k*-means Clustering of Affine Subspaces* in *SODA* (SIAM, 2021), 2649–2659.
3. Har-Peled, S. & Mazumdar, S. *On coresets for k-means and k-median clustering* in *36th Annual ACM Symposium on Theory of Computing,* (2004), 291–300.
4. Agarwal, P. K., Har-Peled, S. & Varadarajan, K. R. Approximating Extent Measures of Points. *J. ACM* **51,** 606–635 (July 2004).
5. Fichtenberger, H., Gillé, M., Schmidt, M., Schwiegelshohn, C. & Sohler, C. *BICO: BIRCH Meets Coresets for k-Means Clustering* in *ESA* **8125** (Springer, 2013), 481–492.
6. Balcan, M.-F. F., Ehrlich, S. & Liang, Y. *Distributed k-means and k-median Clustering on General Topologies* in *NIPS* (2013), 1995–2003.
7. Huang, L., Jiang, S. H.-C., Li, J. & Wu, X. *Epsilon-coresets for clustering (with outliers) in doubling metrics* in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)* (2018), 814–825.
8. Friggstad, Z., Rezapour, M. & Salavatipour, M. R. Local Search Yields a PTAS for k-Means in Doubling Metrics. *SIAM J. Comput.* **48,** 452–480 (2019).
9. Phillips, J. M. Coresets and Sketches. *CoRR* **abs/1601.00617.** arXiv: 1601.00617 (2016).
10. Munteanu, A. & Schwiegelshohn, C. Coresets-Methods and History: A Theoreticians Design Pattern for Approximation and Streaming Algorithms. *KI* **32,** 37–53 (2018).
11. Feldman, D. Core-sets: An updated survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **10** (2020).
12. Har-Peled, S. & Kushal, A. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry* **37,** 3–19 (2007).
13. Feldman, D. & Langberg, M. *A unified framework for approximating and clustering data* in *STOC* https://arxiv.org/abs/1106.1379 (ACM, 2011), 569–578.
14. Sohler, C. & Woodruff, D. P. *Strong Coresets for k-Median and Subspace Approximation: Goodbye Dimension* in *FOCS* (IEEE Computer Society, 2018), 802–813.
15. Huang, L. & Vishnoi, N. K. *Coresets for clustering in euclidean spaces: Importance sampling is nearly optimal* in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (2020), 1416–1429.

16. Baker, D. *et al. Coresets for clustering in graphs of bounded treewidth* in *International Conference on Machine Learning* (2020), 569–579.

17. Braverman, V., Jiang, S. H.-C., Krauthgamer, R. & Wu, X. *Coresets for Clustering in Excluded-minor Graphs and Beyond* in *SODA* (SIAM, 2021), 2679–2696.

18. Braverman, V., Jiang, S. H.-C., Krauthgamer, R. & Wu, X. *Coresets for ordered weighted clustering* in *International Conference on Machine Learning* (2019), 744–753.

19. Braverman, V., Jiang, S. H.-C., Krauthgamer, R. & Wu, X. Coresets for Clustering with Missing Values. *arXiv preprint arXiv:2106.16112* (2021).

20. Langberg, M. & Schulman, L. J. *Universal epsilon-approximators for Integrals* in *SODA* (SIAM, 2010), 598–607.

21. Huang, L. & Vishnoi, N. K. *Coresets for clustering in Euclidean spaces: importance sampling is nearly optimal* in *STOC* (ACM, 2020), 1416–1429.

22. Bachem, O., Lucic, M. & Lattanzi, S. *One-shot coresets: The case of k-clustering* in *International conference on artificial intelligence and statistics* (2018), 784–792.

23. Fortunato, S. Community detection in graphs. *Physics reports* **486,** 75–174 (2010).

24. Herman, I., Melançon, G. & Marshall, M. S. Graph Visualization and Navigation in Information Visualization: A Survey. *IEEE Trans. Vis. Comput. Graph.* **6,** 24–43 (2000).

25. Shekhar, S. & Liu, D.-R. CCAM: A Connectivity-Clustered Access Method for Networks and Network Computations. *IEEE Trans. Knowl. Data Eng.* **9,** 102–119 (1997).

26. Yiu, M. L. & Mamoulis, N. *Clustering Objects on a Spatial Network* in *SIGMOD Conference* (ACM, 2004), 443–454.

27. Rattigan, M. J., Maier, M. & Jensen, D. *Graph clustering with network structure indices* in *Proceedings of the 24th international conference on Machine learning* (2007), 783–790.

28. Cui, W., Zhou, H., Qu, H., Wong, P. C. & Li, X. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics* **14,** 1277–1284 (2008).

29. Tansel, B. C., Francis, R. L. & Lowe, T. J. State of the art—location on networks: a survey, Part I and II. *Management Science* **29,** 482–497 (1983).

30. Chen, K. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing* **39,** 923–947 (2009).

31. Robertson, N. & Seymour, P. D. Graph minors. II. Algorithmic aspects of tree-width. *Journal of algorithms* **7,** 309–322 (1986).

32. Kloks, T. *Treewidth: computations and approximations* (Springer Science & Business Media, 1994).

33. Maniu, S., Senellart, P. & Jog, S. *An Experimental Study of the Treewidth of Real-World Graph Data* in *ICDT* **127** (Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019), 12:1–12:18.

34. Bousquet, N. & Thomassé, S. VC-dimension and Erdős–Pósa property. *Discrete Mathematics* **338,** 2302–2317 (2015).

35. Jain, K., Mahdian, M. & Saberi, A. *A new greedy approach for facility location problems* in *STOC* (ACM, 2002), 731–740.

36. Byrka, J., Pensyl, T., Rybicki, B., Srinivasan, A. & Trinh, K. An Improved Approximation for k-Median and Positive Correlation in Budgeted Optimization. *ACM Trans. Algorithms* **13,** 23:1–23:31 (2017).

37. Cohen-Addad, V., Klein, P. N. & Mathieu, C. Local Search Yields Approximation Schemes for k-Means and k-Median in Euclidean and Minor-Free Metrics. *SIAM J. Comput.* **48,** 644–667 (2019).

38. Cohen-Addad, V., Pilipczuk, M. & Pilipczuk, M. *Efficient Approximation Schemes for Uniform-Cost Clustering Problems in Planar Graphs* in *ESA* **144** (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019), 33:1–33:14.

39. Thorup, M. Quick k-Median, k-Center, and Facility Location for Sparse Graphs. *SIAM J. Comput.* **34,** 405–432 (2005).

40. Kearns, M. J. & Vazirani, U. V. *An introduction to computational learning theory* (MIT press, 1994).

41. Feldman, D., Schmidt, M. & Sohler, C. Turning Big Data Into Tiny Data: Constant-Size Coresets for k-Means, PCA, and Projective Clustering. *SIAM J. Comput.* **49,** 601–657 (2020).

42. Varadarajan, K. R. & Xiao, X. *On the Sensitivity of Shape Fitting Problems* in *FSTTCS* **18** (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012), 486–497.

43. Sack, J. & Urrutia, J. *Handbook of Computational Geometry* (Elsevier, 1999).

44. Cohen-Addad, V., Klein, P. N. & Mathieu, C. Local search yields approximation schemes for k-means and k-median in Euclidean and minor-free metrics. *SIAM Journal on Computing* **48,** 644–667 (2019).

45. Li, M., Miller, G. L. & Peng, R. *Iterative Row Sampling* in *FOCS* (IEEE Computer Society, 2013), 127–136.

46. Clarkson, K. L. & Woodruff, D. P. *Sketching for M-Estimators: A Unified Approach to Robust Regression* in *SODA* (SIAM, 2015), 921–939.

47. Munteanu, A., Schwiegelshohn, C., Sohler, C. & Woodruff, D. P. *On Coresets for Logistic Regression* in *NeurIPS* (2018), 6562–6571.

48. Braverman, V., Feldman, D. & Lang, H. New Frameworks for Offline and Streaming Coreset Constructions. *CoRR* **abs/1612.00889.** eprint: 1612.00889 (2016).

49. Narayanan, S. & Nelson, J. *Optimal terminal dimensionality reduction in Euclidean space* in *STOC* (ACM, 2019), 1064–1069.

50. Thorup, M. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM* **51,** 993–1024 (2004).

51. Abraham, I. & Gavoille, C. *Object location using path separators* in *PODC* (ACM, 2006), 188–197.

52. Johnson, W. B. & Lindenstrauss, J. in *Conference in modern analysis and probability (New Haven, Conn., 1982)* 189–206 (Amer. Math. Soc., 1984).

53. Cohen, M. B., Elder, S., Musco, C., Musco, C. & Persu, M. *Dimensionality reduction for k-means clustering and low rank approximation* in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing* (2015), 163–172.

54. Becchetti, L., Bury, M., Cohen-Addad, V., Grandoni, F. & Schwiegelshohn, C. *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* (2019), 1039–1050.

55. Makarychev, K., Makarychev, Y. & Razenshteyn, I. *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* (2019), 1027–1038.

56. Har-Peled, S. *On Complexity, Sampling, and ϵ-Nets and ϵ-Samples* (American Mathematical Soc., 2011).

57. Vapnik, V. N. & Chervonenkis, A. Y. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and its Applications* **16,** 264–280 (1971).

58. Mettu, R. R. & Plaxton, C. G. Optimal Time Bounds for Approximate Clustering. *Mach. Learn.* **56,** 35–60 (2004).

59. Jain, K. & Vazirani, V. V. Approximation algorithms for metric facility location and k-Median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM* **48,** 274–296 (2001).

60. Eisenstat, D., Klein, P. N. & Mathieu, C. *Approximating k-center in planar graphs* in *SODA* (SIAM, 2014), 617–627.

61. Klein, P. & Mozes, S. *Optimization Algorithms for Planar Graphs* Book draft, http://www.planarity.org. 2012.

62. Chierichetti, F., Kumar, R., Lattanzi, S. & Vassilvitskii, S. *Fair Clustering Through Fairlets* in *NIPS* (2017), 5036–5044.

63. Chandola, V., Banerjee, A. & Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **41,** 15:1–15:58 (2009).

64. Yager, R. R. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking. *IEEE Trans. Syst. Man Cybern.* **18,** 183–190 (1988).

65. Agarwal, P. K. & Procopiuc, C. M. Exact and Approximation Algorithms for Clustering. *Algorithmica* **33,** 201–226 (2002).

66. Megiddo, N. & Supowit, K. On the Complexity of Some Common Geometric Location Problems. *SIAM Journal on Computing* **13,** 182–196 (1984).

67. Tamir, A. The k-centrum multi-facility location problem. *Discrete Applied Mathematics* **109,** 293–307 (2001).

68. Aouad, A. & Segev, D. The ordered k-median problem: surrogate models and approximation algorithms. *Mathematical Programming,* 1–29 (2018).

69. Byrka, J., Sornat, K. & Spoerhase, J. *Constant-factor approximation for ordered k-median* in *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing* (2018), 620–631.

70. Chakrabarty, D. & Swamy, C. *Interpolating between k-Median and k-Center: Approximation Algorithms for Ordered k-Median* in *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)* **107** (Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018), 29:1–29:14.

71. Chakrabarty, D. & Swamy, C. Approximation Algorithms for Minimum Norm and Ordered Optimization Problems. *CoRR* **abs/1811.05022.** arXiv: 1811.05022 (2018).

72. Allison, P. D. *Missing data* (Sage publications, 2001).

73. Little, R. J. & Rubin, D. B. *Statistical analysis with missing data* (John Wiley & Sons, 2019).

74. Hathaway, R. J. & Bezdek, J. C. Fuzzy c-means clustering of incomplete data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **31,** 735–744 (2001).

75. Wagstaff, K. in *Classification, clustering, and data mining applications* 649–658 (Springer, 2004).

76. Chi, J. T., Chi, E. C. & Baraniuk, R. G. *k*-POD: A Method for k-Means Clustering of Missing Data. *The American Statistician* **70,** 91–99 (2016).

77. Wang, S. *et al.* *K*-Means Clustering With Incomplete Data. *IEEE Access* **7,** 69162–69171 (2019).

78. Himmelspach, L. & Conrad, S. *Clustering approaches for data with missing values: Comparison and evaluation* in *ICDIM* (IEEE, 2010), 19–28.

79. Arthur, D. & Vassilvitskii, S. *k-means++: the advantages of careful seeding* in *SODA* (SIAM, 2007), 1027–1035.

80. Gao, J., Langberg, M. & Schulman, L. J. Analysis of Incomplete Data and an Intrinsic-Dimension Helly Theorem. *Discret. Comput. Geom.* **40,** 537–560 (2008).

81. Gao, J., Langberg, M. & Schulman, L. J. Clustering lines in high-dimensional space: Classification of incomplete data. *ACM Trans. Algorithms* **7,** 8:1–8:26 (2010).

82. Lee, E. & Schulman, L. J. *Clustering Affine Subspaces: Hardness and Algorithms* in *SODA* (SIAM, 2013), 810–827.

83. Har-Peled, S. & Mazumdar, S. *On coresets for k-means and k-median clustering* in *STOC* https://arxiv.org/abs/1810.12826 (ACM, 2004), 291–300.

84. Frahling, G. & Sohler, C. *Coresets in dynamic geometric data streams* in *STOC* (ACM, 2005), 209–217.

85. Braverman, V., Frahling, G., Lang, H., Sohler, C. & Yang, L. F. *Clustering High Dimensional Dynamic Data Streams* in *ICML* **70** (PMLR, 2017), 576–585.

86. Balcan, M.-F., Ehrlich, S. & Liang, Y. *Distributed k-means and k-median clustering on general communication topologies* in *NIPS* (2013), 1995–2003.

87. Reddi, S. J., Póczos, B. & Smola, A. J. *Communication Efficient Coresets for Empirical Loss Minimization* in *UAI* (AUAI Press, 2015), 752–761.

88. Bachem, O., Lucic, M. & Krause, A. *Scalable k-Means Clustering via Lightweight Coresets* in *KDD* (ACM, 2018), 1119–1127.

89. Chan, T. M. Dynamic Coresets. *Discret. Comput. Geom.* **42,** 469–488 (2009).

90. Henzinger, M. & Kale, S. *Fully-Dynamic Coresets* in *ESA* **173** (Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020), 57:1–57:21.

91. Mussay, B., Osadchy, M., Braverman, V., Zhou, S. & Feldman, D. *Data-Independent Neural Pruning via Coresets* in *ICLR* (OpenReview.net, 2020).

92. Varadarajan, K. & Xiao, X. *A near-linear algorithm for projective clustering integer points* in *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms* (2012), 1329–1342.

93. Gonzalez, T. F. Clustering to minimize the maximum intercluster distance. *Theoretical computer science* **38,** 293–306 (1985).

94. Indyk, P. *Better algorithms for high-dimensional proximity problems via asymmetric embeddings* in *SODA* (ACM/SIAM, 2003), 539–545.

95. Lucic, M., Faulkner, M., Krause, A. & Feldman, D. Training gaussian mixture models at scale via coresets. *The Journal of Machine Learning Research* **18,** 5885–5909 (2017).

96. Feldman, D., Kfir, Z. & Wu, X. Coresets for gaussian mixture models of any shape. *arXiv preprint arXiv:1906.04895* (2019).

97. Munteanu, A., Schwiegelshohn, C., Sohler, C. & Woodruff, D. *On Coresets for Logistic Regression* in *Advances in Neural Information Processing Systems* (eds Bengio, S. *et al.*) **31** (Curran Associates, Inc., 2018).

98. Huang, L., Jiang, S. & Vishnoi, N. *Coresets for Clustering with Fairness Constraints* in *Advances in Neural Information Processing Systems* **32** (Curran Associates, Inc., 2019).

99. Jubran, I., Tukan, M., Maalouf, A. & Feldman, D. *Sets Clustering* in *Proceedings of the 37th International Conference on Machine Learning* **119** (PMLR, 13–18 Jul 2020), 4994–5005.

100. Van Handel, R. *Probability in high dimension* tech. rep. (PRINCETON UNIV NJ, 2014).

101. Dinitz, M. & Krauthgamer, R. *Fault-tolerant spanners: better and simpler* in *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing* (2011), 169–178.

102. Duan, R., Gu, Y. & Ren, H. *Approximate Distance Oracles Subject to Multiple Vertex Failures* in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2021), 2497–2516.

103. Karthik, C. & Parter, M. *Deterministic replacement path covering* in *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)* (2021), 704–723.