Approximation Algorithms and Sketches for Clustering

David Saulpic

École Doctorale Informatique, Télécommunications et Électronique, Paris
Sorbonne Université, Laboratoire LIP6, équipe Recherche Opérationnelle

Approximation Algorithms and Sketches for Clustering

David Saulpic

Thèse de doctorat en informatique
présentée et soutenue publiquement le 13 septembre 2022

Jury

Rapporteurs

Robert Krauthgamer	Professeur, Weizmann Institute, Israël
Laurent Viennot	Professeur, INRIA Paris, France

Examinatrices et Examinateur

Cristina Bazgan	Professeure, Université Paris Dauphine, France
Monika Henzinger	Professor, Universität Wien, Autriche
Chris Schwiegelshohn	Assistant Professor, Aarhus University, Danemark

Encadrants

Vincent Cohen-Addad	Research Scientist, Google, France
Christoph Dürr	Professeur, Sorbonne Université, France

Contents

Acknowledgments **7**

1 General Introduction **9**

1.1 Beyond a Single Representative: k-means and k-median 11

1.2 Part I: Growing Roots and Going Through the Trees 14

1.3 Part II: Coping with Big Data 16

1.4 Broad Background 18

1.5 Our Contribution, Chapter by Chapter 21

1.6 How to Read this Thesis 22

Publications of the Author **23**

Preliminaries: Notations and Useful Results. **25**

I Approximation Algorithm via Embeddings Into Tree-like Structures **27**

2 Presentation of the Results and Challenges **29**

2.1 Results Presented in the Part 29

2.2 Using Ultrametrics for Squared Distances 31

3 Approximation Schemes for Clustering in Doubling Metrics **33**

3.1 Introduction and Sketch of Proofs 33

3.2 Preliminaries 40

3.3 Near-Linear Time Approximation Scheme for Facility Location 44

3.4 The (k, z)-Clustering Problem 55

3.5 Other Applications of the Framework . . . 64
3.6 Conclusions . . . 69
3.7 Omitted Proofs . . . 70

4 Scalable Differentially Private Clustering via Quadtrees 77
4.1 Introduction . . . 77
4.2 Preliminaries . . . 80
4.3 Simple algorithm for Differentially Private k-Median . . . 82
4.4 MPC Implementations . . . 86
4.5 Extension to k-Means . . . 90
4.6 Empirical Evaluation for k-Median . . . 97
4.7 Conclusion . . . 101

II Coreset and Sketches for Clustering 103

5 Presentation of our Results and Overview of the Techniques 105
5.1 Introduction . . . 105
5.2 Brief description of Chen's Coreset . . . 112
5.3 VC-dimension: Coresets independent of the size of the input . . . 113
5.4 Further Related Work . . . 115

6 A New Coreset Framework for Clustering 119
6.1 Overview of Our Techniques . . . 119
6.2 The Coreset Construction Algorithm, and Proof of Theorem 5.3 . . . 122
6.3 Sampling inside Groups: Proof of Lemma 6.4 . . . 128
6.4 Dealing with the Few Far points: Sampling from Outer Rings . . . 144
6.5 Partitioning into Well Structured Groups: Proof of Lemma 6.7 . . . 152
6.6 A Coreset of Size $k^2\varepsilon^{-2}$ . . . 154

7 New Coreset Bounds for Various Metric Spaces 159
7.1 Overview of our Techniques . . . 159
7.2 Metrics with Bounded Doubling Dimension . . . 164
7.3 Graphs with Bounded Treewidth . . . 166

7.4 Planar Graphs . . . 171
7.5 Minor-Excluded Graphs . . . 175
7.6 Euclidean Spaces . . . 181

8 Lower Bounds for Coreset 185
8.1 Introduction . . . 185
8.2 A subinstance for the case $k = 1$ . . . 189
8.3 Combining the subinstances . . . 192

9 Deterministic Sketches for Clustering 197
9.1 Introduction and Key Challenges . . . 198
9.2 Deterministic Coresets and Partition Coresets for (k, z)-Clustering . . . 204
9.3 Derandomized Dimension Reduction . . . 211
9.4 Deterministic Coreset via Uniform VC-Dimension . . . 215
9.5 Witness Sets . . . 225
9.6 Computing Approximate Solutions . . . 227
9.7 Omitted Proofs . . . 229

10 Sublinear Algorithms for Power Mean in Euclidean Spaces 231
10.1 Introduction . . . 231
10.2 Sublinear Algorithm and Analysis . . . 236
10.3 Proof of Theorem 10.1 . . . 241
10.4 A Brief Note on the MPC Algorithm From Section 4.4.2 . . . 244
10.5 Probability Amplification . . . 244
10.6 Lower bound . . . 246
10.7 Experiments . . . 247

Conclusion and Open Questions 251

Bibliography 255

Acknowledgments

Firstly, I would like to thank Laurent Viennot and Robert Kraugthgamer, who accepted to review this long thesis and be part of the jury. For the discussions, the accuracy of your comments, your patience and for your time: a thousand thanks. I am also very grateful to the other committee members Cristina Bazgan, Monika Henzinger and Chris Schwiegelshohn. It was a great honour for me.

During the three years of my PhD, I had the constant feeling of growing up, learning new ways of reasoning, becoming more accurate in my thinking: this (and way more) was fueled by the perfect guidance and ingenuity of Vincent Cohen-Addad. I cannot thank you enough for all you brought me.

Christoph Dürr was also always available for answering my imprecise existential questions and providing his help: thank you in particular for sharing your views on theoretical research.

Chris could have been my third advisor: you shared so much with me, from your passion of coresets and theory to post-deadline trashtalk about NBA. I hope you will keep your enthusiasm, and that I will be able to still enjoy a part of it.

I was very lucky to benefit from the precise advice and recommendations of Claire Mathieu at several crucial moments of my studies. Merci beaucoup, this is priceless. I would like to express my gratitude towards Phil Klein, Giuseppe Italiano and Anupam Gupta, who hosted me during several internships. Your passion for research and the theory of algorithms was very communicative, and still has a great influence on me.

I also want to thank my academic family: Frederik Malmann-Trenn, who was like a big brother from scratch, and Simon Mauras and Mathieu Mari, the nicest conference roommates.

Mille merci à Adèle, Anne-Elisabeth, Kostas puis François, pour m'avoir supporté dans le bureau (special thanks to Kostas, who supported my French during those long 3 years). C'était la belle vie sous la serre ! Et aux autres artistes du LIP6, pour les illusions du Buet 2 et de Fontainebleau, et pour toutes les bières qu'on boira ensemble !

Je ne peux pas parler du labo sans parler du CROUS, et tous les gens qui m'y ont accompagné pendant ces quelques années - merci à Marvin, Margot, Parham, Nadjet, Olivier, Thibaut, Nawal, Bruno, Fanny, Martin, Océane, Martin, Niklas. Ces beaux repas vont me manquer. Merci en particulier à Gaspard, qui mérite largement le titre

de G.O. crous. Merci à Emmanuel pour les pauses vélos à moteur, et à Pierre et Matthieu, pour avoir partagé une vision du métier de chercheur qui me porte encore aujourd'hui.

Merci aux collègues de la Cité des sciences pour la bouffée d'oxygène que vous m'avez apporté, pour tous les horoscopes, mots croisés et Contrario. Avec votre bonne humeur, vous m'avez mis une timide bougie dans la main (pas facile celui là). Et merci à Laetitia pour ton encadrement parfait, avoir partagé un peu de ta passion pour la médiation (et beaucoup de chocolats !)

Finalement, merci aux copains qui font des maths sérieuses d'être toujours présents, en haut du Ventoux ou autour d'une raclette. Merci Cyril, Jean, Marc, Remy, Séginus, Simon, Thibaut et Ulysse. J'espère qu'on restera pas trop longtemps éparpillés sur le globe et qu'on pourra vite se retrouver pour manger du fromage et faire des blagues corses !

Chapter 1

General Introduction

Living at the pace of industrial applications, the field of Computer Science is in constant evolution, with the regular apparition of new areas and paradigms. One of its recent shifts is concerned about processing enormous datasets, containing billions of points: this is the so-called "big data" era. Not only computer scientists are concerned: other sciences, such as physics or biology, are now collecting massive amount of data; companies are taking action driven by data; and big data is even becoming a political hot topic on which governments are taking actions [84, 142].

The quantity of data collected and analyzed by numerical devices is indeed drastically increasing. For data analysts, this raises a new challenge: how to actually understand the structure of those endless flows of information, and extract the essential from them? This, in turn, challenges the computer science community: processing manually all that data is a mirage, and it is necessary to have automatic procedures – in other words, algorithms – to treat it.

Those algorithms need to be specifically tailored to the big data paradigm: it is indeed unrealistic to imagine algorithms using substantially more memory than that to store the input data, or substantially more time than to simply read it. Sometimes, even storing the whole data in memory may be impossible, and finding a summary of it is necessary. By contrast, an algorithm that needs a processing time polynomial in the time necessary to read the input was traditionally considered fast enough – as an example, if the time to read the input is x seconds, an algorithm solving a problem in x^{10} seconds was considered fast. Many such algorithms were developed in the past decades, but they are out of scope for processing big data: the time to read a standard big dataset is often larger than $x = 10$ seconds, and no one is willing to wait the result of an algorithm for $x^{10} = 10^{10}$ seconds – roughly 317 years. It is thus crucial to find new algorithms, adapted to those new constraints, in order to process and analyze data.

With the aim of both summarizing and gathering some structural information about a dataset, one of the simplest pieces of information that exists is the mean. When the dataset consists of points in Euclidean space, the algorithm to compute the mean merely consists of taking the sum, coordinate by coordinate, of the input data, and

normalize it by the number of input points. The mean is precisely this normalized sum. If one cannot even read the full input, it is possible to compute a point close to the mean by applying this algorithm to a small random subset of the input. This algorithm is extremely fast, as it does not require more than reading a small random subset of the input, and memory efficient, as one only needs to store the sum. Hence, it answers perfectly the time and memory constraints of the big data paradigm.

In terms of information provided, the mean is sometimes considered as a good *representative* of the full dataset: one can use it as a first approximation to describe the dataset, as it is done for instance when talking about the mean salary of a country, or the mean height of a population. However, as noted in the definition given by INSEE[1] [105], it is often not the best indicator, and the median may be more relevant. For instance, they note that the mean is very impacted by *outliers*: in the case of the salary, it is very impacted by few very high ones, and that furthermore uncertainty on those high values is transferred to the mean – while the median escapes those two shortcomings. One reason for which the mean is so vastly used despite those disadvantages may be the easiness of its computation [105]. This was a good reason when data was processed by hand, but we now have computers and algorithms: they can be of help to solve more complex problems, and to provide a more precise information.

For instance, a limitation of the mean (that the median suffers as well) is the following: using the mean of a dataset, it is implicitly assumed that it is actually meaningful to represent the whole data by a single point. This assumption is of course not always satisfied, as illustrated by the example of Fig. 1.1. In such a case, finding a single representative conceals diversity of the dataset. Using two representatives instead would give a more accurate and faithful summary of it.

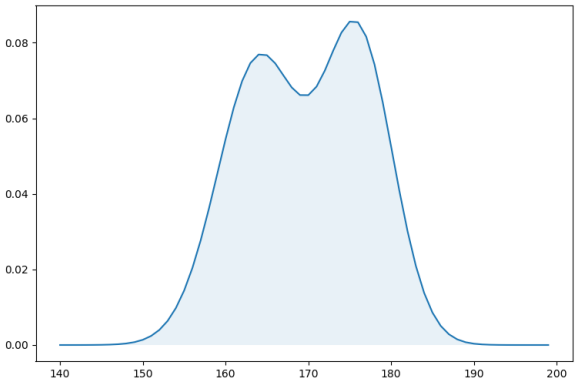


Figure 1.1: Distribution of sizes in France, that exhibits two peaks. This drawing is artificial, based only on the estimated mean and standard deviation of the sizes of French men and women.

The natural question is therefore: can we represent the data by *more* than a single point? Easy to say, but harder to formalize: how to define two means, or two medians, of a dataset? We present in the next section the common answer to this question, by

[1] INSEE is the French national institute for statistics and economical studies.

defining a generalization of those notions to k points, namely k-means and k-median.

Then, the central question of this thesis is the following: how to compute those means and medians, especially in the "big data" regime?

The remaining of this introduction consists of the introduction of the k-median and k-means problem, followed by more specific introductions to the two main parts of the thesis. We then give a broad overview of previous research, and present briefly our contribution chapter by chapter. Finally, we present some preliminaries common to the whole thesis.

1.1 Beyond a Single Representative: k-means and k-median

A natural generalization of the mean was introduced by Hugo Steinhaus in 1956 [152], based on the following idea. In Euclidean spaces, the mean is the point minimizing the *variance*, i.e. the point that minimizes the sum of squared distances to it. In equation, given an input $P \subseteq \mathbb{R}^d$, the mean is the point m minimizing

$$\sum_{p \in P} \text{dist}(p, m)^2, \tag{1.1}$$

where $\text{dist}(p, m)$ is the Euclidean distance between p and m.

Then, if instead one was looking for a set $\mathcal{S}$ of k "means" instead of the mean, it is natural to minimize

$$\sum_{p \in P} \min_{s \in \mathcal{S}} \text{dist}(p, s)^2.$$

In words, each data point from P is assigned to its closest point in $\mathcal{S}$ (those are called "centers"), and contributes the square of the distance to that center. The goal is to find the set $\mathcal{S}$ that minimizes the sum of contribution. Note that this definition of mean can be generalized to any metric space: instead of using the Euclidean distance, one could use any other distance function. For instance, if data points consist of road crossings in a road network, it may be better to use the travel time as a distance. Hence, it is possible to define the mean with Eq. (1.1) for other spaces than the Euclidean one.

Generalizing the notion of median is slightly harder. Its common definition for real number is the middle point of the dataset: this is not clearly defined when the data is instead from the Euclidean plane. For that reason, we will use an alternate way of defining the median, similar to Eq. (1.1): when the dataset P consists of real numbers, the median minimizes

$$\sum_{p \in P} |p - m|. \tag{1.2}$$

To see this, imagine that the point m minimizing Eq. (1.2) has more input point bigger than it, than input points smaller. Then, increasing slightly m decreases the

equation, which contradict the minimality of m. Hence, Eq. (1.2) is minimal when there are as many points bigger and smaller than m: this is precisely the definition of median. Eq. (1.2) is therefore an alternate definition to the median, which it is also valid when the data is from Euclidean plane or even higher dimensions. Analogously to k-means, the median can be generalized to other spaces, and to a higher number of "medians": the k-median problem seeks to find a set $\mathcal{S}$ of size k that minimizes

$$\sum_{p \in P} \min_{s \in \mathcal{S}} \text{dist}(p, s).$$

To settle on a vocabulary, any set $\mathcal{S}$ of size k is a *solution*. The *cost* of a solution is $\sum_{p \in P} \min_{s \in \mathcal{S}} \text{dist}(p, s)$ for k-median, $\sum_{p \in P} \min_{s \in \mathcal{S}} \text{dist}(p, s)^2$ for k-means. The points of $\mathcal{S}$ are called *centers*, and the *cluster* induced by $c \in \mathcal{S}$ is the set of input points closer to c than to any other center of $\mathcal{S}$. In case where a point p is in c's cluster, we sometimes say that c is p's center.

Why studying those problems? A brief data analysis motivation. One motivation for those problems is that centers of a good solution can be used as representatives for the dataset. More precisely, a center c be used in place of its cluster, to find a compressed representation of the dataset. The question is how to assess the quality of that representation.

In the k-median case, the objective is equivalent to minimizing the average distance to the closest center. Hence, we can compare the cost of solution $\mathcal{S}$ to the average pairwise distance in the dataset: when $\mathcal{S}$ is a good solution to k-median, we may be able say that the average distance to the center is smaller compared to the average distance of the dataset. The center are in that sense good *representative* of the dataset.

The k-means objective attracted a particular attention, maybe due to its closeness with the standard mean. An optimal solution minimizes *variance* – another widely used statistical quantity – within each cluster, and centers are representative according to that notion. The k-means objective is appealing for another reason: a set of clusters induced by an optimal (or near-optimal) solution may corresponds to a pre-existing structure of the data. For instance, minimizing the k-means objective allows to recover some *ground-truth* clusters, when those are *well-separated*[2] [114]. Hence, under some conditions the structure induced by a k-means solution is much more interesting than the centers only.

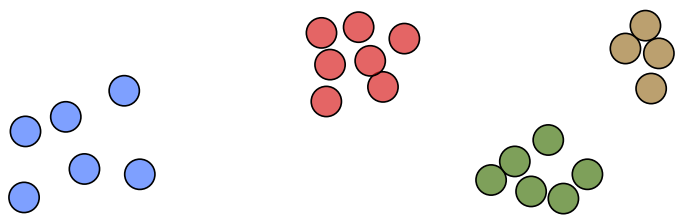


Figure 1.2: A set of points with a clear clustering into 4 clusters, well separated. An optimal k-means solution would recover those 4 ground-truth clusters.

[2] the condition roughly state that points in two separate clusters must be far away compared to the variance within each cluster.

Another motivation for k-means: quantization. Another motivation for k-means is worth mentioning. A common problem in signal processing is to discretize a continuous signal $(s(t))_{t\in\mathbb{R}}$, under the following constraint: the signal is observed at discrete points $t_1, t_2, ...$, and the precise value $s(t_i)$ cannot be stored – maybe, due to calibration constraint of the measure instruments. Instead, when $s(t)$ is in some range Q_i, we can only store a value q_i. For instance, if the signal is made of real number, one way of discretizing is to use $Q_i = [i, i+1)$ and $q_i = i$: a real number x is truncated to keep only its integer part. Quantization is used virtually in all digital signal processing, as a digital representation must be discrete while most signal are continuous. Similarly, it can also be used as means of compression.

The relation with our k-means problem is the following: suppose the Q_i and q_i are not fixed, and one seeks to find the best partition into k sets Q_i, and the best choice of q_i values for those sets. More precisely, the best partition in the following sense: the one that minimizes the *quantization error*, which is the sum of the squared distance from $s(t_i)$ to its corresponding q_{t_i}. It turns out that those bests Q_i and q_i corresponds to the optimal k-means clustering of the points $s(t_1), s(t_2), ...$, as the squared distances from $s(t_i)$ to the center is precisely the quantization error.

For instance, in computer graphics, this may corresponds to the following task: given an image, reduce the number of different colors to k – such as to find a compressed representation of the image. The k colors can be chosen using k-means, to minimize the quantization error. If one wants to minimize the average error instead, then k-median is the right tool.

Hardness of the problems. The k-median and k-means problems are hard to solve in most cases: it is NP-hard, i.e., under the common assumption P $\neq$ NP it is not possible to compute the optimal solution in time polynomial in the input size. This hardness holds even when the data merely consists of points lying in the Euclidean plane (see Mahajan, Nimbhorkar and Varadarajan [128] for k-means, Megiddo and Supowit [135] for k-median), or when $k = 2$ (Dasgupta and Freund [61]).

This implies that it is necessary to go *beyond worst case* in order to get provably good result, or to give up with the objective of computing the optimal solution. For instance, one can try to find an *approximate* solution, instead of the optimal, namely a solution that has cost within a small multiplicative factor to the optimal one – and the goal is to have the smallest multiplicative factor possible. Another approach is to focus on the case where there is a *clear* optimal clustering, making some assumption on the input in order to recover the optimal solution. We describe in more details both the hardness results and the way of bypassing them in Section 1.4. In short, although the problem is hard in the general case, we have many different ways to deal with it.

However, another source of difficulties that we have already mentioned comes from the practical aspect of the problem: we seek algorithms that can be applied to gigantic datasets. The standard polynomial time algorithms are not fast enough: we are looking for algorithms that are at worst near-linear, but that could also run in distributed environment – where the dataset is so big that it cannot be stored in a single

machine – or parallel environments – where the computation is carried simultaneously by different machines.

Goal of this thesis. In this thesis, our broad objective is to present *fast* algorithms with good provable guarantees for the k-median and k-means problem. For that, we study and use different techniques to *simplify* our clustering problem, and be able to solve it fast. The goal is to find ways of replacing the original complex and enormous dataset by a simpler one, using compression schemes that *provably* preserve the structure of the input.

Organization. In the first part, we show fast approximation algorithm constructed by means of replacing the metric space by a tree-like structure. In the second one, we present techniques to reduce drastically the number of distinct input point, and to "turn big data into tiny data".[3]

1.2 Part I: Growing Roots and Going Through the Trees

As mentioned previously, to deal with big data it is crucial to develop fast algorithm. In the first part of the thesis, we consider therefore the following question:

> ▶ **Question 1.** Is there some metric space where it is possible to solve k-median and k-means in near-linear time? ◀

Besides the big data motivation, this question is interesting in its own right, as understanding the complexity and the structure of the problems is an attractive mathematical question.

A very simple class of metric spaces where this is doable is in spaces called *ultrametrics*, which essentially are metric spaces induced by trees as follows. The set P consists of leafs of a tree that has weights on vertices, and the distance between two points is the weight of their lowest-common ancestor.[4] In that case, our problem can easily be solved via a very general technique called *dynamic programming*.

Hence, to solve a k-median or k-means problem, a natural attempt is to transform the input metric space into an ultrametric. If one can find such a transformation that does not distort too much the input, this approach may allow to compute a good solution in near-linear time. Our goal is to apply this approach to Euclidean spaces.

In the Euclidean plane, the go-to method for that transformation is essentially the following: find a box enclosing all the input points, and split it into four rectangles

[3] This catch phrase comes from Feldman, Schmidt and Sohler [78].

[4] For that to define a metric, the weights have to be monotonous, in the sense that the weight of a node is smaller than the weight of its parent.

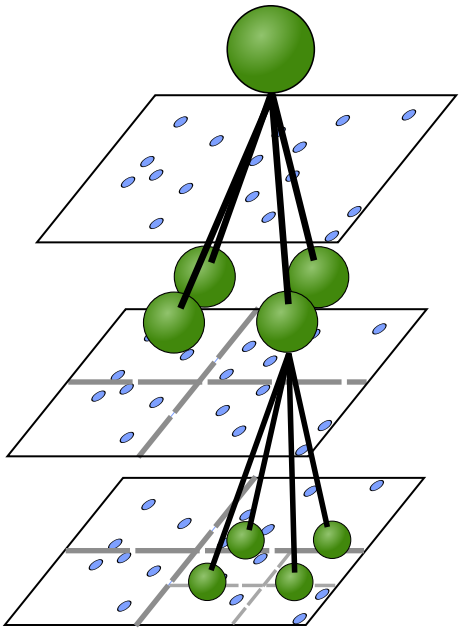


Figure 1.3: A hierarchical partitioning of an input in $\mathbb{R}^2$. The input is represented by the inclined plane, and it is recursively partitioned by the grey dashed lines. The corresponding tree nodes are in green, and the edges are the thick black lines. To lighten the figure, we only recursively split one of the regions.

of roughly the same area. Create a root node for the enclosing box, and a children for each of the rectangle. The weight of the root is set to be the diameter of the input. The recursive application of that construction yields a hierarchical partitioning of the input, which can be represented by a tree – see Fig. 1.3 for a illustration. The ultrametric induced by this tree may be used to replace the original metric. When replacing a metric by another one, we say that we *embed* the original metric into the new one: here, we embed the Euclidean plane into an ultrametric.

It can be shown that this embedding preserves on average distances between the input points, up to some small factor. Furthermore, it can be efficiently computed in near-linear time. Hence, this allows to solve distance-based problems fast, using the tree structure instead of the original metric, with only a constant factor loss: for instance, the k-median problem can be dealt with using this approach. However, distances squared may not be preserved as well as simple distances,[5] and it is not clear how to use the approach for the k-means problem. In the first part of this thesis, we study the efficiency of that ultrametric embedding for clustering, in particular focusing on application to k-means clustering.

More precisely, we study in Chapter 3 the performance of the aforementioned decomposition for clustering in *low-dimensional* Euclidean spaces, and answer positively Question 1 for those. While this low-dimensionality may seem far from the aforementioned "big data" motivations, where data is often very high-dimensional, studying

[5]This is due to the fact that distances are only preserved *on average*. If two points at distance 1 have probability $1/n$ to be at distance n in the split tree, and probability $1 - 1/n$ to be at distance 1, then their distance is on average 2 while their average squared distance is $n + 1 + 1/n$.

the problem with that angle allows us to get insight that can later be used in the large scale, big data setting. We use them in particular to provide a scalable algorithm for k-median and k-means that respect a notion of privacy of the input, in Chapter 4.

This is a key application: as data collection appears everywhere in our lives, people and more generally democracies are concerned with the effect of data analysis can have on privacy. Laws are now enforcing companies to respect some privacy principles while collecting and analyzing data: hence, it becomes necessary to develop data analysis algorithms that respect in some sense the privacy of users. This has been modeled by the notion of *Differential Privacy*, that we will explore in Chapter 4: it turns out that the embedding into ultrametrics and the techniques we introduce in Chapter 3 are particularly suited to that model. Leveraging our techniques, we show and experiment a practical private algorithm for clustering that enjoys provable guarantees.

1.3 Part II: Coping with Big Data

The other theme of this thesis is to find compression schemes for clustering. As described before, datasets are in many practical cases too large to be processed conventionally, as the data simply does not fit into one computer's memory. Henceforth, sketching, compression, and summarization techniques are at the heart of modern data analysis. This has led to new algorithms operating in other models of computation such as streaming, distributed computing or massively-parallel computation (MPC). For these algorithms, finding good small-size representations – also called *sketches* – of the input data is key.

We illustrate the notion of sketches in Fig. 1.4. The figure presents a very simple sketch, in the case where the input lies in the unit ball of the 2-dimensional Euclidean space. Its construction is as follows. Build a grid of the unit ball with granularity ε – i.e., all points whose coordinates are multiples of ε –, and snap every input point to the closest grid point. In place of the input, one can use the grid points, with weights corresponding to the number of snapped points. This is quite a good representation of the initial dataset: indeed, the grid ensures that the distance from any point to its closest grid point is at most ε. Hence, the distance of an input point to any solution $\mathcal{S}$ is the same, up to an additive ε, before and after snapping. In turn, the cost of the solution $\mathcal{S}$ is the same for the sketch and the original input dataset, up to a tiny error $\varepsilon \cdot n$ for k-median, where n is the number of input points.[6] Furthermore, the number of distinct points after snapping is the number of grid points, which is $(1/\varepsilon)^2$: this is independent of n and k, so we consider it to be a very good sketch.

In general, we will be looking for sketches with those properties: we want the sketch to have roughly the same cost as the initial dataset for any k-median (or k-means) solution, and we want the sketch to be as small as possible. Such sketches are known

[6] This additive error is actually too large, but we will consider it satisfactory for sake of simplicity in this introduction.

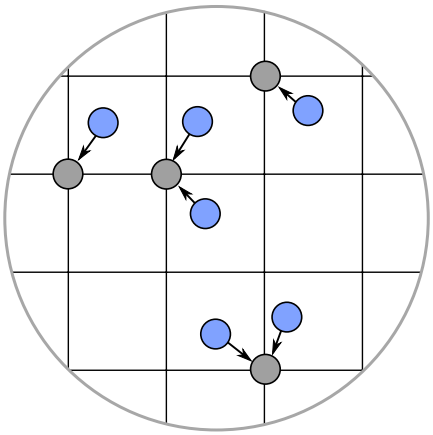


Figure 1.4: A sketch of the unit ball, based on a grid. Blue input points are mapped to the closest grid intersection, in grey. Several input point can be mapped to the same grid point, reducing the number of distinct points – hence *sketching* the input.

as *coresets*, and they are the central object of the second part of this thesis. In this context, the quality of a sketch is usually quantified in terms of its size, i.e.: the dependency on the number of input points n, the dimension d of the underlying metric (when there is one), the number of centers k, and a precision parameter ε.

Coresets allow to turn big data into tiny data, with numerous advantages. First, reducing the size of the input may allow to reinstate the "traditional" algorithms, that are already well analyzed and understood. Second, coreset can be used in settings where there are additional constraints on the memory usage of algorithms: for instance, when the input cannot fit in a single machine and is distributed among several of them, those can merely exchange coresets to communicate their data. This has small size compared to the full input and consequently can be stored and processed in a single machine, instead of the full dataset.

Going back to our example, the grid-based coreset of Fig. 1.4 is great for the Euclidean plane, but unfortunately the size of a grid scales exponentially with the dimension: in $\mathbb{R}^d$, the size of the grid is ε^{-d}, which gives a coreset of the same size. We want at all cost to avoid such a dependency – in big data, the dimension d is often at least a few hundreds, so the grid has enormous size, way larger than the number of atoms in the universe and the initial size of the input. Snapping the input onto the grid would therefore not reduce its size, and this sketching method is useless.

Thankfully, coresets are not doomed to have this exponential dependency in d: recent efforts to design even smaller sketches have resulted in an impressive success story, producing coresets of size *independent* both of the dimension and the number of input points. Those coreset algorithms are very simple, and have high impact in practice. However, it seems that the current set of techniques reached a barrier: they are increasingly tedious to analyze, and for some cases coreset construction cannot be improved using those.

In spite of the success of coreset constructions, our understanding of coreset lower bound is limited. Hence, it is not clear whether the aforementioned difficulties are

inherent to the techniques used, or to the problem itself. In the second part of this thesis, we try to answer systematically the following broad questions:

▶ **Question 2.** What are the best coreset size possible, for k-median and k-means? What particular structure on the metric space is useful to construct small coreset? ◀

1.4 Broad Background

The k-median problem defined above seeks to minimize a sum of distances; the k-means a sum of distances squared. There does not seem to be any particular reason to not consider other powers of distances: hence, we define the common generalization of k-median and k-means that is the main problem studied this thesis.

▶ **Definition 1.1 ((k, z)-clustering problem).** Given an ambient metric space (X, dist), a set of points $P \subseteq X$ (sometimes called *clients*), and positive integers k and z, the goal of the (k, z)-*clustering problem* is to output a set $\mathcal{S}$ of k *centers* (or *facilities*) chosen from X that minimizes

$$\text{cost}(P, \mathcal{S}) := \sum_{p \in P} \min_{c \in \mathcal{S}} (\text{dist}(p, c))^z$$

◀

Computing the optimum solution to (k, z)-clustering can easily be done by enumerating all the possible centers' location and simply keeping the best. However, there are $|X|^k$ many of them: the time required to perform the enumeration is completely impractical. Instead, we are looking for somewhat efficient algorithms, that only require a computation time polynomial in the size of the input.

One of the most common use cases of (k, z)-clustering is when the metric space is a Euclidean space, i.e., $\mathbb{R}^d$ equipped with the ℓ_2 norm. Most of the work of this thesis is therefore dedicated to that particular case: the key parameters here are only the size of the input set P, the dimension d of the space and, of course, the number of clusters k. As the metric space is described implicitly, the size of the input is indeed simply $|P|d$, to write the d coordinates of each input point. In other cases, the size of the input is essentially the size of the metric space X. Note a huge difference: in Euclidean spaces, the metric space is infinite, and it is not even possible to enumerate all possible center's location, hence for instance not obvious how to find the optimum center for $(1, z)$-clustering. In contrast, when the size of the metric is finite, it is easy to find this optimum center in $|X| \cdot |P|$ time.

As we mentioned, the (k, z)-clustering problem is a difficult one: it is not only NP-

	Discrete k-median	Discrete k-means	Euclidean k-median	Euclidean k-means
Lower bound	$1 + 2/e$ [90]	$1 + 8/e$ [90]	1.06 [56]	1.015 [56]
Upper bound	2.675 [37]	6.36 [5]	2.41 [52]	5.96 [52]

Figure 1.5: Approximability of k-median and k-means. The lower-bounds are conditioned on the assumption P $\neq$ NP.

hard to compute the optimal solution, but also NP-hard to compute an approximate one. NP-hardness is a conditional notion of difficulty: solving in polynomial time a NP-hard problem would break a widely believed conjecture, namely P $\neq$ NP. Fig. 1.5 summarizes the current state of our knowledge in terms of polynomial-time approximability: we say that a solution $\mathcal{S}$ is an α-approximation if its cost is at most α times the cost of the optimal solution.

Hence, if one wishes to have a polynomial-time algorithm with very good approximation guarantee for (k, z)-clustering, it is necessary to make some assumption on the input data.

Beyond Worst Case The previous inapproximability results are *worst-case*, in the sense that there *exist* instances where it is not possible to get a good approximation in polynomial time. However, not all instances are equally hard: a line of work tried to defined some natural conditions under which it is possible to recover the optimal (or a near-optimal) solution in polynomial time. For instance, this is doable in discrete metrics, when the optimal solution stays optimal even when all distances are perturbed by a small factor [57]. A notion of *stability* has been defined by Ostrovski, Rabani, Schulman and Swamy [148], inspired by the idea that one should try to solve (k, z)-clustering when there are precisely k identifiable clusters: finding the optimal solution is possible when it is significantly cheaper to cluster with k centers than with $k - 1$ centers [57]. As a last example, if the input consist of well separated cluster in a Euclidean space, in the sense that the distance between points in two different cluster is very large compared to the average cost of the optimal solution, then it is possible to find those optimal clusters using k-means [114].

Restricting the metric space. Another way of bypassing the hardness results is to restrict the input metric space, for instance by considering fixed-dimensional Euclidean spaces. As mentioned, the problem is however already NP-hard in the Euclidean plane, when $d = 2$ ([128], [135]). Nonetheless, it is possible to compute approximate solution as precise as one wishes: for any fixed $\varepsilon > 0$, one can compute in polynomial time a $(1 + \varepsilon)$-approximation to (k, z)-clustering (see Kolliopoulos and Rao [112], Cohen-Addad, Klein, Mathieu [53] or Friggstad, Rezapour and Salavatipour [86]). Here, by polynomial time we mean $|P|^{f(d,\varepsilon)}$ for any function f, as ε and d are considered to be fixed.

A standard generalization of the Euclidean dimension of a space that abstracts out a lot of the geometry and allows us to focus on the most crucial properties is called the *doubling dimension*. The doubling dimension of a metric is the logarithm of the

number of balls of radius $R/2$ necessary to cover any ball of radius R, for any R: this indeed generalizes Euclidean space, as the space $(\mathbb{R}^d, \ell_2)$ has doubling dimension d (up to constant factors). In case of bounded doubling dimension, Friggstad, Rezapour and Salvatipour [86] showed how to compute a $(1+\varepsilon)$-approximation.

It is also possible to compute $(1+\varepsilon)$ approximations in planar graphs, and more generally minor-excluding graphs [53].

Fast and Practical Algorithms The view practitioners have on the problem is however slightly different: instead of looking for an algorithm with close to optimal approximation guarantee, they are looking for *fast* algorithms that *seem* to provide good result. Moreover, most of the practical work focuses on Euclidean spaces.

The most celebrated such algorithm is probably Lloyd's heuristic [123] for k-means, introduced in the context of quantization we mentioned above. This heuristic works as follows: starting from a set of centers, one can define a clustering, by associating each input point to its closest center. Now, for each cluster, the best possible center is the mean of points in the cluster: this defines k new centers. It is possible to iterate many times those two steps, to compute some solution to k-means. Unfortunately, this algorithm may take exponential time to converge, even in two dimensions (see Vattani [158]). Sadly, even when it has converged, this algorithm does not guarantee to provide a good solution, as it is very sensitive to the initial choice of centers (see the study of Milligan [138]). Consequently, finding good initialization method was investigated: Lloyd's algorithm can always be run as a post-processing step of an algorithm, to improve the quality of the solution.

The k-means++ algorithm of Arthur and Vassilvitski [10] is the most common initialization, and is a perfect example of fast algorithm with not-so-good guarantees. It runs in $O(ndk)$ time – same as one iteration of Lloyd's algorithm, and although its approximation guarantee is only $O(\log k)$ it is used as a comparison baseline for most of the k-means practical work.

The k-means++ algorithm is also extremely simple: it computes centers in rounds, as follows. In the first round, an arbitrary input point is selected as a center; then, at each step, one new center is selected with probability proportional to its *squared* distance to the selected centers. After k rounds, this yields an $O(\log k)$ approximation. Furthermore, to make this algorithm work for the more general (k, z)-clustering problem, one simply need to select centers with probability proportional to their distance raised to the power z (see the analysis of Wei [160]).

A vast literature followed on the k-means++ algorithm, either trying to explain why it seems to give solutions way better than the $O(\log k)$ approximation guaranteed, or by modifying it in order to improve performances. For instance, Agarwal, Jaiswal and Pal [2] showed improved guarantees when the instance verifies some form of stability criterion. If one is willing to open slightly more centers, after $O(k)$ rounds the cost of the solution is within a constant of the optimal one that uses only k centers, as shown in [3, 160, 130, 45]. A parallel version of this algorithm was introduced by Bahmani, Moseley, Vattani, Kumar and Vassilvitskii [14], with the same guarantee as the k-means++ algorithm. Bachem, Lucic, Hassani and Krause [11] showed how

to approximate the k-means++ output distribution in sublinear time. Finally, Lattanzi and Sohler [117] improved the approximation guarantee of the algorithm: they combined k-means++ with another simple algorithm, Local Search, to get a constant factor approximation.

1.5 Our Contribution, Chapter by Chapter

• Chapter 2 is a mere introduction to the approximation algorithm techniques that are use in Part I. In this introductive chapter, we present the context of our work in that part, and the challenges we address.

• Chapter 3 presents an approximation scheme for (k, z)-Clustering in Euclidean spaces, namely an algorithm that computes a $(1 + \varepsilon)$-approximation to the problem. For any fixed ε and dimension, this algorithm runs in near-linear time: this gives an answer to Question 1. This is based on a joint work with Vincent Cohen-Addad and Andreas Feldmann [A4], that appeared in the Journal of the ACM.

• Chapter 4 applies the techniques developed in Chapter 3 to get a private algorithm for k-median and k-means with provable approximation guarantee. This chapter is more oriented towards practice: we present a scalable algorithm, able to run in a distributed setting. In particular, we implemented the algorithm and showed its practical efficiency both in terms of speed and quality of the computed solution. This is based on a collaboration with Vincent Cohen-Addad, Alessandro Epasto, Silvio Lattanzi, Vahab Mirrokni, Andres Munoz, Chris Schwiegelshohn and Sergei Vassilvitskii, that will be presented at the conference KDD 22 [A3]

• Chapter 5 is an extensive introduction to coreset and sketches, theme of Part II. We present an overview of the techniques used in the subsequent chapters, as well as a more precise description of the coreset literature.

• In Chapter 6, we present our main framework for coreset construction. The principal contribution of that chapter is to reduce the construction of a coreset to showing the existence of *approximate centroid sets*, that are somewhat the twin of coreset for set of centers. In broad terms, a set $\mathbb{C}$ is an approximate centroid set if for any possible candidate solution $\mathcal{S}$, there is a set $\tilde{\mathcal{S}} \in \mathbb{C}^k$ such that for any point p, $\mathrm{dist}(p, \mathcal{S}) = (1 \pm \varepsilon)\mathrm{dist}(p, \tilde{\mathcal{S}})$. Hence, $\tilde{\mathcal{S}}$ can be used in place of $\mathcal{S}$. We present an algorithm that builds coreset of size $O(\varepsilon^{-2}k \log |\mathbb{C}|)$, when the input admits an approximate centroid set $\mathbb{C}$. Crucially, this algorithm does not need to compute $\mathbb{C}$: its mere existence suffices. Hence, to construct small coresets, one only needs to show the existence of small approximate centroid set, and apply our algorithm.

• In Chapter 7, we show the existence of approximate centroid sets for various metric spaces, resulting in state of the art coreset construction. For discrete metrics, an easy approximate centroid set is the full metric, of size n. This result in coreset of size essentially $O(\varepsilon^{-2}k \log n)$ For metrics of doubling dimension d, their existence is a straightforward consequence of the existence of nets, which yield coreset of size $O(\varepsilon^{-2}kd)$. The Euclidean space $\mathbb{R}^d$ is known to have doubling dimension $\Theta(d)$: the

result carries over. It can be further improved using standard dimension reduction techniques: it is possible to replace the dependency in d by $O(\varepsilon^{-2} \log k)$. We also show that metrics induced by graphs with small *separators* have small centroid set: namely, metric induced by graph of bounded treewidth or excluding a minor. This chapter and the previous one are based on an article published at STOC 2021 with Vincent Cohen-Addad and Chris Schwiegelshohn [A11]

• In Chapter 8, we show that our construction for discrete and doubling metrics are tight: there exist a family of discrete metric space such that any coreset on those must have size at least $\Omega(\varepsilon^{-2} k \log n)$. The proof is based on the optimality of Chernoff bounds. This completes our understanding of coreset for those spaces: upper and lower bounds are tight, and the proofs both rely heavily on Chernoff bounds, simple properties on sum of independent random variables. This is part of a joint work with Vincent Cohen-Addad, Kasper Green Larsen and Chris Schwiegelshohn [A8], that was presented at STOC 2022.

• Chapter 9 presents a determinstic coreset construction. The previous coreset constructions are randomized, and succeed with probability $1 - \delta$. However, we do not know how to verify that the outcome of a randomized coreset construction is indeed a valid coreset: hence, determinism may be a desirable property. We present such coresets construction for various metric spaces, as in Chapter 7.

In particular, to achieve deterministic bounds similar to the randomized one in Euclidean spaces, one needs to remove any dependency on the dimension d: one of the technical ingredients of the chapter is to show deterministic dimension reduction for clustering. This is of independent interest, and provides another way of sketching Euclidean input. This is a collaboration with Vincent Cohen-Addad and Chris Schwiegelshohn, currently under submission [A13].

• Finally, in Chapter 10 we use the coreset knowledge developed in the previous chapters to show algorithms running in sublinear time to compute the median, the mean and more generally an approximation to $(1, z)$-clustering. We show that it is enough to consider a *constant* number of input point drawn uniformly at random to compute this solution, and experiment our algorithm to show the practical speed-up it allows. This chapter is based on a work that was presented as a spotlight at NeurIPS 2021, with Vincent Cohen-Addad and Chris Schwiegelshohn [A12].

1.6 How to Read this Thesis

The manuscript is written so that all chapters can be read independently, to allow the reader to directly access a specific result of interest. However, it is recommended to read the introduction of the corresponding part first: the introductory chapters of each part present the problems addressed, and attempt to explain their significance. They also give an overview of the techniques used in the subsequent chapters.

It is also recommended to read the next preliminary section, in which we present

notations and a couple of lemmas used all along the manuscript.

Publications of the Author

[A1] Amariah Becker, Philip N. Klein, and David Saulpic. "A Quasi-Polynomial-Time Approximation Scheme for Vehicle Routing on Planar and Bounded-Genus Graphs". In: *European Symposium on Algorithms, ESA*. 2017.

[A2] Amariah Becker, Philip N Klein, and David Saulpic. "Polynomial-Time Approximation Schemes for k-center, k-median, and Capacitated Vehicle Routing in Bounded Highway Dimension". In: *European Symposium on Algorithms, ESA*. 2018.

[A3] Vincent Cohen-Addad, Alessandro Epasto, Silvio Lattanzi, Vahab Mirrokni, Andres Munoz, David Saulpic, Chris Schwiegelshohn, and Sergei Vassilvitskii. "Scalable Differentially Private Clustering via Hierarchically Separated Trees". In: To Appear at the Conference on Knowledge Discovery and Data Mining (KDD). 2022.

[A4] Vincent Cohen-Addad, Andreas Emil Feldmann, and David Saulpic. "Near-linear Time Approximation Schemes for Clustering in Doubling Metrics". In: *J. ACM*. Vol. 68. 2021.

[A5] Vincent Cohen-Addad, Anupam Gupta, Lunjia Hu, Hoon Oh, and David Saulpic. "An Improved Local Search Algorithm for k-Median". In: *Symposium on Discrete Algorithms, SODA*. 2022.

[A6] Vincent Cohen-Addad, Niklas Hjuler, Nikos Parotsidis, David Saulpic, and Chris Schwiegelshohn. "Fully Dynamic Consistent Facility Location". In: *NeurIPS*. 2019.

[A7] Vincent Cohen-Addad, Adrian Kosowski, Frederik Mallmann-Trenn, and David Saulpic. "On the Power of Louvain in the Stochastic Block Model". In: *NeurIPS*. 2020.

[A8] Vincent Cohen-Addad, Kasper Green Larsen, David Saulpic, and Chris Schwiegelshohn. "Towards Optimal Lower Bounds for k-median and k-means Coresets". In: *STOC '22*. 2022.

[A9] Vincent Cohen-Addad, Frederik Mallmann-Trenn, and David Saulpic. "A Massively Parallel Modularity-Maximizing Algorithm With Provable Guarantees". In: *Symposium on Principles of Distributed Computing (PODC)*. 2022.

[A10] Vincent Cohen-Addad, Frederik Mallmann-Trenn, and David Saulpic. "Community Recovery in the Degree-Heterogeneous Stochastic Block Model". In: *Conference on Learning Theory (COLT)*. 2022.

[A11] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. "A new coreset framework for clustering". In: *Symposium on Theory of Computing (STOC)*. 2021.

[A12] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. "Improved Coresets and Sublinear Algorithms for Power Means in Euclidean Spaces". In: *NeurIPS*. 2021.

[A13] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. "On Deterministic Clustering Sketches". In: submitted.

[A14] Andreas Emil Feldmann and David Saulpic. “Polynomial time approximation schemes for clustering in low highway dimension graphs”. In: *J. Comput. Syst. Sci.* Vol. 122. 2021.

[A15] Niklas Hjuler, Giuseppe F. Italiano, Nikos Parotsidis, and David Saulpic. “Dominating Sets and Connected Dominating Sets in Dynamic Graphs”. In: *International Symposium on Theoretical Aspects of Computer Science, STACS*. 2019.

Preliminaries: Notations and Useful Results

To conclude this introduction, we introduce some notations used all along the thesis. Given a point p and a set $\mathcal{S}$, we define $\text{dist}(p, \mathcal{S}) = \min_{s \in \mathcal{S}} \text{dist}(p, s)$, and $\text{cost}(p, \mathcal{S}) = \text{dist}(p, \mathcal{S})^z$ for (k, z)-clustering problem. Given a set of point P and a solution $\mathcal{S}$, we define $\text{cost}(P, \mathcal{S}) := \sum_{p \in P} \text{cost}(p, \mathcal{S})$ and, in the case where P contains all the points of the metric space, we define $\text{cost}(\mathcal{S}) := \text{cost}(P, \mathcal{S})$. In the case where P is weighted with weight $w : P \to \mathbb{R}^+$, then $\text{cost}(P, \mathcal{S}) := \sum_{p \in P} w(p)\text{cost}(p, \mathcal{S})$.
To deal with powers of distances, we will need a variant of the triangle inequality, stated as follows:

▶ **Lemma 1.2 ([129]).** Let a, b, c be arbitrary points in a metric space with distance function dist, $\mathcal{S}$ be an arbitrary set of points in the same metric space, and let z be a positive integer. Then for any $\varepsilon > 0$

$$\text{dist}(a, b)^z \leq (1+\varepsilon)^{z-1}\text{dist}(a, c)^z + \left(\frac{1+\varepsilon}{\varepsilon}\right)^{z-1} \text{dist}(b, c)^z$$

$$|\text{dist}(a, \mathcal{S})^z - \text{dist}(b, \mathcal{S})^z| \leq \varepsilon \cdot \text{dist}(a, \mathcal{S})^z + \left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1} \text{dist}(a, b)^z.$$

◀

Proof. The proof of the first inequality is from Makarychev, Makarychev and Razenshteyn [129], Corollary A.2. We have, for two real number x, y and $t := \frac{1}{1+\varepsilon}$:

$$(x+y)^z = \frac{1}{t^z}\left(tx + (1-t)\left(\frac{ty}{1-t}\right)\right)^z$$

The right hand side is a convex combination of x and $\frac{ty}{1-t}$, and the function $x \to x^z$ is convex (remember that $z \geq 1$). Hence, using Jensen's inequality, we get

$$\begin{aligned}(x+y)^z &\leq \frac{tx^z}{t^z} + \frac{(1-t)}{t^z} \cdot \left(\frac{ty}{1-t}\right)^z \\ &\leq (1+\varepsilon)^{z-1}x^z + \left(\frac{1}{1-t}\right)^{z-1} y^z \\ &\leq (1+\varepsilon)^{z-1}x^z + \left(\frac{1+\varepsilon}{\varepsilon}\right)^{z-1} y^z.\end{aligned}$$

Applying this with $x = \text{dist}(a, b)$ and $y = \text{dist}(b, c)$ and using the triangle inequality concludes the first statement of the lemma.

For the second part, let $\mathcal{S}(a), \mathcal{S}(b)$ be the closest point to a and b from $\mathcal{S}$, and assume that $\text{dist}(b, \mathcal{S}) \leq \text{dist}(a, \mathcal{S})$. Then:

$$\begin{aligned}
\text{dist}(a, \mathcal{S})^z &\leq \text{dist}(a, \mathcal{S}(b))^z \\
&\leq \left(1 + \frac{\varepsilon}{2z}\right)^{z-1} \cdot \text{dist}(b, \mathcal{S}(b))^z + \left(1 + \frac{2z}{\varepsilon}\right)^{z-1} \cdot \text{dist}(a, b)^z \\
&\leq (1 + \varepsilon) \cdot \text{dist}(b, \mathcal{S}(b))^z + \left(1 + \frac{2z}{\varepsilon}\right)^{z-1} \cdot \text{dist}(a, b)^z \\
&\leq \text{dist}(b, \mathcal{S})^z + \varepsilon \cdot \text{dist}(a, \mathcal{S}(a))^z + \left(1 + \frac{2z}{\varepsilon}\right)^{z-1} \cdot \text{dist}(a, b)^z,
\end{aligned}$$

and so

$$|\text{dist}(a, \mathcal{S})^z - \text{dist}(b, \mathcal{S})^z| = \text{dist}(a, \mathcal{S})^z - \text{dist}(b, \mathcal{S})^z \leq \varepsilon \cdot \text{dist}(a, \mathcal{S})^z + \left(\frac{2z + \varepsilon}{\varepsilon}\right)^{z-1} \text{dist}(a, b)^z.$$

In the other case, when $\text{dist}(a, \mathcal{S}) \leq \text{dist}(b, \mathcal{S})$:

$$\begin{aligned}
\text{dist}(b, \mathcal{S})^z &\leq \text{dist}(b, \mathcal{S}(a))^z \\
&\leq \left(1 + \frac{\varepsilon}{2z}\right)^{z-1} \cdot \text{dist}(a, \mathcal{S}(a))^z + \left(1 + \frac{2z}{\varepsilon}\right)^{z-1} \cdot \text{dist}(a, b)^z \\
&\leq (1 + \varepsilon) \cdot \text{dist}(a, \mathcal{S})^z + \left(1 + \frac{2z}{\varepsilon}\right)^{z-1} \cdot \text{dist}(a, b)^z,
\end{aligned}$$

and so

$$|\text{dist}(a, \mathcal{S})^z - \text{dist}(b, \mathcal{S})^z| = \text{dist}(b, \mathcal{S})^z - \text{dist}(a, \mathcal{S})^z \leq \varepsilon \cdot \text{dist}(a, \mathcal{S})^z + \left(\frac{2z + \varepsilon}{\varepsilon}\right)^{z-1} \text{dist}(a, b)^z.$$

□

All the algorithms presented in this thesis start by computing a constant-factor approximation to the solution. For that, our main tool is a fast algorithm due to Mettu and Plaxton:

> ▶ **Lemma 1.3.** [Mettu and Plaxton, [136]] Given a metric space on n points with oracle access to distances, there exists an algorithm running in time $O(n \cdot \max(k, \log n))$ that computes, with constant probability, a $O(2^z)$-approximation to (k, z)-clustering. ◀

Although their theorem is stated only for k-median, the authors note (similarly as [100]) that the proof directly translate to k-means and other powers, using Lemma 1.2 with $\varepsilon = 1$, losing a 2^z factor in the approximation guarantee.

Part I

Approximation Algorithm via Embeddings Into Tree-like Structures

Chapter 2

Presentation of the Results and Challenges

In this section, our primary goal is to show *fast* algorithms for (k, z)-clustering in Euclidean space, and a generalization of them called doubling metrics,[1] by leveraging embedding into ultrametrics. Maybe the most natural way of constructing such an embedding from the Euclidean Space is to separate the Euclidean input into two parts by a axis-parallel hyperplane, do that again on each part to define four finer parts, and continue recursively to obtain a series of partition of the input into a finer and finer level of granularity, as illustrated in Fig. 1.3.

As we saw in the introduction, this decomposition naturally defines an embedding into an ultrametric, where the distance between two points is the diameter of the smallest part that contains both. While the introduction focused for simplicity on this simple embedding, which is useful end of itself as we will see in Chapter 4, we will need to use a slightly different embedding in order to compute very precise solutions. Indeed, this one inherently yields a (large) constant distortion. In Chapter 3, we will leverage the same recursive decomposition of the Euclidean plane to find a slightly different embedding, allowing for fast and very precise algorithms for (k, z)-clustering.

2.1 Results Presented in the Part

A different viewpoint on the previous embedding is as follows. In each part of the partition, there is a single *portal* placed at the center of the part. The Euclidean metric is distorted: instead of straight lines, a path that connects two points is twisted such that it can enter or exit a part only through portals. In Chapter 3, we use the standard

[1] The doubling dimension of a metric space is the logarithm of the number of balls of radius $R/2$ required to cover any ball of radius R, see Definition 3.8. In particular, d-dimensional Euclidean Spaces have doubling dimension d (up to constants).

idea of placing more than a single portal in each part: instead of distorting distances by a large constant factor, this idea allows to preserve them up to an arbitrary small precision; and preserving the hierarchical tree structure of the decomposition still allows us to solve the problem fast.

Those portals are at the heart of previous results. There has been indeed a large body of work to design good approximation for clustering in Euclidean Spaces of bounded dimension, and more generally in metrics of fixed doubling dimension. In their seminal work introducing the use of portals for k-median, Arora, Raghavan and Rao [9] gave a polynomial time approximation scheme (PTAS) for k-Median in $\mathbb{R}^2$, namely for any $\varepsilon > 0$, they show an algorithm that computes a $(1+\varepsilon)$-approximation in polynomial time.[2] This generalizes to a quasi-polynomial time approximation scheme (QPTAS) for inputs in $\mathbb{R}^d$, with fixed d. This result was improved in two ways. First by Talwar [155] who generalized the QPTAS to any metric space of fixed doubling dimension. Second by Kolliopoulos and Rao [112] who obtained an $f(\varepsilon, d) \cdot n \log^{d+6} n$ time algorithm for k-Median in d-dimensional Euclidean space. Unfortunately, Kolliopoulos and Rao's algorithm relies on the Euclidean structure of the input and does not immediately generalize to low dimensional doubling metric. Thus, using this framework the only result known for k-Median in metrics of fixed doubling dimension was only a QPTAS. Moreover, all those algorithms rely on the decomposition and the expectation-based argument sketched above. As we will see shortly, those algorithms cannot work for the k-Means problem, when distances are squared. Thus no efficient algorithms were known for the k-Means problem, even in the plane.

Somewhat recently, Friggstad, Rezapour and Salavatipour [86] and Cohen-Addad, Klein and Mathieu [53] moved away from this embedding framework, and showed that the classic local search algorithm for the problems gives a $(1+\varepsilon)$-approximation in time $n^{d/\varepsilon^{O(d)}}$ in metrics of doubling dimension d. More recently still, Cohen-Addad [50] showed how to speed up the local search algorithm for Euclidean space to obtain a running time $nk(\log n)^{(d/\varepsilon)^{O(d)}}$.

Nonetheless, obtaining an efficient approximation scheme (namely a $(1+\varepsilon)$-approximation algorithm running in time $f(\varepsilon, d)\text{poly}(n)$) for k-Median and k-Means in metrics of doubling dimension d has remained a major challenge.

The main result of Chapter 3 is the following (informal) theorem:

> ▶ **Informal Theorem (see Theorem 3.1 and Theorem 3.2).** For any $0 < \varepsilon < 1/3$, there exists a $(1+\varepsilon)$-approximation algorithm for (k,z)-clustering problem in metrics of doubling dimension d with running time $O_{\varepsilon,d,z}(n\text{polylog}n)$. ◀

We also show similar results for several variants of the (k,z)-clustering problem, such as Facility Location, k-center or clustering with outliers.

The ultrametric techniques we develop turn out to be useful for other setting, where it is easier to deal with a tree structure than a Euclidean one. In particular, we present

[2] Note that ε is not part of the input: the complexity can be $n^{f(\varepsilon)}$.

clustering algorithms that preserves some form of privacy in Chapter 4. As privacy concerns are growing with the increasing amount of data collected, it is essential to design algorithms that respect the privacy of individuals. Differential privacy is by now the standard model of modeling privacy for algorithms: essentially an algorithm is differentially private if its outcome does not change when a single input point changes.

In Chapter 4, we use the embedding into ultrametrics described above to design a *differentially private* algorithm for (k, z)-clustering. Although many algorithms have been proposed for differentially private clustering, none of them benefits from both theoretical guarantees and practical efficiency. To fill this gap, we show a differentially private algorithm that has provable approximation guarantee and is amenable to large-scale implementation. We leverage for this the malleability of trees together with the techniques developed in Chapter 3 to adapt the ultrametric embedding to (k, z)-clustering and not simply k-median.

> ▶ **Informal Theorem (see Theorem 4.2 and Theorem 4.12).** There exist differentially private algorithms for k-median ($z = 1$) and k-means ($z = 2$) that compute a solution with cost $\text{polylog}(n)\text{OPT} + kd^2 \log^2 n \cdot \Lambda^z$, when all input points have norm at most Λ. The running time of those algorithms is $\tilde{O}(ndk^2)$. ◀

The privacy constraint forces us to have such an additive error: the solution computed should be roughly the same when the dataset consists of n points at the origin, or $n-1$ at the origin and a single one at distance Λ. More details on that constraint are given in the chapter.

The guarantees we obtain are somewhat far from the state-of-the art results for private clustering, for which the multiplicative factor now matches the one of non-private algorithms (See Ghazi, Kumar and Manurangsi [87]). However, those algorithms are far from being practical and even hardly implementable. On the other hand, state-of-the-art implementations either have no theoretical guarantees on the quality of the solution obtained, or cannot be implemented in large-scale scenario where the data is distributed. In contrast, we present a parallel implementation of the algorithm, and we show experimentally that its performances are comparable to the best non-private methods.

2.2 Using Ultrametrics for Squared Distances

We discuss in slightly more details how to work with the embeddings. For simplicity, we limit our introduction here to the ultrametric embedding, instead of the more intricate one with portals, as the general ideas are the same for both. Recall that a hierarchical partitioning is constructed, represented by a tree, and the distance between two points is defined to be the diameter of the smallest part containing both

of them.

It is clear that in this ultra metric, distances can only be larger than in the Euclidean space. Hence, any solution is cheaper in the original metric than in the ultrametric. Therefore, a solution computed on the embedding that has a tiny cost compared to the optimal one is a good solution in the original space as well. To be able to use the ultrametric in place of the original one, one only needs to show that the converse is true as well, and that distances in the ultrametric are not too large compared to the original ones.

Of course, an embedding computed from an arbitrary tree may completely distort distances, by separating early points that are very close. To avoid that issue, the trees are build randomly: at each level, instead of picking a fixed separating hyperplane, it is drawn at random. That way, two points that are close compared to the diameter of the part are likely not to be separated. In turn, the expected distance in the tree metric between the two points is similar to their original distance.

That approach however fails when distances are squared: in that case, the expected squared distance in the ultrametric may be as large as n times larger than in the original metric. Indeed, in a given part of diameter n, two points at distance 1 have probability $1/n$ to be separated: in this case, their distance squared in the ultrametric is n^2. Hence, the expected distortion is at least n.

Our main contribution in this chapter is to show how to bypass this apparent impossibility, and how to use those trees for squared distances. Instead of bounding the expected distortion, as the argument mentioned above, we try to ensure a stronger property, namely that the embedding preserves the distance between all pairs of points. This is equivalent to the following: any pair of points is separated in the tree at a level where parts have diameter comparable to their original distance.

If that were the case, then both distances and squared distances would be preserved in the ultrametric, and solving k-means would henceforth be possible. This ideal case does not happen: instead, we will be able to show the slightly weaker property that *most* of input points are separated from any other point at a level comparable to their distance. Hence, for all those points, the ultrametric can be used. The others pay an extra cost, as their distances are not preserved.

We call those points the *badly-cut* points. We will show that any point has only a tiny probability of being badly-cut. Very briefly, we will leverage this fact to show the existence of a near-optimal solution in the embedding: since each point has a tiny probability of being badly-cut, the expected cost of those in any solution is a tiny fraction of the overall cost. In turn, we can afford a large error for them, and charge it to the cost of the non badly-cut points.

Showing how to deal with the badly-cut points is the main technical contribution of this part.

Chapter 3

Near-Linear Time Approximation Schemes for Clustering in Doubling Metrics

In this chapter, we consider the classic Facility Location, k-Median, and k-Means problems in metric spaces of doubling dimension d. We give nearly linear-time approximation schemes for each problem, namely algorithm that for any $\varepsilon > 0$ computes a $(1+\varepsilon)$-approximation in time near linear in n. The precise complexity of our algorithms is $\widetilde{O}(2^{(1/\varepsilon)^{O(d^2)}}n)$, making a significant improvement over the state-of-the-art algorithms which run in time $n^{(d/\varepsilon)^{O(d)}}$.

Moreover, we show how to extend the techniques used to get the first efficient approximation schemes for the problems of prize-collecting k-Median and k-Means, and efficient bicriteria approximation schemes for k-Median with outliers, k-Means with outliers and k-Center.

As the historical roots of the low dimension (k, z)-clustering problem are in Facility Locations types problems, we adopt the following terminology: input points are *clients*, and they need to be *served* by a set of *facilities*, or centers.

3.1 Introduction and Sketch of Proofs

Formally, our two main theorems are the following.

▶ **Theorem 3.1.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1+\varepsilon)$-approximation algorithm for k-Median in metrics of doubling dimension d with running time $2^{(1/\varepsilon)^{O(d^2)}} n \log^4 n + 2^{O(d)} n \log^9 n$ and success probability at least $1 - 2\varepsilon$. ◀

▶ **Theorem 3.2.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1+\varepsilon)$-approximation algorithm for (k, z)-clustering in metrics of doubling dimension d with running time $2^{(1/\varepsilon)^{O(d^2)}} n \log^5 n + 2^{O(d)} n \log^9 n$ and success probability at least $1 - O(\varepsilon)$. ◀

En route to proving those theorems, we consider the Facility Location problem, in which no bound on the number of opened centers is given, but each center comes with an opening cost. This difference allows us to design a simpler algorithm, which is a nice illustration of our techniques.

▶ **Theorem 3.3.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1+\varepsilon)$-approximation algorithm for Facility Location in metrics of doubling dimension d with running time $2^{(1/\varepsilon)^{O(d^2)}} n + 2^{O(d)} n \log n$ and success probability at least $1 - \varepsilon$. ◀

In all these theorems, we make the common assumption to have access to the distances of the metric in constant time, as, e.g., in [97, 58, 89]. It is possible to remove that assumption using the algorithm of Bartal et al. [22], who showed how to construct in near-linear time an oracle that computes distances up to a $(1+\varepsilon)$-multiplicative error. This is key to our linear-time algorithm, as simply storing the distance matrix takes $O(n^2)$ space.

Note that the double-exponential dependence on d is unavoidable unless P = NP, since the problems are APX-hard in Euclidean space of dimension $d = O(\log n)$. For Euclidean inputs, our algorithms for the k-Means and k-Median problems outperform the ones of Cohen-Addad [50], removing in particular the dependence on k, and the one of Kolliopoulos and Rao [112] when $d > 3$, by removing the dependence on $\log^{d+6} n$. Our algorithm is also more precise than the popular k-means++ algorithm, or the one from Cohen-Addad et al. [54], which run in time $\widetilde{O}(n)$ but have an approximation guarantee $O(\log k)$.

We also note that the success probability can be boosted to $1 - \varepsilon^\delta$ by repeating the algorithm $\log \delta$ times and outputting the best solution encountered.

After proving the three theorems above, we will apply the techniques to related problems. We say an algorithm is an (α, β)-approximation for k-Median or k-Means with Z outliers if its cost is within an α factor of the optimal one and the solution drops βZ outliers. Similarly, an algorithm is an (α, β)-approximation for k-Center if its cost is within an α factor of the optimal one and the solution opens βk centers. In the prize-collecting versions of the problems, each client are either connected to a facility, and pay there distance (or distance squared), or are not connected and pay some fixed penalty.

▶ **Theorem 3.4.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1+\varepsilon)$-approximation algorithm for Prize-Collecting k-Median (resp. k-Means) in metrics of doubling dimension d with running time $2^{(1/\varepsilon)^{O(d^2)}} n \log^4 n + 2^{O(d)} n \log^9 n$ and success probability at least $1-\varepsilon$. ◀

▶ **Theorem 3.5.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1+\varepsilon, 1+O(\varepsilon))$-approximation algorithm for k-Median (resp. k-Means) with Z outliers in metrics of doubling dimension d with running time $2^{(1/\varepsilon)^{O(d^2)}} n \log^6 n + T(n)$ and success probability at least $1-\varepsilon$, where $T(n)$ is the running time to construct a constant-factor approximation. ◀

We note as an aside that our proof of Theorem 3.5 could give an approximation where at most Z outliers are dropped, but $(1+O(\varepsilon))k$ centers are opened. For simplicity, we focused on the previous case only.

▶ **Theorem 3.6.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1+\varepsilon, 1+O(\varepsilon))$-approximation algorithm for k-Center in metrics of doubling dimension d, with running time $2^{(1/\varepsilon)^{O(d^2)}} n \log^6 n + n \log k$ and success probability at least $1-\varepsilon$. ◀

As explained above, this bicriteria is necessary in order to get an efficient algorithm: it is APX-hard to approximate the cost [74], and achieving the optimal cost with $(1+\varepsilon)k$ centers requires a complexity $\Omega(n^{1/\sqrt{\varepsilon}})$ [132]. To the best of our knowledge, this works presents the first linear-time bicriteria approximation scheme for the problem of k-center.

Techniques

To give a detailed insight on our techniques and our contribution we first need to quickly review previous approaches for obtaining approximation schemes on bounded doubling metrics. The general approach, due to Arora [8] and Mitchell [139], which was generalized to doubling metrics by Talwar [155], is the following.

Previous Techniques

The approach consists in randomly partitioning the metric into a constant number of regions, and applying this recursively to each region. The recursion stops when the regions contain only a constant number of input points. This leads to what is called a *split-tree decomposition*: a partition of the space into a finer and finer level of granularity. In Euclidean spaces, this decomposition is often called a *quad-tree decomposition*. A split-tree is the generalization to doubling metrics. The reader who is not familiar with the split-tree decomposition may refer to Section 3.2.2 for a more

formal introduction.

Portals The approach then identifies a specific set of points for each region, called *portals*, which allows to show that there exists a near-optimal solution such that different regions "interplay" only through portals. For example, in the case of the Traveling Salesperson (TSP) problem, it is possible to show that there exists a near-optimal tour that enters and leaves a region only through its portals. In the case of the k-Median problem a client located in a specific region can be assigned to a facility in a different region only through a path that goes to a portal of the region. In other words, clients can "leave" a region only through the portals.

Proving the existence of such a structured near-optimal solution relies on the fact that the probability that two very close points end up in different regions of large diameter is very unlikely. Hence the expected *detour* paid by going through a portal of the region is small compared to the original distance between the two points, if the portals are dense enough.

For the sake of argument, we provide a proof sketch of the standard proof of Arora [8]. We will use a refined version of this idea in later sections. The split-tree recursively divides the input metric (X, dist) into parts of smaller and smaller diameter. The root part consists of the entire point set and the parts at level i are of diameter roughly 2^i. The set of portals of a part of level i is an $\varepsilon_0 2^i$-*net* for some ε_0, which is a small set such that every point of the metric is at distance at most $\varepsilon_0 2^i$ to it. Consider two points x, y and let us bound the expected detour incurred by connecting w to y through portals. This detour is determined by a path that starting from x at the lowest level, in each step connects a point w to its closest net point of the part containing w on the next higher level. This is done until the lowest-level part $R_{x,y}$ (i.e., the part of smallest diameter) is reached, which contains both x and y, from where a similar procedure leads from this level through portals of smaller and smaller levels all the way down to y. If the level of $R_{x,y}$ is i then the detour, i.e., the difference between $\mathrm{dist}(x, y)$ and the length of the path connecting x and y through portals, is $O(\varepsilon_0 2^i)$ by the definition of the net. Moreover, the proof shows that the probability that x and y are not in the same part on level i is at most $\mathrm{dist}(x, y)/2^i$. Thus, the expected detour for connecting x to y is $\sum_{\text{level } i} \Pr[R_{x,y} \text{ is at level } i] \cdot O(\varepsilon_0 2^i) = \sum_{\text{level } i} O(\varepsilon_0 \mathrm{dist}(x, y))$. Hence, setting ε_0 to be some ε divided by the number of levels yields that the expected detour is $O(\varepsilon \mathrm{dist}(x, y))$.

Dynamic programming The portals now act as separators between different parts and allows to apply a dynamic programming (DP) approach for solving the problems. The DP consists of a DP-table entry for each part and for each *configuration* of the portals of the part. Here a configuration is a potential way the near-optimal solution interacts with the part. For example, in the case of TSP, a configuration is the information at which portal the near-optimal tour enters and leaves and how it connects the portals on the outside and inside of the part. For the k-Median problem, a configuration stores how many clients outside (respectively inside) the part connect through each portal and are served by a center located inside (respectively outside). Then the dynamic program proceeds in a bottom-up fashion along the split-tree to fill

up the DP table. The running time of the dynamic program depends exponentially on the number of portals.

How many portals? The challenges that need to be overcome when applying this approach, and in particular to clustering problems, are two-fold. First the "standard" use of the split-tree requires $O((\frac{\log n}{\varepsilon})^d)$ portals per part in order to obtain a $(1+\varepsilon)$-approximation, coming from the fact that the number of levels can be assumed to be logarithmic in the number of input points. This often implies quasi-polynomial time approximation schemes since the running time of the dynamic program has exponential dependence on the number of portals. This is indeed the case in the original paper by Talwar [155] and in the first result on clustering in Euclidean space by Arora et al. [9]. However, in some cases, one can lower the number of portals per part needed. In Euclidean space for example, the celebrated "patching lemma" [7] shows that only a constant number (depending on ε) of portals are needed for TSP. Similarly, Kolliopoulos and Rao [112] showed that for k-Median in Euclidean space only a constant number of portal are needed, if one uses a slightly different decomposition of the metric.

Surprisingly, obtaining such a result for doubling metrics is much more challenging. To the best of our knowledge, this work is the first one to reduce the number of portals to a constant.

A second challenge when working with split-tree decompositions and the k-Means problem is that because the cost of assigning a point to a center is the squared distance, the analysis of Arora, Mitchell, and Talwar does not apply. If two points are separated at a high level of the split-tree, then making a detour to the closest portal may incur an expected cost much higher than the cost of the optimal solution.

Our Contributions

Our contribution can be viewed as a "patching lemma" for clustering problems in doubling metrics. Namely, an approach that allows to solve the problems mentioned above: (1) it shows how to reduce the number of portals to a constant, (2) it works for any clustering objective which is defined as the sum of distances to some constant p (with k-Median and k-Means as prominent special cases), and (3) it works not only for Euclidean but also for doubling metrics.

Our starting point is the notion of *badly cut* points of Cohen-Addad [51] for the capacitated version of the above clustering problems. To provide some intuition on the definition, let us focus on k-median and start with the following observation: consider a center c of the optimal solution and a client p assigned to c. If the diameter of the lowest-level part containing both c and p is of order $\mathrm{dist}(p,c)$ (say at most $\mathrm{dist}(p,c)/\varepsilon^2$), then by taking a large enough but constant size net as a set of portals in each part (say an $\varepsilon^3 2^i$-net for a part of level i), the total detour for the two points is at most $O(\varepsilon \mathrm{dist}(p,c))$, which is acceptable.

The problematic scenario is when the lowest-level part containing c and p is of diameter much larger than $\mathrm{dist}(p,c)$. In this case, it is impossible to afford a detour proportional

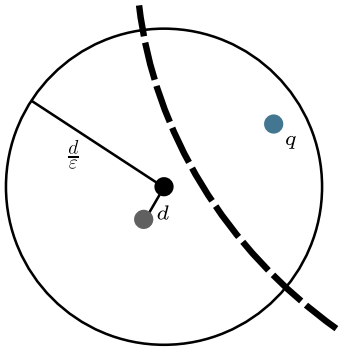


Figure 3.1: Illustration of badly cut for clients. The black point is a client p, the gray one is $\mathcal{L}(p)$, and the point q is arbitrary at distance d/ε from p. The dashed line is the boundary of a part with "large" diameter.

to the diameter of the part in the case of the k-Median and k-Means objectives. To handle this case we first compute a constant approximation $\mathcal{L}$ and use it to guide us towards a $(1+\varepsilon)$-approximation.

Badly cut clients and facilities Consider a client p and the center $\mathcal{L}(p)$ serving p in $\mathcal{L}$ (i.e., $\mathcal{L}(p)$ is closest to p among the centers in $\mathcal{L}$), and call $\mathrm{OPT}(p)$ the facility of an optimum solution OPT that serves p in OPT. Informally, we say that p is *badly cut* if there is a point q in the ball centered at p of radius $\mathrm{dist}(p,\mathcal{L}(p))/\varepsilon$ such that the highest-level part containing p and not q is of diameter much larger than $\mathrm{dist}(p,\mathcal{L}(p))/\varepsilon$ (say greater than $\mathrm{dist}(p,\mathcal{L}(p))/\varepsilon^2$). In other words, there is a point q in this ball such that paying a detour through the portal to connect p to q yields a detour larger than $\varepsilon\mathrm{dist}(p,q)$ (see Figure 3.1).

Similarly, we say that a center c from the solution $\mathcal{L}$ is *badly cut* if there is a point q in the ball centered at c of radius $\mathrm{dist}(c,\mathrm{OPT}(c))/\varepsilon$ (where $\mathrm{OPT}(c)$ is the facility of OPT that is the closest to c) such that the highest-level part containing c and not q is of diameter $\mathrm{dist}(c,\mathrm{OPT}(c))/\varepsilon^2$. The crucial property here is that any client p or any facility c is badly cut with probability $O(\varepsilon^3)$, as we will show.

Using the notion of badly cut We now illustrate how this notion can help us. Assume for simplicity that $\mathrm{OPT}(p)$ is in the ball centered at a client p of radius $\mathrm{dist}(p,\mathcal{L}(p))/\varepsilon$ (if this is not the case then $\mathrm{dist}(p,\mathrm{OPT}(p))$ is much larger than $\mathrm{dist}(p,\mathcal{L}(p))$, so this is a less problematic scenario and a simple idea can handle it). If p is not badly cut, then the lowest-level part containing both p and $\mathrm{OPT}(p)$ is of diameter not much larger than $\mathrm{dist}(p,\mathcal{L}(p))/\varepsilon$. Taking a sufficiently fine net for each part (independent of the number of levels) allows to bound the detour through the portals to reach $\mathrm{OPT}(p)$ from p by at most $\varepsilon\mathrm{dist}(p,\mathcal{L}(p))$. Since $\mathcal{L}$ is an $O(1)$-approximation, this is fine.

If p is badly cut, then we modify the instance by relocating p to $\mathcal{L}(p)$. That is, we will work with the instance where there is no more client at p and there is an additional client at $\mathcal{L}(p)$. We claim that any solution in the modified instance can be lifted to the

original instance at an expected additional cost of $O(\varepsilon^3 \mathrm{OPT})$. This comes from the fact that the cost increase for a solution is, by the triangle inequality, at most the sum of distances of the badly cut clients to their closest facility in the local solution. This is at most $O(\varepsilon^3 \mathrm{OPT})$ in expectation since each client is badly cut with probability at most $O(\varepsilon^3)$ and $\mathcal{L}$ is an $O(1)$-approximation.

Here we should ask, what did we achieve by moving p to $\mathcal{L}(p)$? Note that p should now be assigned to facility f of OPT that is the closest to $\mathcal{L}(p)$. So we can make the following observation: If $\mathcal{L}(p)$ is not badly cut, then the detour through the portals when assigning p to f is fine (namely at most ε times the distance from $\mathcal{L}(p)$ to its closest facility in OPT). Otherwise, if $\mathcal{L}(p)$ is also badly cut, then we simply argue that there exists a near-optimal solution which contains $\mathcal{L}(p)$, in which case p is now served optimally at a cost of 0 (in the new instance).

From bicriteria to opening exactly k centers Since $\mathcal{L}(p)$ is badly cut with probability $O(\varepsilon^3)$, this leads to a solution opening $(1 + O(\varepsilon^3))k$ centers. At first, it looks difficult to then reduce the number of centers to k without increasing the cost of the solution by a factor larger than $(1 + \varepsilon)$. However, and perhaps surprisingly, we show in Lemma 3.24 that this can be avoided: we show that there exists a near-optimal solution that contains the badly cut centers of $\mathcal{L}(p)$.

We can then conclude that a near-optimal solution can be computed by a simple dynamic-programming procedure on the split-tree decomposition to identify the best solution in the modified instance.

Our result on Facility Location in Section 3.3 provides a simple illustration of these ideas — avoiding the bicriteria issue due to the hard bound on the number of opened facilities for the k-Median and k-Means problems. Our main result on k-Median and k-Means is described in Section 3.4. We discuss some extensions of the framework in Section 3.5.

3.1.1 Related work

On clustering problems The clustering problems considered in this chapter are known to be NP-hard, even restricted to inputs lying in the Euclidean plane (see Mahajan et al. [128] or Dasgupta and Freund [61] for k-Means, Megiddo and Supowit [135] for the problems with outliers, and Masuyama et al. [133] for k-Center). The problems of Facility Location and k-Median have been studied since a long time in graphs, see e.g. [109]. The current best approximation ratio for metric Facility Location is 1.488, due to Li [119], whereas it is 2.67 for k-Median, due to Byrka et al. [37].

The problem of k-Means in general graphs also received a lot of attention (see e.g., Kanungo et al. [109]) and the best approximation ratio is 6.357, due to Ahmadian et al. [5].

Clustering problems with outliers where first studied by Charikar et al. [41], who devised an $(O(1), (1+O(\varepsilon))$-approximation for k-Median with outliers and a constant

factor approximation for prize-collecting k-Median. More recently, Friggstad et al. [85] showed that local search provides a bicriteria approximation, where the number of centers is approximate instead of the number of outliers. However, the runtime is $n^{f(\varepsilon,d)}$, and thus we provide a much faster algorithm. To the best of our knowledge, we present the first approximation scheme that preserves the number of centers.

The k-Center problem is known to be NP-hard to approximate within any factor better than 2, a bound that can be achieved by a greedy algorithm [74]. This is related to the problem of covering points with a minimum number of disks (see e.g. [122, 132]). Marx and Pilipczuk [132] proposed an exact algorithm running in time $n^{\sqrt{k}+O(1)}$ to find the maximum number of points covered by k disks and showed a matching lower bound, whereas Liao et al. [122] presented an algorithm running in time $O(mn^{O(1/\varepsilon^2 \log^2 1/\varepsilon)})$ to find a $(1+\varepsilon)$-approximation to the minimal number of disks necessary to cover all the points (where m is the total number of disks and n the number of points). This problem is closely related to k-Center: the optimal value of k-Center on a set P is the minimal number L such that there exist k disks of radius L centered on points of P covering all points of P. Hence, the algorithm from [122] can be directly extended to find a solution to k-Center with $(1+\varepsilon)k$ centers and optimal cost. The local search algorithm of Cohen-Addad et al. [53] can be adapted to k-center and generalizes the last result to any dimension d: in $\mathbb{R}^d$, one can find a solution with optimal cost and $(1+\varepsilon)k$ centers in time $n^{1/\epsilon^{O(d)}}$. Losing on the approximation allows us to present a much faster algorithm.

On doubling dimension Despite their hardness in general metrics, these problems admit a PTAS when the input is restricted to a low dimensional metric space: Friggstad et al. [86] showed that local search gives a $(1+\varepsilon)$-approximation. However, the running time of their algorithm is $n^{(d/\varepsilon)^{O(d)}}$ in metrics with doubling dimension d.

A long line of research exists on filling the gap between results for Euclidean spaces and metrics with bounded doubling dimension. This started with the work of Talwar [155], who gave QPTASs for a long list of problems. The complexity for some of these problems was improved later on: for the Traveling Salesperson problem, Gottlieb [89] gave a near-linear time approximation scheme, Chan et al. [39] gave a PTAS for Steiner Forest, and Gottlieb [89] described an efficient spanner construction. Very recently, Bartal and Gottlieb gave nearly-linear time algorithms for Steiner Tree and Forest [21].

3.2 Preliminaries

3.2.1 Definitions

Since we will transform an input instance of (k, z)-clustering into a tree-like structure, we will need a more formal definition of an instance. An instance to the problem is a 4-tuple (P, F, dist, k), where $(P \cup F, \text{dist})$ is a metric space and k is a positive integer. The goal is to find a set $\mathcal{S} \subseteq F$ such that $|\mathcal{S}| \leq k$ and $\sum_{c \in P} \min_{f \in \mathcal{S}}(\text{dist}(c, f)^z)$ is minimized. We let $n = |P \cup F|$ be the total size of the metric space.

The Facility Location problem is a relaxation of the k-median problem: instead of having a strict limitation on the number of facilities, one can pay for each opened facilities. Formally:

▶ **Definition 3.7 (Facility-location problem).** Given a metric space (X, dist), an instance for the facility location problem is set of clients $P \subset X$, a set of facilities $F \subset X$ and a cost $w_f \geq 0$ for all $f \in F$. The goal is to output a subset of set facilities $\mathcal{S} \subseteq F$ that minimizes

$$\sum_{f \in \mathcal{S}} w_f + \sum_{p \in P} \text{dist}(p, \mathcal{S}).$$ ◀

For those clustering problems, it will also be convenient to name the center serving a client. For a client p and a solution $\mathcal{S}$, we denote $\mathcal{S}(p)$ the center closest to p, and $\mathcal{S}_p := \text{dist}(p, \mathcal{S}(p))$ the distance to it.

Consider a metric space (X, dist). For a point $p \in X$ and an integer $r \geq 0$, we let $\beta(p, r) = \{x \in X \mid \text{dist}(p, x) \leq r\}$ be the *ball* around p with radius r.

▶ **Definition 3.8.** The *doubling dimension* of a metric is the smallest integer d such that any ball of radius $2r$ can be covered by 2^d balls of radius r. ◀

We call Δ the aspect-ratio (sometimes referred to as *spread* in the literature) of the metric, i.e., the ratio between the largest and the smallest non-zero distance.

A crucial property of doubling metrics is the existence of small nets. A δ-*net* of P is a set of points $X \subseteq P$ such that for all $p \in P$ there is an $x \in X$ such that $\text{dist}(p, x) \leq \delta$, and for all $x, y \in X$ we have $\text{dist}(x, y) > \delta$. A net is therefore a set of points not too close to each other, such that every point of the metric is close to a net point. The following lemma bounds the cardinality of a net in doubling metrics.

▶ **Lemma 3.9 (from Gupta et. al [91]).** Let (P, dist) by a metric space with doubling dimension d and diameter D, and let X be a δ-net of V. Then $|X| \leq 2^{d \cdot \lceil \log_2(D/\delta) \rceil}$. ◀

Another property of doubling metrics that will be useful for our purpose is the existence of low-stretch spanners with a linear number of edges. More precisely, Har-Peled and Mendel [97] showed that one can find a graph (called a *spanner*) in the input metric that has $O(n)$ edges such that distances in the graph approximate the original distances up to a constant factor. This construction takes time $2^{O(d)}n$. We will make use of these spanners only for computing constant-factor approximations of our problems: for this purpose, we will therefore assume that the number of edges is $m = 2^{O(d)}n$.

Lastly, the doubling metrics we consider in this chapter are discrete. Hence, this does not strictly speaking generalize the Euclidean version of (k, z)-clustering, where the centers can be placed everywhere. However, up to losing a polylogarithmic factor in the running time, it is possible to reduce the continuous Euclidean problem to this discrete setting by computing a set of candidate centers that approximate the best set of centers in $\mathbb{R}^d$ [134]. Less formally, the only part where discreteness is important is in the dynamic program: it is actually possible to deal directly with infinite Euclidean Space without losing anything in the complexity, by slightly adapting the dynamic program. We explain how to do that in the relevant section, and will consider for simplicity that the input is discrete.

3.2.2 Decomposition of Metric Spaces

As pointed out in our techniques section, we will make use of hierarchical decompositions of the input metric. We define a *hierarchical decomposition* (sometimes simply a decomposition) of a metric (X, dist) as a collection of partitions $\mathcal{D} = \{\mathcal{B}_0, \ldots, \mathcal{B}_{|\mathcal{D}|}\}$ that satisfies the following:

- each $\mathcal{B}_i$ is a partition of X,
- $\mathcal{B}_i$ is a refinement of $\mathcal{B}_{i+1}$, namely for each part $B \in \mathcal{B}_i$ there exists a part $B' \in \mathcal{B}_{i+1}$ that contains B,
- $\mathcal{B}_0$ contains a singleton set for each $x \in X$, while $\mathcal{B}_{|\mathcal{D}|}$ is a trivial partition that contains only one set, namely X.

We define the *ith level* of the decomposition to be the partition $\mathcal{B}_i$, and call $B \in \mathcal{B}_i$ a level-i part. If $B' \in \mathcal{B}_{i-1}$ is such that $B' \subset B$, we say that B' is a *subpart* of B.

For a given decomposition $\mathcal{D} = \{\mathcal{B}_0, \ldots, \mathcal{B}_{|\mathcal{D}|}\}$, we say that a point x is *cut from y at level j* if j is the maximum integer such that x is in some $B \in \mathcal{B}_j$ and y is in some $B' \in \mathcal{B}_j$ with $B \neq B'$. For a point x and radius r we say that the ball $\beta(x, r)$ is *cut* by $\mathcal{D}$ at level j if j is the maximum level for which some point of the ball is cut from x at level j.

A key ingredient for our result is the following lemma, that introduces some properties of the hierarchical decomposition (sometimes referred to as *split-tree*) proposed by Talwar [155] for low-doubling metrics. As mentioned in the introduction, this decomposition generalizes the *quad-tree decomposition* to doubling metrics.

▶ **Lemma 3.10 (Reformulation of [155, 20]).** For any metric (X, dist) of doubling dimension d and any $\rho > 0$, there is a randomized hierarchical decomposition $\mathcal{D}$ such that the diameter of a part $B \in \mathcal{B}_i$ is at most 2^{i+1}, $|\mathcal{D}| \leq \lceil \log_2(\text{diam}(X)) \rceil$, each part $B \in \mathcal{B}_i$ is refined in at most $2^{O(d)}$ parts at level $i-1$, and:

1. **Scaling probability**: for any $x \in X$, radius r, and level i, we have
$$\Pr[\mathcal{D} \text{ cuts } \beta(x,r) \text{ at a level } i] \leq 2^{2d+2} r/2^i.$$
2. **Portal set:** every set $B \in \mathcal{B}_i$ where $\mathcal{B}_i \in \mathcal{D}$ comes with a set of *portals* $\mathcal{P}_B \subseteq B$ that is
   (a) **concise:** the size of the portal set is bounded by $|\mathcal{P}_B| \leq 1/\rho^d$; and
   (b) **precise:** for every point $x \in B$ there is a portal $p \in \mathcal{P}_B$ close-by, i.e., $\text{dist}(x,p) \leq \rho 2^{i+1}$; and
   (c) **nested:** any portal of level $i+1$ that lies in B is also a portal of B, i.e., for every $p \in \mathcal{P}_{B'} \cap B$ where $B' \in \mathcal{B}_{i+1}$ we have $p \in \mathcal{P}_B$.

Moreover, this decomposition can be found in time $(1/\rho)^{O(d)} n \log \Delta$. ◀

3.2.3 Formal Definition of Badly Cut Points

As sketched in the introduction, the notion of badly cut lies at the heart of our analysis. We define it formally here. We denote $\kappa(\varepsilon, z) = \varepsilon^2 \frac{z}{(z+\varepsilon)^z}$ and $\tau(\varepsilon, d, z) = 2d + 2 + \log(1/\kappa(\varepsilon, z))$, two parameters that are used throughout this chapter.

▶ **Definition 3.11.** Let $(P \cup F, \text{dist})$ be a metric with doubling dimension d, let $\mathcal{D}$ be a hierarchical decomposition of the metric (for any $\rho > 0$), and $\varepsilon > 0$. Let also $\mathcal{L}$ be a solution to the instance for any of the problems Facility Location, k-Median, or k-Means.
A client $p \in P$ is *badly cut w.r.t.* $\mathcal{D}$ if the ball $\beta(p, 3\text{dist}(p,\mathcal{L})/\varepsilon)$ is cut as some level j greater than $\log(3\text{dist}(p,\mathcal{L})/\varepsilon) + \tau(\varepsilon, d, z)$, where $\text{dist}(p,\mathcal{L})$ is the distance from p to the closest facility of $\mathcal{L}$.
Similarly, a center $f \in \mathcal{L}$ is *badly cut w.r.t.* $\mathcal{D}$ if $\beta(f, 3\text{dist}(f, \text{OPT}))$ is cut at some level j greater than $\log(3\text{dist}(f, \text{OPT})) + \tau(\varepsilon, d, z)$, where $\text{dist}(f, \text{OPT})$ is the distance from f to the closest facility of OPT. ◀

In the following, when $\mathcal{D}$ is clear from the context we simply say badly cut. The following lemma bounds the probability of being badly cut.

▶ **Lemma 3.12.** Let $(P \cup F, \text{dist})$ be a metric, and $\mathcal{D}$ a random hierarchical decomposition given by Lemma 3.10. Let p be any point in $P \cup F$. The probability that p is badly cut is at most $\kappa(\varepsilon, z)$. ◀

Proof. Consider first a point $p \in P$. By Property 1, the probability that a ball $\beta(p, r)$ is cut at level at least j is at most $2^{2d+2}r/2^j$. Hence the probability that a ball $\beta(p, 3\text{dist}(p, \mathcal{L})/\varepsilon)$ is cut at a level j greater than $\log(3\text{dist}(p, \mathcal{L})/\varepsilon) + 2 + 2d + \log(1/\kappa(\varepsilon, z))$ is at most $\kappa(\varepsilon, z)$.

The proof for $p \in F$ is identical. □

3.2.4 Preprocessing

In the following, we will work with the slightly more general version of the clustering problems where there is some *demand* on each input point: there is a function $\chi : P \mapsto \{0, \ldots, n\}$ and the goal is to minimize $\sum_{p \in P} \chi(p) \cdot \min_{f \in \mathcal{S}} \text{dist}(p, f) + \sum_{f \in \mathcal{S}} w_f$ for the Facility Location problem, or $\sum_{p \in P} \chi(p) \cdot \min_{s \in \mathcal{S}} \text{dist}(p, s)^z$ for the (k, z)-clustering problem. For simplicity, we will consider in the proof that the client set is actually a multiset, where a client p appears $\chi(p)$ times.

We will preprocess the input instance to transform it into several instances of the more general clustering problem, ensuring that the aspect-ratio Δ of each instance is polynomial. We defer this construction to Section 3.7.1.

3.3 A Near-Linear Time Approximation Scheme for Non-Uniform Facility Location

To demonstrate the utility of the notion of badly cut, we show how to use it to get a near-linear time approximation scheme for Facility Location in metrics of bounded doubling dimension. In this context we refer to centers in the set F of the input as facilities.

We first show a structural lemma that allows to focus on instances that do not contain any badly cut client. For this, we essentially rely on Property 1 of Lemma 3.10, regardless of the value of ρ chosen. Then, we use the second property of decomposition to show that, for a particular choice of ρ, these instances have *portal-respecting* solutions that are nearly optimal, and that can be computed with a dynamic program. We conclude by providing a fast dynamic program, that takes advantage of all the structure exhibited before.

3.3.1 An instance with small distortion

Consider a metric space (P, dist) and an instance $\mathcal{I}$ of the Facility Location problem on (P, dist). Here we generalize slightly, as explained in Section 3.2.4, and restrict the set of candidate center to a subset of P. Our first step is to show that, given $\mathcal{I}$, a randomized decomposition $\mathcal{D}$ of (P, dist) (for any value of $\rho > 0$) and any solution $\mathcal{L}$ for $\mathcal{I}$ on (P, dist), we can build an instance $\mathcal{I}_{\mathcal{D}}$ on the same metric (but different client set) such that any solution S has a similar cost in $\mathcal{I}$ and in $\mathcal{I}_{\mathcal{D}}$, and more importantly $\mathcal{I}_{\mathcal{D}}$ does not contain any badly cut client with respect to $\mathcal{D}$. Note that to ensure that property, $\mathcal{I}_{\mathcal{D}}$ must depend on the randomness of $\mathcal{D}$.

Let $\mathrm{cost}_{\mathcal{I}_0}(\mathcal{S}) = \sum_{p \in P} \min_{f \in \mathcal{S}}(\mathrm{dist}(p, f))^z$ be the cost incurred by only the distances to the facilities in a solution $\mathcal{S}$ to an instance $\mathcal{I}_0$, and fix $\varepsilon > 0$ and a solution $\mathcal{L}$. For any instance $\mathcal{I}_{\mathcal{D}}$ on (P, dist), we let

$$\nu_{\mathcal{I}_{\mathcal{D}}} = \max_{\text{solution } \mathcal{S}} \left\{ \mathrm{cost}_{\mathcal{I}}(\mathcal{S}) - (1+2\varepsilon)\mathrm{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}), (1-2\varepsilon)\mathrm{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}) - \mathrm{cost}_{\mathcal{I}}(\mathcal{S}) \right\}.$$

In the particular case of Facility Location, $z = 1$, but we allow the cost to be more general in order to make the proof adjustable to k-Means. Let $B_{\mathcal{D}}$ denotes the set of badly cut facilities (w.r.t $\mathcal{D}$) of the solution $\mathcal{L}$. As explained above, our goal is to construct an instance with the following *small distortion* property. We say that $\mathcal{I}_{\mathcal{D}}$ has *small distortion w.r.t.* $\mathcal{I}$ if: $\sum_{f \in B_{\mathcal{D}}} w_f \leq \varepsilon \cdot \sum_{f \in \mathcal{L}} w_f$, $\nu_{\mathcal{I}_{\mathcal{D}}} \leq \varepsilon \mathrm{cost}_{\mathcal{I}}(\mathcal{L})$, and there exists a solution $\mathcal{S}$ that contains $B_{\mathcal{D}}$ with

$$\mathrm{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}) \leq (1 + O(\varepsilon))\mathrm{cost}_{\mathcal{I}}(\mathrm{OPT}) + O(\varepsilon)\mathrm{cost}_{\mathcal{I}}(\mathcal{L}). \tag{3.1}$$

When $\mathcal{I}$ is clear from the context we simply say that $\mathcal{I}_{\mathcal{D}}$ has small distortion. In words, the first condition asks that only a tiny mass of the facilities of $\mathcal{L}$ is badly-cut, the second one that the cost of any solution $\mathcal{S}$ is nearly the same in the instance $\mathcal{I}_{\mathcal{D}}$ as in $\mathcal{I}$ and the third that there exists a solution that contains all badly cut clients and that have, in $\mathcal{I}_{\mathcal{D}}$, a small cost.

In particular, we will show that $\mathrm{OPT}' = \mathrm{OPT} \cup B_{\mathcal{D}}$ (where OPT is the optimal solution for the instance $\mathcal{I}$) fulfills the condition of (3.1).

Construction of instance $\mathcal{I}_{\mathcal{D}}$. In the following, we will always work with a particular $\mathcal{I}_{\mathcal{D}}$ constructed from $\mathcal{I}$ and $\mathcal{L}$ as follows: $\mathcal{I}$ is transformed such that every badly cut client p is moved to $\mathcal{L}(p)$, namely, $\chi(\mathcal{L}(p))$ is increased by $\chi(p)$ after which we set $\chi(p) = 0$. Recall however that we treat the client set as a multiset, so that $\mathrm{cost}_{\mathcal{I}_0}(\mathcal{S})$ counts the distance from p to the closest facility $\chi(p)$ times.

What we would like to prove is that the optimal solution in $\mathcal{I}$ can be transformed to a solution in $\mathcal{I}_{\mathcal{D}}$ with a small additional cost, and vice versa. The intuition behind this is the following: a client of the solution L is badly cut with probability $\kappa(\varepsilon, z)$ (from Lemma 3.12), hence every client contributes with $\kappa(\varepsilon, z)\mathrm{dist}(p, \mathcal{L})$ to transform any solution S for the instance $\mathcal{I}$ to a solution for the instance $\mathcal{I}_{\mathcal{D}}$, and vice versa.

However, we will need to convert a particular solution in $\mathcal{I}_{\mathcal{D}}$ (think of it as $\mathrm{OPT}_{\mathcal{I}_{\mathcal{D}}}$) to a solution in $\mathcal{I}$: this particular solution depends in the randomness of $\mathcal{D}$, and this short

argument does not apply because of dependency issues. It is nevertheless possible to prove that $\mathcal{I}_\mathcal{D}$ has a small distortion, as done in the following lemma.

▶ **Lemma 3.13.** Given an instance $\mathcal{I}$ of Facility Location, a randomized decomposition $\mathcal{D}$, an ε such that $0 < \varepsilon < 1/4$ and a solution $\mathcal{L}$, let $\mathcal{I}_\mathcal{D}$ be the instance obtained from $\mathcal{I}$ by moving every badly cut client p to $\mathcal{L}(p)$ (as described above). The probability that $\mathcal{I}_\mathcal{D}$ has small distortion is at least $1 - \varepsilon$, where the solution fulfilling (3.1) is $\mathrm{OPT}' = \mathrm{OPT} \cup B_\mathcal{D}$. ◀

Proof. To show the lemma, we will show that $\mathbb{E}\left[\sum_{f\in B_\mathcal{D}} w_f\right] \leq \varepsilon^2 \sum_{f\in L} w_f/2$ and $\mathbb{E}\left[\nu_{\mathcal{I}_\mathcal{D}}\right] \leq \varepsilon^2 \mathrm{cost}(\mathcal{L})/2$. Then, Markov's inequality and a union bound over the probabilities of failure yield that with probability $1-\varepsilon$, $\sum_{f\in B_\mathcal{D}} w_f \leq \varepsilon \cdot \sum_{f\in\mathcal{L}} w_f$ and $\nu_{\mathcal{I}_\mathcal{D}} \leq \varepsilon \mathrm{cost}_\mathcal{I}(\mathcal{L})$. Since $\mathrm{OPT} \subseteq \mathrm{OPT}'$ and $(1-2\varepsilon)\mathrm{cost}_{\mathcal{I}_\mathcal{D}}(\mathrm{OPT}') - \mathrm{cost}_\mathcal{I}(\mathrm{OPT}') \leq \nu_{\mathcal{I}_\mathcal{D}} \leq \varepsilon \mathrm{cost}_\mathcal{I}(\mathcal{L})$, the cost incurred by connecting clients to facilities in OPT' is

$$\begin{aligned}
\mathrm{cost}_{\mathcal{I}_\mathcal{D}}(\mathrm{OPT}') &\leq (1 + \frac{2\varepsilon}{1-2\varepsilon})(\mathrm{cost}_\mathcal{I}(\mathrm{OPT}') + \varepsilon \mathrm{cost}_\mathcal{I}(\mathcal{L})) && (\text{as } \nu_{\mathcal{I}_\mathcal{D}} \leq \varepsilon \mathrm{cost}_\mathcal{I}(\mathcal{L})) \\
&< (1+4\varepsilon)(\mathrm{cost}_\mathcal{I}(\mathrm{OPT}') + \varepsilon \mathrm{cost}_\mathcal{I}(\mathcal{L})) && (\text{as } \varepsilon < 1/4) \\
&\leq (1+4\varepsilon)(\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + \varepsilon \mathrm{cost}_\mathcal{I}(\mathcal{L})) && (\text{as } \mathrm{OPT} \subseteq \mathrm{OPT}'),
\end{aligned}$$

which shows that $\mathcal{I}_\mathcal{D}$ has small distortion.

Note that $\mathbb{E}\left[\sum_{f\in B_\mathcal{D}} w_f\right] = \sum_{f\in\mathcal{L}} \Pr[f \text{ badly cut}] \cdot w_f \leq \varepsilon^2 \sum_{f\in\mathcal{L}} w_f/2$ is immediate from Lemma 3.12. It remains to show that $\mathbb{E}\left[\nu_{\mathcal{I}_\mathcal{D}}\right] \leq \varepsilon^2 \mathrm{cost}(\mathcal{L})/2$. For the sake of lightening equations, we will denote by $\sum_{\text{bcc. } p}$ the sum over all badly cut clients p.

By definition, we have that for any solution $\mathcal{S}$,

$$\begin{aligned}
\mathrm{cost}(\mathcal{S}) - \mathrm{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}) &\leq \sum_{\text{bcc. } p} \mathrm{dist}(p,\mathcal{S})^z - \mathrm{dist}(\mathcal{S},\mathcal{L}(p))^z \\
&\leq \sum_{\text{bcc. } p} 2\varepsilon \cdot \mathrm{dist}(\mathcal{S},\mathcal{L}(p))^z + (1+z/\varepsilon)^z \mathrm{dist}(p,\mathcal{L}(p))^z
\end{aligned}$$

using the generalized triangle inequality from Lemma 1.2.

Subtracting $\sum_{\text{bcc. } p} 2\varepsilon \cdot \mathrm{dist}(\mathcal{S},\mathcal{L}(p))^z \leq \sum_{p\in P} 2\varepsilon \cdot \mathrm{dist}(\mathcal{S},\mathcal{L}(p))^z$ from the right and left side, respectively, yields

$$\mathrm{cost}(\mathcal{S}) - (1+2\varepsilon)\mathrm{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}) \leq \sum_{\text{bcc. } c} (1+z/\varepsilon)^z \mathrm{dist}(p,\mathcal{L}(p))^z$$

Similarly, we have that

$$\begin{aligned}
\mathrm{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}) - \mathrm{cost}(\mathcal{S}) &\leq \sum_{\text{bcc. } p} \mathrm{dist}(\mathcal{S},\mathcal{L}(p))^z - \mathrm{dist}(p,\mathcal{S})^z \\
&\leq \sum_{\text{bcc. } p} 2\varepsilon \cdot \mathrm{dist}(p,\mathcal{S})^z + (1+z/\varepsilon)^z \mathrm{dist}(p,\mathcal{L}(p))^z
\end{aligned}$$

and we conclude

$$(1-2\varepsilon)\text{cost}_{\mathcal{I}_\mathcal{D}}(S) - \text{cost}(S) \leq \sum_{\text{bcc. } p} (1+z/\varepsilon)^z \text{dist}(p,\mathcal{L}(p))^z$$

Therefore,

$$\nu_{\mathcal{I}_\mathcal{D}} \leq \sum_{\text{bcc. } c} (1+z/\varepsilon)^z \text{dist}(p,\mathcal{L}(p))^z.$$

Taking expectation (over the random choice of $\mathcal{D}$), we get:

$$E[\nu_{\mathcal{I}_\mathcal{D}}] \leq \sum_{\text{client } p} Pr[p \text{ badly cut}] \cdot (1+z/\varepsilon)^z \text{dist}(p,\mathcal{L}(p))^z.$$

Applying Lemma 3.12 and using $\kappa(\varepsilon,z) = \varepsilon^2(\frac{z}{z+\varepsilon})^z$, we conclude $E[\nu_{\mathcal{I}_\mathcal{D}}] \leq \varepsilon^2 \cdot \text{cost}(\mathcal{L})$. The lemma follows for a sufficiently small ε. □

3.3.2 Portal Respecting Solution

In the following, we fix an instance $\mathcal{I}$, a decomposition $\mathcal{D}$, and a solution $\mathcal{L}$. By Lemma 3.13, $\mathcal{I}_\mathcal{D}$ has small distortion with probability at least $1-\varepsilon$ and so we condition on this event from now on.

We explore the structure that this conditioning can give to solutions. We will show that in the solution $\text{OPT}' = \text{OPT} \cup B_\mathcal{D}$ with small cost, each client p is cut from its serving facility f at a level at most $\log(3\text{dist}(p,\mathcal{L})/\varepsilon + 4\text{dist}(p,\text{OPT}))) + \tau(\varepsilon,d,z)$. This will allow us to consider *portal-respecting* solution, where every client to facility path goes in and out parts of the decomposition only at designated portals. Indeed, the detour incurred by using a portal-respecting path instead of a direct connection depends on the level where its extremities are cut, as proven in Lemma 3.35. Hence, ensuring that this level stays small implies that the detour made is small (in our case, $O(\varepsilon(\text{dist}(p,\mathcal{L}) + \text{dist}(p,\text{OPT})))$. Such a solution can be computed by a dynamic program that we will present afterwards.

Recall that $\text{dist}(p,\mathcal{L})$ and $\text{dist}(p,\text{OPT})$ are the distances from the *original* position of p to $\mathcal{L}$ and OPT, although p may have been moved to $\mathcal{L}(p)$, and $B_\mathcal{D}$ is the set of badly cut facilities of $\mathcal{L}$ w.r.t $\mathcal{D}$.

▶ **Lemma 3.14.** Let $\mathcal{I}$ be an instance of Facility Location with a randomized decomposition $\mathcal{D}$, $\varepsilon < 1/4$ and $\mathcal{L}$ be a solution for $\mathcal{I}$, such that $\mathcal{I}_\mathcal{D}$ has small distortion. Let $\text{OPT}' = \text{OPT} \cup B_\mathcal{D}$, and for any client p in $\mathcal{I}_\mathcal{D}$, let $\text{OPT}'(p)$ be the closest facility to p in OPT′. Then p and $\text{OPT}'(p)$ are cut in $\mathcal{D}$ at level at most $\log(3\text{dist}(p,\mathcal{L})/\varepsilon + 4\text{dist}(p,\text{OPT})) + \tau(\varepsilon,d,z)$. ◀

Proof. Let p be a client. To find the level at which p and $\text{OPT}'(p)$ are separated, we distinguish between two cases: either p in $\mathcal{I}$ is badly cut w.r.t. $\mathcal{D}$, or not.

If p is badly cut, then it is now located at $\mathcal{L}(p)$ in the instance $\mathcal{I}_\mathcal{D}$. In that case, either:

1. $\mathcal{L}(p)$ is also badly cut, and therefore $\mathcal{L}(p) \in B_{\mathcal{D}} \subseteq \mathrm{OPT}'$ and so $\mathrm{OPT}'(p) = \mathcal{L}(p)$. Since p and $\mathcal{L}(p)$ are collocated, it follows that p and $\mathrm{OPT}'(p)$ are never cut.

2. $\mathcal{L}(p)$ is not badly cut: Definition 3.11 implies that $\mathcal{L}(p)$ and $\mathrm{OPT}(\mathcal{L}(p))$ are cut at a level at most $\log(3\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT})) + \tau(\varepsilon, d, z)$. By triangle inequality, $\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT}) \leq \mathrm{dist}(p, \mathcal{L}) + \mathrm{dist}(p, \mathrm{OPT})$, and thus p (located at $\mathcal{L}(p)$ in $\mathcal{I}_{\mathcal{D}}$) and $\mathrm{OPT}'(p)$ are also cut at level at most $\log\big(3\mathrm{dist}(p, \mathcal{L}) + 3\mathrm{dist}(p, \mathrm{OPT})\big) + \tau(\varepsilon, d, z)$.

We now turn to the case where p is not badly cut. In this case p is not moved to $\mathcal{L}(p)$ and the ball $\beta(c, 3\mathrm{dist}(p, \mathcal{L})/\varepsilon)$ is cut at level at most $\log(3\mathrm{dist}(p, \mathcal{L})/\varepsilon) + \tau(\varepsilon, d, z)$. We make a case distinction according to $\mathrm{dist}(p, \mathrm{OPT})$ and $\mathrm{dist}(p, \mathcal{L})$.

1. If $\mathrm{dist}(p, \mathcal{L}) \leq \varepsilon\mathrm{dist}(p, \mathrm{OPT})$, then we have the following. If $\mathcal{L}(p)$ is badly cut, $\mathcal{L}(p)$ is open in OPT' and therefore $\mathrm{OPT}'_c = \mathrm{dist}(p, \mathcal{L})$. Moreover, since p is not badly cut the ball $\beta(p, \mathrm{dist}(p, \mathcal{L}))$ is cut at level at most $\log(3\mathrm{dist}(p, \mathcal{L})/\varepsilon) + \tau(\varepsilon, d, z)$. Therefore p and $\mathrm{OPT}'(p)$ are cut at level at most $\log(3\mathrm{dist}(p, \mathcal{L})/\varepsilon) + \tau(\varepsilon, d, z)$.

   Now consider the case where $\mathcal{L}(p)$ is not badly cut. Both p and $\mathrm{OPT}'(p)$ lie in the ball centered at $\mathcal{L}(p)$ and of diameter $2\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT})$: indeed, we use $\mathrm{dist}(p, \mathcal{L}) \leq \varepsilon\mathrm{dist}(p, \mathrm{OPT})$ to derive

$$\begin{aligned}\mathrm{dist}(p, \mathcal{L}(p)) &\leq \varepsilon\mathrm{dist}(p, \mathrm{OPT}(p)) \leq \varepsilon\mathrm{dist}(p, \mathrm{OPT}(\mathcal{L}(p)))\\ &\leq \varepsilon\mathrm{dist}(p, \mathcal{L}(p)) + \varepsilon\mathrm{dist}\big(\mathcal{L}(p), \mathrm{OPT}(\mathcal{L}(p))\big),\end{aligned}$$

   and therefore $\mathrm{dist}(p, \mathcal{L}(p)) \leq \frac{\varepsilon}{1-\varepsilon}\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT}) \leq 2\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT})$, since $\varepsilon \leq 1/4$. On the other hand,

$$\begin{aligned}\mathrm{dist}(\mathrm{OPT}'(p), \mathcal{L}(p)) &\leq \mathrm{dist}(p, \mathrm{OPT}'(p)) + \mathrm{dist}(p, \mathcal{L}(p))\\ &\leq \mathrm{dist}(p, \mathrm{OPT}(p)) + \mathrm{dist}(p, \mathcal{L}(p))\\ &\leq \mathrm{dist}(p, \mathrm{OPT}(\mathcal{L}(p))) + \mathrm{dist}(p, \mathcal{L}(p))\\ &\leq 2\mathrm{dist}(p, \mathcal{L}(p)) + \mathrm{dist}(\mathcal{L}(p), \mathrm{OPT}(\mathcal{L}(p)))\\ &\leq \left(1 + \frac{2\varepsilon}{1-\varepsilon}\right)\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT}),\end{aligned}$$

   which is smaller than $2\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT})$ for any $\varepsilon \leq 1/4$. Hence we have $p, \mathrm{OPT}'(p) \in \beta(\mathcal{L}(p), 2\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT}))$.

   By definition of badly cut, p and $\mathrm{OPT}'(p)$ are therefore cut at level at most $\log(3\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT})) + \tau(\varepsilon, d, z)$. Since $\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT}) \leq \mathrm{dist}(\mathcal{L}(p), \mathrm{OPT}(p)) \leq \mathrm{dist}(\mathcal{L}(p), p) + \mathrm{dist}(p, \mathrm{OPT}(p)) \leq (1+\varepsilon)\mathrm{dist}(p, \mathrm{OPT})$ as $\mathrm{dist}(p, \mathcal{L}) \leq \varepsilon\mathrm{dist}(p, \mathrm{OPT})$, we have that $\log(3\mathrm{dist}(\mathcal{L}(p), \mathrm{OPT})) \leq \log(4\mathrm{dist}(p, \mathrm{OPT}))$. Hence p and $\mathrm{OPT}'(p)$ are cut at level at most $\log(4\mathrm{dist}(p, \mathrm{OPT})) + \tau(\varepsilon, d, z)$.

2. If $\mathrm{dist}(p, \mathrm{OPT}) \leq \mathrm{dist}(p, \mathcal{L})/\varepsilon$, then since p is not badly cut the ball $\beta(p, \mathrm{dist}(p, \mathcal{L})/\varepsilon)$ in which lies $\mathrm{dist}(p, \mathrm{OPT})$ is cut at level at most $\log(3\mathrm{dist}(p, \mathcal{L})/\varepsilon) + \tau(\varepsilon, d, z)$.

In all cases, p and $\mathcal{L}(p)$ are cut at level at most $\log(3\text{dist}(p,\mathcal{L})/\varepsilon + 4\text{dist}(p,\text{OPT})) + \tau(\varepsilon,d,z)$. This concludes the proof. □

We then aim at proving that there exists a near-optimal "portal-respecting" solution, as we define below. A *path* between two points x and y is a sequence of points $w_1,\dots,w_k$, where $x = w_1$ and $y = w_k$, and its length is $\sum \text{dist}(w_j, w_{j+1})$. A solution can be seen as a set of facilities, together with a path for each client that connects it to a facility, and the cost of the solution is given by the sum over all path lengths. We say that a path $w_1,\dots,w_k$ is *portal-respecting* if for every pair w_j, w_{j+1}, whenever w_j and w_{j+1} lie in different parts $B, B' \in \mathcal{B}_i$ of the decomposition $\mathcal{D}$ on some level i, then these nodes are also portals at this level, i.e., $w_j, w_{j+1} \in \mathcal{P}_B \cup \mathcal{P}_{B'}$. As explained in Lemma 3.35, if two points x and y are cut at level i, then there exists a portal-respecting path from x to y of length at most $\text{dist}(u,v) + 16\rho 2^i$. We define a *portal-respecting* solution to be a solutions such that each path from a client to its closest facility in the solution is portal-respecting. The cost of a portal-respecting solution is the sum of the length of the path raised to the power z.

The dynamic program will a portal-respecting solution with optimal cost. Therefore, we need to prove that the optimal portal-respecting solution is close to the optimal solution. We actually show something slightly stronger. Given a solution S, we define $b(\mathcal{S}) := (1+z/\varepsilon)^{z-1}2^z \sum_{p,i:\ p \text{ and } \mathcal{S}(p) \text{ cut at level } i} \left(\varepsilon^2 2^{-4-\tau(\varepsilon,d,z)} \cdot 2^i\right)^z$: one can see $b(\mathcal{S})$ as a *budget*, given by the fact that clients are not badly cut. Next we show a structural lemma, that bounds the cost of a structured solution and of its budget.

▶ **Lemma 3.15 (Structural lemma).** Given an instance $\mathcal{I}$, an ε such that $0 < \varepsilon \le 1/4$ and a solution $\mathcal{L}$, and $\rho = \frac{\varepsilon^2}{128z} 2^{-\tau(\varepsilon,d,z)}$, it holds with probability $1-\varepsilon$ (over a decomposition $\mathcal{D}$ with parameter ρ) that there exists a portal-respecting solution $\mathcal{S}$ in $\mathcal{I}_\mathcal{D}$ such that $\text{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}) + b(\mathcal{S}) = (1+O(\varepsilon))\text{cost}_\mathcal{I}(\text{OPT}) + O(\varepsilon \text{cost}_\mathcal{I}(\mathcal{L}))$. ◀

Proof. From Lemma 3.13, with probability $1-\varepsilon$ it holds that the instance $\mathcal{I}_\mathcal{D}$ has small distortion, and $\text{cost}_{\mathcal{I}_\mathcal{D}}(\text{OPT}') \le (1+4\varepsilon)(\text{cost}_\mathcal{I}(\text{OPT}) + \varepsilon \text{cost}_\mathcal{I}(\mathcal{L}))$.

We now bound the cost of making OPT' portal respecting by applying Lemma 3.14. Since each client p of $\mathcal{I}_\mathcal{D}$ is cut from $\text{OPT}'(p)$ at level at most $\log(3\text{dist}(p,\mathcal{L})/\varepsilon + 4\text{dist}(p,\text{OPT})) + \tau(\varepsilon,d,z)$, we have from Lemma 3.35 that the length of the portal-respecting assignment of p to $\text{OPT}'(p)$ is at most

$$O\left(\text{dist}(p,\text{OPT}'(p)) + 64\rho 2^{\tau(\varepsilon,d,z)}(\text{dist}(p,\mathcal{L})/\varepsilon + \text{dist}(p,\text{OPT}))\right).$$

Raised to the power z to compute the cost of the path, this is at most (using Lemma 1.2)

$$O\left((1+\varepsilon)\text{cost}(p,\text{OPT}'(p)) + (1+z/\varepsilon)^{z-1}\left(64\rho 2^{\tau(\varepsilon,d,z)}(\text{dist}(p,\mathcal{L})/\varepsilon + \text{dist}(p,\text{OPT}))\right)^z\right).$$

Choosing $\rho = \frac{\varepsilon^2}{128z} 2^{-\tau(\varepsilon,d,z)}$ ensures that the portal-respecting cost of p is at most $(1+O(\varepsilon))\text{cost}(p,\text{OPT}) + O(\varepsilon)\text{cost}(p,\mathcal{L})$. Summing over all clients p gives a total cost

of $(1+O(\varepsilon))\text{cost}_{\mathcal{I}}(\text{OPT})+O(\varepsilon)\text{cost}_{\mathcal{I}}(\mathcal{L})$. The resulting portal respecting tour is the solution $\mathcal{S}$ we are looking for.

We proceed the same way to bound $b(\text{OPT}')$:

$$\begin{aligned} b(\text{OPT}') &= (1+z/\varepsilon)^{z-1}2^z \sum_p \left(4\varepsilon^2 2^{-4-\tau(\varepsilon,d,z)} \cdot 2^{\tau(\varepsilon,d,z)}(\text{dist}(p,\mathcal{L})/\varepsilon + \text{dist}(p,\text{OPT}))\right)^z \\ &\leq (1+z/\varepsilon)^{z-1}2^z \sum_p (\varepsilon/2)^z(\text{cost}(p,\mathcal{L}) + \text{cost}(p,\text{OPT})) \\ &\leq \varepsilon(\text{cost}_{\mathcal{I}}(\mathcal{L}) + \text{cost}_{\mathcal{I}}(\text{OPT})). \end{aligned}$$

Hence $\text{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}) + b(\mathcal{S}) = (1+O(\varepsilon))\text{cost}_{\mathcal{I}}(\text{OPT}) + O(\varepsilon\text{cost}_{\mathcal{I}}(\mathcal{L}))$. □

3.3.3 The Algorithm

Our algorithm starts by computing a $O(2^d)$-approximation $\mathcal{L}$ in time $O(n \log n)$, with the procedure described in Section 3.7.3. Essentially, this procedure reduces the problem to solving Facility Location in a tree, which can be done with a simple dynamic program.

The algorithm then computes a hierarchical decomposition $\mathcal{D}$, as explained in the Section 3.2.2, with parameter $\rho = \frac{\varepsilon^2}{128z}2^{-\tau(\varepsilon,d,z)}$.

Given $\mathcal{L}$ and the decomposition $\mathcal{D}$, our algorithm finds all the badly cut clients as follows. For each client p, to determine whether p is badly cut or not, the algorithm checks whether the decomposition cuts $\beta(p, 3\text{dist}(p,\mathcal{L})/\varepsilon)$ at a level higher than $\log(3\text{dist}(p,\mathcal{L})/\varepsilon)+\tau(\varepsilon,d,z)$, making p badly cut. This can be done efficiently, since p is in exactly one part at each level, by verifying whether p is at distance smaller than $3\text{dist}(p,\mathcal{L})/\varepsilon$ to such a part of too high level. Thus, the algorithm finds all the badly cut clients in near-linear time.

The next step of the algorithm is to compute instance $\mathcal{I}_{\mathcal{D}}$ by moving every badly cut client p to its facility in $\mathcal{L}$. This can be done in linear time.

A first attempt at a dynamic program. We now turn to the description of the dynamic program (DP) for obtaining the best portal-respecting solution of $\mathcal{I}_{\mathcal{D}}$. This is the standard dynamic program for Facility Location and we only describe it for the sake of completeness. The reader familiar with this can therefore skip to the analysis.

There is a table entry for each part of the decomposition, and two vectors of length $|\mathcal{P}_B|$, where $\mathcal{P}_B$ is the set of portals in the part B. We call such a triplet a configuration. Each configuration given by a part B and vectors $\langle \ell_1, \ldots, \ell_{|\mathcal{P}_B|}\rangle$ and $\langle s_1, \ldots, s_{|\mathcal{P}_B|}\rangle$ (called the *portal parameters*), encodes a possible interface between part B and a solution for which the ith portal has distance ℓ_i to the closest facility inside of B, and distance s_i to its closest facility outside of B. The value stored for such a configuration in a table entry is the minimal cost for a solution with facilities respecting the constraints induced by the vectors on the distances between the solution and the

portals inside the part (as described below).

To fill the table, we use a dynamic program following the lines of Arora et al. [9] or Kolliopoulos and Rao [112]. If a part has no descendant (meaning the part contains a single point), computing the solution given the configuration is straightforward: either a center is opened on this point or not, and it is easy for both cases to check whether they are consistent with the configuration, where only the distances to portals inside the part need to be verified. At a higher level of the decomposition, a solution is simply obtained by going over all the sets of parameter values for all the children parts. It is immediate to see whether sets of parameter values for the children can lead to a consistent solution:

- for each portal p_i of the parent part, there must be one portal p_j of a child part such that the distance from p_i to a center inside the parent part prescribed by the configuration (given by the value ℓ_i for the parent part) corresponds to $\text{dist}(p_i, p_j)$ plus the distance from p_j to a center inside the child part (given by the value ℓ_j for the child part);
- for each portal p_j of a child part there must exist either:
  - a portal p_i of the parent part such that the distance from p_j to a center outside its part prescribed by the configuration (given by the value s_j of the child part) is $\text{dist}(p_i, p_j)$ plus the distance from p_i to a center outside of the part (given by the value s_i of the parent part), or
  - a portal p_i of another child part such that this distance given by the value s_j of the first child part is $\text{dist}(p_i, p_j)$ plus the distance from p_i to a center inside the child part (given by the value ℓ_i of this child part).

The runtime of this algorithm depends on the number of possible distances determining the number of possible portal parameters. Even if the aspect ratio is polynomial, there can be a large number of possible distances, so that the number of configurations might be exponential. Using the budget given by Lemma 3.15, one can approximate the distances and obtain an efficient algorithm, as we show next.

A faster dynamic program. We now describe a faster dynamic program. Given a facility location solution $\mathcal{S}$ and a part B, we define the *internal* cost of $\mathcal{S}$ as the opening cost of the facilities in $\mathcal{S} \cap B$ plus the service cost of the clients in B.

Consider a level where the diameter of the parts is say D. A table entry of the dynamic program is

$$T[B, \ell_1, \ldots, \ell_{|\mathcal{P}_B|}, s_1, \ldots, s_{|\mathcal{P}_B|}, f],$$

where

- B is a part;
- each of ℓ_i and s_i is a multiple of $\varepsilon^2 2^{-4-\tau(\varepsilon,d,z)} D$ in the range $[0, D/\varepsilon + D]$, whose intended meaning is to represent the portal parameters for B;

- f is a boolean value that is 1 if and only if the closest facility to any of the portals of B is at distance larger than D/ε (in particular, there is no facility within the part and the portal parameters are irrelevant).

The intended value of the table entry is

- if $f = 1$: the number of clients inside B, and
- if $f = 0$: the cost of a solution that abides by the portal parameters and that has minimum internal cost.

A few remarks are in order. Since the diameter of the part is D we can afford a detour of εD, and so we only need to store the approximate portal parameters.

In the case where $f = 1$, the diameter of the part B is D and every facility is at distance at least $\mathrm{OPT} \geq D/\varepsilon$ from any portal B, while any point of B is at distance at most εD from a portal. Hence up to losing an additive $D + \varepsilon D \leq O(\varepsilon \mathrm{OPT})$ in the cost of the computed solution, we may assume that all the points of the part are assigned to the same facility. So the algorithm is not required to store the precise distance to the closest facility outside the part, and it uses the flag f to reflect that it is in this scenario. The algorithm then treats this whole part as a single client (weighted by the number of clients inside the part and located at an arbitrary location inside the part) to be considered at higher levels.

If on the other hand the closest facility to any portal of B is at distance less than D/ε, we have that for any portal of the part the closest facility is at distance at most $D/\varepsilon + D$ (since D is the diameter of the part).

We now describe the recurrence relationship to populate the table entries. In the base case, a part B contains at most one point and so at most $|\mathcal{P}_B| + 1$ distinct facilities – one at each portal, and one on the point inside the region. It can thus be solved by going over all $2^{|\mathcal{P}_B|+1}$ possibilities.[1] Otherwise, a given table entry for a part B is computed by considering the pre-computed solutions for the $2^{O(d)}$ child parts of B.

If $f = 1$, then the value of the entry is simply the number of clients within B and should be computed by summing up the entries of the child parts of B. In the remaining case when $f = 0$, a set of table entries is said to be *compatible* with a table entry for B if the set contains exactly one entry for any of the child parts of B and:

- for each portal p_i of the parent part, there must be one portal p_j of a child part such that the distance from p_i to a center inside the parent part prescribed by the configuration (given by the value ℓ_i for the parent part) corresponds to $\mathrm{dist}(p_i, p_j)$ plus the distance from p_j to a center inside the child part (given by the value ℓ_j for the child part);
- for each child part that has $f = 0$, and for each portal p_j of that child part, there must exist either:

[1] In the Euclidean case, one can discretize the region simply by taking an $\varepsilon^2 2^{-4-\tau(\varepsilon,d,z)} D$-net, which has size $\varepsilon^{-2} 2^{O(\tau(\varepsilon,d,z))}$.

  - a portal p_i of the parent part such that the distance from p_j to a center outside its part prescribed by the configuration (given by the value s_j of the child part) is $\text{dist}(p_i, p_j)$ plus the distance from p_i to a center outside of the part (given by the value s_i of the parent part), or
  - a portal p_i of another child part such that this distance given by the value s_j of the first child part is $\text{dist}(p_i, p_j)$ plus the distance from p_i to a center inside the child part (given by the value ℓ_i of this child part).

- for each child part that has $f = 1$ and diameter D, and for each portal p_j of that child part, there is no portal p_i either of the parent part or another child part such that: $\text{dist}(p_j, p_i)$ plus the distance from p_i to a center (given by values ℓ_i or s_i) is less than D/ε.

Given such a set of table entries, its value is computed by summing up the values of the entries where $f = 0$, and assigning the points of parts B' that have $f = 1$ to their closest facility outside B' (since there is no facility within B') according to the portal parameters of the other parts. The value of the entry for B is the minimum value over all compatible sets of table entries for child parts.

Analysis – Proof of Theorem 3.3. The following lemmas show that the solution computed by this algorithm is a near-optimal one, and that the complexity is near-linear: this proves Theorem 3.3. We first bound the connection cost in $\mathcal{I}_{\mathcal{D}}$.

▶ **Lemma 3.16.** Let $\mathcal{S}$ be as in Lemma 3.15. The algorithm computes a solution $\mathcal{S}^*$ with cost at most $\text{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}^*) \leq (1 + O(\varepsilon))\text{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}) + b(\mathcal{S})$. ◀

Proof. We show that the solution $\mathcal{S}$ can be adapted to a configuration of the DP with extra cost $b(\mathcal{S})$. For this, let p be a client served by a facility $\mathcal{S}(p)$, and let $w_1, \ldots, w_k$ be the portal-respecting path from p to $\mathcal{S}(p)$ with $w_1 = c$ and $w_k = \mathcal{S}(p)$. The cost contribution of p to $\mathcal{S}$ is therefore $\sum_{i=1}^{k-1} \text{cost}(w_i, w_{i+1})$. For each w_i, let also l_i be the level at which w_i is a portal.

The distance between p and $\mathcal{S}(p)$ is approximated at several places of the DP. Consider any node w_i on the path from p to $\mathcal{S}(p)$:

- When $\text{dist}(w_i, \mathcal{S}(p)) \leq 2^{l_i}/\varepsilon + 2^{l_i}$, the distance between w_i and $\mathcal{S}(p)$ is rounded to the closest multiple of $\varepsilon^2 2^{-4-\tau(\varepsilon,d,z)} 2^{l_i}$. Since the diameters of the parts are geometrically increasing, the sum of all those errors is at most $\varepsilon^2 2^{-4-\tau(\varepsilon,d,z)} 2^{l_c}$, where l_c is the level at which p and $\mathcal{S}(p)$ are cut.

- When $\text{dist}(w_i, \mathcal{S}(p)) \geq 2^{l_i}/\varepsilon + 2^{l_i}$, the whole part is contracted and served by a single facility at distance at least $2^{l_i}/\varepsilon$. The difference for client p is therefore 2^{l_i}. This difference is only paid once, at the the highest level l_j where $\text{dist}(w_j, \mathcal{S}(p)) \geq 2^{l_j}/\varepsilon + 2^{l_j}$. This inequality implies that $2^{l_j} \leq \varepsilon \text{cost}(w_j, \mathcal{S}(p))$, which is smaller than $\varepsilon \sum \text{dist}(w_i, w_{i+1})$.

Hence, the total rounding error induced by the evaluation of $\text{dist}(c, \mathcal{S}(p))$ is $\varepsilon^2 2^{-2-\tau(\varepsilon,d,z)} 2^{l_c} + \varepsilon \sum \text{dist}(w_i, w_{i+1})$. Using Lemma 1.2, this implies a cost error of

$$(1 + z/\varepsilon)^{z-1} 2^z \cdot \left(\varepsilon^{2z} 2^{-4z-\tau(\varepsilon,d,z)\cdot z} 2^{z\cdot l_c} + 2^{z\cdot l_j} \right).$$

The first term, summed over all clients p is precisely equal to the budget $b(\mathcal{S})$. The second one verifies, $(1+z/\varepsilon)^{z-1} 2^z \cdot 2^{z\cdot l_j} \leq (1+z/\varepsilon)^{z-1} 2^z \cdot \varepsilon^z \left(\sum \text{dist}(w_i, w_{i+1})\right)^z$, which is at most ε times the cost of p in the portal-respecting solution $\mathcal{S}$.

Hence, summing over all clients, the additional cost incurred by the DP is at most $b(\mathcal{S}) + 2^z \varepsilon^z \text{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S})$. Since the DP computes a solution with minimal cost, it holds that $\text{cost}_{\mathcal{I}_\mathcal{D}}(S^*) \leq (1 + 4\varepsilon)\text{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}) + b(\mathcal{S})$. □

Combining all previous result, we can now finally bound the connection cost in $\mathcal{I}$ of the solution $\mathcal{S}^*$ computed by the algorithm.

▶ **Corollary 3.17.** Let S^* be the solution computed by the algorithm. With probability $1-\varepsilon$, it holds that $\text{cost}_\mathcal{I}(S^*) = (1+O(\varepsilon))\text{cost}_\mathcal{I}(\text{OPT}) + O(\varepsilon)\text{cost}_\mathcal{I}(\mathcal{L})$. ◀

Proof. Lemma 3.15 ensures that, with probability $1 - \varepsilon$, the cost of $\mathcal{S}$ in $\mathcal{I}_\mathcal{D}$ and $b(\mathcal{S})$ is at most $(1 + O(\varepsilon))\text{cost}_\mathcal{I}(\text{OPT}) + O(\varepsilon \text{cost}_\mathcal{I}(\mathcal{L}))$. Using that $\mathcal{I}_\mathcal{D}$ has small distortion, and combining this with Lemma 3.16 concludes the proof:

$$\begin{aligned}
\text{cost}_\mathcal{I}(\mathcal{S}^*) &\leq (1 + 2\varepsilon)\text{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}^*) + \varepsilon \text{cost}_\mathcal{I}(\mathcal{L}) \\
&\leq (1 + O(\varepsilon))(\text{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}) + b(\mathcal{S})) + \varepsilon \text{cost}_\mathcal{I}(\mathcal{L}) \\
&\leq (1 + O(\varepsilon))\text{cost}_\mathcal{I}(\text{OPT}) + O(\varepsilon)\text{cost}_\mathcal{I}(\mathcal{L})
\end{aligned}$$

□

If $\mathcal{I}_\mathcal{D}$ has small distortion, the facility cost increase due to badly cut clients is bounded by $\sum_{f \in B_\mathcal{D}} w_f \leq \varepsilon \sum_{f \in L} w_f$, since we have $\text{OPT}' = \text{OPT} \cup B_\mathcal{D}$. This observation and the previous corollary conclude the bound on the total cost of $\mathcal{S}^*$, as required.

According to Appendix 3.7.3, we can compute a solution $\mathcal{L}$ with cost $O(2^d)\text{cost}_\mathcal{I}(\text{OPT})$ in $O(n \log n)$ time. Due to the above corollary, we obtain a solution with cost $(1 + O(\varepsilon 2^d))\text{cost}_\mathcal{I}(\text{OPT})$, with probability $1 - \varepsilon$. The probability can be boosted to $1 - \varepsilon^\delta$ by repeating the process δ times. In order to obtain a $(1 + O(\varepsilon))$-approximation, we can *bootstrap* as in Bateni et al. [23], i.e., we repeat the process starting from the computed solution instead of $\mathcal{L}$. Repeating this N times yields a solution of cost $(1 + \sum_{i=1}^{N-1} O(\varepsilon^i) + O(\varepsilon^N 2^d))\text{cost}_\mathcal{I}(\text{OPT})$, with probability $1 - N\varepsilon^\delta$. Taking $N = d$ and $\delta = \log(d/\varepsilon)/\log(1/\varepsilon)$ ensures that, with probability at least $1 - \varepsilon$, the solution has cost $(1 + O(\varepsilon))\text{cost}_\mathcal{I}(\text{OPT})$.

▶ **Lemma 3.18.** This algorithm runs in $2^{(1/\varepsilon)^{O(d^2)}} n + 2^{O(d)} n \log n$ time. ◀

Proof. The preprocessing step (computing $\mathcal{L}$, the hierarchical decomposition $\mathcal{D}$, and the instance $\mathcal{I}_{\mathcal{D}}$) has a running time $O(n \log n)$, as all the steps can be done with this complexity: computing $\mathcal{L}$ takes time $O(n \log n)$ as described in Appendix 3.7.3, and as explained earlier, the hierarchical decomposition $\mathcal{D}$ and the instance $\mathcal{I}_{\mathcal{D}}$ can also be computed with this complexity. The decomposition can moreover be transformed in order to remove part that do not contain any point, as well as degree 2 nodes. This ensures to have $O(n)$ part in total, since there are n leaves and a degree at least 3.

The DP has a linear time complexity: in a part of diameter D, the portal set is an $(\frac{\varepsilon^2}{128z} 2^{-\tau(\varepsilon,d,z)} D)$-net, and hence has size $2^{O\left(d \log(2^{\tau(\varepsilon,d,z)}/\varepsilon)\right)}$ by Lemma 3.9. Since $\tau(\varepsilon, d, z) = 2d + 2 + \log \frac{(z+\varepsilon)^z}{\varepsilon^2 z^z}$, this number can be simplified to $2^{O(d^2 + d \log(1/\varepsilon))}$. Since each portal stores a distance that can take only $\varepsilon^{-2} 2^{2+\tau(\varepsilon,d,z)}$ values, there are at most $T = (\varepsilon^{-2} 2^{\tau(\varepsilon,d,z)})^{2^{O(d^2 + d \log(1/\varepsilon))}} = 2^{2^{O(d^2 + d \log(1/\varepsilon))}}$ possible table entries for a given part.

To fill the table, notice that a part has at most $2^{O(d)}$ children, due to the properties of the hierarchical decomposition. For any given part, going over all the sets of parameter values for all the children parts therefore takes time $T^{2^{O(d)}} = 2^{2^{O(d^2 + d \log(1/\varepsilon))}}$. This dominates the complexity of computing all table entry for one part of the decomposition.

Since the hierarchical decomposition is a tree with n leaves (one per input point) and without degree-one internal nodes (those can be compressed), there are at most n parts in the decomposition: the complexity of the dynamic program is therefore $n \cdot 2^{2^{O(d^2 + d \log(1/\varepsilon))}}$.

The bootstrapping increases that complexity by a factor $\log d$, absorbed in the big-O notation. The total complexity of the algorithm is thus

$$n \cdot 2^{2^{O(d^2 + d \log(1/\varepsilon))}} + 2^{O(d)} n \log n \qquad \square$$

3.4 The (k, z)-Clustering Problem

We aim at using the same approach as for Facility Location, but in the case where the number of opened facility is fixed to k. Again, we will work with the more general version of (k, z)-Clustering as defined in Section 3.2.4, where the instance consists of a set of clients P, a set of candidate centers F, an integer k, and a function $\chi : P \mapsto \{1, \ldots, n\}$ and the goal is to minimize $\sum_{p \in P} \chi(p) \cdot \min_{f \in \mathcal{S}} \operatorname{dist}(p, f)^z$. We will consider in the proof that P is actually a multiset, instead of carrying along the multiplicity χ.

The road-map is as for Facility Location: we show in Lemma 3.20 that an instance $\mathcal{I}_{\mathcal{D}}$ has a *small distortion* with good probability, and then in Lemma 3.23 that if an instance has small distortion then there exists a near-optimal portal-respecting solution. We finally present a dynamic program that computes such a solution.

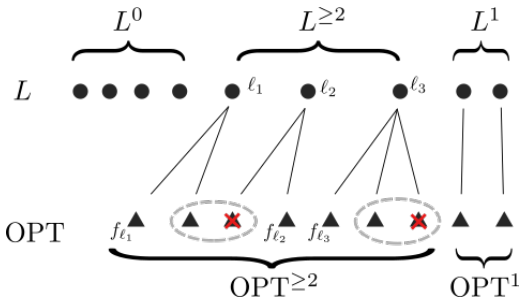


Figure 3.2: Illustration of Step 1. There is a ray between each facility f of OPT and $\mathcal{L}(f)$. Facilities in $\mathcal{H}$ are circled in grey; the removed facilities are crossed out in red.

A key ingredient of the proof for Facility Location was our ability to add all badly-cut facilities to the solution OPT′. This is not directly possible in the case of k-Median and k-Means, as the number of facilities is fixed. Hence, the first step of our proof is to show that one can make some room in OPT, by removing a few centers without increasing the cost by too much.

3.4.1 Towards a Structured Near-Optimal Solution

Let OPT be an optimal solution to $\mathcal{I}$ and $\mathcal{L}$ an approximate solution. We consider the mapping of the facilities of OPT to $\mathcal{L}$ defined as follows: for any $f \in \text{OPT}$, let $\mathcal{L}(f)$ denote the facility of $\mathcal{L}$ that is the closest to f. Recall that for a client p, $\mathcal{L}(p)$ is the facility serving p in $\mathcal{L}$.

For any facility ℓ of $\mathcal{L}$, define $\psi(\ell)$ to be the set of facilities of OPT that are mapped to ℓ, namely, $\psi(\ell) = \{f \in \text{OPT} \mid \mathcal{L}(f) = \ell\}$. Define $\mathcal{L}^1$ to be the set of facilities ℓ of $\mathcal{L}$ for which there exists a unique $f \in \text{OPT}$ such that $\mathcal{L}(f) = \ell$, namely $\mathcal{L}^1 = \{\ell \in \mathcal{L} \mid |\psi(\ell)| = 1\}$. Let $\mathcal{L}^0 = \{\ell \in \mathcal{L} \mid |\psi(\ell)| = 0\}$, and $\mathcal{L}^{\geq 2} = \mathcal{L} \setminus (\mathcal{L}^1 \cup \mathcal{L}^0)$. Similarly, define $\text{OPT}^1 = \{f \in \text{OPT} \mid \mathcal{L}(f) \in \mathcal{L}^1\}$ and $\text{OPT}^{\geq 2} = \{f \in \text{OPT} \mid \mathcal{L}(f) \in \mathcal{L}^{\geq 2}\}$. Note that $|\text{OPT}^{\geq 2}| = |\mathcal{L}^0| + |\mathcal{L}^{\geq 2}|$, since $|\text{OPT}^1| = |\mathcal{L}^1|$ and, w.l.o.g., $|\text{OPT}| = |\mathcal{L}| = k$.

The construction of a structured near-optimal solution is made in 3 steps. The first one defines a solution OPT′ as follows. Start with $\text{OPT}' = \text{OPT}$.

- **Step 1.** For each facility $\ell \in \mathcal{L}^{\geq 2}$, fix one in $\text{OPT}^{\geq 2}$ that is closest to ℓ, breaking ties arbitrarily, and call it f_ℓ. Let $\mathcal{H} \subseteq \text{OPT}^{\geq 2}$ be the set of facilities of $\text{OPT}^{\geq 2}$ that are not the closest to their corresponding facility in $\mathcal{L}^{\geq 2}$, i.e., $f \in \mathcal{H}$ if and only if $f \in \psi(\ell)$ and $f \neq f_\ell$ for some $\ell \in \mathcal{L}^{\geq 2}$. Among the facilities of $\mathcal{H}$, remove from OPT′ the subset of size $\lfloor \varepsilon \cdot |\text{OPT}^{\geq 2}|/2 \rfloor$ that yields the smallest cost increase. Note that this subset is well-defined if $\varepsilon \leq 1$.

This step, illustrated in Figure 3.2, makes room to add the badly cut facilities without violating the constraint on the maximum number of centers, while at the same time ensures near-optimal cost, as the following lemma shows.

▶ **Lemma 3.19.** After Step 1, OPT′ has cost $(1 + O(\varepsilon))\text{cost}(\text{OPT}) + O(\varepsilon)\text{cost}(\mathcal{L})$ ◀

Proof. We claim that for a client p served by $f \in \mathcal{H}$ in the optimum solution OPT, i.e., $f = \text{OPT}(p)$, the detour entailed by the deletion of f is $O(\text{OPT}_p + \mathcal{L}_p))$. Indeed, let f' be the facility of OPT that is closest to $\mathcal{L}(f)$, and recall that $\mathcal{L}(p)$ is the facility that serves p in the solution $\mathcal{L}$. Since $f' \notin \mathcal{H}$, the cost to serve p after the removal of f is at most $\text{dist}(p, f')$, which can be bounded by $\text{dist}(p, f') \leq \text{dist}(p, f) + \text{dist}(f, \mathcal{L}(f)) + \text{dist}(\mathcal{L}(f), f')$. But by definition of f', $\text{dist}(f', \mathcal{L}(f)) \leq \text{dist}(\mathcal{L}(f), f)$, and by definition of the function $\mathcal{L}$ we have $\text{dist}(\mathcal{L}(f), f) \leq \text{dist}(\mathcal{L}(p), f)$, so that $\text{dist}(p, f') \leq \text{dist}(p, f) + 2\text{dist}(f, \mathcal{L}(p))$. Using the triangle inequality finally gives $\text{dist}(p, f') \leq 3\text{dist}(p, f) + 2\text{dist}(p, \mathcal{L}(p))$ which is $O(\text{dist}(p, \text{OPT}) + \text{dist}(p, \mathcal{L}))$. For a facility f of OPT, we denote by $P(f)$ the set of clients served by f, i.e. $P(f) = \{p \in P \mid \text{OPT}(p) = f\}$. The total cost incurred by the removal of f is then $\sum_{p \in P(f)} O(2^z(\text{cost}(p, \text{OPT}) + \text{cost}(p, \mathcal{L}))$, and the cost of removing all of $\mathcal{H}$ is $O(\text{cost}(\text{OPT}) + \text{cost}(\mathcal{L}))$.

Recall that in Step 1 we remove the set $\widehat{\mathcal{H}}$ of size $\lfloor \varepsilon |\text{OPT}^{\geq 2}|/2 \rfloor$ from $\mathcal{H}$, such that $\widehat{\mathcal{H}}$ minimizes the cost increase. We use an averaging argument to bound the cost increase: the sum among all facilities $f \in \mathcal{H}$ of the cost of removing the facility f is less than $O(\text{cost}(\text{OPT}) + \text{cost}(\mathcal{L}))$, and $|\mathcal{H}| = O(1/\varepsilon) \cdot \lfloor \varepsilon |\text{OPT}^{\geq 2}| \rfloor$. Therefore removing $\widehat{\mathcal{H}}$ increases the cost by at most $O(\varepsilon)(\text{cost}(\text{OPT}) + \text{cost}(\mathcal{L}))$, so that Step 1 is not too expensive. □

We can therefore use this solution OPT′ as a proxy for the optimal solution, and henceforth we will denote this solution by OPT. In particular, the badly cut facilities are defined for this solution and not the original OPT.

3.4.2 An instance with small distortion

As in Section 3.3, the algorithm computes a randomized hierarchical decomposition $\mathcal{D}$, and transforms the instance of the problem: every badly cut client p is moved to $\mathcal{L}(p)$, namely, there is no more client at p and we add an extra client at $\mathcal{L}(p)$. Again, we let $\mathcal{I}_\mathcal{D}$ denote the resulting instance and note that $\mathcal{I}_\mathcal{D}$ is a random variable that depends on the randomness of $\mathcal{D}$.

Moreover, similar as for Facility Location, we let $B_\mathcal{D}$ be the set of centers of $\mathcal{L}$ that are badly cut from OPT, i.e., $f \in B_\mathcal{D}$ if the ball $\beta(f, 3\text{dist}(f, \text{OPT}))$ is cut at some level greater than $\log(3\text{dist}(f, \text{OPT})) + \tau(\varepsilon, d, z)$. We call $\text{cost}_\mathcal{I}(\mathcal{S})$ the cost of a solution $\mathcal{S}$ in the original instance $\mathcal{I}$, and $\text{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S})$ its cost in $\mathcal{I}_\mathcal{D}$. We let

$$\nu_{\mathcal{I}_\mathcal{D}} = \max_{\text{solution } \mathcal{S}} \left\{ \text{cost}_\mathcal{I}(\mathcal{S}) - (1 + 2\varepsilon)\text{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}), (1 - 2\varepsilon)\text{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}) - \text{cost}_\mathcal{I}(\mathcal{S}) \right\}.$$

We say that an instance $\mathcal{I}_\mathcal{D}$ has *small distortion* if $\nu_{\mathcal{I}_\mathcal{D}} \leq \varepsilon \text{cost}_\mathcal{I}(\mathcal{L})$, and there exists a solution $\mathcal{S}$ that contains $B_\mathcal{D}$ with $\text{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}) \leq (1 + O(\varepsilon))\text{cost}_\mathcal{I}(\text{OPT}) + O(\varepsilon)\text{cost}_\mathcal{I}(\mathcal{L})$. That is, the condition is the same as for Facility Location, except that we do not need

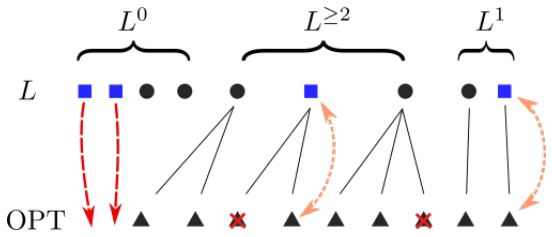


Figure 3.3: Illustration of Steps 2 and 3. Badly-cut facilities are represented by squares. Orange short-dashed arrows represent facilities that are exchanged at Step 2. Red long-dashed arrows represent facilities added to $\mathcal{S}^*$ in Step 3.

a bound on the opening costs. In contrast to the Facility Location problem, here we need to be more careful when identifying the solution fulfilling the latter inequality.

For this, we go on with the next two steps of our construction, defining a solution $\mathcal{S}^*$. Recall that we defined $f_\ell \in \mathrm{OPT}^{\geq 2}$ to be the closest facility to $\ell \in \mathcal{L}^{\geq 2}$, breaking ties arbitrarily. For any $\ell \in \mathcal{L}^1$, we also denote by $f_\ell \in \mathrm{OPT}^1$ the unique facility closest to ℓ. We start with $\mathcal{S}^* = \mathrm{OPT}$ obtained from Step 1. Note that for every $\ell \in \mathcal{L}^1 \cup \mathcal{L}^{\geq 2}$ the closest facility $f_\ell \in \mathrm{OPT}$ is still present after Step 1, since only some of the other facilities in $\mathcal{H}$ were removed.

- **Step 2.** For each badly-cut facility $\ell \in B_{\mathcal{D}} \setminus \mathcal{L}^0$ (i.e., $\psi(\ell) \neq \emptyset$), replace f_ℓ by ℓ in $\mathcal{S}^*$.
- **Step 3.** Add all badly cut facilities of $\mathcal{L}^0$ to $\mathcal{S}^*$.

We show next that $\mathcal{S}^*$ satisfies the conditions for $\mathcal{I}_{\mathcal{D}}$ to have small distortion with good probability.

▶ **Lemma 3.20.** The probability that $\mathcal{I}_{\mathcal{D}}$ has small distortion is at least $1-\varepsilon$, if $\varepsilon \leq 1/4$. ◀

Proof. The proof that $\nu_{\mathcal{I}_{\mathcal{D}}} \leq \varepsilon \mathrm{cost}_{\mathcal{I}}(\mathcal{L})$ with probability at least $1 - \varepsilon/2$ is identical to the one in Lemma 3.13. We thus turn to bound the probability that solution $\mathcal{S}^*$ satisfies the cardinality and cost requirements. Our goal is to show that this happens with probability at least $1 - \varepsilon/2$. Then, taking a union bound over the probabilities of failure yields the proposition.

By Steps 2 and 3, we have that $\mathcal{S}^*$ contains $B_{\mathcal{D}}$. We split the proof of the remaining properties into the following claims.

▶ **Claim 3.21.** With probability at least $1-\varepsilon/4$, the set $\mathcal{S}^*$ is an admissible solution, i.e., $|\mathcal{S}^*| \leq k$. ◀

Proof. We let b be the number of facilities of $\mathcal{L}^0$ that are badly cut. By Lemma 3.12, we have that $\mathbb{E}[b] \leq \kappa(\varepsilon, z)|\mathcal{L}^0| \leq \varepsilon^2|\mathcal{L}^0|/4$. By Markov's inequality, the probability

that $b > \lfloor \varepsilon |\mathcal{L}^0|/2 \rfloor$ is at most $\varepsilon/2$. Now, condition on the event that $b \leq \lfloor \varepsilon |\mathcal{L}^0|/2 \rfloor$. Since $|\mathcal{L}^0| + |\mathcal{L}^{\geq 2}| = |\mathrm{OPT}^2|$, we have that $b \leq \lfloor \varepsilon |\mathrm{OPT}^{\geq 2}|/2 \rfloor$. Moreover, the three steps converting OPT into $\mathcal{S}^*$ ensure that $|\mathcal{S}^*| \leq k + b - \lfloor \varepsilon |\mathrm{OPT}^{\geq 2}|/2 \rfloor$, as Step 1 removes $\lfloor \varepsilon |\mathrm{OPT}^{\geq 2}|/2 \rfloor$ facilities, while Step 2 only swaps facilities so their number does not change, and Step 3 adds b facilities. Combining the two inequalities gives $|\mathcal{S}^*| \leq k$. □

▶ **Claim 3.22.** If $\varepsilon < 1/4$, then with probability at least $1 - \varepsilon/4$, $\mathrm{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}^*) \leq (1 + O(\varepsilon))\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + O(\varepsilon \cdot \mathrm{cost}_\mathcal{I}(\mathcal{L}))$ ◀

Proof. We showed in Lemma 3.19 that the cost increase in $\mathcal{I}$ due to Step 1 is at most $O(\varepsilon)(\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + \mathrm{cost}_\mathcal{I}(\mathcal{L}))$. We will prove below that this implies that also Step 2 leads to a cost increase of $O(\varepsilon)(\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + \mathrm{cost}_\mathcal{I}(\mathcal{L}))$ in $\mathcal{I}$ with good probability. Step 3 can only decrease the cost. Hence we have $\mathrm{cost}_\mathcal{I}(\mathcal{S}^*) \leq (1+O(\varepsilon))\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + O(\varepsilon \cdot \mathrm{cost}_\mathcal{I}(\mathcal{L}))$. To bound the cost of $\mathcal{S}^*$ in $\mathcal{I}_\mathcal{D}$, we use that $(1 - 2\varepsilon)\mathrm{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}^*) - \mathrm{cost}_\mathcal{I}(\mathcal{S}^*) \leq \nu_{\mathcal{I}_\mathcal{D}} \leq \varepsilon \mathrm{cost}_\mathcal{I}(\mathcal{L})$. The cost incurred by connecting clients to facilities in $\mathcal{S}^*$ is

$$\begin{aligned}
\mathrm{cost}_{\mathcal{I}_\mathcal{D}}(\mathcal{S}^*) &\leq (1 + \frac{2\varepsilon}{1-2\varepsilon})(\mathrm{cost}_\mathcal{I}(\mathcal{S}^*) + \varepsilon \cdot \mathrm{cost}_\mathcal{I}(\mathcal{L})) && (\text{as } \nu_{\mathcal{I}_\mathcal{D}} \leq \varepsilon \mathrm{cost}_\mathcal{I}(\mathcal{L})) \\
&\leq (1 + 4\varepsilon)(\mathrm{cost}_\mathcal{I}(\mathcal{S}^*) + \varepsilon \cdot \mathrm{cost}_\mathcal{I}(\mathcal{L})) && (\text{as } \varepsilon \leq 1/4) \\
&\leq (1 + O(\varepsilon))\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + O(\varepsilon \cdot \mathrm{cost}_\mathcal{I}(\mathcal{L})) && (\text{by above bound on } \mathrm{cost}_\mathcal{I}(\mathcal{S}^*)).
\end{aligned}$$

To bound the cost increase of Step 2, we first show that starting with OPT and replacing every $f_\ell \in \mathrm{OPT}$ by $\ell \in \mathcal{L}^1 \cup \mathcal{L}^{\geq 2}$ results in a solution $\mathcal{S}'$ of cost $O(\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + \mathrm{cost}_\mathcal{I}(\mathcal{L}))$. For that, let p be a client that in OPT is served by a facility f_ℓ that is closest to some $\ell \in \mathcal{L}^1 \cup \mathcal{L}^{\geq 2}$. Recall that every facility of OPT that is closest to some facility of $\mathcal{L}^1 \cup \mathcal{L}^{\geq 2}$ is in OPT, as only some of those from $\mathcal{H}$ are removed in Step 1. Hence if p is served by some $\ell' \in \mathcal{L}^1 \cup \mathcal{L}^{\geq 2}$ in the solution $\mathcal{L}$, then this facility ℓ' is in $\mathcal{S}'$ since it will replace the closest facility $f_{\ell'}$. Thus the cost of serving p in $\mathcal{S}'$ is $\mathrm{dist}(p, \ell') = \mathrm{dist}(p, \mathcal{L})$. On the other hand, if p is served by a facility ℓ_0 of $\mathcal{L}^0$ in $\mathcal{L}$, then it is possible to serve it by the facility ℓ that replaces f_ℓ. The serving cost then is $\mathrm{cost}(p, \ell) \leq (\mathrm{dist}(p, f_\ell) + \mathrm{dist}(f_\ell, \ell))^z \leq (\mathrm{dist}(p, f_\ell) + \mathrm{dist}(f_\ell, \ell_0))^z$, using that f_ℓ is the closest facility to ℓ in the last inequality. Using again the triangle inequality, this cost is at most $O(\mathrm{cost}(p, f_\ell) + \mathrm{cost}(p, \ell_0))$. Moreover, any client served by a facility of $\mathcal{H}$ in OPT, i.e., which is not the closest to a facility of $\mathcal{L}$, can in $\mathcal{S}'$ be served by the same facility as in OPT, with cost $\mathrm{cost}(p, \mathrm{OPT})$. Hence the cost of the obtained solution is at most $O(\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + \mathrm{cost}_\mathcal{I}(\mathcal{L})) = O(\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + \mathrm{cost}_\mathcal{I}(\mathcal{L}))$ by Lemma 3.19 and assuming, say, $\varepsilon \leq 1$.

The probability of replacing f_ℓ by $\ell \in \mathcal{L}^1 \cup \mathcal{L}^{\geq 2}$ in Step 2 is the probability that ℓ is badly cut. This is $\kappa(\varepsilon, z)$ by Lemma 3.12. Finally, with linearity of expectation, the expected cost to add the badly cut facilities $\ell \in \mathcal{L}^1 \cup \mathcal{L}^{\geq 2}$ instead of their closest facility f_ℓ of OPT in Step 2 is $O(\kappa(\varepsilon, z)(\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + \mathrm{cost}_\mathcal{I}(\mathcal{L})))$. Markov's inequality thus implies that the cost of this step is at most $O(\varepsilon)(\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + \mathrm{cost}_\mathcal{I}(\mathcal{L}))$ with probability $1 - \frac{O(\kappa(\varepsilon,z))}{\varepsilon} \geq 1 - \varepsilon/4$, since $\kappa(\varepsilon, z) \leq \varepsilon^2/2$. □

Lemma 3.20 follows from taking a union bound over the probabilities of failure of

Claim 3.21 and 3.22. □

3.4.3 Portal respecting solution

We have to prove the same structural lemma as for Facility Location, to say that there exists a portal-respecting solution in $\mathcal{I}_{\mathcal{D}}$ with cost close to $\text{cost}(\mathcal{S}^*)$ where $\mathcal{S}^*$ is the solution obtained from the three steps above. Recall that for any solution $\mathcal{S}$ and client p, $\mathcal{S}_p$ is the distances from the *original* position of p to $\mathcal{S}$ in $\mathcal{I}$, but p may have been moved to $\mathcal{L}(p)$ in $\mathcal{I}_{\mathcal{D}}$. Recall also that OPT is defined after removing some centers in Step 1.

▶ **Lemma 3.23.** Let $\mathcal{I}$ be an instance of k-Median with a randomized decomposition $\mathcal{D}$ (for any parameter $\rho > 0$), and $\mathcal{L}$ be a solution for $\mathcal{I}$, such that $\mathcal{I}_{\mathcal{D}}$ has small distortion. Let $\mathcal{S}^*$ be the solution obtained by applying Steps 1, 2 and 3. Then, for any client p in $\mathcal{I}_{\mathcal{D}}$, p and $\mathcal{S}^*(p)$ are cut at level at most $\log(4\text{dist}(p, \text{OPT}) + 3\text{dist}(p, \mathcal{L})/\varepsilon) + \tau(\varepsilon, d, z)$ in $\mathcal{D}$, whenever $\varepsilon \leq 1/5$, where $\mathcal{S}^*(p)$ is the closest facility to p in $\mathcal{S}^*$. ◀

Proof. The proof of this lemma is very similar to the one of Lemma 3.14. However, since some facilities of OPT were removed in Step 2 to obtain $\mathcal{S}^*$, we need to adapt the proof carefully. In particular, we will use the following inequality. Let p be a client. If $\text{OPT}(p)$ was moved in Step 2, it was replaced by facility ℓ for which $\text{OPT}(p) = f_\ell$ and $\text{dist}(\text{OPT}(p), \ell) \leq \text{dist}(\text{OPT}(p), \mathcal{L}(p))$, since f_ℓ is the closest facility to ℓ. Using the triangle inequality we obtain $\text{dist}(\text{OPT}(p), \mathcal{L}(p)) \leq \text{dist}(p, \text{OPT}(p)) + \text{dist}(p, \mathcal{L}(p))$. On the other hand, as $\ell \in \mathcal{S}^*$ we get $\text{dist}(p, \mathcal{S}^*(p)) \leq \text{dist}(p, \ell) \leq \text{dist}(p, \text{OPT}(p)) + \text{dist}(\text{OPT}(p), \ell)$, again applying the triangle inequality. Putting these inequalities together we obtain

$$\text{dist}(p, \mathcal{S}^*(p)) \leq 2\text{dist}(p, \text{OPT}(p)) + \text{dist}(p, \mathcal{L}(p)). \tag{3.2}$$

Furthermore, if $\text{OPT}(p)$ is not moved in Step 2 we have $\text{OPT}(p) \in \mathcal{S}^*$, and so Inequality (3.2) holds trivially as $\text{dist}(p, \mathcal{S}^*(p)) \leq \text{dist}(p, \text{OPT}(p))$.

To find the level at which p and $\mathcal{S}^*(p)$ are cut, we distinguish between two cases: either p in $\mathcal{I}$ is badly cut w.r.t. $\mathcal{D}$, or not. If p is badly cut, then it is now located at $\mathcal{L}(p)$ in the instance $\mathcal{I}_{\mathcal{D}}$. In that case, either:

1. $\mathcal{L}(p)$ is also badly cut, i.e., $\mathcal{L}(p) \in B_{\mathcal{D}} \subseteq \mathcal{S}^*$, and so $\mathcal{S}^*(p) = \mathcal{L}(p)$. It follows that p and $\mathcal{S}^*(p)$ are collocated, thus they are never cut.

2. $\mathcal{L}(p)$ is not badly cut. Then, since p is now located at $\mathcal{L}(p)$, $\text{dist}(p, \mathcal{S}^*(p)) = \text{dist}(\mathcal{L}(p), \mathcal{S}^*(\mathcal{L}(p)))$. $\text{OPT}(\mathcal{L}(p))$ is not necessarily in $\mathcal{S}^*$: in that case, it was replaced by a facility f that is closer to $\text{OPT}(\mathcal{L}(p))$ than $\mathcal{L}(p)$, and so $\text{dist}(\mathcal{L}(p), f) \leq 2\text{dist}(\mathcal{L}(p), \text{OPT}(\mathcal{L}(p)))$. Hence, either if $\text{OPT}(\mathcal{L}(p))$ is in $\mathcal{S}^*$ or not, it holds that $\text{dist}(\mathcal{L}(p), \mathcal{S}^*) \leq 2\text{dist}(\mathcal{L}(p), \text{OPT})$.

Since $\mathcal{L}(p)$ is not badly cut, the ball $\beta(\mathcal{L}(p), 3\text{dist}(\mathcal{L}(p), \text{OPT}))$ is cut at level at most $\log(3\text{dist}(\mathcal{L}(p), \text{OPT}))+\tau(\varepsilon, d, z)$. By triangle inequality, $\text{dist}(\mathcal{L}(p), \text{OPT}) = \text{dist}(\mathcal{L}(p), \text{OPT}(\mathcal{L}(p))) \leq \text{dist}(p, \mathcal{L})+\text{dist}(p, \text{OPT})$, and thus p and $\mathcal{S}^*(p)$ are also separated at level at most $\log\big(3\text{dist}(p, \mathcal{L}) + 3\text{dist}(p, \text{OPT})\big) + \tau(\varepsilon, d, z)$.

In the other case where p is not badly cut, the ball $\beta(p, 3\text{dist}(p, \mathcal{L})/\varepsilon)$ is cut at level at most $\log(3\text{dist}(p, \mathcal{L})/\varepsilon) + \tau(\varepsilon, d, z)$. We make a case distinction according to $\text{dist}(p, \mathcal{L})$ and $\text{dist}(p, \text{OPT})$.

1. If $\text{dist}(p, \mathcal{L}) \leq \varepsilon\text{dist}(p, \text{OPT})$, then we have the following. If $\mathcal{L}(p)$ is badly cut, $\mathcal{L}(p) \in B_{\mathcal{D}} \subseteq S^*$ and therefore $\mathcal{S}^*_p \leq \text{dist}(p, \mathcal{L})$. Moreover, since p is not badly cut the ball $\beta(p, \text{dist}(p, \mathcal{L}))$ is cut at level at most $\log(3\text{dist}(p, \mathcal{L})/\varepsilon) + \tau(\varepsilon, d, z)$. Therefore p and $\mathcal{S}^*(p)$ are cut at a level below $\log(4\text{dist}(p, \text{OPT})+3\text{dist}(p, \mathcal{L})/\varepsilon)+ \tau(\varepsilon, d, z)$.

   In the case where $\mathcal{L}(p)$ is not badly cut, both p and $\mathcal{S}^*(p)$ lie in the ball centered at $\mathcal{L}(p)$ and of diameter $3\text{dist}(\mathcal{L}(p), \text{OPT})$, which can be seen as follows. We use $\text{dist}(p, \mathcal{L}) \leq \varepsilon\text{dist}(p, \text{OPT})$ to derive

$$\begin{aligned}\text{dist}(p, \mathcal{L}(p)) &\leq \varepsilon\text{dist}(p, \text{OPT}(p)) \leq \varepsilon\text{dist}(p, \text{OPT}(\mathcal{L}(p)))\\ &\leq \varepsilon\text{dist}(p, \mathcal{L}(p)) + \varepsilon\text{dist}\big(\mathcal{L}(p), \text{OPT}(\mathcal{L}(p))\big)\end{aligned}$$

   And therefore, since $\varepsilon \leq 1/4$, $\text{dist}(p, \mathcal{L}(p)) \leq \frac{\varepsilon}{1-\varepsilon}\text{dist}(\mathcal{L}(p), \text{OPT}) \leq \text{dist}(\mathcal{L}(p), \text{OPT})/3$. Using these inequalities we also get

$$\begin{aligned}\text{dist}(\mathcal{S}^*(p), \mathcal{L}(p)) &\leq \text{dist}(\mathcal{S}^*(p), p) + \text{dist}(p, \mathcal{L}(p))\\ &\leq 2\text{dist}(p, \text{OPT}(p)) + 2\text{dist}(p, \mathcal{L}(p)) \quad \text{(using Inequality (3.2))}\\ &\leq 2\text{dist}(p, \text{OPT}(\mathcal{L}(p))) + 2\text{dist}(p, \mathcal{L}(p))\\ &\leq 4\text{dist}(p, \mathcal{L}(p)) + 2\text{dist}(\mathcal{L}(p), \text{OPT}(\mathcal{L}(p)))\\ &\leq \left(2 + \frac{4\varepsilon}{1-\varepsilon}\right)\text{dist}(\mathcal{L}(p), \text{OPT}),\end{aligned}$$

   which is smaller than $3\text{dist}(\mathcal{L}(p), \text{OPT})$ for any $\varepsilon \leq 1/5$. Hence we have $p, \mathcal{S}^*(p) \in \beta(\mathcal{L}(p), 3\text{dist}(\mathcal{L}(p), \text{OPT}))$. Since $\mathcal{L}(p)$ is not badly cut, p and $\mathcal{S}^*(p)$ are cut at level at most $\log(3\text{dist}(\mathcal{L}(p), \text{OPT})) + \tau(\varepsilon, d, z)$. Since $\text{dist}(\mathcal{L}(p), \text{OPT}(\mathcal{L}(p))) \leq \text{dist}(\mathcal{L}(p), \text{OPT}(p)) \leq \text{dist}(\mathcal{L}(p), p) + \text{dist}(p, \text{OPT}(p)) \leq (1 + \varepsilon)\text{dist}(p, \text{OPT})$, we have that $\log(3\text{dist}(\mathcal{L}(p), \text{OPT})) + \tau(\varepsilon, d, z) \leq \log(4\text{dist}(\mathcal{L}(p), \text{OPT})) + \tau(\varepsilon, d, z)$.

2. If $\text{dist}(p, \text{OPT}) \leq \text{dist}(p, \mathcal{L})/\varepsilon$, then $2\text{dist}(p, \text{OPT}) + \text{dist}(p, \mathcal{L}) \leq 3\text{dist}(p, \mathcal{L})/\varepsilon$ and since p is not badly cut, the ball $\beta(p, 2\text{dist}(p, \text{OPT}) + \text{dist}(p, \mathcal{L}))$ is cut at level at most $\log(3\text{dist}(p, \mathcal{L})/\varepsilon) + \tau(\varepsilon, d, z)$. Moreover, $\mathcal{S}^*(p)$ lies in this ball by Inequality (3.2).

In all cases, p and $\mathcal{S}^*(p)$ are separated at level at most $\log(3\text{dist}(p, \mathcal{L})/\varepsilon + 4\text{OPT}(p)) + \tau(\varepsilon, d, z)$, which concludes the lemma. □

Equipped with these two lemmas, we can prove the following lemma, which concludes the section. Note again, that the bounds are for OPT defined after removing some centers in Step 1.

▶ **Lemma 3.24.** Condition on $\mathcal{I}_{\mathcal{D}}$ having small distortion, with $\rho = \frac{\varepsilon^2}{128z}2^{-\tau(\varepsilon,d,z)}$. There exists a portal-respecting solution $\mathcal{S}$ in $\mathcal{I}_{\mathcal{D}}$ such that $\text{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}) + b(\mathcal{S}) \leq (1 + O(\varepsilon))\text{cost}_{\mathcal{I}}(\text{OPT}) + O(\varepsilon\text{cost}_{\mathcal{I}}(\mathcal{L}))$. ◀

Proof. The proof follows exactly the one of Lemma 3.15, making $\mathcal{S}^*$ portal-respecting, and using Lemma 3.20 and Lemma 3.23 to prove that the resulting solution $\mathcal{S}$ in $\mathcal{I}_{\mathcal{D}}$ has $\text{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}) + b(\mathcal{S}) \leq (1 + O(\varepsilon))\text{cost}_{\mathcal{I}}(\text{OPT}) + O(\varepsilon\text{cost}_{\mathcal{I}}(\mathcal{L}))$. □

3.4.4 The Algorithm

The algorithm follows the lines of the one for Facility Location, in Section 3.3.3. It first computes a constant-factor approximation L, then the hierarchical decomposition $\mathcal{D}$ (with parameter $\rho = \frac{\varepsilon^2}{128p}2^{-\tau(\varepsilon,d,z)}$, as in Section 3.3) and constructs instance $\mathcal{I}_{\mathcal{D}}$. A dynamic program is then used to solve efficiently the problem, providing a solution S of cost at most $(1+\varepsilon)\text{cost}_{\mathcal{I}}(\text{OPT})$ – conditioned on the event that the instance $\mathcal{I}_{\mathcal{D}}$ has small distortion.

Dynamic programming. The algorithm proceeds bottom up along the levels of the decomposition. We give an overview of the dynamic program which is a slightly refined version of the one presented for Facility Location in Section 3.3.3. We make use of two additional ideas.

To avoid the dependency on k we proceed as follows. In the standard approach, a cell of the dynamic program is defined by a part of the decomposition $\mathcal{D}$, the portal parameters (as defined in Section 3.3.3), and a value $k_0 \in [k]$. The value of an entry in the table is then the cost of the best solution that uses k_0 centers, given the portal parameters.

For our dynamic program for the k-Median and k-Means problems, we define a cell of the dynamic program by a part B, the portal parameters $\langle \ell_1, \ldots, \ell_{n_p} \rangle$ and $\langle s_1, \ldots, s_{n_p} \rangle$ and a value c_0 in $[\text{cost}(\mathcal{L})/n; (1+\varepsilon)\text{cost}(\mathcal{L})]$. The entry of the cell is equal to the minimum number k_0 of centers that need to be placed in part B in order to achieve cost at most c_0, given the portal parameters. Moreover, we only consider values for c_0 that are powers of $(1 + \varepsilon/\log n)$. The output of the algorithm is the minimum value c_0 such that the root cell has value at most k (i.e., the minimum value such that at most k centers are needed to achieve it).

The DP table can be computed the following way. For the parts that have no descendant, namely the base cases, computing the best clustering given a set of parameters can be done easily: there is at most one client in the part, and verifying that the parameter values for the centers inside the part are consistent can be done easily. At

a higher level of the decomposition, a solution is obtained by going over all the sets of parameter values for all the children parts. It is immediate to see whether sets of parameter values for the children can lead to a consistent solution (similar to [112, 9]). Since there are at most $2^{O(d)}$ children parts, this gives a running time of $q^{2^{O(d)}}$, where q is the total number of parameter values.

This strategy would lead to a running time of $f(\varepsilon, d)n \log^{2^{O(d)}} n$. We can however treat the children in order, instead of naively testing all parameter values for them. We use a classical transformation of the dynamic program, in which the first table is filled using an auxiliary dynamic program. A cell of this auxiliary DP is a value c_0 in $[\text{cost}(\mathcal{L})/n; (1+\varepsilon)\text{cost}(\mathcal{L})]$, a part C, one of its children C_i, and the portal parameters for the portals of C and all its children before C_i in the given order. The entry of the cell is equal to the minimum number of centers k_0 that need to be placed in the children parts following C_i to achieve a cost of c_0 given the portal parameters. To fill this table, one can try all possible sets of parameters for the following children, see whether they can lead to a consistent solution, and compute the minimum value among them.

Analysis – proof of Theorem 3.1 and Theorem 3.2. We first show that the solution computed by the algorithm gives a $(1+O(\varepsilon))$-approximation, and then prove the claim on the complexity.

▶ **Lemma 3.25.** Let $\mathcal{S}^*$ be the solution computed by the algorithm. With probability $1 - 2\varepsilon$, it holds that $\text{cost}_{\mathcal{I}}(\mathcal{S}^*) = (1 + O(\varepsilon))\text{cost}_{\mathcal{I}}(\text{OPT}) + O(\varepsilon\text{cost}_{\mathcal{I}}(\mathcal{L}))$ ◀

Proof. With probability $1 - 2\varepsilon$, $\mathcal{I}_{\mathcal{D}}$ has small distortion (Lemma 3.20). Following Lemma 3.24, let $\mathcal{S}$ be a portal-respecting solution such that $\text{cost}_{\mathcal{I}}(\mathcal{S}) + b(\mathcal{S}) \leq (1 + O(\varepsilon))\text{cost}_{\mathcal{I}}(\text{OPT}) + O(\varepsilon\text{cost}_{\mathcal{I}}(\mathcal{L}))$.

As in Lemma 3.16, $\mathcal{S}$ can be adapted to a configuration of the DP with a small extra cost. The cost incurred to the rounding of distances can be charged either to $b(\mathcal{S})$ or is a $O(\varepsilon)\text{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S})$, as in Lemma 3.16. The cost to round the value c_0 is a $(1 + \varepsilon/\log n)$ factor at every level of the decomposition. Since there are $O(\log n)$ of them, the total factor is $(1 + \varepsilon/\log n)^{O(\log n)} = 1 + O(\varepsilon)$. Hence, we have the following:

$$\begin{aligned}
\text{cost}_{\mathcal{I}}(\mathcal{S}^*) &= (1 + O(\varepsilon))\text{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}^*) && \text{(as } \mathcal{I}_{\mathcal{D}} \text{ has small distortion)} \\
&= (1 + O(\varepsilon))(\text{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}) + b(\mathcal{S})) && \text{(by previous paragraph)} \\
&\leq (1 + O(\varepsilon))\text{cost}_{\mathcal{I}}(\text{OPT}) + O(\varepsilon\text{cost}_{\mathcal{I}}(\mathcal{L})) && \text{(by definition of } \mathcal{S})
\end{aligned}$$

□

▶ **Lemma 3.26.** The running time of the DP is $2^{(1/\varepsilon)^{O(d^2)}} \cdot n \log^4 n$. ◀

Proof. The number of cells in the auxiliary DP is given by the number of parts ($O(n)$) , the number of children of a part ($2^{O(d)}$), the number of portal parameters ($(1/\varepsilon)^{2^{O(d^2)}/\varepsilon}$) and the possible values for c_0 ($O(\log^2 n)$): it is therefore $n \cdot 2^{O(d)} \cdot (1/\varepsilon)^{2^{O(d^2)}/\varepsilon} \cdot \log^2 n$.

The complexity to fill the table adds a factor $(1/\varepsilon)^{2^{O(d^2)}/\varepsilon} \cdot \log^2 n$, to try all possible combination of portal parameters and value of c_0 . Hence, the overall running time of the DP is $n \cdot (1/\varepsilon)^{2^{O(d^2)}/\varepsilon} \cdot \log^4 n = 2^{(1/\varepsilon)^{O(d^2)}} \cdot n \log^4 n$. □

The proof of Theorem 3.1 and Theorem 3.2 are completed by the following lemma, which bounds the running time of the preprocessing steps.

▶ **Lemma 3.27.** For k-Median and (k, z)-clustering, the total running time of the algorithms are respectively $2^{O(d)} n \log^9 n + 2^{(1/\varepsilon)^{O(d^2)}} n \log^4 n$ and $2^{O(d)} n \log^9 n + 2^{(1/\varepsilon)^{O(d^2)}} n \log^5 n$ ◀

Proof. We need to bound the running time of three steps: computing an approximation, computing the hierarchical decomposition, and running the dynamic program.

For k-Median, a constant-factor approximation can be computed in $O(m \log^9 n) = 2^{O(d)} n \log^9 n$ time with Thorup's algorithm [156]. The split-tree decomposition can be found in $2^{O(d)} n \log n$ time as explained in Section 3.2. Moreover, as explained in Lemma 3.26, the dynamic program runs in time $2^{(1/\varepsilon)^{O(d^2)}} n \log^4 n$, ending the proof of the Theorem 3.1.

Another step is required for higher powers. It is indeed not known how to find a constant-factor approximation in near-linear time. However, one can notice that a c-approximation for k-Median is an nc^z-approximation for (k, z)-Clustering, using Holder's inequality. Moreover, notice that starting from a solution $\mathcal{S}$, our algorithm finds with probability $1 - \varepsilon$ a solution with cost $(1 + O(\varepsilon))\text{cost}(\text{OPT}) + O(\varepsilon)\text{cost}(\mathcal{S})$ in time $2^{(1/\varepsilon)^{O(d^2)}} n \log^4 n$, as for k-Median. By repeating $\log \log n$ times, the success probability is boosted to $1 - \varepsilon/\log n$.

Repeating this algorithm N times, using in step $i + 1$ the solution given at step i, gives thus a solution of cost $(1 + O(\varepsilon))\text{cost}(\text{OPT}) + O(\varepsilon^N)\text{cost}(\mathcal{S})$, with probability $1 - N\varepsilon/\log n$. Starting with $\text{cost}(\mathcal{S}) = O(n)\text{cost}(\text{OPT})$ and taking $N = O(\log n)$ ensures to find a solution for (k, z)-Clustering with cost $(1 + O(\varepsilon))\text{cost}(\text{OPT})$ with probability $1 - \varepsilon$. The complexity for (k, z)-Clustering is therefore the same as for k-Median, with an additional $\log n \cdot \log \log n$ factor for the dynamic program term. This concludes the proof of Theorem 3.2. □

3.5 Other Applications of the Framework

Our techniques can be generalized to variants of the clustering problems where *outliers* are taken into account. We consider here two of them: k-Median with Outliers and its Lagrangian relaxation, Prize-Collecting k-Median. It can also be used to find a bicreteria approximation to k-Center.

3.5.1 Prize-Collecting k-Median

In the "prize-collecting" version of the problems, it is possible not to serve a client p by paying a penalty π_p (these problems are also called clustering "with penalties"). For a solution $\mathcal{S}$, we call an *outlier for* $\mathcal{S}$ a client that is not served by $\mathcal{S}$. Formally, an instance is a quintuple $(P, F, \text{dist}, \pi, k)$ where $(P \cup F, \text{dist})$ is a metric, k is an integer and $\pi : P \to \mathbb{R}^+$ the penalty function, and the goal is to find $\mathcal{S} = (\mathcal{S}_F, \mathcal{S}_O)$ with $\mathcal{S}_F \subseteq F$ and $\mathcal{S}_O \subseteq P$ such that $|\mathcal{S}_F| = k$ and $\text{cost}(\mathcal{S}_0, \mathcal{S}_F) = \sum_{p \in P \setminus S_O} \text{dist}(p, \mathcal{S}_F) + \sum_{p \in \mathcal{S}_O} \pi_p$ is minimized. $\text{cost}(\mathcal{S}_F)$ denotes the cost of solution $\mathcal{S}_F$ with the best choice of outliers (which is easy to determine – $\mathcal{S}_O$ consists of all clients that have $\pi_p \leq \text{cost}(p, \mathcal{S}_F)$.)

Looking at the Prize-Collecting k-Median problem, we aim at applying the framework from Section 3.4. Let $\mathcal{L} = (\text{dist}(p, \mathcal{L}), \mathcal{L}_O)$ be an approximate solution. We define *badly cut* for outliers as we did for centers: an outlier p of $\mathcal{L}_O$ is *badly cut w.r.t.* $\mathcal{D}$ if the ball $\beta(p, 3\text{dist}(p, \text{OPT}))$ is cut at some level j greater than $i + \tau(\varepsilon, d, z)$, where $\text{dist}(p, \text{OPT})$ is the distance from p to the closest facility of the optimum solution OPT. Hence, Lemma 3.12 extends directly, and the probability that an outlier in $\mathcal{L}_O$ is badly cut is $\kappa(\varepsilon, z)$. Note that in the case where p is an outlier both in $\mathcal{L}$ and in OPT, there is no particular need of preserving any distance from p to another point: in any solution we consider, p will pay the penalty π_p rather than connecting to a center.

We now turn to the previous framework, showing how to construct a near-optimal solution containing all badly-cut centers of $\mathcal{L}$. For that we transfer the definitions of the mappings $\text{dist}(p, \mathcal{L}), \psi$ ($\text{dist}(p, \mathcal{L})$ maps a client to its closest center of $\mathcal{L}$, and $\psi(\ell) = \{f \in \text{OPT} \mid \mathcal{L}(f) = \ell\}$) and of the sets $\mathcal{L}^0, \mathcal{L}^1, \mathcal{L}^{\geq 2}, \text{OPT}^1$, and $\text{OPT}^{\geq 2}$. We will show that this framework, with only a few modifications, leads to an approximation scheme for Prize-Collecting k-Median. Let $\mathcal{T} = \text{OPT}$. As in Section 3.4, we start by removing a few centers from the optimal solution, without increasing the cost too much:

- **Step 1.** Among the facilities of $\text{OPT}^{\geq 2}$ that are not the closest of their corresponding facility in $\mathcal{L}^{\geq 2}$, remove from $\mathcal{T}$ the subset $\widehat{\mathcal{H}}$ of size $\lfloor \varepsilon \cdot |\text{OPT}^{\geq 2}|/2 \rfloor$ that yields the smallest cost increase, i.e. the smallest value of $\sum_{p \in P \setminus \mathcal{L}_O : \text{OPT}(p) \in \widehat{\mathcal{H}}} \text{dist}(p, \mathcal{T} \setminus \widehat{\mathcal{H}}) + \sum_{p \in L_O : \text{OPT}(p) \in \widehat{\mathcal{H}}} \pi_p$.

The function minimized by $\widehat{\mathcal{H}}$ corresponds to redirecting all clients served in the

local solution to a center of $\mathcal{T} \setminus \widehat{\mathcal{H}}$ and paying the penalty for clients $p \in \mathcal{L}_O$ such that $\mathrm{OPT}(p) \in \widehat{\mathcal{H}}$. Those clients are thus considered as outliers in the constructed solution.

▶ **Lemma 3.28.** After step 1, $\mathcal{T}$ has cost $(1+O(\varepsilon))\mathrm{cost}(\mathrm{OPT}) + O(\varepsilon)\mathrm{cost}(\mathcal{L})$. ◀

Proof sketch. The proof is essentially the same as Lemma 3.19, with an averaging argument: the only difference comes from the cost of removing a center from $\mathcal{T}$. For any client p, the cost of removing $\mathrm{OPT}(p)$ from $\mathcal{T}$ is $O(\mathrm{cost}(p,\mathrm{OPT}) + \mathrm{cost}(p,L))$: if $p \notin \mathcal{L}^0$, the argument is the same as in Lemma 3.19, and if $p \in \mathcal{L}_O$ the cost is π_p, which is what p pays in $\mathcal{L}$. Hence the proof follows. □

Again, we denote now by OPT this solution $\mathcal{T}$ and define the instance $\mathcal{I}_\mathcal{D}$ according to this solution. Recall that $B_\mathcal{D}$ is the set of badly cut centers of $\mathrm{dist}(p,\mathcal{L})$, and denote $O_\mathcal{D}$ the set of badly cut outliers of $\mathcal{L}$. We say that an instance $\mathcal{I}_\mathcal{D}$ has *small distortion* if $\nu_{\mathcal{I}_\mathcal{D}} \leq \varepsilon\mathrm{cost}(\mathcal{L})$ and there exists a solution $\mathcal{S}$ that contains $B_\mathcal{D}$ as centers and $O_\mathcal{D}$ as outliers with $\mathrm{cost}_\mathcal{I}(\mathcal{S}) \leq (1+\varepsilon)\mathrm{cost}_\mathcal{I}(\mathrm{OPT}) + \varepsilon\mathrm{cost}_\mathcal{I}(\mathcal{L})$.

To deal with the badly cut centers, there is only one hurdle to be able to apply the proof of Lemma 3.24. Indeed, when a center of OPT that serves a client p is deleted during the construction, the cost of reassigning p is bounded in Lemma 3.24 by $\mathrm{dist}(p,\mathcal{S})^z$. However this is not possible to do when p is an outlier for $\mathcal{S}$: there is no control on $\mathrm{dist}(p,\mathcal{S})$, and hence one has to pay the penalty π_p. It is thus necessary to find a mechanism that ensures to pay this penalty only with a probability ε for each client p. Similar to Section 3.4, this is achieved with the following three steps:

- **Step 2.** For each badly-cut facility $f \in \mathcal{L}$ for which $\psi(f) \neq \emptyset$, let $f' \in \psi(f)$ be the closest to f. Replace f' by f in $\mathcal{T}^*$. For all clients $p \in \mathcal{L}_O$ such that $\mathrm{OPT}(p) = f'$, add p as outliers.
- **Step 3.** Add all badly cut facility f' of $\mathcal{L}^0$ to $\mathcal{T}^*$
- **Step 4.** Add all badly cut outliers of $\mathcal{L}$ to the outliers of $\mathcal{T}^*$.

We show next that $\mathcal{T}^*$ satisfies the conditions for $\mathcal{I}_\mathcal{D}$ to have small distortion with good probability.

▶ **Lemma 3.29.** The probability that $\mathcal{I}_\mathcal{D}$ has small distortion is at least $1 - \varepsilon$. ◀

Proof. When bounding the cost increase due to Step 2, it is necessary to add as outliers all clients served by f' that are outliers in $\mathcal{L}$. Since f' is deleted from $\mathcal{T}^*$ with probability $\kappa(\varepsilon,z)$, the expected cost due to this is $\sum_{p\in\mathcal{L}_O} \kappa(\varepsilon,z)\cdot\pi_p \leq \kappa(\varepsilon,z)\mathrm{cost}_\mathcal{I}(\mathcal{L})$. Using Markov's inequality, this is at most $(\varepsilon/3)\mathrm{cost}_\mathcal{I}(\mathcal{L})$ with probability $1-\varepsilon/3$.

Step 3 does not involve outliers at all. Hence, Claim 3.21 and 3.22 are still valid. Combined with the previous observation about Step 2, this proves that after Step 3, $\mathcal{T}^*$ contains at most k centers — including the ones in $B_{\mathcal{D}}$ — and has cost at most $(1+\varepsilon)\text{cost}_{\mathcal{I}}(\text{OPT}) + (\varepsilon/3)\text{cost}_{\mathcal{I}}(\mathcal{L})$ with probability at least $1-\varepsilon/3$.

Step 4 implies that all outliers in $O_{\mathcal{D}}$ are also outliers in the constructed solution. Since an outlier of $\mathcal{L}$ is badly cut with probability $\kappa(\varepsilon, z)$, the expected cost increase due to this step is at most $\kappa(\varepsilon, z)\text{cost}_{\mathcal{I}}(\mathcal{L})$. Using again Markov's inequality, this cost is at most $(\varepsilon/3)\text{cost}_{\mathcal{I}}(\mathcal{L})$ with probability $1-\varepsilon/3$.

By union-bound, the solution $\mathcal{T}^*$ has cost at most $(1+\varepsilon)\text{cost}_{\mathcal{I}}(\text{OPT}) + \varepsilon\text{cost}_{\mathcal{I}}(\mathcal{L})$ with probability $1-\varepsilon$. Hence, $\mathcal{I}_{\mathcal{D}}$ has small distortion with probability $1-\varepsilon$. □

Given an instance with low distortion, it is again possible to prove that there exists a near optimal portal-respecting solution, and the same DP as for k-Median can find it.

Therefore, using the polynomial time algorithm of Charikar et al. [41] to compute a constant-factor approximation, the algorithm presented in Section 3.4 can be straightforwardly adapted, concluding the proof of Theorem 3.4.

3.5.2 k-Median with Outliers

In the k-Median with Outliers problem, the number of outliers allowed is bounded by some given integer Z. We do not manage to respect this bound together with having at most k facilities and a near-optimal solution: we need to relax it a little bit, and achieve a bicriteria approximation, with k facilities and $(1+O(\varepsilon))Z$ outliers. For this, our framework applies nearly without a change.

The first step in the previous construction does not apply directly: the "cost" of removing a center is not well defined. In order to fix this part, Step 1 is randomized: among the facilities of $\text{OPT}^{\geq 2}$ that are not the closest of their corresponding facility in $\mathcal{L}^{\geq 2}$, remove from $\mathcal{T}^*$ a random subset $\widehat{\mathcal{H}}$ of size $\lfloor \varepsilon \cdot |\text{OPT}^{\geq 2}|/2 \rfloor$.

▶ **Lemma 3.30.** After the randomized Step 1, $\mathcal{T}^*$ has expected cost $(1 + O(\varepsilon))\text{cost}(\text{OPT}) + O(\varepsilon)\text{cost}(\mathcal{L})$ ◀

Proof. Since there are at least $|\text{OPT}^{\geq 2}|/2$ facilities of $\text{OPT}^{\geq 2}$ that are not the closest of their corresponding facility in $\mathcal{L}^{\geq 2}$, the probability to remove one of them is $O(\varepsilon)$. Hence, every outlier of $\mathcal{L}$ that is served in OPT must be added as an outlier in $\mathcal{T}^*$ with probability $O(\varepsilon)$ – when its serving center in OPT is deleted. Hence, the expected number of outliers added is $O(\varepsilon Z)$.

Moreover, the proof of Lemma 3.19 shows that the sum of the cost of deleting all possible facilities is at most $O(\text{cost}(\text{OPT}) + \text{cost}(\mathcal{L}))$ (adding a point as outlier whenever it is necessary). Removing each one of them with probability $O(\varepsilon)$ ensures that the expected cost of $\mathcal{T}^*$ after step 1 is $(1+O(\varepsilon))\text{cost}(\text{OPT}) + O(\varepsilon)\text{cost}(\mathcal{L})$. □

The three following steps are the same as in the previous section, and the proof follows: with constant probability, the instance $\mathcal{I}_{\mathcal{D}}$ has small distortion (defined as for k-Median with penalties), and one can use a dynamic program to solve the problem on it. The DP is very similar to the one for k-Median. The only difference is the addition of a number x to each table entry, which is a power of $(1+\varepsilon/\log(n/\varepsilon))$, and represents the (rounded) number of outliers allowed in the subproblem. This adds a factor $\log^2(n/\varepsilon)/\varepsilon$ to the complexity.

It is possible to compute a constant factor approximation S in polynomial time (using Krishnaswamy et al. [113]). Hence, this algorithm is a polynomial time bicriteria approximation scheme for k-Median with outliers. As in Section 3.4, this directly extends to k-Means with outliers.

This concludes the proof of Theorem 3.5.

3.5.3 k-Center

In the k-Center problem, the goal is to place k centers such as to minimize the largest distance from a point to its serving center. We propose a bicriteria approximation, allowing the algorithm to open $(1+O(\varepsilon))k$ centers.

For this, we change slightly the definition of badly-cut. Given a solution $\mathcal{L}$ with cost γ and a hierarchical decomposition $\mathcal{D}$, a center f of $\mathcal{L}$ is *badly cut w.r.t* $\mathcal{D}$ if the ball $\beta(f,2^i)$ is cut at some level j greater than $i+\tau(\varepsilon,d,z)$, for i such that $2^{i-1}\leq 2\gamma<2^i$.

Note that Lemma 3.12 still holds with this definition : a center f is badly cut with probability at most $\kappa(\varepsilon,z)$. Let $B_{\mathcal{D}}$ be the set of badly cut centers. We assume in the following that $\mathcal{L}$ is a 2-approximation, i.e. $\gamma\leq 2\mathrm{OPT}$.

We make the crucial following observation, using the doubling property of the metric. Let f be a center of $\mathcal{L}$. By definition of doubling dimension, the ball $\beta(f,\gamma)$ can be covered by 2^d balls of radius $\gamma/2\leq \mathrm{OPT}$. Let $\mathcal{C}_f$ be the set of centers of such balls, such that $\beta(f,\gamma)\subseteq \bigcup_{f'\in\mathcal{C}_f}\beta(f',\gamma/2)$.

Given an instance $\mathcal{I}$, we construct $\mathcal{I}_{\mathcal{D}}$ the following way: for each badly cut facility f, force all the facilities in $\mathcal{C}_f$ to be opened in any solution on $\mathcal{I}_{\mathcal{D}}$, and remove all the clients in $\beta(f,\gamma)$ from the instance. We let $\mathcal{C}=\bigcup_{f \text{ badly cut}}\mathcal{C}_f$. The structural lemma of this section is the following:

▶ **Lemma 3.31.** It holds that for all solution $\mathcal{S}$ of $\mathcal{I}_{\mathcal{D}}$:

- $\mathrm{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S})\leq \mathrm{cost}_{\mathcal{I}}(\mathcal{S})$
- $\mathrm{cost}_{\mathcal{I}}(\mathcal{S}\cup\mathcal{C})\leq \max(\mathrm{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}),\mathrm{OPT})$ ◀

Proof. Since the instance $\mathcal{I}_{\mathcal{D}}$ contains a subset of clients of $\mathcal{I}$, it holds that $\mathrm{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S})\leq$

$\text{cost}_{\mathcal{I}}(\mathcal{S})$.

Let $\mathcal{S}$ be a solution in $\mathcal{I}_{\mathcal{D}}$. It serves all client in $\mathcal{I}$ but the one removed: these ones are served by $\mathcal{C}$ at a cost $\gamma/2 \leq \text{OPT}$. Hence, the cost of $\mathcal{S} \cup \mathcal{C}$ is at most $\max(\text{cost}_{\mathcal{I}_{\mathcal{D}}}(S), \text{OPT})$. □

We now show, in a similar fashion as Lemma 3.14, that the clients in $\mathcal{I}_{\mathcal{D}}$ are cut from their serving facility of OPT at a controlled level. Recall that OPT is defined for instance $\mathcal{I}$.

▶ **Lemma 3.32.** Let p be a client in $\mathcal{I}_{\mathcal{D}}$ and $\text{OPT}(p)$ its serving facility in OPT. p and $\text{OPT}(p)$ are cut at level at most $\log(2\gamma) + \tau(\varepsilon, d, z)$. ◀

Proof. Let p be a client, $\mathcal{L}(p)$ its serving center in $\mathcal{L}$ and $\text{OPT}(p)$ its serving center in OPT. If p is still a client in $\mathcal{I}_{\mathcal{D}}$, it means that $\mathcal{L}(p)$ is not badly cut. Observe that $\text{dist}(\mathcal{L}(p), \text{OPT}(p)) \leq \text{dist}(p, \mathcal{L}(p)) + \text{dist}(p, \text{OPT}(p)) \leq \gamma + \text{OPT} \leq 2\gamma$

Let i such that $2^{i-1} \leq 2\gamma \leq 2^i$. Since $\mathcal{L}(p)$ is not badly cut, the ball $\beta(\mathcal{L}(p), 2^i)$ is not badly cut neither: hence, p and $\text{OPT}(p)$ (that are in this ball) are cut at level at most $i + \tau(\varepsilon, d, z) \leq \log(2\gamma) + \tau(\varepsilon, d, z)$. □

This lemma is stronger than Lemma 3.14 and 3.23: it allows us to consider only levels of the decomposition with diameter less than $2^{1+\tau(\varepsilon,d,z)}\gamma$.

Since the set $\mathcal{C}$ has expected size $\kappa(\varepsilon, z)k$, Markov's inequality ensures that with probability $1 - \varepsilon$ this set has size $O(\varepsilon)k$. If every part with diameter D of the hierarchical decomposition is equipped with a ρD-net (for $\rho = \varepsilon 2^{-\tau(\varepsilon,d,z)}$), Lemma 3.32 ensure that there exists a portal-respecting solution S with cost $\text{cost}_{\mathcal{I}_{\mathcal{D}}}(\mathcal{S}) \leq \text{OPT} + O(\varepsilon)\gamma = (1 + O(\varepsilon))\text{OPT}$. Lemma 3.31 ensures that lifting this solution back to $\mathcal{I}$ and adding $\mathcal{C}$ as centers gives a near-optimal solution.

Using the same algorithm as for k-Median to compute a good portal-respecting solution, and computing a 2-approximation with a simple greedy algorithm (see e.g. [74]), that runs in time $O(n \log k)$ concludes the proof of Theorem 3.6.

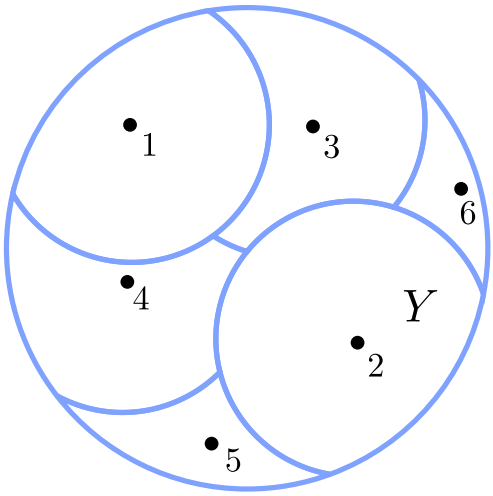
3.6 Conclusions

We demonstrated the usefulness of the notion of *badly-cut* nodes for clustering problems, in metrics with bounded doubling dimension. The techniques we developed allow for near-linear time approximation schemes for many clustering problems, both for Euclidean Spaces and the wider notion of metrics with bounded doubling dimension.

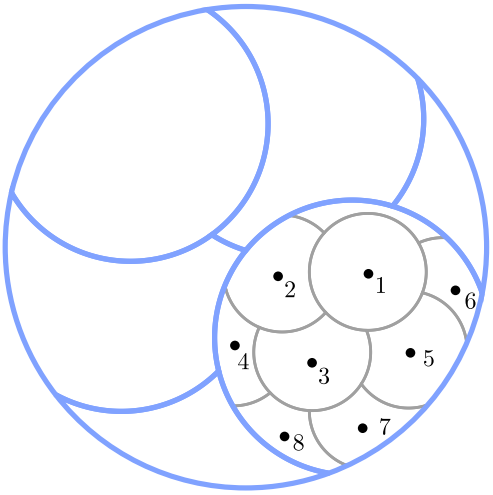
An interesting further question would be to improve our results in the special case

of Euclidean spaces of bounded dimension. By applying our techniques directly, one could get an improved running time of $2^{(1/\varepsilon)^{O(d)}} \cdot n \cdot \text{polylog}(n)$ (instead of $2^{(1/\varepsilon)^{O(d^2)}} \cdot n \cdot \text{polylog}(n)$), by noting that in the Euclidean case the probability that a ball of radius r is cut at some level i is only $O(dr/2^i)$, instead of $O(2^d r/2^i)$ in Lemma 3.10.

However, we do not know how to get a better running time in terms of n. We leave open the question whether it is possible to get a linear running time, matching the result for the Traveling Salesperson problem by Bartal and Gottlieb [20].



(a) The first step of the algorithm: the metric is decomposed into regions. The dots are $Y_{\log(\Delta)}$, and the numbers indicate the order in which they are considered by the algorithm.



(b) The algorithm recursively divides regions. We illustrated it only for region Y. The dots are $Y_{\log(\Delta)-1} \cap Y$.

3.7 Omitted Proofs

3.7.1 Proofs for Section 3.2

We present now the construction of the hierarchical decomposition from Talwar [155] – with a proof slightly adapted to our purposes.

Proof of Lemma 3.10. We present the algorithm constructing the hierarchical decomposition, and prove the lemma as a second step.

Without loss of generality, assume that the smallest distance in the metric is 1: the aspect-ratio Δ is therefore the diameter of the metric. Start from a hierarchy of nets $Y_0 := V, \ldots, Y_{\log(\Delta)}$ such that Y_i is a 2^{i-2}-net of Y_{i-1}. Moreover, let τ be a random number $\tau \in [1/2, 1)$. The hierarchical decomposition $\mathcal{D}$ is defined inductively, starting from $\mathcal{B}_{\lceil \log \Delta \rceil} = \{P\}$ as follows. To partition a part B at level i into subparts at level $i-1$, do the following: for each $y \in Y_{i-1} \cap B$, define $B \cap \beta(y, \tau 2^{i-1})$ to be a part at level $i-1$ and remove $B \cap \beta(y, \tau 2^{i-1})$ from B.

The portals can be defined as follows, once the decomposition is given: for the largest part $\{P\}$, the set of portals is a $\rho \mathrm{diam}(P)$-net of P. Inductively, given the portals $\mathcal{P}_{B'}$ of a part B' at level $i+1$, portals of a subpart B of B' are defined to be a $\rho 2^{i+1}$ net of B that contains $B \cap P_{B'}$. This is possible, as all points of $B \cap \mathcal{P}_{B'}$ are at distance at least $\rho 2^{i+2}$ of each other – because $\mathcal{P}_{B'}$ is a $\rho 2^{i+2}$ net.

When we assume access to the distances through an oracle, it is possible to construct this hierarchy and augment it with the desired set of portals in time $(1/\rho)^{O(d)} n \log(\Delta)$

[97, 58]. Their construction ensures the conciseness, preciseness and nestedness properties – essentially, those portals are nets of each part.

We prove now that this hierarchical decomposition has the required properties. The construction ensures that a part B at level $i-1$ is split in at most $|B \cap Y_{i-1}|$ parts of level i, which is $2^{O(d)}$ following Property 3.9. Proving the scaling property requires a bit more work. Let $x \in X, r > 0$ as in the statement.

The two ingredients needed for this part stem from the construction of the decomposition: the diameter of any part at level i is at most 2^{i+1}, and the minimum distance between two points of Y_i is bigger than 2^{i-2}.

These two properties are enough in order to prove our lemma. Let i be a level such that $2^i \leq r$: then $r/2^i \geq 1$ so there is nothing to prove. Otherwise, we bound the probability as follows: let $y_1, y_2, ...$ be the elements of Y_i in the order considered by the algorithm. We say that $\beta(x,r)$ is cut by y_ℓ if $\forall j < \ell, \beta(x,r) \cap \beta(y_j, \tau 2^i) = \emptyset$ and $\beta(x,r) \cap \beta(y_\ell, \tau 2^i) \notin \{\emptyset, \beta(x,r)\}$. In words, $\beta(x,r)$ is cut by y_ℓ when the part induced by y_ℓ is the first one to include a portion of $\beta(x,r)$. We have:

$$\Pr[\mathcal{D} \text{ cuts } \beta(x,r) \text{ at a level } i] \leq \sum_\ell \Pr\left[\beta(x,r) \text{ is cut by } y_\ell\right].$$

We proceed in two steps. First, let us count the number of level i parts that could possibly cut $\beta(x,r)$. We will then bound the probability each of them actually cuts $\beta(x,r)$. Any level i part is included in a ball $\beta(y, 2^{i+1})$ for some $y \in Y_i$; therefore if $\text{dist}(x,y) > r + 2^{i+1}$ then y cannot cut $\beta(x,r)$. So it is required that $\text{dist}(x,y) \leq r + 2^{i+1} \leq 4 \cdot 2^i$. But since the minimum distance between two points of Y_i is 2^{i-2}, and Y_i has doubling dimension d, we have $|Y_i \cap \beta(x, 4 \cdot 2^i)| \leq 2^{d \log(2^{i+2}/2^{i-2})} = 2^{4d}$. Thus there is only a bounded number of parts to consider.

We prove for each of them that the probability that it cuts $\beta(x,r)$ is $O(r/2^i)$. A union-bound on all the possible parts is then enough to conclude. Let therefore $y \in Y_i \cap \beta(x, 4 \cdot 2^i)$, and x_m and x_M be the respective closest and farthest point of $\beta(x,r)$ from y. A necessary condition for y's part to cut $\beta(x,r)$ is the following: the radius $\tau 2^{i+1}$ of the part has to be in the interval $[d(y,x_m), d(y,x_M))$. Indeed, if the closest point from $\beta(x,r)$ is not in $B\left(y, \tau 2^{i+1}\right)$, then no point of $\beta(x,r)$ can be. Similarly, if the farthest point is in the ball, then all points are in the part induced by y, as $\beta(x,r)$ was not cut by any part before. Since $x_m, x_M \in \beta(x,r)$ this interval has size $2r$, and the radius of the part is picked uniformly in $[2^i/2, 2^i)$. Therefore the probability that the radius of the part falls in $(d(y,x_m), d(y,x_M))$ is at most $4r/2^i$. And finally, the probability that y's part cuts $\beta(x,r)$ is indeed $4r/2^i$.

By a union-bound over all the parts that could possibly cut $\beta(x,r)$ we obtain the claimed probability $\Pr[\mathcal{C} \text{ cuts } \beta(x,r) \text{ at a level } i] = 2^{2d+2} r/2^i$. □

▶ **Lemma 3.33.** Given an instance $\mathcal{I}$ for the (k,z)-clustering problem on a metric (V, dist) with n points, $\varepsilon > 0$, and an α-approximation on $\mathcal{I}$, there exists a linear-time algorithm that outputs a set of instances $\mathcal{I}_1, \ldots, \mathcal{I}_m$ on metrics

$(V_1, \text{dist}_1), \ldots, (V_m, \text{dist}_m)$, respectively, such that

- $V_1, \ldots, V_m$ is a partition of V
- for all i, (V_i, dist_i) has aspect-ratio $O\left(n^4(\alpha/\varepsilon)^{1/z}\right)$,
- if (V, dist') is the metric where distances between points of the same part V_i are given by dist_i while distances between points of different parts is set to ∞, and OPT is the optimum solution to $\mathcal{I}$, then
  - there exists a solution on (V, dist') with cost $(1 + \varepsilon/n)\text{cost}(\text{OPT})$, and
  - any solution on (V, dist') of cost X induces a solution of cost at most $X + \varepsilon\text{cost}(\text{OPT})/n$ in $\mathcal{I}$.

◀

Proof. The cost of the constant-factor approximation is an estimate γ on the cost of the optimum solution OPT: $\gamma = \alpha\text{cost}(\text{OPT})$. It is then possible to replace all distances longer than $(2\gamma)^{1/z}$ by ∞: distances longer than $\gamma^{1/z}$ will indeed never be used by a solution with cost better than γ, so the cost of these solutions is preserved after this transformation. The distance matrix do not respect the triangle inequality anymore: thus we replace it with its metric closure. We say that two vertices are *connected* if their distance is not ∞, and call a connected component any maximal set of connected vertices. The transformation ensures that any connected component has diameter at most $2n\text{cost}(\text{OPT})^{1/z}$, and that every cluster of OPT is contained inside a single connected component. Moreover, any connected component has doubling dimension $2d$: indeed, a subspace of a metric with doubling dimension d has a doubling dimension at most $2d$. Note also that this transformation can be made implicitly: every time the algorithm queries an edge, it can replace the result by ∞ if necessary.

To identify the connected component, the algorithm builds a spanner with the algorithm of [97]: the connected components of the spanner are exactly the ones of our metric, and can be found in linear time.

Then, for the i-th connected component, the algorithm defines an instance $\mathcal{I}_i$ of the more general version of the clustering problem by the following way. It first sets $\chi(v) = 1$ for all vertex v. Then, it iterates over all edges, it contracts every edge (u, v) with length less than $1/n^3 \cdot (\varepsilon\gamma/\alpha)^{1/z}$ to form a new vertex w and sets $\chi(w) = \chi(u) + \chi(v)$.

Now, we aim at reconstructing a metric from this graph. We will do it in an approximate way: for all connected points u, v of connected component i, we set $\text{dist}_i(u, v)$ to be 0 if u and v are merged in the graph, and otherwise $\text{dist}(u, v)$. This ensures that $1/n^3 \cdot (\varepsilon\gamma/\alpha)^{1/z} \leq \text{dist}_i(u, v) \leq 2n\gamma^{1/z}$, hence the aspect-ratio of the instance $\mathcal{I}_i$ is $O\left(n^4(\alpha/\varepsilon)^{1/z}\right)$. Moreover, if u and v are not contracted, their distance is exactly preserved. Otherwise, their distance in the new metric is 0. In the original metric, their distance is at most $n \cdot 1/n^3 \cdot (\varepsilon\gamma/\alpha)^{1/z}$, as there are at most n edges on their shortest-path, and each of them has been contracted. Hence, every distance is preserved up to an additive $O(\varepsilon \cdot \text{cost}(\text{OPT})^{1/z}/n^2)$, and so each cost (distance raised to

the z) is preserved up to an additive $O(\varepsilon \text{cost}(\text{OPT})/n^{2z})$.

Since every cluster of OPT is contained inside a single connected component, this ensures that OPT induces a solution of cost $(1+\varepsilon/n^{2z-1})\text{cost}(\text{OPT})$ on $\bigcup \mathcal{I}_i$. Moreover, lifting a solution in $\bigcup \mathcal{I}_i$ to $\mathcal{I}$ costs at most $\varepsilon \text{cost}(\text{OPT})/n^{2z}$ per pair (client, center) and therefore $\varepsilon \text{cost}(\text{OPT})/n^{2z-1}$ in total. □

If the problem considered is Facility Location, it is easy to merge the solutions on subinstances: since there is no cardinality constraint, the global solution is simply the union of all the solutions. The hard constraint on k makes things a bit harder. Note that the dynamic program presented in Section 3.4 naturally handles it without any increase in its complexity: however, for completeness we present now a direct reduction.

▶ **Lemma 3.34.** Given a set of instances $(\mathcal{I}_1, \text{dist}_1)$, ..., $(\mathcal{I}_m, \text{dist}_m)$ for (k,z)-clustering problem given by Lemma 3.33 and an algorithm running in time $n_i(\log n_i)^\alpha t(\Delta)$ to solve (k,z)-clustering on instances with n_i points and aspect-ratio Δ, there exists an algorithm that runs in time $O\left(n(\log n)^{\alpha+2} t\left(O(n^4 \cdot (\alpha/\varepsilon)^{1/z})\right)\right)$ to solve the problem on $\bigcup \mathcal{I}_i$. ◀

Proof. First, note that the optimal solution in $\bigcup \mathcal{I}_i$ has cost at most $O\left(n^{5z}\alpha/\varepsilon\right)$, since the maximal distance in any of $\mathcal{I}_1, \dots \mathcal{I}_m$ is $O\left(n^4(\alpha/\varepsilon)^{1/z}\right)$. Using this fact, we build a simple dynamic program to prove the lemma. For all $i \leq m$ and $j \leq \log_{1+\varepsilon/\log n}\left(n^{5z}\alpha/\varepsilon\right)$, let $k_{i,j}$ be the minimal k' such that the cost of P with k' centers in $\mathcal{I}_i$ is at most $(1+\varepsilon/\log n)^j$. $k_{i,j}$ can be computed with a simple binary search, using the fact that the cost of a solution is decreasing with k'.

Given all the $k_{i,j}$, a simple dynamic program can compute $k_{\geq i,j}$, the minimal number of centers needed to have a cost at most $(1+\varepsilon)^j$ on $\mathcal{I}_i, \dots \mathcal{I}_m$ (the $\varepsilon/\log n$ becomes a simple ε because of the accumulation of errors). The solution for our problem is $(1+\varepsilon)^j$, where j is the minimal index such that $k_{\geq 1,j} \leq k$.

The complexity of computing $k_{i,j}$ is $O(\log k \cdot n_i(\log n)^\alpha t(O(n^4/\varepsilon)))$, hence the complexity of computing all the $k_{i,j}$ is $O(n(\log n)^{\alpha+2} t(O(n^4/\varepsilon))$. The complexity of the dynamic program computing $k_{\geq i,j}$ is then simply $O(m \log n) = O(n \log n)$, which concludes the proof. □

3.7.2 Portal Respecting Paths and Solutions

Recall that any part $B \in \mathcal{B}_i$ of the decomposition $\mathcal{D} = \{\mathcal{B}_0, \dots, \mathcal{B}_{|\mathcal{D}|}\}$ comes with a set of portals $\mathcal{P}_B$ with the properties listed in Lemma 3.10. In a portal-respecting solution, every client connects to a facility by going in and out of parts of the decomposition only at designated portals. More concretely, a *path* in a metric between two nodes u and v is given by a sequence of nodes $w_1, \dots, w_k$, where $u = w_1, v = w_k$, and its length is $\sum_{j=1}^{k-1} \text{dist}(w_j, w_{j+1})$. A solution can be seen as a set of facilities, together

with a path for each client that connects it to a facility, and the cost of the solution is given by the sum over all path lengths. We say a path $w_1, \ldots, w_k$ is *portal-respecting* if for every pair w_j, w_{j+1}, whenever w_j and w_{j+1} lie in different parts $B, B' \in \mathcal{B}_i$ of the decomposition $\mathcal{D}$ on some level i, then these nodes are also portals at this level, i.e., $w_j, w_{j+1} \in \mathcal{P}_B \cup \mathcal{P}_{B'}$. As explained in Lemma 3.35, if two vertices u and v are cut at level i, then there exists a portal-respecting path from u to v of length at most $\text{dist}(u, v) + 16\rho 2^i$. We define a *portal-respecting* solution to be a solution such that each path from a client to its closest facility in the solution is portal-respecting.

▶ **Lemma 3.35.** If two vertices u and v are cut by a decomposition at level i, there exists a portal-respecting path of length $\text{dist}(u, v) + 16\rho 2^i$ that connects u to v. ◀

Proof. If u and v are cut on level i, then they lie in the same part $B \in \mathcal{B}_{i+1}$ on level $i + 1$ of the decomposition $\mathcal{D}$. As each part on level 0 of $\mathcal{D}$ is a singleton set, both u and v are portals on that level. Now consider the path that starts in $u = w_1$, and for each $j \geq 1$ connects w_j to the closest portal $w_{j+1} \in \mathcal{P}_B$ of the part $B \in \mathcal{B}_j$ on the next level j, until level $i + 1$ is reached. This yields a portal-respecting path P_u, as portals are nested, i.e., each w_j is a portal on every level less than j. A similar procedure finds a portal-respecting path P_v from the other endpoint v up to level $i+1$ through portals of levels below $i+1$. Since u and v lie in the same part on level $i+1$, we may obtain a portal-respecting path from u to v by first following P_u up to level $i + 1$, then connecting to the endpoint of P_v that is a portals of level $i + 1$, and then following P_v all the way down to v. The length of this portal-respecting path is at most $\text{dist}(u, v) + 4 \sum_{j \leq i+1} \rho 2^{j+1} = \text{dist}(u, v) + 32\rho 2^i$, due to the triangle inequality and the preciseness property of the portals (cf. Lemma 3.10). □

3.7.3 Computing a Constant-Factor Approximation to Facility Location

We first explain how to compute the optimal value for Facility Location in a tree metric. More precisely, the input is as follows: we are given a rooted tree T, whose leaves are exactly the clients and candidate facilities (i.e., the leaf set is $C \cup F$), and each internal node has at least two children. For any internal vertex x, there is a label d_x. The labels are such that if y is a descendant of x, then $d_y \leq d_x$. The distance between two leaves u, v is $d_{lca(u,v)}$, where $lca(u, v)$ is the lowest common ancestor of u and v. For an internal node x, we let $T(x)$ be the set of leaves that are descendant of x.

In such metric, it is possible to compute the optimal solution with a dynamic program using the following observation: for an internal node x, if there is a facility opened in $T(x)$, then all clients of $T(x)$ are served by some facility in $T(x)$. This is simply because going to a facility outside of $T(x)$ costs more than d_x, while the distance to any facility inside $T(x)$ is at most d_x.

Hence, if $F(x)$ is the facility location cost of clients in $T(x)$ when a facility is opened in the set $T(x)$, we have the following recurrence for a node x with two children x_1 and x_2:

$$F(x) = \min\{F(x_1) + F(x_2), F(x_1) + d_x \cdot |T(x_2) \cap C|, F(x_2) + d_x \cdot |T(x_1) \cap C|\}$$

The first term corresponds to the case when there are facilities opened in $T(x_1)$ and $T(x_2)$, the second one to when there is no facility opened in $T(x_2)$ and the third to when there is no facility opened in $T(x_1)$. The base case is for a leaf: $F(x) = \infty$ when $x \notin F$, $F(x) = w_x$ when $x \in F$.

Computing that DP table takes $O(n)$ time, since there are $O(n)$ vertices in the tree.

Using this algorithm, one can compute a constant factor approximation in doubling metrics as follows. First compute a tree where leaves are input points, such that the distances in the tree are the same as in the original input up to a factor $3n^2$, in time $O(n \log n)$ using the algorithm described in Corollary 3.5 from Har-Peled and Mendel [97]. Using that tree, it is possible to compute an $O(n^2)$-approximate solution to Facility Location, using the algorithm described above. Now, from Lemma 3.33 we can assume that the aspect-ratio of the instance is $O(n^7/\varepsilon)$. Hence, it is possible to compute a tree metric that preserves all distances by a factor $2^{O(d)}$, using the same construction as in the proof of Lemma 3.10 in Appendix 3.7.1 with parameter $\rho = 1$. It is standard to transform that tree into a binary tree, to make it corresponds with tree metrics as described above. Therefore, we can apply again the previous algorithm to get a $2^{O(d)}$-approximation.

The total time needed by this approximation algorithm is $O(n \log n)$.

Chapter 4

Scalable Differentially Private Clustering via Quadtrees

In this chapter, we study the private k-median and k-means clustering problem in d dimensional Euclidean space. By leveraging the techniques introduced in the previous chapter, we give a theoretically efficient and easy to implement algorithm. Although its worst-case guarantee is worse than that of state of the art private clustering methods, the algorithm we propose runs in time $\tilde{O}(nkd)$ and scales to very large datasets. We also show that our method is amenable to parallelization in large-scale distributed computing environments. In particular, we show that our private algorithm can be implemented in a logarithmic number of MPC rounds in the sublinear memory regime.

Finally, we complement our theoretical analysis with an empirical evaluation demonstrating the algorithm's efficiency and accuracy in comparison to other privacy clustering baselines.

4.1 Introduction

As noted in a white paper about the interface between privacy and data collection [79], *"individual privacy is essential for the functioning of democratic society and for the individuals who compose it"*. This need for privacy is endangered by constant collection and diffusion of highly detailed personal data, which is mostly unregulated yet. When collecting and analyzing this data, it is desirable to use algorithms respecting some form of privacy: for instance, algorithms whose outcome are not impacted by the presence or absence of a particular data point. That way, sharing the result of the algorithm may not harm individuals. Of course, a perfect privacy would lead to useless algorithm, that would not depend at all on the input: there is therefore a trade-off between the privacy ensured by an algorithm and the utility of its result.

The notion of *Differential Privacy (DP)* has emerged as a de facto standard to mathematically define this trade-off. It was introduced by Dwork, McSherry, Nissim and Smith in 2006 [67], and is as follows. Differential Privacy is characterized by the notion of neighboring datasets, X and X', generally assumed to be differing on a single user's data. An algorithm $\mathcal{A}$ is ϵ-differentially private if the probability of observing any particular outcome S when run on X vs X' is bounded: $Pr[\mathcal{A}(X) = S] \leq e^{\epsilon} Pr[\mathcal{A}(X') = S]$. Hence, changing a single entry from the input dataset cannot change too much the output distribution of a DP algorithm. This means that the outcome of the algorithm cannot heavily depend too much on a single entry: in a sense, the algorithm respect the privacy of each entry, as the outcome does not depend on the presence or not of a given entry in the dataset. Note in particular that this notion of privacy does not depend on the computational power of some adversary, or on some conjectured hardness.

For an introduction, see the book by Dwork and Roth [68]. At a high level, DP forces an algorithm to not focus on any individual training example, rather capturing global trends present in the data.

Private Clustering As described in the introductory Chapter 1, clustering is a key data-analysis routine, and releasing a clustering may leak privacy of the input. In turn, differentially private clustering has been well studied, with a number of works giving polynomial time approximately optimal algorithms for different versions of the problem, including k-median and k-means [17, 146, 87, 40, 124]. From an analysis standpoint, the quality of those algorithm is evaluated in term of their approximation guarantees: an algorithm is an (α, β)-approximation if the cost of the solution is at most $\alpha \mathrm{OPT} + \beta$.

This additive error is necessary, in contrast with the non-private setting: any ε-DP algorithm for k-median must have an additive error $\frac{\Lambda k d}{\varepsilon}$, where the input is guaranteed to lie in a ball of diameter Λ. To see the reason of such an additive error, consider the following collection of neighboring datasets: in each of them, there are n points at the origin and a single one at distance Λ. The solution computed should be roughly the same on all those, forcing to ignore the point at distance Λ. Hence, all centers of the solutions should be at 0, leading to a solution with cost Λ – while the optimal k-median on each of those dataset would have cost 0, as there are only two distinct points on each of them. Analyzing formally this example, and generalizing it leads to a lower bound on the additive error of $\frac{\Lambda k d}{\varepsilon}$, as shown by Nguyen, Chaturvedi and Xu [146].

All else being equal, we aim for algorithms that minimize α and β, and recent work by Ghazi, Kumar and Manurangsi [87] has made a lot of progress in that direction. However, in a push to minimize α and β, algorithms often pay in added complexity and running time. In fact, all known differentially private clustering algorithms with theoretical guarantees have this shortcoming: they have superlinear running times and do not scale to large datasets. Hence there is a big gap between optimal algorithms in theory and those that can be used in practice.

Our Results and Techniques. In this chapter, we aim to address the above shortcomings, and design practical differentially private clustering algorithms, with provable approximation guarantees, that are fast, and amenable to scale up via parallel implementation. Toward this goal, we take an approach that has been successful in the non-private clustering literature. While there are constant-approximate algorithms for k-median and k-means, see [5, 109, 108], most practical implementations use the k-means++ algorithm of [10], which has a $\Theta(\log k)$ approximation ratio. The reason for the success of k-means++ is two-fold. First, it is fast, running in linear time, and second, it performs well empirically despite the logarithmic worst-case guarantee. The methods we introduce in this work, while different from k-means++, have the same characteristics: they are fast, and perform *much* better than their worst-case guarantees, significantly outperforming all other implementations. In particular, they run in near-linear time, are amenable to parallel implementation in logarithmic number of rounds, and output high-quality private clusters in practice.

Our first contribution is an efficient and scalable algorithm for differentially private k-median. Our starting point is an embedding of the input points into a tree using a randomly-shifted quadtree.[1] As we saw in previous chapter, such tree embeddings can approximately preserve pairwise distances. Input points are mapped to leaf of the embedding: one key property is that, in order to solve k-median on the tree embedding, one only needs to know the number of point mapped to each leaf – and not which point is mapped to it. Our key insight for using those embeddings in a private manner is that it is possible to truncate the tree embedding, so that leaves represent sets of points of large enough cardinality. That way, each input point is represented in the tree embedding the same way as many other points: hence, a single point does not contribute much to the structure of a solution. Using this insight and carefully adding random noise to the cardinality of each leaf, we obtain our differentially private k-median algorithm.

Our second contribution is a parallel implementation of our algorithm in the classic massively parallel computing (MPC) model. This model is a theoretical abstraction of real-world systems like MapReduce [62], Hadoop [161], Spark [162] and Dryad [106] and it is the standard for analyzing algorithms for large-scale parallel computing [110, 88, 24]. Interestingly we show that our algorithm can be efficiently implemented using a logarithmic number of MPC parallel rounds for k-median clustering. To the best of our knowledge, our algorithms are the first differentially private algorithms for k-median that can be efficiently parallelized.

Third, we complement our theoretical results with an in-depth experimental analysis of the performance of our k-median algorithm. We demonstrate that not only our algorithms scale to large datasets where, until now, differential private clustering with provable guarantees remained elusive, but also we outperform other state-of-the-art private clustering baselines in medium-sized datasets. In particular, we show that in practice compared to prior work with theoretical guarantees such as [17], our parallel algorithm can scale to more than 40 times larger datasets, improves the cost by up to a factor of 2, and obtains solutions within small single digit constant factor of

[1] We note here that this technique has already been used in prior work for private clustering [17] to find a $\tilde{O}(n)$-sized set of candidate centers, to then run a polynomial time local search algorithm. In contrast, we use this structure to directly compute a solution.

non-private baselines.

Finally, we adapt those techniques to the k-means problem. This poses an additional challenge because randomly-shifted quadtrees do not preserve squared distances accurately and so we cannot apply our approach directly. For that, we adapt the technique introduced in Chapter 3 to our setting. The key observation behind our approach is that even if randomly-shifted quadtrees do not preserve well all the squared distances they accurately preserve most of them.

Suppose that we are given in input some solution S for our problem (later will clarify how to obtain it), then we show that for most centers in S it is true that the distances to these centers are approximately preserved by the quadtree. So points living in the clusters of these centers can be clustered using the tree embedding, and the remaining points can be clustered using the few centers that are not preserved in the solution S. Interestingly, we prove that we can use this approach iteratively starting with a solution to the differentially private 1-means problem as S. In Section 4.5 we show that this approach leads to an efficient differentially private algorithm for k-means.

4.2 Preliminaries

Notations. For two points p and q in $\mathbb{R}^d$, we let $\text{dist}(p,q) := \|p-q\| = \sqrt{\sum_{i=1}^{d}(p_i-q_i)^2}$ be the Euclidean distance between p and q. Given $r \geq 0$, we define $\beta(x,r) = \{y \in \mathbb{R}^d | \text{ dist}(x,y) \leq r\}$ as the closed ball around x of radius r. As we mentioned in the introduction, it is necessary to assume that the input P is contained in the open ball $\beta(0,\Lambda)$.

We say that a solution C is (α,β)-approximate for (k,z)-clustering if the $\text{cost}(P,C) \leq \alpha\text{cost}(P,\text{OPT}) + \beta$. We will seek solutions where α is $\tilde{O}(\text{poly}(\log(n)))$ and β is $O(\text{poly}(k,d,\log(n))\cdot\Lambda^z)$. Such high values for β are unfortunately necessary, as shown by Chaturvedi, Nguyen and Xu [146].

The goal of this chapter is to show private algorithms for k-median and k-means ($z=1$ and $z=2$) that are *efficient* easy to parallelize, where by efficient we mean time $\tilde{O}(n \cdot \text{poly}(\log(n),k,d,\log(\Lambda)))$. Notice that celebrated k-means++ [10] satisfies all the requirements with its $O(nkd)$ running time and $O(\log(k))$ approximation, except that it is not private.

Differential privacy. Our formal definition of a private algorithm is the following:

> ▶ **Definition 4.1.** Let $\varepsilon > 0$ be a real number, and $\mathcal{A}$ be a randomized algorithm, that takes a dataset D as input an output $\mathcal{A}(D)$.
> $\mathcal{A}$ is ε-differentially private if for any pair of datasets D_1, D_2 that differ on a

single element, and for all possible set S of outcomes of $\mathcal{A}$,

$$\Pr[\mathcal{A}(D_1) \in S] \leq \exp(\varepsilon) \Pr[\mathcal{A}(D_2) \in S].$$ ◀

A standard differentially private algorithm is the Laplace Mechanism (see [68]). We say that a random variable follows distribution $\mathrm{Lap}(b)$ if its probability density function is $\frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$. With a slight abuse of notation, we use $\mathrm{Lap}(b)$ to denote a variable that follows such a distribution. Example 3.1 in [68] shows that the following algorithm for counting queries is ε-DP : $\mathcal{A}(X) = |X| + \mathrm{Lap}(1/\varepsilon)$.

Note that the notion of differential privacy is only a model, and our result should not be used blindly to preserve privacy of users. We emphasize in particular that the privacy notion is with respect to a single user's data: hence, this model does not necessarily ensure privacy for a *group* of people.

Composition properties. We will make use of standard composition properties of differentially private algorithms, described in [68]. The algorithm $\mathcal{A}$ that applies successively two algorithm $\mathcal{A}_1$ and $\mathcal{A}_2$ that are respectively ε_1-DP and ε_2-DP is itself $(\varepsilon_1 + \varepsilon_2)$-DP. If the $\mathcal{A}_1$ and $\mathcal{A}_2$ run on two distinct parts of the dataset, then $\mathcal{A}$ is $\max(\varepsilon_1, \varepsilon_2)$-DP.

Lastly, if $\mathcal{A} : D_1 \times D_2 \to Z$ satisfies that for all $X \in D_1$, the algorithm $\mathcal{A}(X, \cdot)$ is ε_1-DP, and some algorithm $\mathcal{B} : D_2 \to D_1$ is ε_2-DP, then the algorithm $Y \to \mathcal{A}(\mathcal{B}(Y), Y)$ is $(\varepsilon_1 + \varepsilon_2)$-DP.

Randomly-shifted Quadtrees. A quadtree is a binary tree T, such that each node x in the tree corresponds to a region $T(x)$ of $\mathbb{R}^d$. To distinguish from the input, we call tree nodes *cells*. Each cell is a hyper-rectangle. For a cell c with children c_1, c_2, the region spanned by c is the union of those spanned by c_1 and c_2, i.e., $T(c) = T(c_1) \cup T(c_2)$.

A shifted quadtree is constructed as follows. Start from a root cell containing the entire d-dimensional hypercube $[-\Lambda, \Lambda]^d$ at depth 0, and proceed recursively. Let c be a cell at depth $d \cdot i + j$, with $0 \leq j < d$. The j-th coordinate of the region spanned by c is comprised in $[m, M]$. The children of c are constructed as follows: let x be some random number in $[m + \frac{M-m}{3}, M - \frac{M-m}{3}]$. c_1 comprises all points of c that have their j-th coordinate at most x, and c_2 the remaining points[2]. Note that the diameter of cells is divided by at least $3/2$ every d levels. Denote by $\mathrm{diam}(c)$ the diameter of cell c.

A quadtree $\mathcal{T}$ induces a metric: for two points p, q, we define $\mathrm{dist}_{\mathcal{T}}(p, q) = \mathrm{diam}(c)$ where c is the smallest cell that contains both p and q. We will frequently use that the expected distortion between two points p and q in a shifted quadtree of depth $d \cdot \alpha$, for any α, is $\mathbb{E}_{\mathcal{T}}[\mathrm{dist}_{\mathcal{T}}(p, q)] \leq d^{3/2} \alpha \mathrm{dist}(p, q)$. This property is similar to the

[2] Another standard way of defining quadtree is to have 2^d-regular trees, and to split along the d-dimensions at each step. We are more comfortable working with binary trees, which allows for a simpler dynamic program.

scaling probability of the hierarchical decomposition of Lemma 3.10: it is only more precise, as it is possible to use the geometry of Euclidean Spaces to get a polynomial dependency in d instead of an exponential one, as in Lemma 3.10. For a proof, we refer to Lemma 4.13. In our case, we stop the construction when reaching cells of diameter Λ/n. Hence, $\alpha = \log n$ and the expected distortion is $O(d^{3/2} \log n)$. Such trees are often called Hierarchically Separated Trees (HST) in the literature.

Dimension Reduction. For clustering problems, it is possible to use the Johnson-Lindenstrauss Lemma – which is oblivious to the data, hence private – to reduce the dimension to $O(\log k)$ (see [129]). Hence, we can apply all our algorithm in such a dimension, replacing dependency in d by $\log k$. To compute centers in the original space, we can extract the clusters from the low-dimensional solution and compute privately the 1-median (or 1-mean) of the cluster in the original d-dimensional space. This adds an additive error $O(kd)$. This is not as simple in the MPC model, where even finding the 1-median is not easy. For that model, we need additional guarantees on the projected space to be able to recover centers in the original space. We managed to show that those guarantees hold when projecting onto an $O(\log n)$-dimensional space, hence replacing dependencies in d by $O(\log n)$.

4.3 Simple algorithm for Differentially Private k-Median

4.3.1 Algorithm

A simple way of solving k-median is to embed the input points into an ultrametric. Those are sufficiently simple as to admit a dynamic program for computing an optimum. The approximation factor of this algorithm is therefore the distortion incurred by the embedding. We adapt this approach to incorporate privacy as follows. First, we embed the input into a quadtree, which is a hierarchical decomposition of $\mathbb{R}^d$ via axis-aligned hyperplanes. We then add noise on the quadtree to enforce privacy. Subsequently, we run the dynamic program on the quadtree. Unfortunately, a naive implementation of the dynamic program on that noisy quadtree falls short of the nearly linear time algorithm we are hoping for: indeed, due to the addition of noise, the size of the quadtree cannot be bounded. In particular, there are many quadtree cells that do not contain any point of P but that have a noisy weight bigger than zero: the size of the noisy quadtree cannot be easily related to the input size. To cope with that, we show it is possible to trim the noisy quadtree cells containing few points.

The trimmed and private quadtree is constructed by Algorithm 2. Then, the dynamic program to solve k-median on the tree is presented in Algorithm 3. The combination of those two gives Algorithm 1, which is our main algorithm:

▶ **Theorem 4.2.** Algorithm 1 is ε-DP and computes a solution to k-median with expected cost at most $O(d^{3/2}\log n)\cdot \mathrm{OPT} + \frac{d^2\cdot\log^2 n\cdot k}{\varepsilon}\cdot\Lambda$. Furthermore, it runs in time $\tilde{O}(ndk^2)$. ◀

Algorithm 1 DP-kMedian(P)

Compute a shifted quadtree T. Let r be the root of T.
w = MakePrivate(T, P).
Compute v, S =DynamicProgram-kMedian(T, w, r)
Return S_k

Algorithm 2 MakePrivate(T, P)

Input: a quadtree T, a set of points P
Output: for each cell c of T with diameter more than Λ/n, a private count of $|c \cap P|$.
let Q be a queue, initially containing only the root of T.
Let $w : T \to \mathbb{N}$, initiated with $\forall c,\ w(c) = 0$.
while Q is not empty **do**
    Let $c = Q$.pop()
    if $\mathrm{diam}(c) > \Lambda/n$ **then**
        let $w(c) = |T(c) \cap P| + \mathrm{Lap}(d\log n/\varepsilon)$
        if $w(c) > 2d\log n/\varepsilon$ **then**
            Add c's children to Q
        end if
    end if
end while
Return w

4.3.2 Analysis

▶ **Lemma 4.3.** Step 3 of Algorithm 1 computes a solution with expected cost $\mathrm{OPT}_T + k\cdot d^2\log^2 n/\varepsilon\cdot\Lambda$, where OPT_T is the optimal solution on the metric induced by the tree T, and the expectation is taken over the realization of the variables Lap. ◀

Proof. In a quadtree metric, we have the following property. For a cell c, three points $x, y \in T(c)$ and $z \notin T(c)$, it holds that $\mathrm{dist}(x, y) \le \mathrm{diam}(T(c)) \le \mathrm{dist}(x, z)$. Hence, if there is a center in $T(c)$, then all clients of $T(c)$ can be served by some center in $T(c)$. Moreover, if there is no center in $T(c)$, points in $T(c)$ are at distance at least $\mathrm{diam}(T(c))$ of a center.

Let $F(c, k')$ be the expected cost of the value $v_{k'}$ returned by 'DynamicProgram-k-Median(T, k', c), where the probability is taken over the privacy randomness. and S

Algorithm 3 DynamicProgram-kMedian(T, w, c)

Input: A quadtree T, a weight function on T's node w, and a cell c.
Ouput: For each $k' \leq k$, a solution $S_{k'}$ for k'-median on $T(c)$ and its value $v_{k'}$.
Set $v_0 \leftarrow w(c) \cdot \mathrm{diam}(c)$ and $S_0 \leftarrow \emptyset$.
if $w(c) < \frac{2d \log n}{\varepsilon}$ **then**
    For all $k' \geq 1$, set $v_{k'} \leftarrow 0$ and $S_{k'} \leftarrow k'$ copies of the center of c.
else
    Let c_1, c_2 be the two children of cell c
    Let v^i, S^i be the output of DynamicProgram-kMedian(T, w, c_i).
    for all k', let $(k_1, k_2) = \mathrm{argmin}_{k_1, k_2 \geq 1 : k_1 + k_2 = k'} v^1_{k_1} + v^2_{k_2}$, and do $v_{k'} \leftarrow v^1_{k_1} + v^2_{k_2}$, $S_{k'} \leftarrow S^1_{k_1} \cup S^2_{k_2}$.
end if
Return v, S.

be any solution. We show by induction that for any cell c of height h in the tree with $T(c) \cap S \neq \emptyset$,

$$F(c, |S \cap T(c)|) \leq \mathrm{cost}_T(T(c), S) + \frac{d \log n}{\varepsilon} \cdot \Lambda \cdot |S \cap T(c)| \cdot h.$$

This is true by design of DynamicProgram-kMedian for any leaf that contains a center of S.

For an internal node c with two children c_1, c_2 such that $T(c_1) \cap S \neq \emptyset$ and $T(c_2) \cap S \neq \emptyset$: it holds that $F(c, S \cap T(c)) \leq F(c_1, |S \cap T(c_1)|) + F(c_2, |S \cap T(c_2)|)$. Hence, by induction:

$$\begin{aligned}
&F(c, |S \cap T(c)|) \leq F(c_1, |S \cap T(c_1)|) + F(c_2, |S \cap T(c_2)|) \\
&\leq \mathrm{cost}_T(T(c_1), S) + \frac{d \log n}{\varepsilon} \cdot \Lambda \cdot |S \cap T(c_1)| \cdot (h-1) \\
&\qquad + \mathrm{cost}_T(T(c_2), S) + \frac{d \log n}{\varepsilon} \cdot \Lambda \cdot |S \cap T(c_2)| \cdot (h-1) \\
&\leq \mathrm{cost}_T(T(c), S) + \frac{d \log n}{\varepsilon} \cdot \Lambda \cdot |S \cap T(c)| \cdot h,
\end{aligned}$$

where the last lines uses $\mathrm{cost}(T(c), S) = \mathrm{cost}(T(c_1), S) + \mathrm{cost}(T(c_2), S)$.

The the last case is when $S \cap T(c_1) = \emptyset, S \cap T(c_2) \neq \emptyset$. Let h be the height of c. We have $S \cap T(c_2) = S \cap T(c)$, and so:

$$\begin{aligned}
&F(c, |S \cap T(c)|) \leq F(c_1, 0) + F(c_2, |T(c_2) \cap S|) \\
&\quad \leq T(c_1) \cdot \mathrm{diam}(c_1) + \mathbb{E}[\mathrm{Lap}(\varepsilon/(d \log n)) \cdot \mathrm{diam}(c_1)] \\
&\qquad + \mathrm{cost}_T(T(c_2), S) + \frac{d \log n}{\varepsilon} \cdot \Lambda \cdot |S \cap T(c_2)| \cdot (h-1) \\
&\quad \leq \mathrm{cost}_T(T(c), S) + \frac{d \log n}{\varepsilon} \cdot |S \cap T(c)| \cdot \Lambda \cdot h.
\end{aligned}$$

This shows that the value computed by the algorithm is at most $\mathrm{cost}_T(\mathrm{OPT}) + \frac{d^2 \log^2 n}{\varepsilon} \cdot \Lambda \cdot k$. Now, we need to show the converse: the value computed corresponds to an actual solution.

This is done inductively as well. For any k' and cell c one can compute a solution S for $T(c)$ with k' centers and expected cost at most $F(c,k') + \frac{d\log n}{\varepsilon}\cdot k'$. For that, the base cases are when $k'=0$, and then $\emptyset$ works, or when $w(c) \le \frac{2d\log n}{\varepsilon}$, where the center of the cell works. Otherwise, it is enough to find k_1, k_2 such that $k_1+k_2=k'$ and $F(c,k') = F(c_1,k_1)+F(c_2,k_2)$. Let S_i be the solution computed for $T(c_i)$ with k_i centers: the solution for $T(c)$ is simply $S_1 \cup S_2$. By induction, its cost is at most

$$F(c_1,k_1)+F(c_2,k_2)+\frac{d\log n}{\varepsilon}\cdot\Lambda\cdot(k_1+k_2) = F(c,k')+\frac{d\log n}{\varepsilon}\cdot\Lambda\cdot k'.$$

□

▶ **Lemma 4.4.** Algorithm 2 is ε-DP. ◀

Proof. We show by induction that for a tree T of depth h, MakePrivate(T,P) is $\left(\frac{\varepsilon}{d\log n}\cdot h\right)$-DP.

When the root of the tree has diameter at most Λ/n, the algorithm returns the zero function, which is 0-DP. Let $\mathcal{T}$ be a tree of depth h rooted at r with $\operatorname{diam}(r) > \Lambda/n$), and let r_1, r_2 be the two children of r. Computing $w(r)$ is $\frac{\varepsilon}{d\log n}$-DP, by property of the Laplace Mechanism.

Now, by induction hypothesis, MakePrivate$(T(r_1),P)$ and MakePrivate$(T(r_2),P)$ are $\left(\frac{\varepsilon}{d\log n}\cdot(h-1)\right)$-DP. Since they are computed on two disjoint sets, the union of the two results is $\left(\frac{\varepsilon}{d\log n}\cdot(h-1)\right)$-DP as well. Notice that the algorithm MakePrivate(T,P) boils down to computing $w(r)$, MakePrivate$(T(r_1),P)$ and MakePrivate$(T(r_2),P)$. Hence, by composition MakePrivate(T,P) is $\left(\frac{\varepsilon}{d\log n}\cdot h\right)$-DP.

□

Combining Lemmas 4.3, 4.4 and properties of quadtrees, we conclude the proof of Theorem 4.2:

Proof of Theorem 4.2. We start by proving the approximation guarantee. For this, note that the key property of quadtrees is that $\mathbb{E}_{\mathcal{T}}[\operatorname{dist}_{\mathcal{T}}(p,q)] \le O\left(d^{3/2}\log n\right)\operatorname{dist}(p,q)$, where the expectation is taken on the tree randomness (see Lemma 4.13 in the appendix). Hence, the optimal k-median solution is only distorted by an $O\left(d^{3/2}\log n\right)$ factor: $\mathrm{OPT}_T \le O\left(d^{3/2}\log n\right)\mathrm{OPT}$.

Combined with Lemma 4.3, this shows the approximation guarantee of the whole algorithm. Lemma 4.4 shows the privacy guarantee. What therefore remains is to bound the running time.

Computing the cells of the quadtree containing some points of P can be done in a top-down manner in time $O(nd\log n)$ as follows. Let c be a cell at depth $d\cdot i+j$ with $j<d$, and c_1, c_2 be the two children of c. Given $T(c)\cap P$, it is easy to compute $T(c_1)\cap P$ and $T(c_2)\cap P$ in time $O(|T(c)\cap P|)$, by partitioning $T(c)\cap P$ according to

the value of their j-th coordinate. Since there are $O(d \log n)$ levels in the tree, this is done in time $O(nd \log n)$.

Hence, the running time of Algorithm 2 is bounded by $\tilde{O}(nd)$ plus the time to process empty cells added to Q. There are at most $nd \log n$ empty cells with a non-empty parent added – one per level of the tree and per point of P. Each of them gives rise to a Galton-Watson process: each node adds its two children with probability $\Pr[\mathrm{Lap}(d \log n/\varepsilon) > 2d \log n/\varepsilon] = e^{-2} < 1/2$. By standard properties of Galton-Watson process, this goes on for a constant number of steps. Therefore, there are at most $\tilde{O}(nd)$ empty cells added to Q, which concludes the running time bound for Algorithm 2.

Let N be the number of cells that have a non-zero value of w. We claim that $N = \tilde{O}(nd)$ and that the running time of Algorithm 3 is $O(Nk^2)$. For the first claim, note that N is equal to the number of cells added to Q, which is $\tilde{O}(nd)$ as explained previously. For the second claim, notice that there are at most kN different calls to DynamicProgram-kMedian, each being treated in time $O(k)$. Hence, the complexity of Algorithm 3 is $O(Nk^2) = \tilde{O}(ndk^2)$. This concludes the proof. □

4.4 MPC Implementations

Brief description of MPC We briefly summarize the MPC model [24]. The input data has size $N = nd$, where n is the number of points, and d the dimension. We have m machines, each with local memory s in terms of words (of $O(\log(ms))$ bits). We assume that each word can store one input point dimension. We work in the *fully-scalable* MPC framework [6] where the memory is sublinear in N. More precisely, the machine memory is $s = \Omega\left(N^{\delta}\right)$ for some constant $\delta \in (0, 1)$, and the number of machines m is such that $m \cdot s = \Omega\left(N^{1+\gamma}\right)$, for some $\gamma > 0$.

The communication between machines is as follows. At the beginning of the computation, the input data is distributed arbitrarily in the local memory of machines, the computation proceeds in parallel rounds where each machine can send (and receive) arbitrary messages to any machine, subject to the total messages space of the messages received (or sent) is less than s. In case some machine receives more than s messages, the whole algorithm fails.

For our MPC algorithm we assume that $k \ll s$. This ensures that the final solution of size kd fits in the memory of one machine, which is common for real world applications.

More formally we assume that there are $m = \Omega(n^{1-\delta+\gamma})$ machines each with memory $s = \Omega(n^{\delta} d \log(n))$, and $k \leq n^{\gamma}$, with $\delta - \gamma > \varepsilon$ for some constant ε.

In that section, we first show the following *low dimensional* theorem:

▶ **Theorem 4.5.** Assuming $k \leq n^{\gamma}$, there exists a $O(d \log n)$ rounds MPC algorithm using $m = O(n^{1-\delta+\gamma})$ machines each with memory $s = O(n^{\delta} d \log(n))$ that simulates exactly the private k-median from Theorems 4.2. ◀

This algorithm is suited for low dimensional spaces, as the number of rounds depends on d. We show in Section 4.4.2 how to replace this dependency by a $O(\log k)$, both in the number of rounds and in the approximation ratio.

We then show how to use dimension reduction, to replace dependencies in d by $\log k$:

▶ **Theorem 4.6.** Assuming $k \leq n^{\gamma}$, there exists a $O(\log k \cdot \log n)$ rounds MPC algorithm using $m = O(n^{1-\delta+\gamma})$ machines each with memory $s = O(n^{\delta} d \log(n))$ that computes a solution to k-median with cost at most

$$O\left(\log^{3/2} k \log n \cdot \mathrm{OPT} + \frac{\log^3 k \log^3 n \cdot k}{\varepsilon} \cdot \Lambda + \frac{kd \log k}{\varepsilon} \cdot \Lambda\right). \quad ◀$$

4.4.1 Algorithm for Low Dimensional Inputs

We now describe a high level view of our algorithm which as we can prove simulates exactly (with high probability) our private k-median algorithm. The algorithm uses a shared hash function h to compute the quadtree consistently over the machines. Informally, first, each machine computes over the points stored, all the cells which the points belong to in the tree at each level. To compute the total count of each cell, one can use the algorithm from Andoni et al. [6] (section E.3 of the arxiv version), that computes in a constant number of rounds the number of points in each cell. At the end of that algorithm, the size of each cell is stored in some unspecified machine. To organize the quadtree data in order to be able to process it, we use a shared function r such that a machine $r(c)$ is responsible for all computations related to cell c. We will need care to ensure that no machine is responsible for more cell than what its memory allows.

Then the computation proceeds bottom-up solving the dynamic programming problem in $O(d \log n)$ rounds.[3] Finally, the computation proceeds over the tree top-down in other $O(d \log n)$ rounds to extract the solution.

[3] We note that a more careful and intricate implementation of the dynamic program that requires only $O(d)$ rounds can be achieved. We decided to chose simplicity rather than saving one log factor.

Algorithm 4 MPC-quadtree(P)

1: Each machine receives an arbitrary set of n^δ points of P.
2: Each machine, for each point p received, computes, using h_s, the quadtree cell $c_i(p)$ in which the point p is at level $i \in [d \log(n)]$.
3: Compute the count of every cell.
4: Send the count of cell c to machine $r(c)$, for all c.

Using the algorithm from Andoni et al. [6], one can compute in $O(1)$ steps the count for each cell of the quadtree. Hence, we have the following result:

▶ **Fact 4.7.** Algorithm 4 runs in $O(1)$ many rounds. ◀

At the end of Algorithm 4, we are given a quadtree, represented as follows: each cell c is represented by a machine $r(c)$, which stores a count of input nodes in the cell and pointers towards each children. r is a surjection from a set of $O(nd \log n)$ cells to m machine: we chose it in order to ensure that for any machine $\mathcal{M}$, $|r^{-1}(\mathcal{M})| \leq \frac{O(nd \log n)}{m}$.

We now explain in more details how to implement the algorithm from Theorem 4.2, given that representation of the quadtree.

First, it is straightforward to implement Algorithm 2 in 1 rounds – as each cell only needs to compute the DP count of points in the cell. Next, Algorithm 3 is straightforwardly implemented in $O(d \log n)$ rounds, as computing the output vector v of the dynamic program for a cell only requires knowing those of its children – and it is therefore easy to simulate bottom-up the dynamic program.

What remains to be proven is that no machine gets responsible for more cell than it can afford in memory. More precisely, every time a machine is responsible for a cell, it stores $O(k)$ memory words, for the simulation of the dynamic program. Hence, we need to show that no machine is responsible for more than s/k many cells.

▶ **Fact 4.8.** No machine is responsible for more than s/k many cell. ◀

Proof. Our choice of m and mapping r ensures that a given machine gets responsible for at most $\frac{O(nd \log n)}{m} = O\left(n^{\delta-\gamma} d \log n\right)$. Similarly, our constraints on k and s ensures $\frac{s}{k} = \Omega(n^{\delta-\gamma} d \log n)$, which concludes the proof.

□

Combining those two facts concludes Theorem 4.5.

4.4.2 k-Median in $O(\log n)$-MPC rounds via dimension reduction

The goal of this section is to use standard dimension-reduction techniques to remove the dependency in the dimension from Theorem 4.5 and show Theorem 4.6.

For that, one can use dimension reduction techniques to project the dataset onto $O(\log k)$ dimensions, while preserving the cost of any clustering.

However, the output of our algorithm should be a set of centers in $\mathbb{R}^d$, and not a clustering: an additional step is therefore needed, once clusters have been computed in $\mathbb{R}^{O(\log k)}$, to *project back* and find centers in the original space. For k-means, this can easily be done using differentially-private mean [87]. We show in the following how to do the equivalent for k-median.

We draw here a connection with the second part of the thesis, that develops and uses *coreset*. More precisely, in Chapter 10, we show how to compute an approximate solution to 1-median by only considering an uniform sample of constant size. Therefore, in the MPC setting it is enough to sample a constant number of points from each cluster computed in low dimension, and send them to a machine that can compute a median for them in the original high dimensional space.

For that last step, we rely on the following result.

▶ **Lemma 4.9 (Corollary 54 in [87]).** For every $\varepsilon > 0$, there is an ε-DP polynomial time algorithm for 1-median such that, with probability $1 - \beta$, the additive error is $O\left(\frac{d\Lambda}{\varepsilon}\text{polylog}\left(\frac{1}{\beta}\right)\right)$ ◀

We consider the following algorithm, simplified variant of Algorithm 9.

Algorithm 5 Finding the median via uniform sampling

Input: A dataset P, an α-approximate median $\mathfrak{a}$ for P with cost $\mathcal{C}$, and parameters t, d_{close}, r_{small}.

1. Sample a set Ω of t points uniformly at random.
2. Remove from Ω all points at distance less than $\Delta = \frac{d_{close}}{\alpha} \cdot \frac{\mathcal{C}}{|P|}$, and add to Ω the point $\mathfrak{a}$ with multiplicity equal to the number of removed points.
3. Define rings R_i such that $R_i \cap \Omega$ contains all the points at distance $(2^i \cdot \Delta, 2^{i+1} \cdot \Delta]$ from $\mathfrak{a}$, for $i \in \{1, ..., \log(|P|\alpha/\mu_2)\}$. Let R_0 be $\{\mathfrak{a}\}$, with multiplicity defined in step 2.
4. If $|R_i \cap \Omega| < r_{small} \cdot |\Omega| + \text{Lap}(1/\varepsilon)$, remove all points in $R_i \cap \Omega$ from Ω.
5. Solve the problem on the set Ω, using algorithm given by Lemma 4.9 with $\beta = 1/k$.

▶ **Lemma 4.10.** Algorithm Algorithm 5 is 2ε-DP. ◀

Proof. First, the set of rings selected at step 4 is ε-DP: the selection of one ring is ε-DP, by Laplace mechanism, and since the rings are disjoint the composition of DP

mechanisms ensures that the full set of selected rings is ε-DP.

Now, given a selected set of rings, the set Ω varies by at most one point when the input P varies by a single point. Since the algorithm used in step 5 is ε-DP, by composition, the whole algorithm is 2ε-DP. □

We will see in Section 10.4 that this algorithm computes an $O(1)$-approximation to 1-median on P, with $t = \text{polylog}(|P|)$. Assuming this result for now (we defer the proof to Section 10.4), we can easily use it to *project back* the centers, and conclude the proof of Theorem 4.6.

Proof of Theorem 4.6. Using Johnshon-Lindenstrauss lemma, it is possible to project the points onto a space of dimension $\tilde{d} = O(\log k)$, preserving the cost of any clustering up to a constant factor (see Makarychev et al. [129]). In that projected space, the algorithm from Theorem 4.5 computes privately a solution with cost $O(\log^{3/2} k \log n) \cdot \text{OPT} + \frac{\log^3 k \log^3 n \cdot k}{\varepsilon} \cdot \Lambda$, but centers are not points in $\mathbb{R}^d$ – they are nodes of the quadtree.

To compute good centers in $\mathbb{R}^d$ from the quadtree solution, we use Algorithm 5: in each cluster induced by the quadtree solution, sample the set Ω. Since Ω has size $O(\log^3 k \log^2 n)$, it can be sent it to a centralizing machine, that in turn can run Algorithm 5. The additional additive error is $O\left(\frac{d\Lambda \log k}{\varepsilon}\right)$ in any cluster, hence in total $O\left(\frac{kd\Lambda \log k}{\varepsilon}\right)$. To sample the set Ω, each machine can send to the centralizing one the number of points it stores from P, and the centralizing computes the number of points to be sampled in each machine.[4] □

4.5 Extension to k-Means

The previous sections showed simple proofs in the case of k-Median. Here, we show how to extend them in the case of k-Means, using some form of *badly-cut* clients and facilities. We will not present MPC implementations of those algorithms, focusing for simplicity on the standard computation model.

As we already saw in the previous chapter, although the quadtree decomposition approximates distances well in expectation, it works poorly for squared distances. Indeed, two points p, q have probability $\frac{d \cdot \text{dist}(p,q)}{2^i}$ to be cut at level i: hence, the expected distance squared between p and q is $d \cdot \text{dist}(p,q) \cdot \sum_i d2^i$ (the squared distance when cut at level i is $d2^{2i}$, as the diameter of a level i cell is $\sqrt{d}2^i$), which means that the distance squared can be distorted by an arbitrarily large factor in expectation.

However, observe that p and q have tiny probability to be cut at a level way higher

[4] For instance, the centralizing machine can sample as set R of μ_1 points from $\{1, ..., |P|\}$. Then, if machine i stores n_i points from P, it computes a uniform sample $R \cap (\sum_{j<i} n_i, \sum_{j \le i} n_i]$ many points. The union of those sample is uniform.

than $\log(d \cdot \text{dist}(p,q))$. Hence, there is a tiny probability that points are cut from their optimal center at a high-level. The question is then: what to do when this happens? Here we want to avoid routing in the tree since the squared distance could be arbitrary large and we may want to deal with such points in a different way. To do so, we use a baseline solution L to guide our decisions on points for which the tree distance to their closest center in the optimum solution badly approximates the true distance, let call them *badly cut* points. Since we don't know the optimum solution, we don't know the badly cut points and so we will use L as a proxy for finding the potential bad points.

We show that the solution computed by our algorithm is good with respect to a solution that contains all facilities of L for which the quadtree distances are not good approximations of the true distances. We call those facilities *badly-cut* facilities. To bound the cost of a client c, we distinguish three cases. Either the distance from a point to optimal center is good in the tree, and we are happy because we can serve it nicely in the tree. Or its closest center of L is not badly-cut, in which case we argue that the distance to the optimal center cannot be too high compared to its optimal cost. In the last case, where the closest center of L is badly-cut, we simply assign the point to L since we are working with a solution containing all centers of L. This happens with some tiny probability, and will not be too costly overall, i.e.: only a tiny fraction of the cost of L. Using this reasoning, we show the following lemma:

▶ **Lemma 4.11.** Given an arbitrary solution L and $\alpha > 0$, there exist an ε-DP algorithm A that takes as input a set of points and computes a solution for k-means with at most $k + \frac{\alpha}{2} \cdot |L|$ centers and, with probability $1 - \pi$, costs at most $\frac{O(d^9 \log^2 n)}{\alpha^6 \pi^6} \cdot \text{OPT} + \alpha \cdot \text{cost}(L) + kd^2 \log^2 n/\varepsilon \cdot \Lambda^2$. ◀

Applying repeatedly this algorithm yields the following theorem:

▶ **Theorem 4.12.** For any $\alpha > 0$, there exists an ε-DP algorithm A that takes as input a set of points and computes a solution for k-means with at most $(1+\alpha)k$ centers and, with probability $3/4$, costs at most $\text{poly}(d, \log n, 1/\alpha) \cdot \text{OPT} + kd^2 \log^2 n/\varepsilon \cdot \Lambda^2$. ◀

Proof. We consider the following algorithm:

Algorithm 6 k-means algorithm with extra centers

1: **Input:** A set of clients X.**Output:** A set of k-means centers C.
2: $L \leftarrow 0$, $\varepsilon' \leftarrow \varepsilon / \log n$
3: **for** $\log n$ steps **do**
4: $\quad L' \leftarrow$ Solution computed by A as described by Lemma 4.11, setting $\pi = \frac{1}{4 \log n}$, with privacy parameter ε'.[5]
5: $\quad L \leftarrow L'$
6: **end for**
7: Return L

We argue that the above algorithm produces a solution satisfying the claims of Theorem 4.12. We have that the initial solution has cost at most $n\Lambda^2$. Then, by repeatedly applying Lemma 4.11, we obtain solutions of geometrically decreasing cost. More precisely, after i iterations, we claim that with probability $1 - \frac{i}{4 \log n}$, L has size at most $k \cdot \sum_{j=0}^{i} \alpha^i$, and the cost of L is at most $\text{poly}(d, \log n, 1/\alpha) \cdot \text{OPT} + \alpha^i n\Lambda + kd^2 \log^2 n/\varepsilon' \cdot \Lambda^2$. This is true when $i = 0$, and the induction follows directly from applying Lemma 4.11.

It follows that the final solution computed after $\log n$ steps has cost at most $\text{poly}(d, \log n)$ times OPT plus additive $kd^2 \log^3 n/\varepsilon \cdot \Lambda^2$. Moreover, since the initial solution L is fixed regardless of the input data (hence it is 0-DP), and since $\varepsilon' = \varepsilon/\log n$, the algorithm is by composition ε-DP. Finally, the number of centers is at most $k(1 + \alpha)$ as desired.

□

Hence, the key is to prove Lemma 4.11. For that, we first show how to preprocess the instance such as to be able to solve the problem via dynamic programming. We then use that preprocessing to build an algorithm showing Lemma 4.11. Finally, we show how to remove the extra αk centers by using the reverse greedy algorithm of Chrobak et al. [46].

4.5.1 Preprocessing the Instance Using Badly-Cut

Before describing the ideas behind the extension to k-means, we introduce some notations. We say that two points p, q are *cut at level* i when their lowest common ancestor in the tree is at level in $(d \cdot (i-1), d \cdot i]$, i.e., the diameter of that common ancestor is in $(\sqrt{d} \cdot 2^{i-1}, \sqrt{d} \cdot 2^i]$. In that case, the distance in the quadtree metric between p and q is at most $\sqrt{d}2^i$. We say that a ball $\beta(p, r)$ is cut at level i if i is the largest integer such that there exists a point q with $\text{dist}(p, q) \leq r$ and p and q are cut at level i.

We have the following lemma:

> ▶ **Lemma 4.13.** [Reformulation of Lemma 11.3 [94]] For any i, radius r and point p, we have that $\Pr[\beta(p, r) \text{ is cut at level } i] = O\left(\frac{dr}{2^i}\right)$. ◀

Formalization Let $P \subseteq \mathbb{R}^d$ be an instance of the k-means problem in $\mathbb{R}^d$. Let OPT be an optimal solution to P and L be an arbitrary solution. For a given client c, we let $L(c)$ (resp. $\text{OPT}(c)$) denote the center of L (resp. OPT) that is the closest to c in solutions L (resp. OPT).

Fix some α_C and $\alpha_F > 0$, that will be specified later. For a quadtree decomposition $\mathcal{T}$, we say that a client c is *badly-cut* if the ball $\beta(c, \text{dist}(c, \text{OPT}))$ is cut at a level higher than $\log(\text{dist}(c, \text{OPT}) \cdot d/\alpha_C)$ – note that this is for the analysis only, since we don't know this algorithmically. We say that a center $f \in L$ is *badly-cut* if for some

i, the ball $\beta(f, 2^i)$ is cut at a level higher than $i + \log(d \log n/\alpha_F)$. [6] As L and $\mathcal{T}$ will be fixed all along that section, we simply say that a point or a center is badly-cut. Notice that we do not know which clients are badly-cut. It is however possible to compute the badly-cut centers, since it depends only on L and $\mathcal{T}$. It is explained how to perform this step in time $\tilde{O}(nd)$ in Section 3.3.3: essentially, for each center $f \in L$, one can go up the tree and and at each level ℓ compute the distance D to the separating hyperplane: if $\lceil \log D \rceil + \log(d \log n/\alpha_F) < \ell$, then the ball $\beta\left(f, 2^{\lceil \log D \rceil}\right)$ is badly cut. Otherwise, no ball is badly cut at that level.

Our algorithm computes a randomized quadtree $\mathcal{T}$, and finds the badly-cut center $\mathcal{B}_\mathcal{T}$. It removes from the input each cluster associated with a center of $\mathcal{B}_\mathcal{T}$. Let $P_\mathcal{T}$ be the remaining points. $P_\mathcal{T}$ is a random variable that depends on the randomness of $\mathcal{T}$. Given a solution S for k-means on $P_\mathcal{T}$, the algorithm's output is $S \cup \mathcal{B}_\mathcal{T}$.

We call $\text{cost}(P, S)$ the cost of any solution S in the original input P, and $\text{cost}(P_\mathcal{T}, S)$ its cost in $P_\mathcal{T}$.

A key property for our analysis is a bound on the probability of being badly-cut, which is the equivalent of Lemma 3.12:

▶ **Lemma 4.14.** Any client p has probability at most α_C to be badly-cut. Similarly, a center $f \in L$ has probability at most α_F to be badly-cut. ◀

Proof. Consider first a point $p \in P$. By Lemma 4.13, the probability that a ball $\beta(p, r)$ is cut at level at least j is at most $dr/2^j$. Hence the probability that a ball $\beta(p, \text{dist}(p, \text{OPT}))$ is cut at a level j greater than $\log(\text{dist}(p, \text{OPT})) + \log(d/\alpha_C)$ is at most α_C. The proof for $f \in F$ is identical. □

Using that lemma, one can bound the cost of the clusters of facilities from $\mathcal{B}_\mathcal{T}$, as well as the cost of badly-cut clients:

▶ **Lemma 4.15.** For any $\pi \in (0, 1)$, it holds with probability $1 - \pi$ that:

$$\sum_{f \in \mathcal{B}_\mathcal{T}} \sum_{p : L(p) = f} \text{cost}(p, f) \leq 3/\pi \cdot \alpha_F \text{cost}(P, L),$$

$$\sum_{p \text{ badly-cut}} \text{cost}(p, L) \leq 3/\pi \cdot \alpha_C \text{cost}(P, L),$$

and $|\mathcal{B}_\mathcal{T}| \leq 3/\pi \cdot \alpha_F |L|$ ◀

Proof. Using Lemma 4.14, we have

$$\mathbb{E}[\sum_{f, p : L(p) = f} \text{cost}(p, f)] = \sum_{p \in P} \Pr[L(p) \in \mathcal{B}_\mathcal{T}] \text{cost}(p, L) \leq \alpha_F \text{cost}(P, L).$$

[6] Note that this definition of badly-cut is slightly different from that of Chapter 3.

Using Markov's inequality, with probability $1 - \pi/3$ the first bullet of the lemma holds. For the same reason, the second bullet hold with probability $1 - \pi/3$ as well. Similarly, $\mathbb{E}[|\mathcal{B}_\mathcal{T}|] = \sum_{f \in L} \Pr[f \in \mathcal{B}_\mathcal{T}] = \alpha_F |L|$, so applying again Markov's inequality gives that the third bullet holds with probability $1 - \pi/3$. A union-bound concludes the proof. □

▶ **Lemma 4.16.** When $\alpha_F = \frac{\pi\alpha}{6}$, $\alpha_C = \frac{\alpha^3\pi^3}{144 d^3 \log^2 n}$ and Lemma 4.15 holds:

$$\mathrm{cost}_\mathcal{T}(P_\mathcal{T}, \mathrm{OPT}) \leq \frac{\alpha}{2}\mathrm{cost}(P, L) + \frac{O(d^9 \log^4 n)}{\alpha^6 \pi^6} \cdot \mathrm{cost}(P, \mathrm{OPT}). \quad ◀$$

Proof. We start by showing different bounds for $\mathrm{cost}_\mathcal{T}(c, \mathrm{OPT})$, according to whether c is badly-cut or not.

When a client p is not badly-cut, we directly have that: $\mathrm{dist}_\mathcal{T}(p, \mathrm{OPT}) \leq \frac{d\sqrt{d} \cdot \mathrm{dist}(p, \mathrm{OPT})}{\alpha_C}$, since the lowest common ancestor of p and $\mathrm{OPT}(p)$ has diameter at most $\frac{d\sqrt{d} \cdot \mathrm{dist}(p, \mathrm{OPT})}{\alpha_C}$.

In the case where $p \in P_\mathcal{T}$ is badly-cut, we proceed differently. We use that $L(p)$ is not badly-cut as follows. Both p and $\mathrm{OPT}(p)$ are contained in the ball $\beta(L(p), \mathrm{dist}(p, L) + \mathrm{dist}(p, \mathrm{OPT}))$, since $\mathrm{dist}(L(p), \mathrm{OPT}) \leq \mathrm{dist}(p, L) + \mathrm{dist}(p, \mathrm{OPT})$. Let $i = \lceil \log(\mathrm{dist}(p, L) + \mathrm{dist}(p, \mathrm{OPT})) \rceil$. Since $L(p)$ is not badly-cut, the ball $\beta(L(p), 2^i)$ contains p and $\mathrm{OPT}(p)$ and is cut at level at most $i + \log(d \log n / \alpha_F)$. Hence, $\mathrm{dist}_\mathcal{T}(p, \mathrm{OPT}) \leq \frac{d^{3/2} \log n}{\alpha_F} 2^i \leq \frac{2d^{3/2} \log n}{\alpha_F} \cdot (\mathrm{dist}(p, L) + \mathrm{dist}(p, \mathrm{OPT}))$.

Since $\mathrm{cost}_\mathcal{T}(p, \mathrm{OPT}) = \mathrm{dist}_\mathcal{T}(p, \mathrm{OPT})^2$, this implies that

$$\begin{aligned}
\mathrm{cost}_\mathcal{T}(p, \mathrm{OPT}) &\leq \frac{4d^3 \log^2 n}{\alpha_F^2} \cdot (2\mathrm{cost}(p, L) + 2\mathrm{cost}(p, \mathrm{OPT})) \\
&\leq \frac{8d^3 \log^2 n}{\alpha_F^2} \cdot (\mathrm{cost}(p, L) + \mathrm{cost}(p, \mathrm{OPT}))
\end{aligned}$$

Hence, we have that:

$$\begin{aligned}
\mathrm{cost}_\mathcal{T}(P_\mathcal{T}, \mathrm{OPT}) &= \sum_{p \in P_\mathcal{T} : p \text{ badly-cut}} \mathrm{cost}_\mathcal{T}(p, \mathrm{OPT}) + \sum_{p \in P_\mathcal{T} : p \text{ not badly-cut}} \mathrm{cost}_\mathcal{T}(p, \mathrm{OPT}) \\
&\leq \sum_{p \in P_\mathcal{T} : \text{ badly-cut}} \frac{8d^3 \log^2 n}{\alpha_F^2} \cdot (\mathrm{cost}(p, \mathrm{OPT}) + \mathrm{cost}(p, L)) \\
&\quad + \sum_{p \in P_\mathcal{T} : p \text{ not badly-cut}} \frac{d^3 \mathrm{cost}(p, \mathrm{OPT})}{\alpha_C^2} \\
&\leq \left(\frac{8d^3 \log^2 n}{\alpha_F^2} + \frac{d^3}{\alpha_C^2} \right) \cdot \mathrm{cost}(P, \mathrm{OPT}) + \sum_{p \in P : \text{ badly-cut}} \frac{8d^3 \log^2 n}{\alpha_F^2} \cdot \mathrm{cost}(p, L).
\end{aligned}$$

Using now Lemma 4.15, we get:

$$\begin{aligned}
&\mathrm{cost}_{\mathcal{T}}(P_{\mathcal{T}}, \mathrm{OPT} \cup \mathcal{B}_{\mathcal{T}}) \\
\leq &\left(\frac{8d^3 \log^2 n}{\alpha_F^2} + \frac{d^3}{\alpha_C^2}\right) \cdot \mathrm{cost}(P, \mathrm{OPT}) + \frac{8d^3 \log^2 n}{\alpha_F^2} \cdot \frac{3\alpha_C}{\pi} \mathrm{cost}(P, L) \\
\leq &\frac{\alpha}{2} \mathrm{cost}(P, L) + \frac{O(d^9 \log^4 n)}{\alpha^6 \pi^6} \cdot \mathrm{cost}(P, \mathrm{OPT}).
\end{aligned}$$

□

4.5.2 The private k-means algorithm

That lemma shows that it is enough to compute the optimal solution on $P_{\mathcal{T}}$, and add to it the centers of $\mathcal{B}_{\mathcal{T}}$ which can be done by an algorithm similar to the one for k-Median:

Algorithm 7 DP-bicriteria-kMeans(P, L)

1: Compute a shifted quadtree $\mathcal{T}$. Let r be the root of $\mathcal{T}$.
2: Compute the instance $P_{\mathcal{T}}$, $w = \mathrm{MakePrivate}(\mathcal{T}, P_{\mathcal{T}})$.
3: Compute $s = \mathrm{DynamicProgram\text{-}KMeans}(\mathcal{T}, w, k, r)$
4: Use the dynamic program table to find a solution S with cost s
5: **Return** $S \cup \mathcal{B}_{\mathcal{T}}$

The algorithm DynamicProgram-KMeans is exactly the same as DynamicProgram-KMedian (Algorithm 3), except that it returns $w(c) \cdot \mathrm{diam}(c)^2$ at step 3, to fit the k-means cost. We can now turn to the proof of Lemma 4.11, to show the guarantees ensured by this algorithm.

Proof of Lemma 4.11. We start by showing the quality of approximation. As for k-median, the solution S computed at step 5 of Algorithm 7 is optimal for $P_{\mathcal{B}}$ in the quadtree metric with the additional noise. Hence, its cost verifies $\mathrm{cost}(P_{\mathcal{T}}, S) \leq \mathrm{cost}_{\mathcal{T}}(P_{\mathcal{T}}, S) \leq \mathrm{cost}_{\mathcal{T}}(P_{\mathcal{T}}, \mathrm{OPT}) + \frac{kd^2 \log^2 n}{\varepsilon} \cdot \Lambda^2$.

Now, with probability $1 - \pi$ Lemma 4.15 holds. In that case, using Lemma 4.16, the cost of $P_{\mathcal{T}}$ is at most

$$\mathrm{cost}(P_{\mathcal{T}}, S) \leq \frac{\alpha}{2} \cdot \mathrm{cost}(P, L) + \frac{O(d^9 \log^4 n)}{\alpha^6 \pi^6} \cdot \mathrm{cost}(P, \mathrm{OPT}) + \frac{kd^2 \log^2 n}{\varepsilon} \cdot \Lambda^2.$$

Moreover, Lemma 4.15 ensures that $\sum_{f \in \mathcal{B}_{\mathcal{T}}} \sum_{c : L(c) = f} \mathrm{cost}(c, f) \leq \frac{\alpha}{2} \mathrm{cost}(P, L)$. Hence, combining those bounds concludes the lemma.

We now turn to the privacy guarantee. $\mathcal{T}$ is computed obliviously to the data. Hence, when P changes by one point, $P_{\mathcal{T}}$ changes by at most one point as well – depending whether this point is served by a badly-cut center in L. As for k-median, Step 3 of the algorithm therefore ensures that the solution computed at step 5 is ε-DP. □

4.5.3 Going from $(1+\alpha)k$ centers to k

In this last section, we show how to get a true solution to k-means, removing the extra αk centers.

For that, we can use the reverse greedy algorithm of Chrobak et al. [46].[7] This algorithm starts from a set of centers, and iteratively removes the one leading to the smallest cost increase, until there are k centers remaining. It can be implemented in a private manner as follows: let S be a set of $O(k)$ centers computed privately. For any center s of S, let $w(s)$ to be the size of S's cluster, plus a Laplace noise $\mathrm{Lap}(1/\varepsilon)$. Let P_S be the resulting instance. Informally, any solution on P_S induces a solution with similar cost on P, with an additive error $\pm\ \mathrm{cost}(P,S) + k\cdot\Lambda^2/\varepsilon$ – see Lemma 4.18. Further, since S is private, P_S is private as well – see Lemma 4.17.

On the weighted instance P_S, the reverse greedy algorithm finds a solution with k centers that is an $O(\log k)$-approximation of the optimal solution $\mathcal{A}$ for that instance, using that P_S contains $O(k)$ distinct points. This is Theorem 2.2 in [46].

Now, the optimal solution $\mathcal{A}$ on P_S has cost in P at most $\mathrm{OPT} + \mathrm{cost}(P,S) + k\Lambda^2/\varepsilon$, by Lemma 4.18. Hence, combined with Theorem 4.12, $\mathcal{A}$ has cost $\mathrm{cost}(P,\mathcal{A}) \leq \mathrm{poly}(d,\log n,1/\alpha)\cdot\mathrm{OPT} + kd^2\log^2 n\log k/\varepsilon^2\Lambda^2$. The solution computed by the reversed greedy has therefore cost at most $\mathrm{poly}(d,\log n,1/\alpha)\cdot\mathrm{OPT} + kd^2\log^2 n\log^2 k/\varepsilon^2\Lambda^2$.

Before formalizing the argument, we show the two crucial lemmas.

▶ **Lemma 4.17.** If solution S is computed via an ε-DP algorithm, then the algorithm computing P_S is 2ε-DP. ◀

Proof. Fix some solution S. By properties of the Laplace mechanism (see 4.2), for any center s of S the value of $w(s)$ is computed on s's cluster is ε-DP. Since all clusters are disjoint, the instance P_S is computed in an ε-DP way.

Now, S is not fixed but given privately to the algorithm. By composition, the algorithm that computes P_S is 2ε-DP. □

▶ **Lemma 4.18.** Let S be any solution, and P_S computed as described previously. With high probability, for any set of k centers T, $|\mathrm{cost}(P,T) - \mathrm{cost}(P_S,T)| \leq \frac{1}{2}\cdot\mathrm{cost}(P_S,T) + 10\mathrm{cost}(P,S) + k\log n\Lambda^2/\varepsilon$. ◀

Proof. Without the addition of a Laplace noise, the cost difference between the two solution can be bounded using Lemma 1.2.

For any point $p \in P$ served by some center $s \in S$, we can apply Lemma 1.2 with

[7] Although the algorithm is stated only for k-median, its adaptation to k-means is straightforward using Lemma 1.2.

$\varepsilon = 1/2$ (and $z = 2$), to get:

$$|\text{cost}(p, T) - \text{cost}(s, T)| \leq \frac{1}{2} \cdot \text{cost}(s, T) + 10\text{cost}(p, s).$$

Moreover, w.h.p the total noise added is smaller than $k \log n/\varepsilon$, hence contributes at most $k \log n\Lambda^2/\varepsilon$ to the cost. Summing over all p concludes the proof. □

▶ **Lemma 4.19.** Let $\mathcal{S}$ be the solution computed by Theorem 4.12, and $P_{\mathcal{S}}$ the instance computed as described previously. Applying the reverse greedy algorithm on instance $P_{\mathcal{S}}$ is 2ε-DP and yields a solution with k centers and cost at most $\text{poly}(d, \log n, 1/\alpha) \cdot \text{OPT} + kd^2 \log^2 n \log^2 k/\varepsilon^2\Lambda^2$ ◀

Proof. First, the algorithm is 2ε-DP, as shown by Lemma 4.17.

Second, let $\mathcal{A}$ be the optimal solution on the instance $P_{\mathcal{S}}$, and OPT be the optimal cost for the full set of points P. Applying Lemma 4.18, we get:

$$\begin{aligned}\text{cost}(P_{\mathcal{S}}, \mathcal{A}) &\leq \text{cost}(P_{\mathcal{S}}, \text{OPT}) \\ &\leq 2\text{cost}(P, \text{OPT}) + 20\text{cost}(P, \mathcal{S}) + 2k \log n\Lambda^2/\varepsilon \\ &= \text{poly}(d, \log n, 1/\alpha) \cdot \text{OPT} + kd^2 \log^2 n \log k/\varepsilon^2\Lambda^2.\end{aligned}$$

We can now bound the cost of the solution computed by the reverse greedy algorithm. As $P_{\mathcal{S}}$ is made of $O(k)$ many distinct points, the reverse greedy computes a solution $\tilde{\mathcal{S}}$ with cost at most $\text{cost}(P_{\mathcal{S}}, \tilde{\mathcal{S}}) = O(\log k)\text{cost}(P_S, \mathcal{A})$. Applying again Lemma 4.18, we get

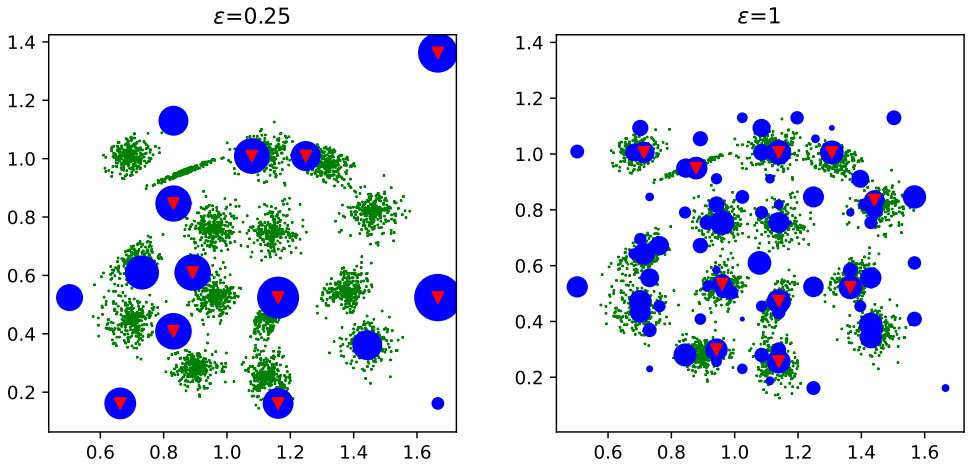
$$\begin{aligned}\text{cost}(P, \tilde{\mathcal{S}}) &\leq \frac{3}{2} \cdot \text{cost}(P_S, \tilde{\mathcal{S}}) + 10\text{cost}(P, \mathcal{S}) + k \log n\Lambda^2/\varepsilon \\ &= \text{poly}(d, \log n, 1/\alpha) \cdot \text{OPT} + kd^2 \log^2 n \log k/\varepsilon^2\Lambda^2,\end{aligned}$$

which concludes the proof. □

4.6 Empirical Evaluation for k-Median

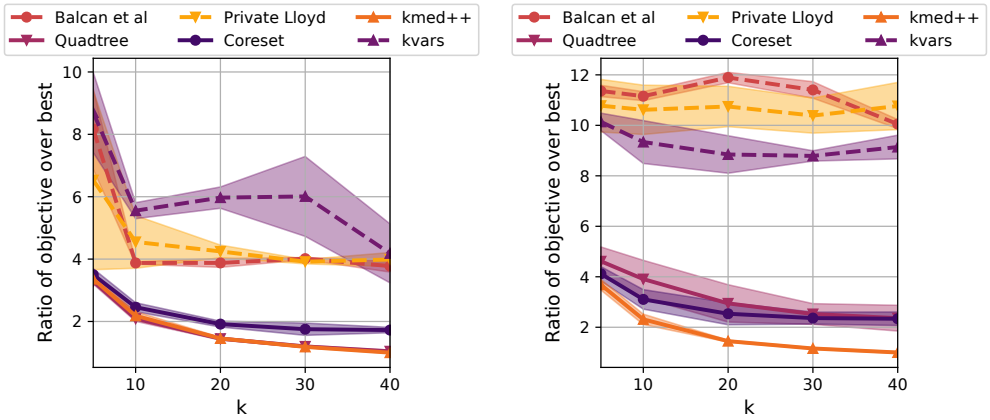
In this section, we present an empirical evaluation of our algorithm for the k-median objective. To the best of our knowledge, this is the first comprehensive empirical evaluation of private k-median algorithms as the majority of of experimental results has previously focused on k-means. All datasets used here are publicly-available.

Datasets. We used the following well known, real-world datasets from the UCI Repository [66] that are standard in clustering experiments `SKYNTYPE` [27] ($n = 245057, d = 4$), `SHUTTLE` [66] ($n = 58000, d = 9$), `COVERTYPE` [28] ($n = 581012, d = 54$) and `HIGGS` [19] ($n = 11000000$, $d = 28$). Finally, we use a publicly available synthetic datasets `SYNTHETIC` ($n = 5000$, $d = 2$) [83] for visualizing clustering results.



(a) (b)

Figure 4.1: Visualization of our algorithm. Original dataset in green. Leaves of the tree scaled by their weight in blue and centers found by our algorithm in red. (a) $\epsilon = 0.25$ and (b) $\epsilon = 1.0$.



(a) (b)

Figure 4.2: Comparison of algorithms on (a) `SKYNTYPE` and (b) `SHUTTLE`, with $\varepsilon = 0.5$.

Experimental details To simplify the stopping condition of Algorithm 2 we parameterize our algorithm by a depth parameter α and weight parameter β. We grow all of our trees to a max depth of αd and stop splitting the tree when $w(c) < \frac{10\beta d}{\epsilon}$ instead of $2d \log n/\epsilon$. This threshold was chosen to decrease the chance of potentially splitting empty cells multiple times and does not affect the privacy properties of the mechanism. The implementation for building the tree embedding was done using C++ in a large-scale distributed infrastructure. The dynamic program for solving the optimization problem in the tree was done in a single machine.

For all algorithms we report the average of 10 runs. We varied the number of centers, k, from 5 to 40, and the parameter ϵ from 0.25 to 1.

Non-private baseline We compare the results of our algorithm against a non-private implementation of k-median++ [10] (**kmed++**) with 10 iterations of Lloyd's algorithm. Each iteration was done by optimizing the k-median objective exactly using Python's BFGS optimizer.

Private baselines To the best of our knowledge all private baselines for clustering algorithms have focused on the k-means problem. However, using a private 1-median algorithm, it is possible to adapt some of the prior work to solve the private k-median problem.

As a first step we implement a subroutine of the 1-median problem using the objective perturbation framework of [107]. The algorithm described in [107] requires a smooth loss function. We therefore modified the k-median objective to the $\frac{1}{\lambda}$-smooth k-median objective $f_\lambda \colon x \mapsto \|x\| + 2\lambda \log\left((1 + e^{-\frac{\|x\|}{\lambda}})/2\right)$ which converges to $\|x\|$ as $\lambda \mapsto 0$.

Given this tool we implemented the following algorithms.

- **Quadtree** : A version of Algorithm 4. After finding the centers using the tree, we ran 4 iterations of the Lloyd algorithm using the private 1-median implementation described above using at most 20k points from each cluster for the optimization step to allow it to fit in memory. We split the privacy budget ϵ uniformly: using $\epsilon/5$ to build the tree and $\epsilon/5$ per Lloyd's iteration. We tune the parameters $\alpha \in \{10, 12, 14\}$ and $\beta \in \{6, 8, 10\}$. The hyper-parameters for the 1-median solver were set to $\lambda = 0.2$ and γ to $0.01 * \sqrt{d}/n$ (γ is a bound on the gradient norm of the optimizer defined in [107]).
- **Private Lloyd**: a private implementation of Lloyd's algorithm. This algorithm has no approximation guarantee. The initial centers are chosen randomly in the space, and at each iteration, each point is assigned to the nearest center, and centers are recomputed using the private 1-median algorithm. We chose the number of iteration to be 7, as a tradeoff between the quality of approximation found and the privacy noise added. Here, the hyper-parameters for the 1-median solver were $\lambda = 1$ and $\gamma = 0.01\sqrt{d}/n$.
- **Balcan et al**: the private algorithm of [17]. This algorithm finds a small set of candidate center, and then runs a private local-search algorithm. To compute the candidate centers, the algorithm essentially projects randomly onto a $O(\log n)$-dimensional subspace, and discretize the space by taking a fine-grained grid. The solution computed has worst case cost at most $\log(n)^{3/2} \cdot \mathrm{OPT} + \mathrm{poly}(d, k, \log n)$. We modified the code available online [16] to adapt it to k-median, by using our 1-median implementation with $\lambda = 1$ and $\gamma = 0.01\sqrt{d}/n$.
- **kvars**: A private instantiation of the kvariates heuristic algorithm of [147], which is a private generalization of the k-means++ algorithm [10]. The algorithm uses a sub-routine that splits data into computation nodes. We hash each point using SimHash [42] to assign them to one of 500 computation nodes.
- **Coreset**[8]: A heuristic algorithm for private k-means clustering that creates a

[8] https://ai.googleblog.com/2021/10/practical-differentially-private.html

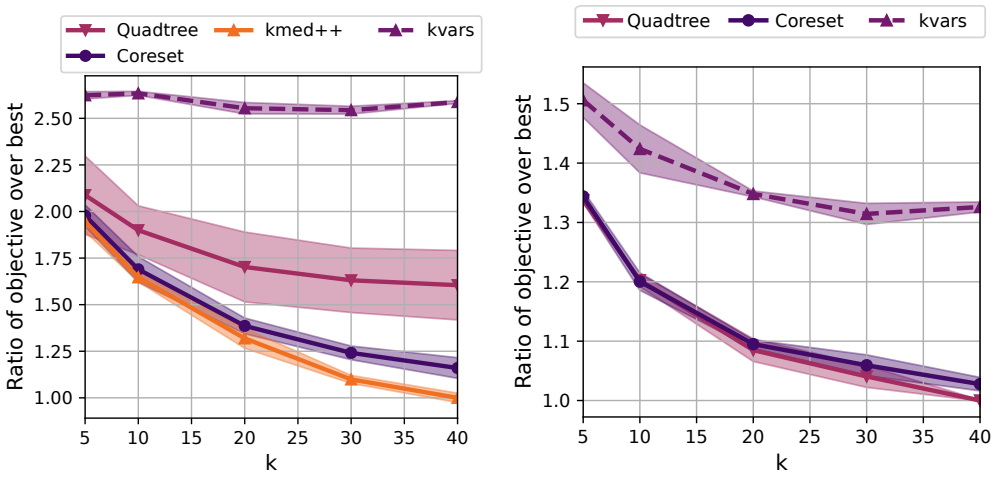


Figure 4.3: Objective function as a function of k datasets (a) `COVERTYPE` and (b) `HIGGS`, for $\varepsilon = 0.5$.

coreset via recursive partitioning using locality sensitive hashing. We modified the heuristic to handle k-median with our private 1-median implementation, with $\lambda = 0.2$ and $\gamma = 0.01\sqrt{d}/n$.

Other baselines not evaluated We describe here other potential candidate baselines which we found not feasible to compare against. Since our work focuses on scalability, we do not compare against algorithms with impractically large running times like the algorithm of [153, 87] which have state-of-the-art theoretical approximations but that have not previously been implemented.[9] We also did not compare against [146], as it lacks guarantees for k-median and the baseline [17] showed comparable performance with their algorithm. Finally, we do not compare with the heuristic GUPT [140] as it does not provide an explicit aggregation procedure for k-median.

Results We begin by showing a visualization of our algorithm on the `SYNTHETIC` dataset of 2 dimensions to give intuition on the effect of privacy on constructing the tree embedding. Figure 4.1(a) shows the centers returned by our algorithm for $\epsilon = 0.25$ and $\epsilon = 1$. This example illustrates that that as ϵ increases, our tree embedding captures the geometry of the dataset more correctly.

We now discuss the quality of the clusterings returned by each algorithm. We begin evaluating all baselines on the small datasets `SKYNTYPE` and `SHUTTLE`. Figure 4.2 shows the quality of each algorithm for $\epsilon = 0.5$. The plots are normalized by the best clustering objective. There are several points worth noting in this plot. First, the performance of the Balcan et al. algorithm which has the best approximation guarantees is consistently outperformed by our algorithm and the coreset algorithm. Second, notice that on `SKYNTYPE` our algorithm achieves a performance that is essentially the

[9] Private communication with the authors of [87] confirmed that there is no practical implementation of this algorithm available.

same as the non-private baseline.

For the large datasets `COVERTYPE` and `HIGGS`, it was impossible for us to run the Balcan et al. approach. On `HIGGS`, even the baseline **kmed++** did not finish. Therefore, we only compare our algorithm against the coreset and kvars baselines. Figure 4.3 shows the results. Here we see that our algorithm has the strongest performances on HIGGS while on `COVERTYPE` it is comparable to the coreset heuristic and slightly worse for large k.

We compare only the quality of the solutions computed and not the running time, as the parallel implementation has a large overhead and it never runs really fast. However, our implementation does run and provide apparently good results on large scale datasets on which other private algorithms do not terminate or give really poor results – to the notable exception of the Coreset algorithm, which does not enjoy theoretical guarantees.

In summary, our empirical evaluation confirms that our approach, which is the only method that has both theoretical performance guarantees and can be made to scale to large datasets, consistently performs well on a wide variety of examples, achieving accuracy much higher than the worst case analysis would indicate.

4.7 Conclusion

We have presented practical and scalable differentially private algorithms for k-median with worst case approximation guarantees. Although their worst-case performance is worse than state of the art methods, they are parallelizable, easy to implement in distributed settings, and empirically perform better than any other algorithm with approximation guarantees. Furthermore, we have presented an extension of those algorithms to the k-means objective, with a theoretical analysis. A natural open question is to close this gap between theory and practice: finding scalable methods that have even better worst-case guarantees.

Part II

Coreset and Sketches for Clustering

Chapter 5

Presentation of our Results and Overview of the Techniques

This chapter is an extensive introduction to the coresets results we show in this part of the thesis. We first introduce the specific questions we consider, and then present our results. Afterwards, we give a precise description of two types of coreset constructions: the first one due to Chen [44], from which we draw much inspiration, the second to Feldman and Langberg [76], which is the basis ingredient for much of (if not all) the previous best coresets constructions. Finally, we will give a broader overview of the history of coreset studies.

5.1 Introduction

Our main concern and tool in this part is coreset for (k, z)-clustering, small sketches of the input for clustering. Formally:

▶ **Definition 5.1.** An ε-coreset for the (k, z)-clustering problem in a metric space (X, dist) for clients P is a weighted subset Ω of P with weights $w : \Omega \to \mathbb{R}_+$ such that, for any set $\mathcal{S} \subset X$, $|\mathcal{S}| = k$,

$$|\sum_{p \in P} \mathrm{cost}(p, \mathcal{S}) - \sum_{p \in \Omega} w(p)\mathrm{cost}(p, \mathcal{S})| \leq \varepsilon \cdot \sum_{p \in P} \mathrm{cost}(p, \mathcal{S}).$$

◀

In words, an ε-coreset is a (weighted) set Ω that has same cost as the input P for *any* possible solution $\mathcal{S}$ (up to a $(1 \pm \varepsilon)$ factor). As explained in Chapter 1, a small coreset allows to "turn big data into tiny data", which has numerous advantages: compressing

the input in order to be able to store it, or speeding-up any algorithm by running it on the coreset rather than the full dataset.

The study of coreset for clustering started with the work of Har-Peled and Mazumdar [96]. They gave a construction based on snapping points to a grid – essentially what we described in the introduction. This is specifically tailored to Euclidean spaces, and has a prohibited exponential dependency in the dimension.

The first breakthrough in coreset history is due to Chen [44], who introduced sampling in the coreset toolbox. The basic approach is to devise a non-uniform sampling distribution that picks points according to their cost contribution in an arbitrary constant factor approximation. In a nutshell, the analysis shows that, for a given set $\mathcal{S}$ of k centers, it happens with high probability that the sampled instance Ω with appropriate weights has roughly the same cost as the original instance, i.e. $\text{cost}(\Omega, \mathcal{S}) \in (1 \pm \varepsilon)\text{cost}(X, \mathcal{S})$. Then, to show that the set Ω is an ε-coreset, it is necessary to take a union-bound over these events for all possible set of k centers. We give more details on Chen's work in Section 5.2.

All the current best analysis rely on those ideas: sample proportionate to the cost in some solution, and use a union-bound to show that the sample preserves the cost of any solution. Bounding the size of the union-bound is the main hurdle faced by this approach: indeed, there may be infinitely many possible set of centers, as in Euclidean spaces.

The state-of-the-art analysis relies on VC-dimension to overcome this hurdle. Informally, given a ground set X and a set $\mathcal{R}$ of subsets of X, the VC dimension of $(X, \mathcal{R})$ is a complexity measure that is directly related to how well a uniform sample of X preserves relative size of sets from $\mathcal{R}$. For coresets, we are looking to preserve the number points whose closest center of $\mathcal{S}$ is at distance R, for any R: if a sample preserves those, it is indeed a coreset. It is henceforth not surprising to see VC dimension helping in the design of coresets. In metric spaces where the VC dimension of the aforementioned sets[1] is D, it can be shown that taking $O_{\varepsilon,z}(k \cdot D \log k)$ samples yields a coreset [78], although tighter bounds are achievable in certain cases. For instance, in d dimensional Euclidean spaces D is in $O(kd \log k)$ [12], which would yield coresets of size $O_{\varepsilon,z}(k^2 \cdot d \log^2 k)$, but Huang and Vishnoi [100] and Braverman, Jiang, Krauthgamer and Wu [34] showed the existence of a coreset with $O(k \cdot \log^2 k \cdot \varepsilon^{-2z-2})$ points.

This VC-dimension based analysis was proven powerful in various metric spaces, such as doubling spaces by Huang, Jiang, Li and Wu [98], graphs of bounded treewidth by Baker, Braverman, Huang, Jiang, Krauthgamer, Wu [15] or the shortest-path metric of a graph excluding a fixed minor by Braverman et al. [34]. However, range spaces of even heavily constrained metrics do not necessarily have small VC-dimension (e.g. bounded doubling dimension does not imply bounded VC-dimension or vice versa [98, 120]), and applying previous techniques requires heavy additional machinery to adapt the VC-dimension approach to them. Moreover, the bounds provided are far from the

[1] To be more precise, Feldman et al. [78] consider weighted distances: not simply points at distance R to $\mathcal{S}$, but the points whose weighted distance is at most R: $\{p : \text{dist}(p, \mathcal{S}) \leq w_p R\}$. In Chapter 9, we generalize their result to the case where the weights are 1.

bound obtained for Euclidean spaces: their dependency in k is at least $\Omega(k^2)$, leaving a significant gap to the best lower bounds of $\Omega(k)$. We thus ask:

> ▶ **Question 3.** Is it possible to design coresets whose size are near-linear in k, and with no dependency in $|P|$, for doubling metrics, minor-free metrics, bounded-treewidth metrics? Are the current roadblocks specific to the analysis through VC-dimension, or inherent to the problem? ◀

These questions are the subject of Chapter 6 and Chapter 7. To answer positively, we present in Chapter 6 a new framework to analyze importance sampling. Its analysis stems from first principles, and it can be applied in a black-box fashion to any metric space that admits an *approximate centroid set* (see Definition 5.2) of bounded size. We show in Chapter 7 that all previously mentioned spaces satisfy this condition, and our construction improves on the best-known coreset size. More precisely, we recover (and improve) all previous results for (k, z)-clustering such as Euclidean spaces, ℓ_p spaces for $p \in [1, 2)$, finite n-point metrics, while also giving the first coresets with size near-linear in k and ε^{-z} for a number of other metrics such as doubling spaces, minor free metrics, and graphs with bounded treewidth.

One application of small coresets is the design *sublinear* algorithms, that do not have access to the full input. If only allowing access to a few random input points, we show in Chapter 10 how to compute a $(1 + \varepsilon)$-approximation to $(1, z)$-clustering – the generalization of mean to arbitrary powers. It is well known that the mean of $1/\varepsilon^2$ points taken uniformly at random is a $(1 + \varepsilon)$-approximation to the mean of the full input: we generalize this result to arbitrary powers. This is particularly helpful in practice, as the running time of the algorithm does not depend on n at all. We complement our theoretical study with experiments, showing that the quality of the solution computed on our small sample is indeed good and that the running time is way lower than standard algorithms.

Although numerous great work focused on improving the coreset construction or on applying them, our understanding of coreset lower bounds is limited, and there is a significant gap between the best upper and lower bounds on the possible coreset size. For example, even for Euclidean k-means, nothing beyond the trivial $\Omega(k)$ lower bound is known. We thus ask:

> ▶ **Question 4.** What is the best coreset size one could hope for? ◀

In Chapter 8, we settle the problem in general discrete metrics, by drawing a strong tie between coreset and concentration inequalities for sum of random variables. Those are already key to our coreset constructions: we complement those results by showing that the tightness of Chernoff bounds implies that our upper bounds are actually tight.

Interestingly – and somewhat frustratingly – all the state-of-the-art results are heavily reliant on randomization, and the aforementioned concentration inequalities for random variables. Concretely, to achieve a success probability $1 - \pi$, coreset constructions require $\Omega_{k,\varepsilon}(\log(1/\pi))$ points. In particular, to construct a coreset that

offers the desired guarantees with *high probability* (i.e., probability $1 - \text{poly}(1/n)$), the dependency in n strikes back. At the same time, we know from the randomized constructions that there exist coresets with size independent on n. Designing deterministic coreset is interesting from both practical and theoretical standpoints: from a practical perspective, it makes sure that the results can be reproduced;[2] from a theoretical standpoint, understanding the power of randomness in algorithms is a very basic question and a recurring theme in theoretical computer science: at what cost can we make an algorithm deterministic? We ask:

> ▶ **Question 5.** Is it possible to deterministically construct k-median and k-means coreset of size matching the best randomized results? ◀

Finding deterministic coresets of small size is explicitly asked as an open question in recent surveys on the coreset literature [75, 143].

To the best of our knowledge, no deterministic coreset construction has been claimed for other spaces than Euclidean. The best determinstic constructions for Euclidean k-median (and more generally, (k, z)-Clustering) yields coresets of size $O(\text{poly}(k) \cdot \varepsilon^{-(d+O(z))})$ [95], which compares very unfavourably to the best randomized result of size $\tilde{O}(k\varepsilon^{-2-\max(2,z)})$ we show in Chapter 7. A notable exception is for k-means, where it is possible to leverage the algebraic structure of the problem to construct coresets of size $k^{\varepsilon^{-2}\log(1/\varepsilon)}$ [78]. Unfortunately, this construction is very specific to k-means, requires additional effort to be implemented deterministically and cannot be used for k-median, and importantly, is still very far from the best known randomized bounds.

Suppose we were to relax the problem of deterministically computing a coreset and aim for a Las-Vegas algorithm.[3] For closely related problems such as locality sensitive hashing (see Ahle [4] and references therein), this is achievable. For clustering, all known algorithms are Monte Carlo algorithms. The main barrier to obtaining a Las Vegas algorithm (a challenging and interesting open problem as well) is that we currently do not know how to verify that a candidate set of points is indeed a valid coreset without evaluating all possible choices of k centers and verifying that the cost of each set of k centers is preserved. This stands in stark contrast with other related techniques for which derandomization techniques have been designed such as the Johnson Lindenstrauss lemma [72], matrix approximation [31], or regression [149].

As a side result, we also present deterministic dimension reduction for (k, z)-Clustering in Euclidean space. Dimension reduction is another important class of sketches: it allows to avoid some curse of dimensionality, by reducing the ambient dimension to $O(\varepsilon^{-2}\log k)$. Yet, those results are based on random projection (see Makarychev, Makarychev and Razenshteyn [129] and Becchetti, Bury, Cohen-Addad, Grandoni and Schwiegelshohn [25]). We present a deterministic procedure with similar guarantees in Chapter 9.

[2] Obtaining similar probabilistic outcomes over different architectures, computers, programming languages is a challenge.

[3] Recall that a Las Vegas algorithm returns a solution in expected polynomial time, whereas a Monte Carlo algorithm only returns a solution with a small probability of failure.

5.1.1 Results Presented in the Part

We now state more precisely our answers to the questions previously asked. First, we present our framework for constructing coresets. It requires the existence of a particular discretization of the set of possible centers, as described in the following definition.

▶ **Definition 5.2.** Let (X, dist) be a metric space, $P \subseteq X$ a set of clients and two positive integers k and z. Let $\varepsilon > 0$ be a precision parameter. Given a set of $O(k)$ centers $\mathcal{A}$, a set $\mathbb{C}$ is an *$\mathcal{A}$-approximate centroid set* for (k, z)-clustering on P if it satisfies the following property.
For every set of k centers $\mathcal{S} \in X^k$, there exists $\tilde{\mathcal{S}} \in \mathbb{C}^k$ such that for all points $p \in P$ that satisfies $\mathrm{cost}(p, \mathcal{S}) \leq \left(\frac{8z}{\varepsilon}\right)^z \mathrm{cost}(p, \mathcal{A})$, or $\mathrm{cost}(p, \tilde{\mathcal{S}}) \leq \left(\frac{8z}{\varepsilon}\right)^z \mathrm{cost}(p, \mathcal{A})$, it holds

$$|\mathrm{cost}(p, \mathcal{S}) - \mathrm{cost}(p, \tilde{\mathcal{S}})| \leq \frac{\varepsilon}{z \log(z/\varepsilon)} \left(\mathrm{cost}(p, \mathcal{S}) + \mathrm{cost}(p, \mathcal{A})\right). \qquad \blacktriangleleft$$

This definition is slightly different from Matousek's one [134], in that we seek to preserve distances only for interesting points, and allow an error $\varepsilon \mathrm{cost}(p, \mathcal{A})$. This difference is crucial in some of our applications. Our main theorem is:

▶ **Theorem 5.3.** Let (X, dist) be a metric space, $P \subseteq X$ a set of clients with n distinct points and two positive integers k and z. Let $\varepsilon > 0$ be a precision parameter, and $\pi \in (0, 1)$. Let also $\mathcal{A}$ be a constant-factor approximation for (k, z)-clustering on P.
Suppose there exists an $\mathcal{A}$-approximate centroid set $\mathbb{C}$ for (k, z)-clustering on P. Then, there exists an algorithm running in time $O(n)$ that constructs with probability at least $1 - \pi$ a coreset of size

$$O\left(\frac{2^{O(z \log z)} \cdot \log^4 1/\varepsilon}{\min(\varepsilon^2, \varepsilon^z)} \left(k \log |\mathbb{C}| + \log\log(1/\varepsilon) + \log(1/\pi)\right)\right)$$

with positive weights for the (k, z)-clustering problem. $\blacktriangleleft$

When applying this theorem to particular metric spaces, the running time is dominated by the construction of the constant-factor approximation $\mathcal{A}$, which can be done for instance in $\tilde{O}(k|P|)$ given oracle access to the distances using Lemma 1.3.

If one wishes to trade a factor ε^{-z} for a factor k, we also present coresets of size $k \cdot 2^{O(z \log z)} \frac{\log^3(1/\varepsilon)}{\varepsilon^2} \left(k \log |\mathbb{C}| + \log(1/\pi)\right)$, as explained in Section 6.6.

Theorem 5.3 reduces the construction of coreset to showing the mere existence of an approximate centroid set – crucially, it is not necessary to compute the approximate centroid set, only to know its existence. In Chapter 7, we illustrate the power of this theorem by applying it to several metric spaces, achieving the following (simplified) size bounds (we ignore $\mathrm{poly}\log(1/\varepsilon)$ and $2^{O(z \log z)}$ factors): let $\Gamma = \min(\varepsilon^{-2} + \varepsilon^{-z}, k\varepsilon^{-2})$.

- $O\left(\Gamma \cdot k \log n\right)$, since general discrete metric spaces have an approximate centroid set of size n (the full space). This improves on the bound from Feldman and Langberg [76] $O\left(\varepsilon^{-2z} k \log k \log n\right)$.
- $O\left(\Gamma \cdot k \left(d + \log k\right)\right)$ for metric spaces with doubling dimension d. This improves over the $O(k^3 d \varepsilon^{-2})$ from Huang et al. [98]. See Corollary 7.6.
- $O\left(\Gamma \cdot k\varepsilon^{-2} \cdot \log k\right)$ for Euclidean spaces, see Corollary 7.20. This improves on the recent result from Huand and Vishnoi [100], who achieve $O\left(\varepsilon^{-2z-2} k \log^2 k\right)$.
- $O\left(\Gamma \cdot k \left(\log^2 k + \frac{\log k}{\varepsilon^4}\right)\right)$ for a family of graphs excluding a fixed minor, see Corollary 7.15. This improves on Braverman et al. [34], whose coreset has size $\widetilde{O}(k^2/\varepsilon^4)$.
- $O\left(\Gamma \cdot k \left(\log^2 k + \frac{\log k}{\varepsilon^3}\right)\right)$ for Planar Graphs, which is a particular family excluding a fixed minor for which we can save a $1/\varepsilon$ factor and present a simpler, instructive proof.
- $O\left(\Gamma \cdot k \left(t + \log k\right)\right)$ in graphs with treewidth t, see Corollary 7.9. This improves upon the work of Baker et al. [15], that construct coreset with size $\widetilde{O}(k^2 t/\varepsilon^2)$.
- $O(k\varepsilon^{-2z} \cdot \min(d, \varepsilon^{-2} \log k))$ in $\mathbb{R}^d$ with ℓ_p distance, for $p \in [1, 2)$, see Corollary 7.21. This improves on Huang and Vishnoi [100], who presented a coreset of size $O(k \log k \cdot \varepsilon^{-4z} \cdot \min(d, \varepsilon^{-2} \log k))$.

In Chapter 8, we settle the complexity of the problem for several cases. First, for finite n-point metrics, we prove the following theorem.

> ▶ **Informal Theorem (See Theorem 8.2).** For any $0 < \varepsilon < 1/2$, k and n, there exists a finite n point metric such that any ε-coreset for (k, z)-clustering consists of at least $\Omega\left(\frac{k}{\varepsilon^2} \log n\right)$ points.
> For the k-median and k-means objective, this matches the upper bounds up to polylog$(1/\varepsilon)$ factors. ◀

Our result improves over the $\Omega(k\varepsilon^{-1} \log n)$ lower bound of Baker, Braverman, Huang, Jiang, Krauthgamer, and Wu [15].

A simple corollary of that theorem is that our bounds for metrics with doubling dimension D are tight as well:

> ▶ **Informal Corollary (See Corollary 8.10).** For any $0 < \varepsilon < 1/2$, k and D, there exists a metric space with doubling dimension D such that any ε-coreset for (k, z)-clustering consists of at least $\Omega\left(\frac{kD}{\varepsilon^2}\right)$ points.
> For the k-median and k-means objective, this matches the upper bounds up to $\log k$ and polylog$(1/\varepsilon)$ factors. ◀

In Chapter 9, we answer Question 5 and present a deterministic implementation of our framework for Euclidean spaces. We show the following theorem:

▶ **Informal Theorem (See Corollary 9.20 for a formal statement).** In the Euclidean space $\mathbb{R}^d$, there exist a deterministic algorithm running in deterministic time $n^2 \cdot k^{k \cdot \varepsilon^{-O(z)}} + n^{\varepsilon^{-O(z)}}$ that constructs an ε-coreset for (k, z)-clustering of size $k^2 \log^2 k \cdot \varepsilon^{-O(1)}$. ◀

An interesting feature of our construction is that it works for any space where the *uniform* VC-dimension of balls is bounded. Compared to the framework of Feldman and Langberg, this simplifies a lot: they need to bound the VC-dimension of a set of balls in a weighted metric, as we describe in Section 5.3. As mentioned e.g. in Baker et al. [15], bounding the uniform VC-dimension is much easier than its weighted counterpart. Hence, applying our techniques instead of Feldman and Langberg ones (maybe with randomization if one wants a faster algorithm) may yield simpler coresets, if one targets suboptimal size $\text{poly}(k, \varepsilon^{-1})$. In particular, we apply our algorithm to minor-excluded graphs, which yields a much simpler answer to Question 3 than the one we provide in Chapter 7 (although with a worse size).

To show that result in Euclidean spaces, one of the crux is to reduce the dimension: all the known techniques to reduce the dimension to $O_\varepsilon(k)$ or $O_\varepsilon(\log k)$ heavily require randomization. The only exception is for k-means, where principal component analysis can be used to reduce to $O_\varepsilon(k)$ dimensions: unfortunately, this does not carry over to other powers. Hence, our main technical contribution is to show the following theorem.

▶ **Informal Theorem (See Theorem 9.14 for a formal statement).** For (k, z)-clustering in Euclidean space, one can reduce the dimension to $O(\varepsilon^{-O(z)} \log k)$ in deterministic time $O\left(n^{\varepsilon^{-O(z)}} \cdot d\right)$ time. ◀

Finally, we apply coreset for $(1, z)$-clustering in order to compute very efficiently a near optimal solution to $(1, z)$-clustering, that we call the *power mean*. More precisely, we construct an algorithm that has low query complexity: instead of considering the full input, our algorithm is only allowed to access a few number of random input points. This number is the query complexity of the algorithm: the smaller, the less information it needs about the input dataset.

▶ **Informal Theorem (See Theorem 10.1 and Theorem 10.2 for formal statements).** There exists an algorithm with query complexity $\tilde{O}(\varepsilon^{-z-5} \log^2 1/\delta)$ that computes a $(1 + \varepsilon)$ approximate solution to the high dimensional power mean problem with probability at least $1 - \delta$.
Furthermore, any algorithm that computes with probability more than $4/5$ a $(1 + \varepsilon)$-approximation for a one-dimensional power mean has query complexity $\Omega(\varepsilon^{-z+1})$. ◀

5.2 Brief description of Chen's Coreset

It is enlightening to describe Chen's algorithm and analysis [44] in greater details, as it is our source of inspiration. The algorithm is stated for k-median in Algorithm 8. In words, the algorithm computes a constant factor approximation, then places rings around each center and takes as a coreset a uniform sample of each ring, with weight for a point p inversely proportional to the probability of sampling p.

Algorithm 8 Ke Chen's Coreset Algorithm

Compute a constant factor approximation $\mathcal{A}$ with centers $c_1, ..., c_k$ and clusters $C_1, ..., C_k$.
Let $\Delta := \frac{1}{|P|}\sum_{p\in P} \text{cost}(p, \mathcal{A})$ be the average cost of the solution: for all $j \in \{1, ..., \log(|P|/\varepsilon)\}$, let $R_{i,j} := \big\{p \in C_i : \text{dist}(p, c_i) \in [2^j\varepsilon\Delta, 2^{j+1}\varepsilon\Delta)\big\}$. Let also $R_{i,0} := \{p \in C_i : \text{dist}(p, c_i) < \varepsilon\Delta)\}$.
Let $\Omega_{i,j}$ be a uniform sample of size $\delta := \varepsilon^{-2}k\log(2n)$ from $R_{i,j}$, with weights $\frac{|R_{i,j}|}{\delta}$.
return $\Omega = \bigcup \Omega_{i,j}$

Let Ω be the outcome of Algorithm 8, and $\mathcal{S}$ be any solution: we wish to show that $|\text{cost}(P, \mathcal{S}) - \text{cost}(\Omega, \mathcal{S})| \leq \varepsilon\text{cost}(P, \mathcal{S})$ with high probability. For this, we proceed ring by ring, and aim at bounding

$$|\text{cost}(R_{i,j}, \mathcal{S}) - \text{cost}(\Omega_{i,j}, \mathcal{S})|. \tag{5.1}$$

Since rings partition the input (by definition, there is no point that pays more than $|P|$ times the average cost), providing a guarantee for all rings will be enough to conclude for the whole input. To deal with a ring, note that $\text{cost}(\Omega_{i,j}, \mathcal{S})$ is a sum of independent random variable: for $\ell \in \{1, ..., \delta\}$, let X_ℓ be the distance from the ℓ-th sampled point to $\mathcal{S}$. Then, we have

$$\text{cost}(\Omega_{i,j}, \mathcal{S}) = \sum_{\ell=1}^{\delta} \frac{|R_{i,j}|}{\delta} X_\ell,$$
$$\mathbb{E}[\text{cost}(\Omega_{i,j}, \mathcal{S})] = \text{cost}(R_{i,j}, \mathcal{S}).$$

Hence, Eq. (5.1) is the deviation between a sum of independent identically distributed random variables and its expectation. The go-to tool to bound this type of expression are Chernoff bounds and their variants. Without diving into details, here one can note that since we focus on the ring $R_{i,j}$, triangle inequality ensures that all X_ℓ have the same value, up to an additive $\pm 2^{j+1}\varepsilon\Delta$: in that case, one can show that if $\delta \geq \varepsilon^{-2}\log(2/\lambda)$, then it holds with probability $1 - \lambda$ that

$$|\text{cost}(R_{i,j}, \mathcal{S}) - \text{cost}(\Omega_{i,j}, \mathcal{S})| \leq \varepsilon|R_{i,j}|2^{j+1}\varepsilon\Delta \leq 2\varepsilon\text{cost}(R_{i,j}, c_i), \tag{5.2}$$

where the last inequality holds because all points in $R_{i,j}$ are at distance at least $2^j\varepsilon\Delta$ to c_i. With our choice of $\delta = \varepsilon^{-2}k\log(2n)$, the previous guarantee holds therefore with probability $1 - n^{-k}$.

Since rings partition the input, summing Eq. (5.2) over all rings concludes the approximation guarantee:

$$\begin{aligned}|\text{cost}(P,\mathcal{S}) - \text{cost}(\Omega,\mathcal{S})| &\leq \sum_{i=1}^{k}\sum_{j} 2\varepsilon\text{cost}(R_{i,j},c_i)\\ &\leq 2\varepsilon\text{cost}(P,C) = O(\varepsilon)\text{cost}(P,\mathcal{S}),\end{aligned}$$

where the last inequality uses that $\mathcal{A}$ is a constant-factor approximation. Hence, for a fixed solution $\mathcal{S}$, the coreset guarantee holds with probability $1-n^{-k}$. As there are n^k many possible solutions $\mathcal{S}$ (consisting of k centers picked among a metric with n points), a union-bound ensures that the coreset guarantee holds for any solution $\mathcal{S}$ with constant probability.

Finally, the size of the coreset is equal to the number of rings times δ, i.e., $O(k^2\varepsilon^{-2}\log^2 n)$. The running time is $O(nk)$ plus the time to compute $\mathcal{A}$.

The key ingredients of the proof are the following: group points such that each point pays roughly the same in any solution, to allows the use of concentration inequality, and union bound over all candidate solutions. In the discrete case, the union-bound is trivial: it is not as easy for the Euclidean case, or in restricted metrics, when one wishes to remove dependency in n.

Along this part, we will see several ideas to get rid on the dependency in n, by dealing separately with rings with $j > \log(1/\varepsilon)$. We will also see how to deal with rings not one-by-one but by groups, such as to limit the number of different uniform samples necessary. Finally, we will present ways of limiting the number of candidate solutions, such as to limit the lose incurred by the union-bound step.

5.3 VC-dimension: Coresets independent of the size of the input

Since it is at the heart of virtually all modern coresets construction before ours, we present briefly the sensitivity framework. We highlight here the key ideas, following Feldman's presentation [75]. This can be seen as a way to perform the previously described union-bound over all solutions for Euclidean spaces and other metrics.

A powerful hammer to bound the number of samples required for a given task is the notion of VC-dimension, that is a measure that is directly related to the size of a uniform sample that satisfies some desirable property. In particular, one may consider the following property – very similar to the coreset guarantee we are shooting for:

$$\forall\mathcal{S}, \left|\sum_{p\in P} w(p)f(p,\mathcal{S}) - \sum_{q\in\Omega} u(q)f(q,\mathcal{S})\right| \leq \varepsilon, \tag{5.3}$$

where P is the input set and w a weight on each point, Ω is a subset of P, u are weights

on the sample, and f may be an arbitrary function. If $f(p, \mathcal{S}) = \frac{\text{cost}(p,\mathcal{S})}{\sum w(q)\text{cost}(q,\mathcal{S})}$, then Eq. (5.3) is exactly a coreset guarantee.

The PAC learning theory [29, 121] provides a bound on the size of a uniform sample to satisfy Eq. (5.3), in the case where the weights w are uniform. To satisfy the equation with probability $1-\pi$, the bound depends polynomially on the maximum contribution of a point $\sup_{p\in P,\mathcal{S}} |f(p, \mathcal{S})|$, on the sum of weights $\sum_{p\in P} w(p)$, $1/\varepsilon, \log 1/\pi$ and on a measure of complexity similar to the VC dimension.

Formulated that way, the bound the sample depends on is actually not satisfying, as $\sum_{p\in P} w(p)$ depends on $|P|$. The key hindsight from Feldman and Langberg [76] is that one can save that factor by rescaling the weights and changing the functions f. In particular, for any weight $m(p)$, defining $g(p, \mathcal{S}) := \frac{w(p)f(x,\mathcal{S})}{m(p)}$ ensures that $\sum_{p\in P} m(p)g(p, \mathcal{S}) = \sum_{p\in P} w(p)f(p, \mathcal{S})$, but the sample size now depends on $\sup |g(p, \mathcal{S})|$ and $\sum m(p)$. Additionally, the sample is not uniform anymore, and the sample distribution needs to be proportionate to $m(p)$.

To jointly minimize them, the new weights are chosen in order to minimize the total weight under the constraint $\sup |g(p, \mathcal{S})| = 1$. That is, $m(p) = w(p) \sup_{\mathcal{S}} |f(p, \mathcal{S})|$.

In our case, this means that $m(p)$ is $w(p) \sup_{\mathcal{S}} \frac{\text{cost}(p,\mathcal{S})}{\sum w(q)\text{cost}(q,\mathcal{S})}$. That is, a point is sampled proportionate to its maximum relative contribution in any solution: this is the sensitivity, or importance, of a point. Although it is hard to compute exactly, some standard techniques exist to approximate the sensitivity. For sake of simplicity, we will assume to have access to those sensitivity values.

Using this framework, the coreset size depends on two crucial values:

- the sum of sensitivities, which is often $O(k)$ [78, 157, 34]. The size of the coreset depends near-linearly in that quantity.
- The VC dimension of the *scaled* set of functions $g(p, \mathcal{S})$. Without going into details, the crux to apply the sensitivity framework is to compute that dimension.

As we already mentioned, the fact that the functions $g(p, \mathcal{S})$ are scaled by a non-uniform value makes it harder. For clustering problems, bounding the VC-dimension of the set of functions $f(p, \mathcal{S})$ boils down to bounding the dimension of $\cup_s \{p : \text{dist}(p, s) \geq r\}$, which is the simply the set of balls' complement. Instead, for scaled function, the set is $\cup_s \{p : \frac{\text{dist}(p,s)}{m(p)} \geq r\}$, which is harder to conceptualize and to work with. We present in Chapter 9 a construction that only requires to bound the VC-dimension when weights are uniforms.

This framework had been applied to a large variety of settings. For Euclidean k-means and k-median, an upper bound of $D \in O(kd \log k)$ is implicit in the work of [29] and Eisenstat and Angluin [69]. This bound was recently shown to be tight by Csikos, Mustafa and Kupavskii [59]. The dependency on d may be replaced with a dependency on $\log k$ [78, 151, 25, 100], using techniques that we explain in Section 7.6. Thus $O(k \log^2 k)$ is a natural barrier for known techniques in Euclidean spaces.

This VC dimension is also bounded in bounded-treewidth graph by Baker et al. [15], in minor-free graphs by Braverman et al. [34], in the case where input points are lines

by Marom and Feldman [131], for projective clustering by Feldman et al. [78], ...

As mentioned in introduction, this framework reached a limit in some cases. The VC dimension of the scaled function may be actually unbounded. This is the case for instance in doubling metrics, where heavy additional work has to be performed to bound the dimension of a slightly perturbed metric, instead of the original one (see Huang et al. [98]). Even so, the coreset obtained has size $\tilde{O}(k^3 d\varepsilon^{-2})$, still far from the current best in Euclidean Spaces, and is tight for that technique. The VC dimension also depends on the number of input points in the case of minor-free graph.

To conclude this introduction on the sensitivity framework, we mention the extension of the framework presented by Braverman et al. [34]. This is based on the following idea: many coreset construction have a dependency in $\log n$, where n is the size of the input. If the construction is applied to a coreset of size $\log n$, then it should yield a coreset of size $\log \log n$ instead: iterating this procedure may remove entirely that dependency.

Of course, things are not that simple, as the n may actually be the total weight of the input – which does not decrease after computing a coreset. However, in some cases such as Euclidean spaces or minor-free graphs, the dependency is actually in the number of distinct input point, regardless of their weight: in those cases, constructing iteratively coresets of coresets is an elegant way of removing dependency in n.

We note that this new idea is not specific to the sensitivity framework: we will also make use of it in some of our coreset constructions.

5.4 Further Related Work

We already surveyed most of the relevant bounds for coresets for k-means and k-median. A complete overview over all of these upper bounds is given in Table 5.1, further pointers to coreset literature can be found in surveys [143, 75]. Lower bounds are presented in Table 5.2. For the remainder of the section, we highlight differences to previous techniques.

The early coreset results mainly considered input data embedded in constant dimensional Euclidean spaces [82, 95, 93]. These coresets relied on low-dimensional geometric decompositions inducing coresets of sizes typically of order at least $k \cdot \varepsilon^{-d}$. These techniques were replaced by *importance sampling* schemes, initiated by the seminal work of Chen [44] described in Section 5.2. While the early coreset papers [95, 96] were heavily reliant on the structure of Euclidean spaces, Chen gave the first coreset of size $O(k^2\varepsilon^{-2}\log^2 n)$ for general n-point metrics.

The state of the art importance sampling techniques in Euclidean spaces are now based on the framework described in Section 5.3. This applies as well to many other metric, as described in Section 5.3.

Additional Euclidean Results. In collaboration with Cohen-Addad, Larsen and Schwiegelshohn [A8], we showed specific results for Euclidean coresets, that are beyond the scope of this manuscript. First, we prove a lower bound of $\Omega\left(k\varepsilon^{-2}\right)$ for (k,z)-clustering coresets in Euclidean Spaces. Second, we improved our construction of Chapter 6 in the special Euclidean case using chaining techniques, to get coreset of size $\tilde{O}\left(k\varepsilon^{-3}\right)$ for Euclidean k-median, and $\tilde{O}_z\left(k^2\varepsilon^{-2}\right)$ for general powers z. This improves respectively over the $\tilde{O}\left(k\varepsilon^{-4}\right)$ and $\tilde{O}_z\left(k^2\varepsilon^{-2}\right)$ of Corollary 7.20.

Related work beyond coresets. So far we only described works that aim at giving better coreset construction for unconstrained k-median and k-means in some metric space. Nevertheless, there is a rich literature on further related questions. As a tool for data compression, coresets feature heavily in streaming literature. Some papers consider a slightly weaker guarantee of summarizing the data set such that a $(1+\varepsilon)$ approximation can be maintained and extracted. Such notions are often referred to as *weak coresets* or streaming coresets, see [76, 77]. Further papers focus on maintaining coresets with little overhead in various streaming and distributed models, see [18, 32, 33, 82, 81]. Other related work considers generalizations of k-median and k-means by either adding capacity constraints [55, 99, 150], or considering more general objective functions [13, 35]. Coresets have also been studied for many other problems: we cite non-comprehensively Determinant Maximization [103], Diversity Maximization [38, 104] logistic regression [101, 144], dependency networks [141], or low-rank approximation [126].

Reference	Size (Number of Points)
Euclidean space	
Har-Peled, Mazumdar (STOC'04) [96]	$O(k \cdot \varepsilon^{-d} \cdot \log n)$
Har-Peled, Kushal (DCG'07) [95]	$O(k^3 \cdot \varepsilon^{-(d+1)})$
Chen (Sicomp'09) [44]	$O(k^2 \cdot d \cdot \varepsilon^{-2} \log n)$
Langberg, Schulman (SODA'10) [115]	$O(k^3 \cdot d^2 \cdot \varepsilon^{-2})$
Feldman, Langberg (STOC'11) [76]	$O(k \cdot d \cdot \log k \cdot \varepsilon^{-2z})$
Feldman, Schmidt, Sohler (Sicomp'20) [78]	$O(k^3 \cdot \log k \cdot \varepsilon^{-4})$
Sohler and Woodruff (FOCS'18) [151]	$O(k^2 \cdot \log k \cdot \varepsilon^{-O(z)})$
Becchetti, Bury, Cohen-Addad, Grandoni, Schwiegelshohn (STOC'19) [25]	$O(k \cdot \log^2 k \cdot \varepsilon^{-8})$
Huang, Vishnoi (STOC'20) [100]	$O(k \cdot \log^2 k \cdot \varepsilon^{-2-2z})$
Braverman, Jiang, Krauthgamer, Wu (SODA'21) [34]	$\tilde{O}(k^2 \cdot \varepsilon^{-4})$
Corollary 7.20	$O\left(k \cdot \log k \cdot \varepsilon^{-2-\max(2,z)}\right)$
Corollary 9.20 (deterministic)	$O\left(k^2 \cdot \log^2 k \cdot \varepsilon^{-O(z)}\right)$
General n-point metrics, $ddim$ denotes the doubling dimension	
Chen (Sicomp'09) [44]	$O(k^2 \cdot \varepsilon^{-2} \cdot \log^2 n)$
Feldman, Langberg (STOC'11) [76]	$O(k \cdot \log k \cdot \log n \cdot \varepsilon^{-2z})$
Huang, Jiang, Li, Wu (FOCS'18) [98]	$O(k^3 \cdot ddim \cdot \varepsilon^{-2})$
Corollary 7.6	$O(k \cdot (ddim + \log k) \cdot \varepsilon^{-\max(2,z)})$
Graph with n vertices, t denotes the treewidth	
Baker, Braverman, Huang, Jiang, Krauthgamer, Wu (ICML'20) [15]	$\tilde{O}(k^2 \cdot t/\varepsilon^2)$
Corollary 7.9	$O(k \cdot (t + \log k) \cdot \varepsilon^{-\max(2,z)})$
Theorem 9.19 (deterministic)	$O\left(k^2 t \log^2 k \cdot \varepsilon^{-5}\right)$
Graph with n vertices, excluding a fixed minor H	
Bravermann, Jian, Krauthgamer, Wu (SODA'21) [34]	$\tilde{O}_{\lvert H\rvert}\left(k^2 \cdot \varepsilon^{-4}\right)$
Corollary 7.15	$O_{\lvert H\rvert}\left(k \cdot (\log^2 k + \frac{\log k}{\varepsilon^4}) \cdot \varepsilon^{-\max(2,z)}\right)$
Theorem 9.19 (deterministic)	$O\left(k^2 \cdot \lvert H\rvert \cdot \log^2 k \cdot \varepsilon^{-5}\right)$

Table 5.1: Comparison of coreset sizes for (k, z)-Clustering in various metrics. Dependencies on $2^{O(z)}$ and polylogε^{-1} are omitted from all references. Additionally, we may trade a factor ε^{-z+2} for a factor k in any construction with $z > 2$. [95, 96] only applies to k-means and k-median, [25, 78] only applies to k-means. [151] runs in exponential time, which has been addressed by Feng et al. [80]. Aside from [95, 96], the algorithms are randomized and succeed with constant probability. Although the results are claimed only for k-Median in [15], it seems that they can be generalized to any power. The main difference is in the computation of a constant factor approximation.

Metric Space	Best lower bound	Our result
Discrete Metrics	$\Omega(k\varepsilon^{-1}\log n)$[15]	$\Omega(k\varepsilon^{-2}\log n)$* Theorem 8.2
doubling dimension D	-	$\Omega(k\varepsilon^{-2}D)$* Corollary 8.10
Euclidean k-median	$\Omega(k\varepsilon^{-1/2})$ [15]	$\Omega(k\varepsilon^{-2})^{\dagger}$
Euclidean k-means	-	$\Omega(k\varepsilon^{-2}))^{\dagger}$
Euclidean	$\Omega(k2^{z/100})$ [100]	$\Omega(k\varepsilon^{-2}))^{\dagger}$

Table 5.2: Lower bounds on coreset sizes. Results marked with * are tight. Results marker with † are from a collaboration with Cohen-Addad, Larsen and Schwiegelshohn [A8], beyond the scope of this thesis.

Chapter 6

A New Coreset Framework for Clustering

In this chapter, we present how to compute a coreset of small size, provided the existence of a small approximate centroid set: we show Theorem 5.3. The roadmap is as follows: first, we proceed to a high-level description of our result. We then present formally the algorithm in Section 6.2, and describe the construction of a coreset for well structured instances in Section 6.3 and Section 6.4. Those are the bulk of the chapter. We show how to reduce to a well structured instance in Section 6.5. Finally, we complement this result in Section 6.6, showing how to construct a coreset with a different tradeoff between k and ε.

6.1 Overview of Our Techniques

Our proof of Theorem 5.3 is arguably from first principles. We start with a quick overview of its ingredients. The approach consists in first reducing to a well structured instance, that consists of a set of centers $\mathcal{A}$ inducing k clusters, all having roughly the same cost, and where every point is at the same distance of $\mathcal{A}$, up to a factor 2. Then we show it is enough to perform importance sampling on all these clusters.

Reducing to a structured instance. Like most coreset constructions, we initially compute a constant factor approximation $\mathcal{A}$ to the problem. Instead of considering mere rings as in Chen's analysis presented in Section 5.2, we partition points into groups such that the following conditions are satisfied, for a given group G:

- For all clusters, the cost of the intersection of the cluster with the group is at least half the average; i.e. $\forall C_i,\ \mathrm{cost}(C_i \cap G, \mathcal{A}) \geq \frac{\mathrm{cost}(G,\mathcal{A})}{2k}$.

- In every cluster C_i, there exists a cost $r_{G,i}$ such that the points in the intersection of the cluster with the group cost $r_{G,i}$ (up to constant factors), i.e. $\forall p \in C_i \cap G, r_{G,i} \leq \text{cost}(p, \mathcal{A}) \leq 2r_{G_i}$.

As a comparison with Section 5.2, note that while the second condition is satisfied in each ring, the first one is not, as a ring contains points of a single cluster. Instead, we group several rings together, to reduce the number of uniform samples needed in the algorithm from $\Omega_\varepsilon(k)$ to $O_\varepsilon(1)$.

We then compute coresets for each group and output the union. In some sense, this preprocessing step identifies canonical instances for coresets; any algorithm that produces improved coresets for instances satisfying the aforementioned regularity condition can be combined with our preprocessing steps to produce improved coreset in general.

Importance Sampling in Groups. The first technical challenge is to analyse the importance sampling procedure for structured instances.

The arguably simplest way to attempt to analyse importance sampling is by first showing that for any fixed solution $\mathcal{S}$ we need a set Ω of δ samples from G to show that with good enough probability

$$\sum_{p \in \Omega} \text{cost}(p, \mathcal{S}) \frac{\text{cost}(G, \mathcal{A})}{\text{cost}(p, \mathcal{A}) \cdot \delta} = (1 \pm \varepsilon) \cdot \text{cost}(G, \mathcal{S}), \tag{6.1}$$

and then applying a union bound over the validity of Eq. (6.1) for all solutions S. This union bound is typically achieved using VC-dimension techniques.

Using this simple estimator, most analyses of importance sampling procedures require a sample size of at least k points to approximate the cost of a single given solution. To illustrate this, consider an instance where a single cluster C is isolated from all the others. Clearly, if $\mathcal{S}$ does not have a center close to C, the cost will be extremely large, requiring some point of C to be contained in the sample. One way to remedy this is by picking a point p' proportionate to $\frac{\text{cost}(p', \mathcal{A})}{\text{cost}(\mathcal{A})} + \frac{1}{|C_i|}$ rather than $\frac{\text{cost}(p', \mathcal{A})}{\text{cost}(\mathcal{A})}$, where C_i is the cluster to which p' is assigned, see for instance Feldman et al. [78]. This analysis always leads to coreset of size quadratic in k at best[1]. Our analysis of importance sampling for structured instances will allow us to bypass both the quadratic dependency on k, and the need to bound on the VC-dimension of the range space induced by balls in the metric space.

Our high level idea is to use two union bounds. The first one will deal with clusters that are very expensive compared to their cost in $\mathcal{A}$. The second one will focus on solutions in which clusters have roughly the same cost as they do in $\mathcal{A}$. For the former case, we observe that if a cluster C_i is served by a center in solution $\mathcal{S}$ that is very far away, then we can easily bound its cost in $\mathcal{S}$ as long as our sample approximates the size of every cluster. Specifically, assume that there exists a point p in C_i with

[1] A linear dependency on k can be achieved using a different analysis, see [76, 100] for examples. This approach does not seem to generalize to arbitrary metrics.

distance to $\mathcal{S}$ at least $\Omega(1)\cdot\varepsilon^{-1}\cdot\text{dist}(p,c_i)$. Then, since we are working with structured instances, all points of C_i are roughly at the same distance of c_i and that this distance is negligible compared to $\text{dist}(p,\mathcal{S})$, all points of C_i are nearly at the same distance of $\mathcal{S}$. Conditioned on the event $\mathcal{E}$ that the sample Ω preserves the size of all clusters, the cost of C_i in solution $\mathcal{S}$ is preserved as well. Note that this event $\mathcal{E}$ is independent of the solution $\mathcal{S}$ and thus we require no enumeration of solutions to preserve the cost of expensive clusters. Proving that $\mathcal{E}$ holds is a straightforward application of concentration bounds.

The second observation is that points with $\text{dist}(p,\mathcal{S}) \leq \varepsilon\cdot\text{dist}(p,\mathcal{A})$ are so cheap that their cost is preserved by the sampling with an error at most $\varepsilon\cdot\text{cost}(\mathcal{A})$. Indeed, their cost in $\mathcal{S}$ cannot be more than $\varepsilon\cdot\text{cost}(\mathcal{A})$: it is easy to show that the same bound holds for the coreset.

The intermediate cases, i.e. points p such that $\varepsilon^{-1}\cdot\text{dist}(p,\mathcal{A}) > \text{dist}(p,\mathcal{S}) > \varepsilon\cdot\text{dist}(p,\mathcal{A})$, but not so far as to simply use event $\mathcal{E}$ to bound the cost, is the hardest part of the analysis. We let I be the set of such points. With some basic calculations, one can therefore show that the variance of the cost estimator Eq. (6.1) is of the order

$$\frac{1}{\delta}\sum_{p\in I}\left(\text{cost}(p,\mathcal{S})\cdot\varepsilon^{-z}\cdot\text{cost}(p,\mathcal{A})\cdot\frac{\text{cost}(G,\mathcal{A})}{\text{cost}(p,\mathcal{A})}\right) \in O(\frac{\varepsilon^{-z}}{\delta})\cdot(\text{cost}(G,\mathcal{A})+\text{cost}(G,\mathcal{S}))^2.$$

Thus, standard concentration bounds give an additive error of $\varepsilon\cdot(\text{cost}(\mathcal{A})+\text{cost}(\mathcal{S}))$ when the number of samples in each group is $\delta = O(\varepsilon^{-2-z})$. We will present this analysis in Section 6.3.5: although non-optimal, it gives a shorter, hopefully enlightening proof. Moreover, it yields coresets of size $\tilde{O}\left(k\varepsilon^{-2-z}\right)$, already improving the state-of-the-art in many cases.

To improve this number of necessary samples to $O(\varepsilon^{-\max(2,z)})$, we use a different estimator defined as follows. For every cluster C_i, let q_i be the point of C_i that is the closest to $\mathcal{S}$. We then consider

$$\sum_{p\in C_i\cap\Omega}(\text{cost}(p,\mathcal{S})-\text{cost}(q_i,\mathcal{S}))\cdot\frac{\text{cost}(G,\mathcal{A})}{\text{cost}(p,\mathcal{A})\cdot\delta} \tag{6.2}$$

$$+\sum_{p\in C_i\cap\Omega}\text{cost}(q_i,\mathcal{S})\cdot\frac{\text{cost}(G,\mathcal{A})}{\text{cost}(p,\mathcal{A})\cdot\delta} \tag{6.3}$$

Conditioned on event $\mathcal{E}$, the estimator in Equation 6.3 is always concentrated around its expectation, as $\text{cost}(q_i,\mathcal{S})$ is fixed for $\mathcal{S}$. The first estimator in Equation 6.2 now has a reduced variance. Specifically, the Estimator 6.2 has variance at most $\frac{\varepsilon^{-z+2}\text{cost}(\mathcal{A})\text{cost}(\mathcal{S})}{\delta}$ when $z>2$, and $\frac{\text{cost}(\mathcal{A})^2}{\delta}$ when $z\in\{1,2\}$, which ultimately allows us to show that $O\left(\varepsilon^{-\max(2,z)}\right)$ samples are enough to achieve an additive error of $\varepsilon\cdot(\text{cost}(\mathcal{S})+\text{cost}(\mathcal{A}))$. This technique is somewhat related to (and inspired by) chaining arguments (see e.g. Talagrand [154] for more on chaining). The key difference is while chaining is generally applied to improve over basic union bounds, our estimator is designed to reduce the variance.

Preserving the Cost of Points not in Well-Structured Groups Unfortunately, it is not possible to decompose the entire point set into groups. Given an initial

solution $\mathcal{A}$ and a cluster $C \in \mathcal{A}$ with center c, this is possible for all the points at distance at most $\varepsilon^{-O(z)} \cdot \frac{\text{cost}(C,c)}{|C|}$. The remaining points are now both far from their respective center in $\mathcal{A}$ and only a small fraction of the point set – as there cannot be many point really further away than the average distance. In the following, let P_{far} denote these points.

For any given subset of these far away points and a candidate solution $\mathcal{S}$, one can use that either the points pay at most what they do in $\mathcal{A}$, or an increase in their cost significantly increases the overall cost. In the former case, standard importance sampling preserves the cost with a very small sample size. In the latter case, a significant cost by a point p in P_{far} also implies that all points close to the center c have to significantly increase the cost. Since P_{far} corresponds to only a tiny fraction of the points in the cluster, it turns out to be possible to charge the error made for points in P_{far} to the total cluster.

A Union-bound to Preserve all Solutions As pictured in the previous paragraphs, the cost of points with either very small or very large distance to $\mathcal{S}$ is preserved for any solution $\mathcal{S}$ with high probability.

The guarantee we have for interesting points is weaker: their cost is preserved by the coreset with high probability for any *fixed* solution $\mathcal{S}$. Hence, for this to hold for any solution, we need to take a union-bound over the probability of failure for all possible solution $\mathcal{S}$. However, the union-bound is necessary only for these interesting points : this explains the introduction of the approximate centroid set in Definition 5.2. Assuming the existence of a set $\mathbb{C}$ such as in Definition 5.2, one can take a union-bound over the failure of the construction for all solution in $\mathbb{C}^k$, to ensure that the cost of interesting points is preserved for all these solutions. To extend this result to *any* solution $\mathcal{S}$, one can take the solution $\tilde{\mathcal{S}}$ in $\mathbb{C}$ that approximates best $\mathcal{S}$, and relate the cost of interesting points in $\mathcal{S}$ to their cost in $\tilde{\mathcal{S}}$ with a tiny error. Since the cost of interesting points in $\tilde{\mathcal{S}}$ is preserved in the coreset, the cost of these points in $\mathcal{S}$ is preserved as well.

6.2 The Coreset Construction Algorithm, and Proof of Theorem 5.3

6.2.1 Partitioning an Instance into Groups: Definitions

As sketched in the introductory chapter, the algorithm partitions the input points into structured groups. We give here the useful definitions.

Fix a metric space $I = (X, \text{dist})$, positive integers k, z and a set of clients P. For a solution $\mathcal{S}$ of (k, z)-clustering on P and a center $c \in \mathcal{S}$, c's cluster consists of all points closer to c than to any other center of $\mathcal{S}$.

Fix as well some $\varepsilon > 0$, and let $\mathcal{A}$ be any solution for (k, z)-clustering on P with k centers. Let $C_1, ..., C_k$ be the clusters induced by the centers of $\mathcal{A}$.

- the average cost of a cluster C_i is $\Delta_{C_i} = \frac{\text{cost}(C_i, \mathcal{A})}{|C_i|}$
- For all i, j, the *ring* $R_{i,j}$ is the set of points $p \in C_i$ such that
$$2^j \Delta_{C_i} \leq \text{cost}(p, \mathcal{A}) \leq 2^{j+1} \Delta_{C_i}.$$
- The *inner ring* $R_I(C_i) := \cup_{j \leq 2z \log(\varepsilon/z)} R_{i,j}$ (resp. *outer ring*, $R_O(C_i) := \cup_{j > 2z \log(z/\varepsilon)} R_{i,j}$) of a cluster C_i consists of the points of C_i with cost at most $(\varepsilon/z)^{2z} \Delta_{C_i}$ (resp. at least $(z/\varepsilon)^{2z} \Delta_{C_i}$). The *main ring* $R_M(C_i)$ consists of all the other points of C_i. We let $R_I^{\mathcal{A}}$ (resp. $R_O^{\mathcal{A}}$) be the union of inner (resp. outer rings) of the clusters induced by $\mathcal{A}$.

In words, a ring consist of points from a given cluster that have same cost, up to a factor 2. The inner ring is the union of all points that have cost essentially negligible, and the outer ring the points that are very expensive – but there must be few of them, by Markov's inequality.

Rings are gathered into *groups*, such that two rings in the same group have same cost (again, up to a factor 2).

- for each j, R_j is defined to be $\cup_{i=1}^{k} R_{i,j}$.
- For each j, the main rings $R_{i,j}$ are gathered into *groups* $G_{j,b}$ defined as follows:
$$G_{j,b} := \Big\{ p \mid \exists i,\ p \in R_{i,j} \text{ and } \Big(\frac{\varepsilon}{4z}\Big)^z \cdot \frac{\text{cost}(R_j, \mathcal{A})}{k} \cdot 2^b \leq \text{cost}(R_{i,j}, \mathcal{A}) \leq \Big(\frac{\varepsilon}{4z}\Big)^z \cdot 2^{b+1} \cdot \frac{\text{cost}(R_j, \mathcal{A})}{k} \Big\}.$$
- For any j, let $G_{j,min} := \cup_{b \leq 0} G_{j,b}$ be the union of the cheapest groups, and $G_{j,max} := \cup_{b \geq z \log \frac{4z}{\varepsilon}} G_{j,b}$ be the union of the most expensive ones. The set of interesting groups is made of $G_{j,min}, G_{j,max}$, and $G_{j,b}$ for all $0 < b < z \log \frac{4z}{\varepsilon}$.
- The set of outer rings is also partitioned into *outer groups*:
$$G_b^O = \Big\{ p \mid \exists i,\ p \in R_O(C_i), \text{ and } \Big(\frac{\varepsilon}{4z}\Big)^z \cdot \frac{\text{cost}(R_O^{\mathcal{A}}, \mathcal{A})}{k} \cdot 2^b \leq \text{cost}(R_O(C_i), \mathcal{A}) \leq \Big(\frac{\varepsilon}{4z}\Big)^z \cdot 2^{b+1} \cdot \frac{\text{cost}(R_O^{\mathcal{A}}, \mathcal{A})}{k} \Big\}.$$
- We let as well $G_{min}^O = \cup_{b \leq 0} G_b^O$ and $G_{max}^O = \cup_{b \geq z \log \frac{4z}{\varepsilon}} G_b^O$. The interesting outer groups are G_{min}^O, G_{max}^O and all G_b^O with $0 < b < z \log \frac{4z}{\varepsilon}$.

Intuitively, grouping points by groups is helpful, as all points in the same ring can pay the same additive error. Since there are very few groups, it turns out possible to construct a coreset for each group, and then take the union of the group's coreset. This is essentially the algorithm we propose.

We note few facts about the partitioning:

▶ **Fact 6.1.** There exist at most $O(z\log(z/\varepsilon))$ many non-empty R_j that are not included in inner or outer ring, i.e., not in $R_I^{\mathcal{A}}$ nor in $R_O^{\mathcal{A}}$. ◀

The number of different non-empty interesting groups is bounded as well:

▶ **Fact 6.2.** There exists at most $O(z^2\log^2(z/\varepsilon))$ many interesting $G_{j,b}$. ◀

This is simply due to the fact that j can take only interesting values between $2z\log(\varepsilon/z)$ and $2z\log(z/\varepsilon)$, and interesting b between 0 and $z\log(4z/\varepsilon)$.

Similarly, by the definition of the outer groups, we have also that

▶ **Fact 6.3.** There exists at most $O(z\log(z/\varepsilon))$ many interesting outer groups. ◀

For simplicity, we will drop mention of "interesting" : when considering any group, it will implicitly be an interesting group.

6.2.2 The algorithm

For an initial metric space (X, dist), set of clients P and $\varepsilon > 0$, our algorithm essentially consists of the following steps: given a solution $\mathcal{A}$, it processes the input in order to reduce the number of different groups. Then, the algorithm computes a coreset of the points inside each group using the following `GroupSample` procedure. The final coreset is made of the union of the coresets for all groups.

The `GroupSample` procedure takes as input a group of points G as defined in Section 6.2.1, a set of centers $\mathcal{A}$ inducing clusters $\tilde{C}_1, \tilde{C}_2, ..., \tilde{C}_k$ on G and an integer δ. Note importantly that the definition of clusters $\tilde{C}_i$ says that they are only made of points from the group G. The output of `GroupSample` is a set of weighted points, computed as follows: a point $p \in \tilde{C}_i$ is sampled with probability $\frac{\delta\cdot\mathrm{cost}(\tilde{C}_i,\mathcal{A})}{|\tilde{C}_i|\cdot\mathrm{cost}(G,\mathcal{A})}$, and the weight of any sampled point is rescaled by a factor $\frac{|\tilde{C}_i|\cdot\mathrm{cost}(G,\mathcal{A})}{\delta\mathrm{cost}(\tilde{C}_i,\mathcal{A})}$.[2]

The properties of the `GroupSample` procedure are captured by the following lemma.

▶ **Lemma 6.4.** Let (X, dist) be a metric space, k, z be two positive integers and G be a group of clients and $\mathcal{A}$ be a solution to (k, z)-clustering on G with k centers such that:

- for every cluster $\tilde{C}$ induced by $\mathcal{A}$ on G, all points of $\tilde{C}$ have the same cost in $\mathcal{A}$, up to a factor 2: $\forall p, q \in \tilde{C},\ \mathrm{cost}(p, \mathcal{A}) \leq 2\mathrm{cost}(q, \mathcal{A})$.

[2]Note that this is essentially importance sampling, as each point in a cluster $\tilde{C}_i$ have cost roughly equal to the average. We chose this different distribution for simplicity in our proofs.

- for all clusters $\tilde{C}$ induced by $\mathcal{A}$ on G, it holds that $\frac{\text{cost}(G,\mathcal{A})}{2k} \leq \text{cost}(\tilde{C},\mathcal{A})$.

Let $\mathbb{C}$ be a $\mathcal{A}$-approximate centroid set for (k,z)-clustering on G.
Then, there exists an algorithm `GroupSample`, running in time $O(|G|)$ that constructs a set Ω of size δ such that, with probability $1 - \exp\left(k\log|\mathbb{C}| - 2^{O(z\log z)} \cdot \frac{\min(\varepsilon^2,\varepsilon^z)}{\log^2 1/\varepsilon} \cdot \delta\right)$ it holds that for all set $\mathcal{S}$ of k centers:

$$|\text{cost}(G,\mathcal{S}) - \text{cost}(\Omega,\mathcal{S})| = O(\varepsilon)\left(\text{cost}(G,\mathcal{S}) + \text{cost}(G,\mathcal{A})\right). \quad \blacktriangleleft$$

We further require the `SensitivitySample` procedure, which we will apply to some of the points not consider by the calls to `GroupSample`. From a group G, this procedure merely picks δ points p with probability $\frac{\text{cost}(p,\mathcal{A})}{\text{cost}(G,\mathcal{A})}$. Each of the δ sampled points has a weight $\frac{\text{cost}(G,\mathcal{A})}{\delta\cdot\text{cost}(p,\mathcal{A})}$.

The key property of `SensitivitySample` is given in the following lemma.

▶ **Lemma 6.5.** Let (X,dist) be a metric space, k,z be two positive integers, P be a set of clients and $\mathcal{A}$ be a $c_{\mathcal{A}}$-approximate solution solution to (k,z)-clustering on P.
Let G be either a group G_b^O or $G_{\max}^O$. Suppose moreover that there is a $\mathcal{A}$-approximate centroid set $\mathbb{C}$ for (k,z)-clustering on G .
Then, there exists an algorithm `SensitivitySample` running in time $O(|G|)$ that constructs a set Ω of size δ such that it holds with probability $1 - \exp\left(k\log|\mathbb{C}| - 2^{O(z\log z)} \cdot \frac{\varepsilon^2}{\log^2 1/\varepsilon} \cdot \delta\right)$ that, for all sets $\mathcal{S}$ of k centers:

$$|\text{cost}(G,\mathcal{S}) - \text{cost}(\Omega,\mathcal{S})| = \frac{\varepsilon}{z\log z/\varepsilon} \cdot \left(\text{cost}(\mathcal{S}) + \text{cost}(\mathcal{A})\right). \quad \blacktriangleleft$$

An interesting feature of Lemma 6.5 is that the probability does not depend on ε^{-z}, as it does in Lemma 6.4. In particular, it matches the lower bound for Euclidean and doubling metrics that was mentioned in Chapter 5.

Using the two algorithms `GroupSample` and `SensitivitySample`, we can formally present the whole algorithm:

Input: A metric space (X,dist), a set $P \subseteq X$, $k,z > 0$, a solution $\mathcal{A}$ to (k,z)-clustering on P, and ε such that $0 < \varepsilon < 1/3$.
Output: A coreset. Namely, a set of points $\Omega \subseteq P \cup \mathcal{A}$ and a weight function $w : \Omega \mapsto \mathbb{R}_+$ such that for any set of k centers $\mathcal{S}$, $\text{cost}(P,\mathcal{S}) = (1 \pm \varepsilon)\text{cost}(\Omega,\mathcal{S})$.

1. Set the weights of all the centers of $\mathcal{A}$ to 0.
2. **Partition the remaining instance into groups**:
   (a) For each cluster C of $\mathcal{A}$ with center c, remove $R_I(C)$ and increase the weight of c by $|R_I(C)|$.
   (b) For each cluster C with center c in solution $\mathcal{A}$, the algorithm discards also

all of $C \cap \cup_j G_{j,min}$ and $R_O(C) \cap G^O_{min}$, and increases the weight of c by the number of points discarded in cluster c.

(c) Let $\mathcal{D}$ be the set of points discarded at those steps, and P_1 be the weighted set of centers that have positive weights.

3. **Sampling from well structured groups:** For every j such that $z\log(\varepsilon/z) \leq j \leq 2z\log(z/\varepsilon)$ and every group $G_{j,b} \notin G_{j,min}$, compute a coreset $\Omega_{j,b}$ of size

$$\delta = O\left(2^{O(z\log z)}\frac{\log^2 1/\varepsilon}{\min(\varepsilon^2,\varepsilon^z)}\left(k\log|\mathbb{C}| + \log\log(1/\varepsilon) + \log(1/\pi)\right)\right)$$

using the `GroupSample` procedure.

4. **Sampling from the outer rings:** From each group $G^O_1, ..., G^O_{max}$, compute a coreset Ω^O_b of size

$$\delta = O\left(\frac{2^{O(z\log z)}\cdot\log^2(1/\varepsilon)}{\varepsilon^2}(k\log|\mathbb{C}| + \log\log(1/\varepsilon) + \log(1/\pi))\right)$$

using the `SensitivitySample` procedure.

5. **Output:**

- A coreset consisting of $\mathcal{A} \cup_{j,b} \Omega_{j,b} \cup_i \Omega^O_i$.
- Weights: weights for $\mathcal{A}$ defined throughout the algorithm, weights for $\Omega_{j,b}$ defined by the `GroupSample` procedure, weights for Ω_O defined by the `SensitivitySample` procedure.

▶ **Remark 6.6.** Instead of using the `GroupSample` procedure, one could use any coreset construction tailored for the well structured group. Improving on that step would improve the final coreset bound: if the size of the coreset produced for a group is T, then the total coreset has size

$$\widetilde{O}\left(T + \frac{2^{O(z\log z)}}{\varepsilon^2}\cdot k\log|\mathbb{C}|\right)$$ ◀

6.2.3 Proof of the Main Theorem 5.3

As we prove in Section 6.5, the outcome of the partitioning step, $\mathcal{D}$ and P_1, satisfies the following lemma, that deals with the inner ring, and the groups $G_{j,min}$ and G^O_{min}:

▶ **Lemma 6.7.** Let (X, dist) be a metric space with a set of clients P, k, z be two positive integers, and $\varepsilon \in \mathbb{R}^*_+$. For every solution $\mathcal{S}$, it holds that

$$|\text{cost}(\mathcal{D},\mathcal{S}) - \text{cost}(P_1,\mathcal{S})| = O(\varepsilon)\text{cost}(\mathcal{S}),$$

where $\mathcal{D}$ and P_1 are defined in Step 2 of the algorithm. ◀

Combining properties of the partitioning procedure, Lemma 6.4, Lemma 6.5 and Lemma 6.7 allows to prove Theorem 5.3:

Proof of Theorem 5.3. Let Ω be the output of the algorithm described above, and $\delta = O\left(2^{O(z\log z)}\frac{\log^2 1/\varepsilon}{\min(\varepsilon^2,\varepsilon^z)}\left(k\log|\mathbb{C}| + \log\log(1/\varepsilon) + \log(1/\pi)\right)\right)$ as defined in step 3 of the algorithm. Due to Fact 6.2 and Fact 6.3, Ω has size $O(z^2\log^2(z/\varepsilon)\cdot\delta + |\mathcal{A}|)$, and non-negative weights by construction.

We now turn to analysing the quality of the coreset. Any group $G_{j,b}$ for $b > 0$ satisfies Lemma 6.4: the cost of any point $p \in G_{j,b}\cap C_i$ satisfies $2^j\Delta_{C_i} \leq \text{cost}(p,\mathcal{A}) \leq 2^{j+1}\Delta_{C_i}$, and

- for $b \in \left\{0,...,z\log\frac{4z}{\varepsilon}\right\}$, the cost of all clusters induced by $\mathcal{A}$ on $G_{j,b}$ are equal up to a factor 2, hence for all i $\frac{\text{cost}(G_{j,b},\mathcal{A})}{2k} \leq \text{cost}(C_i\cap G_{j,b},\mathcal{A})$
- for $b = max$, it holds that $\frac{\text{cost}(G_{j,max},\mathcal{A})}{2k} \leq \frac{\text{cost}(R_j,\mathcal{A})}{2k} \leq \text{cost}(C_i\cap G_{j,max},\mathcal{A})$.

Hence, Lemma 6.4 ensures that, with probability $1-\exp\left(k\log|\mathbb{C}| - 2^{O(z\log z)}\cdot\frac{\min(\varepsilon^2,\varepsilon^z)}{\log^2 1/\varepsilon}\cdot\delta\right)$, the coreset $\Omega_{j,b}$ constructed for $G_{j,b}$ satisfies for any solution $\mathcal{S}$

$$|\text{cost}(G_{j,b},\mathcal{S}) - \text{cost}(\Omega_{j,b},\mathcal{S})| = O(\varepsilon)\left(\text{cost}(G_{j,b},\mathcal{S}) + \text{cost}(G_{j,b},\mathcal{A})\right).$$

Similarly, Lemma 6.5 ensures that, with probability $1-\exp\left(\log|\mathbb{C}| - 2^{O(z\log z)}\cdot\frac{\varepsilon^2}{\log^2 1/\varepsilon}\cdot\delta\right)$, the coreset Ω_b^O constructed for G_b^O satisfies for any solution $\mathcal{S}$

$$|\text{cost}(G_b^O,\mathcal{S}) - \text{cost}(\Omega_b^O,\mathcal{S})| = \frac{\varepsilon}{z\log(z/\varepsilon)}\left(\text{cost}(\mathcal{S}) + \text{cost}(\mathcal{A})\right).$$

Taking a union-bound over the failure probability of Lemma 6.5 and of Lemma 6.4 applied to all groups $G_{j,b}$ with $z\log(\varepsilon/z) \leq j \leq 2z\log(z/\varepsilon)$ and all G_i^O implies that, with probability

$$1 - z^2\log^2(z/\varepsilon)\exp\left(k\log|\mathbb{C}| - 2^{O(z\log z)}\cdot\frac{\min(\varepsilon^2,\varepsilon^z)}{\log^2 1/\varepsilon}\cdot\delta\right)$$
$$-z\log(z/\varepsilon)\exp\left(\log|\mathbb{C}| - 2^{O(z\log z)}\frac{\varepsilon^2}{\log^2 1/\varepsilon}\cdot\delta\right)$$

for any solution $\mathcal{S}$,

$$\begin{aligned}
&|\text{cost}(\mathcal{S}) - \text{cost}(\Omega,\ \mathcal{S})| \\
\leq\ &|\text{cost}(\mathcal{D},\mathcal{S}) - \text{cost}(P_1,\mathcal{S})| + \sum_{j,b}|\text{cost}(G_{j,b},\mathcal{S}) - \text{cost}(G_{j,b}\cap\Omega,\mathcal{S})| \\
&\qquad + \sum_i |\text{cost}(G_b^O,\mathcal{S}) - \text{cost}(G_b^O\cap\Omega,\mathcal{S})| \\
\leq\ &O(\varepsilon)\text{cost}(\mathcal{S}) + O(\varepsilon)\text{cost}(\mathcal{A}) \leq O(\varepsilon)\text{cost}(\mathcal{S})
\end{aligned}$$

where the penultimate inequality uses Lemma 6.7, and the last one that $\mathcal{A}$ is a constant-factor approximation.

For $\delta = 2^{O(z \log z)} \frac{\log^2 1/\varepsilon}{\min(\varepsilon^2, \varepsilon^z)} (k \log |\mathbb{C}| + \log\log(1/\varepsilon) + \log(1/\pi))$, this probability can be simplified to

$$1 - \exp\left(2(\log z + \log\log(z/\varepsilon)) + k \log |\mathbb{C}| - 2^{O(z \log z)} \cdot \frac{\min(\varepsilon^2, \varepsilon^z)}{\log^2 1/\varepsilon} \cdot \delta\right) = 1 - \pi.$$

The complexity of this algorithm is:

- $O(n)$ to compute the groups: given all distances from a client to its center, computing the average cost of all clusters costs $O(n)$, hence partitioning into R_j cost $O(n)$ as well, and then decomposing R_j into groups is also done in $O(n)$ time;
- plus the cost to compute the coreset in the groups, which is $\sum_{j,b} O(|G_{j,b}|) + \sum_i O(|G_b^Q|) = O(n)$

Hence, the total complexity is $O(n)$. □

6.3 Sampling inside Groups: Proof of Lemma 6.4

The goal of this section is to show the existence of an algorithm `GroupSample` that satisfies Lemma 6.4, that we restate here for convenience:

▶ **Lemma 6.4.** Let (X, dist) be a metric space, k, z be two positive integers and G be a group of clients and $\mathcal{A}$ be a solution to (k, z)-clustering on G with k centers such that:

- for every cluster $\tilde{C}$ induced by $\mathcal{A}$ on G, all points of $\tilde{C}$ have the same cost in $\mathcal{A}$, up to a factor 2: $\forall p, q \in \tilde{C},\ \text{cost}(p, \mathcal{A}) \leq 2\text{cost}(q, \mathcal{A})$.
- for all clusters $\tilde{C}$ induced by $\mathcal{A}$ on G, it holds that $\frac{\text{cost}(G,\mathcal{A})}{2k} \leq \text{cost}(\tilde{C}, \mathcal{A})$.

Let $\mathbb{C}$ be a $\mathcal{A}$-approximate centroid set for (k, z)-clustering on G.
Then, there exists an algorithm `GroupSample`, running in time $O(|G|)$ that constructs a set Ω of size δ such that, with probability $1 - \exp\left(k \log |\mathbb{C}| - 2^{O(z \log z)} \cdot \frac{\min(\varepsilon^2, \varepsilon^z)}{\log^2 1/\varepsilon} \cdot \delta\right)$ it holds that for all set $\mathcal{S}$ of k centers:

$$|\text{cost}(G, \mathcal{S}) - \text{cost}(\Omega, \mathcal{S})| = O(\varepsilon) \left(\text{cost}(G, \mathcal{S}) + \text{cost}(G, \mathcal{A})\right). \quad ◀$$

6.3.1 Description of the GroupSample Algorithm

The `GroupSample` merely consists of importance sampling in rounds, i.e. there are δ rounds in which one point of G is sampled. Let $\tilde{C}_1, \tilde{C}_2, ...$ be the clusters induced by $\mathcal{A}$ on G: the probability of sampling point $p \in \tilde{C}_i$ is $\frac{\text{cost}(\tilde{C}_i,\mathcal{A})}{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}$ – recall that all clusters $\tilde{C}_i$ contain only points from the group G. The weight of any sampled point is rescaled by a factor $\frac{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}{\delta\text{cost}(\tilde{C}_i,\mathcal{A})}$. If there are m copies of a point, it is sampled in a round with probability $\frac{m\cdot\text{cost}(\tilde{C}_i,\mathcal{A})}{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}$ (which is equivalent to sampling each copy with probability $\frac{\text{cost}(\tilde{C}_i,\mathcal{A})}{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}$). In what follows, each copies will be considered independently.

▶ **Definition 6.8.** We denote $f(p) := \frac{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}{\delta\text{cost}(\tilde{C}_i,\mathcal{A})}$ the scaling factor of the weight of a point $p \in \tilde{C}_i$. ◀

6.3.2 Organization of the Proof

To analyze the sampling procedure of `GroupSample`, we consider different cost ranges $I_{\ell,\mathcal{S}}$ induced by a solution $\mathcal{S}$ as follows. A point p of G is in $I_{\ell,\mathcal{S}}$ if $2^\ell \cdot \text{cost}(p,\mathcal{A}) \leq \text{cost}(p,\mathcal{S}) \leq 2^{\ell+1} \cdot \text{cost}(p,\mathcal{A})$. We distinguish between the following cases.

- $\ell \leq \log \varepsilon/2$. We call all $I_{\ell,\mathcal{S}}$ in this range *tiny*. The union of all tiny $I_{\ell,\mathcal{S}}$ is denoted by $I_{tiny,\mathcal{S}}$.
- $\log \varepsilon/2 \leq \ell \leq z \log(8z/\varepsilon)$. We call all $I_{\ell,\mathcal{S}}$ in this range *interesting*.
- $\ell \geq z \log(4z/\varepsilon)$. We call all $I_{\ell,\mathcal{S}}$ in this range *huge*.

Note that interesting and huge ranges intersect. This is to give us some slack in the proof: for a solution $\mathcal{S}$, we will deal with huge ranges before relating $\mathcal{S}$ to its representative $\tilde{\mathcal{S}}$ from $\mathbb{C}^k$. Due to the approximation, some non-huge range for $\mathcal{S}$ can become huge for $\tilde{\mathcal{S}}$: however, due to our definition, they stay in the interesting ranges.

A simple observation leads to the next fact.

▶ **Fact 6.9.** Given a solution $\mathcal{S}$, there are at most $O(z \log z/\varepsilon)$ interesting $I_{\ell,\mathcal{S}}$. ◀

Bounding the difference in cost of $G \cap I_{\ell,\mathcal{S}}$ requires different arguments depending on the type of $I_{\ell,\mathcal{S}}$.

The two easy cases are tiny and huge, so we will first proceed to prove those. Proving the interesting case is arguably both the main challenge and our main technical contribution.

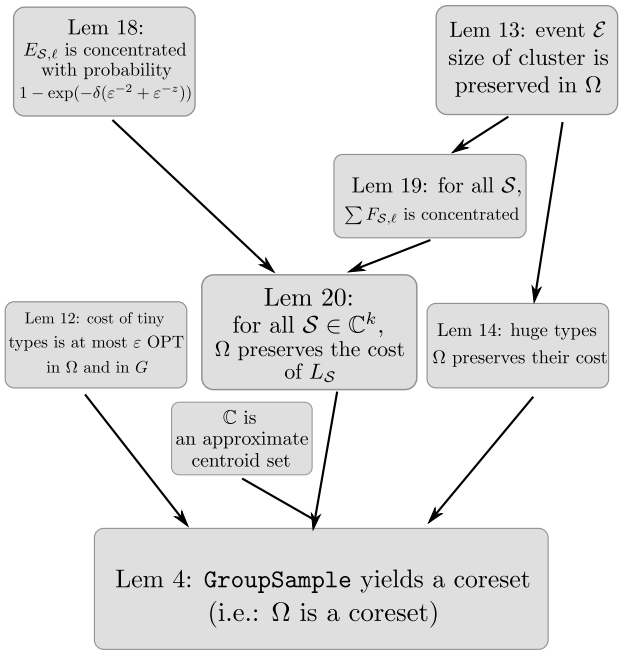


Figure 6.1: Arrangement of Lemmas of Section 6.3 to prove Lemma 6.4.

For the proof, we will rely on Bernstein's concentration inequality:

▶ **Theorem 6.10 (Bernstein's Inequality).** Let $X_1, \dots X_\delta$ be non-negative independent random variables. Let $S = \sum_{i=1}^{\delta} X_i$. If there exists an almost-sure upper bound $M \geq X_i$, then

$$\mathbb{P}\left[|S - \mathbb{E}[S]| \geq t\right] \leq \exp\left(-\frac{t^2}{2\sum_{i=1}^{\delta}\left(\mathbb{E}[X_i^2] - \sum \mathbb{E}[X_i]^2\right) + \frac{2}{3}\cdot M \cdot t}\right).$$

◀

In application of that bound, we will simply drop the $E[X_i]^2$ terms from the denominator, as it only improves the bound, and in our applications the second moment will dominate in all important cases.

In what follows, we fix k, z, G and $\mathcal{A}$, as in the assumptions of Lemma 6.4. Let $\tilde{C}_1, ..., \tilde{C}_k$ be the clusters induced by $\mathcal{A}$ on G. The assumptions imply the following fact:

▶ **Fact 6.11.** For any $p \in \tilde{C}_i$, $\frac{\text{cost}(\tilde{C}_i,\mathcal{A})}{2|\tilde{C}_i|} \leq \text{cost}(p,\mathcal{A}) \leq \frac{2\text{cost}(\tilde{C}_i,\mathcal{A})}{|\tilde{C}_i|}$. ◀

We will start with the tiny type, as it is mostly divorced from the others. We will then show that the weight of each cluster is preserved, which implies the cost huge type is preserved as well.

6.3.3 Dealing with Tiny Type

▶ **Lemma 6.12.** It holds that, for any solution $\mathcal{S}$,

$$\max\left(\sum_{p\in I_{tiny,\mathcal{S}}}\text{cost}(p,\mathcal{S}),\sum_{p\in I_{tiny,\mathcal{S}}\cap\Omega}f(p)\text{cost}(p,\mathcal{S})\right)\leq\varepsilon\cdot\text{cost}(G,\mathcal{A}).$$ ◀

Proof. By definition of $I_{tiny,\mathcal{S}}$, $\sum_{p\in I_{tiny,\mathcal{S}}}\text{cost}(p,\mathcal{S})\leq\sum_{p\in I_{tiny,\mathcal{S}}}\frac{\varepsilon}{2}\cdot\text{cost}(p,\mathcal{A})\leq\frac{\varepsilon}{2}\cdot\text{cost}(G,\mathcal{A})$. Similarly, we have for the other term

$$\begin{aligned}\sum_{p\in I_{tiny,\mathcal{S}}\cap\Omega}f(p)\cdot\text{cost}(p,\mathcal{S})&\leq\sum_{p\in I_{tiny,\mathcal{S}}\cap\Omega}f(p)\frac{\varepsilon}{2}\cdot\text{cost}(p,\mathcal{A})\\&\leq\frac{\varepsilon}{2}\sum_{i=1}^{k}\sum_{p\in\tilde{C}_i\cap I_{tiny,\mathcal{S}}\cap\Omega}\frac{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}{\delta\text{cost}(\tilde{C}_i,\mathcal{A})}\cdot\frac{2\cdot\text{cost}(\tilde{C}_i,\mathcal{A})}{|\tilde{C}_i|}\\&\leq\varepsilon\cdot\frac{|I_{tiny,\mathcal{S}}\cap\Omega|}{\delta}\text{cost}(G,\mathcal{A})\\&\leq\varepsilon\cdot\text{cost}(G,\mathcal{A}).\end{aligned}$$

where the last inequality uses that Ω contains δ points. □

6.3.4 Preserving the Weight of Clusters, and the Huge Type

We now consider the huge ranges. For this, we first show that, given we sampled enough points, $|\tilde{C}_i|$ is well approximated for every cluster $\tilde{C}_i$. This lemma will also be used later for the interesting points. We define event $\mathcal{E}$ to be: For all cluster $\tilde{C}_i$ induced by $\mathcal{A}$ on G,

$$\sum_{p\in\tilde{C}_i\cap\Omega}\frac{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}{\text{cost}(\tilde{C}_i,\mathcal{A})\cdot\delta}=(1\pm\varepsilon)\cdot|\tilde{C}_i|$$

▶ **Lemma 6.13.** We have that with probability at least $1-k\cdot z^2\log^2(z/\varepsilon)\exp\left(-O(1)\frac{\varepsilon^2}{k}\delta\right)$, event $\mathcal{E}$ happens. ◀

Proof. Consider any cluster $\tilde{C}_i$ induced by $\mathcal{A}$ on G. The expected number of points sampled from $\tilde{C}_i$ is then at least

$$\mu_i:=\sum_{p\in\tilde{C}_i}\frac{\delta\text{cost}(\tilde{C}_i,\mathcal{A})}{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}=\frac{\delta\text{cost}(\tilde{C}_i,\mathcal{A})}{\text{cost}(G,\mathcal{A})}\geq\frac{\delta}{2k},$$

where the inequality holds by assumption on G. Define the indicator variable of point p from the sample being drawn from $\tilde{C}_i$ as $\mathcal{P}_i(p)$. Using Chernoff bounds, we therefore have

$$\mathbb{P}\left[\left|\sum_{p\in G\cap\Omega} P_i(p) - \mu_i\right| \geq \varepsilon\cdot\mu_i\right] \leq \exp\left(-\frac{\varepsilon^2\cdot\mu_i}{3}\right) \leq \exp\left(-\frac{\varepsilon^2\delta}{6k}\right). \tag{6.4}$$

Now, rescaling $P_i(p)$ by a factor $\frac{|\tilde{C}_i|\cdot\mathrm{cost}(G,\mathcal{A})}{\delta\mathrm{cost}(\tilde{C}_i,\mathcal{A})}$ implies that approximating μ_i up to a $(1\pm\varepsilon)$ factor also approximates $|\tilde{C}_i|$ up to a $(1\pm\varepsilon)$ factor.

The final result follows by applying a union bound for all clusters in all groups. □

We now show that for any cluster $\tilde{C}_i$ with a non-empty huge range, Lemma 6.13 implies that the cost is well approximated – without the need of going through the approximate solution $\tilde{\mathcal{S}}$.

▶ **Lemma 6.14.** Condition on event $\mathcal{E}$. Then, for any solution $\mathcal{S}$, and any i such that there exists $\ell \geq z\log(4z/\varepsilon)$ and a point $p\in\tilde{C}_i$ with $\mathrm{cost}(p,\mathcal{S}) \geq 2^\ell\mathrm{cost}(p,\mathcal{A})$, we have:

$$\left|\mathrm{cost}(\tilde{C}_i,\mathcal{S}) - \sum_{p\in\Omega\cap C_i} f(p)\cdot\mathrm{cost}(p,\mathcal{S})\right| \leq 7\varepsilon\cdot\mathrm{cost}(\tilde{C}_i,\mathcal{S}).$$ ◀

Proof. Let $p\in\tilde{C}_i$ as given in the statement. Using the structure of clusters in a group, this implies for any $q\in\tilde{C}_i$: $\mathrm{cost}(p,q) \leq (\mathrm{dist}(p,\mathcal{A})+\mathrm{dist}(q,\mathcal{A}))^z \leq 3^z\cdot\mathrm{cost}(p,\mathcal{A}) \leq 3^z\cdot 2^{(\ell - z\log(4z/\varepsilon))}\mathrm{cost}(p,\mathcal{A}) \leq (3\varepsilon/4z)^z\cdot\mathrm{cost}(p,\mathcal{S})$. By Lemma 1.2, we therefore have for any point $q\in\tilde{C}_i$

$$\begin{aligned}
\mathrm{cost}(p,\mathcal{S}) &\leq (1+\varepsilon/2z)^{z-1}\,\mathrm{cost}(q,\mathcal{S}) + (1+2z/\varepsilon)^{z-1}\,\mathrm{cost}(p,q)\\
&\leq (1+\varepsilon)\,\mathrm{cost}(q,\mathcal{S}) + \varepsilon\cdot\mathrm{cost}(p,\mathcal{S})\\
\Rightarrow \mathrm{cost}(q,\mathcal{S}) &\geq \frac{1-\varepsilon}{1+\varepsilon}\mathrm{cost}(p,S) \geq (1-2\varepsilon)\mathrm{cost}(p,\mathcal{S})
\end{aligned}$$

By a similar calculation, we can also derive an upper bound of $\mathrm{cost}(q,\mathcal{S}) \leq \mathrm{cost}(p,\mathcal{S})\cdot(1+2\varepsilon)$. Hence, we have

$$\begin{aligned}
\sum_{q\in\Omega\cap\tilde{C}_i}\frac{|\tilde{C}_i|\cdot\mathrm{cost}(G,\mathcal{A})}{\mathrm{cost}(\tilde{C}_i,\mathcal{A})\cdot\delta}\cdot\mathrm{cost}(q,\mathcal{S}) &= (1\pm2\varepsilon)\cdot\mathrm{cost}(p,\mathcal{S})\cdot\sum_{q\in\Omega\cap\tilde{C}_i}\frac{|\tilde{C}_i|\cdot\mathrm{cost}(G,\mathcal{A})}{\mathrm{cost}(\tilde{C}_i,\mathcal{A})\cdot\delta}\\
&= (1\pm2\varepsilon)\cdot\mathrm{cost}(p,\mathcal{S})\cdot(1\pm\varepsilon)\cdot|\tilde{C}_i| \quad (\text{Event }\mathcal{E})\\
&= (1\pm2\varepsilon)\cdot(1\pm\varepsilon)\cdot(1\pm2\varepsilon)\cdot\mathrm{cost}(\tilde{C}_i,\mathcal{S})\\
&= (1\pm7\varepsilon)\cdot\mathrm{cost}(\tilde{C}_i,\mathcal{S}).
\end{aligned}$$

□

6.3.5 Bounding Interesting $I_{\ell,\mathcal{S}}$: a Simple but Suboptimal Analysis.

Now we move onto the most involved case, presenting first a suboptimal analysis of `GroupSample` for the interesting types. As explained in the introduction, our main goal is to design a good estimator and apply Bernstein's inequality to it.

Since the clusters intersecting a huge $I_{\ell,\mathcal{S}}$ are dealt with by Lemma 6.14, we only need to focus on the *interesting clusters*, namely clusters $\tilde{C}$ that satisfy

$$\nexists p \in \tilde{C} \mid \text{cost}(p,\mathcal{S}) \geq \left(\frac{8z}{\varepsilon}\right)^z \cdot \text{cost}(p,\mathcal{A}). \tag{6.5}$$

In other words, a clustering is interesting only if it does not have any point in a huge $I_{\ell,\mathcal{S}}$. This restriction will be crucial to our analysis. Let $L_{\mathcal{S}}$ be a set of interesting clusters (possibly not all of them).[3] For simplicity, we will assimilate $L_{\mathcal{S}}$ and the points contained in the clusters of $L_{\mathcal{S}}$.

We present here a first attempt to show that the cost of interesting points is preserved. Although suboptimal, it serves as a good warm-up for our improved bound.

In this first attempt, we will use the simple estimator $E(L_{\mathcal{S}}) := \sum_{p\in L_{\mathcal{S}}\cap\Omega} f(p)\text{cost}(p,\mathcal{S})$ as an estimator of the cost for points in $L_{\mathcal{S}}$. Note that by choice of the weights $f(p)$, this estimator is unbiased: $\mathbb{E}[E(L_{\mathcal{S}})] = \sum_{p\in L_{\mathcal{S}}} \text{cost}(p,\mathcal{S})$, precisely the quantity we seek to estimate.

To show concentration, we rely on Bernstein's inequality from Theorem 6.10. Hence, the key part of our proof is to bound the variance of the estimator.

▶ **Lemma 6.15.** Let G be a group of points, and $\mathcal{A}$ be a solution. Let $\mathbb{C}$ be an $\mathcal{A}$-approximate centroid set, as in Definition 5.2. It holds with probability

$$1 - \exp\left(k\log|\mathbb{C}| - \frac{\varepsilon^{2+z}}{2^{O(z\log z)}\log^2 1/\varepsilon}\cdot\delta\right)$$

that, for all solution $\tilde{\mathcal{S}} \in \mathbb{C}^k$ and any set of interesting clusters $L_{\tilde{\mathcal{S}}}$ induced by $\mathcal{A}$ on G:

$$|E(L_{\mathcal{S}}) - \mathbb{E}\left[E(L_{\mathcal{S}})\right]| \leq \frac{\varepsilon}{z\log z/\varepsilon}\cdot\text{cost}(G,\mathcal{A})$$

◀

Proof. First, we fix some solution $\mathcal{S}$ and some set of interesting clusters $L_{\mathcal{S}}$, verifying Eq. (6.5). We express $E(L_{\mathcal{S}})$ as a sum of i.i.d variables : $E(L_{\mathcal{S}}) = \sum_{j=1}^{\delta} X_j$, where $X_j = f(\Omega_j)\text{cost}(\Omega_j,\mathcal{S})$ when the j-th sampled point is $\Omega_j \in L_{\mathcal{S}}$, $X_j = 0$ otherwise. Recall that, due to Fact 6.11, the probability that the j-th sampled point is p from some cluster $\tilde{C}$ satisfies $\mathbb{P}[\Omega_j = p] = \frac{\text{cost}(\tilde{C},\mathcal{A})}{|\tilde{C}|\cdot\text{cost}(G,\mathcal{A})} \leq \frac{2\text{cost}(p,\mathcal{A})}{\text{cost}(G,\mathcal{A})}$. From the same fact, $f(p) \leq \frac{2\text{cost}(G,\mathcal{A})}{\delta\text{cost}(p,\mathcal{A})}$.

[3] We define $L_{\mathcal{S}}$ to contain only huge clusters but not all of them in order to relate the cost of solutions from the approximate centroid set $\mathbb{C}$ to the cost of any solution, as it will become clear in Section 6.3.7.

We will rely on Bernstein's inequality (Theorem 6.10). To do this, we need an upper bound on the variance of $E(L_{\mathcal{S}})$, as well as an almost sure upper bound M on every sample. We first bound $\mathbb{E}[X_i^2]$:

$$\begin{aligned}
\mathbb{E}[X_i^2] &= \mathbb{E}\left[\left(f(\Omega_i)\text{cost}(\Omega_i,\mathcal{S})\right)^2\right] \\
&= \sum_{p\in L_{\mathcal{S}}} \left(f(p)\text{cost}(p,\mathcal{S})\right)^2 \Pr\left[\Omega_i = p\right] \\
&\leq \sum_{p\in L_{\mathcal{S}}} \text{cost}(p,\mathcal{S}) \cdot \left(\frac{4z}{\varepsilon}\right)^z \cdot \text{cost}(p,\mathcal{A}) \cdot \left(\frac{2\text{cost}(G,\mathcal{A})}{\delta\text{cost}(p,\mathcal{A})}\right)^2 \frac{2\text{cost}(p,\mathcal{A})}{\text{cost}(G,\mathcal{A})} \\
&\leq \left(\frac{4z}{\varepsilon}\right)^z \cdot \frac{\text{cost}(G,\mathcal{A})}{\delta^2} \sum_{p\in L_{\mathcal{S}}} \text{cost}(p,\mathcal{S}) \\
&\leq \left(\frac{4z}{\varepsilon}\right)^z \cdot \frac{\text{cost}(G,\mathcal{A})\text{cost}(G,\mathcal{S})}{\delta^2} \\
&\leq \left(\frac{4z}{\varepsilon}\right)^z \cdot \frac{\left(\text{cost}(G,\mathcal{A}) + \text{cost}(G,\mathcal{S})\right)^2}{\delta^2}
\end{aligned}$$

Where, in the third line, we upper bounded only one of the $\text{cost}(p,\mathcal{S})$ by $(4z/\varepsilon)^z\text{cost}(p,\mathcal{A})$. Hence, it holds that $\sum_{i=1}^{\delta}\mathbb{E}[X_i^2] \leq \left(\frac{4z}{\varepsilon}\right)^z \cdot \frac{(\text{cost}(G,\mathcal{A})+\text{cost}(G,\mathcal{S}))^2}{\delta}$.

To apply Bernstein's inequality, we also need an upper-bound on the value of X_i: using $\text{cost}(p,\mathcal{S}) \leq \left(\frac{4z}{\varepsilon}\right)^z \text{cost}(p,\mathcal{A})$ and $f(p) \leq \frac{2\text{cost}(G,\mathcal{A})}{\delta\text{cost}(p,\mathcal{A})}$ we get

$$X_i \leq M := 2^{O(z\log z)} \cdot \varepsilon^{-z} \frac{\left(\text{cost}(G,\mathcal{A}) + \text{cost}(G,\mathcal{S})\right)}{\delta}$$

Applying Bernstein's inequality with those bounds on the variance and the value of the X_i, we then have:

$$\begin{aligned}
&\mathbb{P}\left[\left|E(L_{\mathcal{S}}) - \mathbb{E}[E(L_{\mathcal{S}})]\right| > \frac{\varepsilon}{z\log z/\varepsilon} \cdot \left(\text{cost}(G,\mathcal{A}) + \text{cost}(G,\mathcal{S})\right)\right] \\
\leq\ & \exp\left(-\frac{\frac{\varepsilon^2}{z^2\log^2 z/\varepsilon} \cdot \left(\text{cost}(G,\mathcal{A}) + \text{cost}(G,\mathcal{S})\right)^2}{2\sum_{i=1}^{\delta}\text{Var}[X_i] + \frac{1}{3}M \cdot \frac{\varepsilon}{z\log z/\varepsilon} \cdot \left(\text{cost}(G,\mathcal{A}) + \text{cost}(G,\mathcal{S})\right)}\right) \\
\leq\ & \exp\left(-\frac{\varepsilon^{2+z}}{2^{O(z\log z)}\log^2 1/\varepsilon} \cdot \delta\right)
\end{aligned}$$

Hence, for a fixed solution $\mathcal{S}$ and a fixed set of interesting clusters $L_{\mathcal{S}}$, it holds with probability $1 - \exp\left(-\frac{\varepsilon^{2+z}}{2^{O(z\log z)}\log^2 1/\varepsilon} \cdot \delta\right)$ that $|E(L_{\mathcal{S}}) - \mathbb{E}[E(L_{\mathcal{S}})]| > \frac{\varepsilon}{z\log z/\varepsilon} \cdot (\text{cost}(G,\mathcal{A}) + \text{cost}(G,\mathcal{S}))$.

Doing a union-bound over the $\mathbb{C}^k$ many solutions $\mathcal{S}$ and the 2^k many sets of interesting clusters concludes the lemma: it holds with probability $1-\exp\left(k\log\mathbb{C} - \frac{\varepsilon^{2+z}}{2^{O(z\log z)}\log^2 1/\varepsilon} \cdot \delta\right)$ that, for any solution $\mathcal{S} \in \mathbb{C}^k$ and any set of interesting clusters $L_{\mathcal{S}}$, $|E(L_{\mathcal{S}}) - \mathbb{E}[E(L_{\mathcal{S}})]| > \frac{\varepsilon}{z\log z/\varepsilon} \cdot (\text{cost}(G,\mathcal{A}) + \text{cost}(G,\mathcal{S}))$. □

In order to apply Lemma 6.15, note that the quantity $|E(L_{\mathcal{S}}) - \mathbb{E}[E(L_{\mathcal{S}})]|$ is equal to $|\text{cost}(L_{\mathcal{S}} \cap \Omega, \mathcal{S}) - \text{cost}(L_{\mathcal{S}}, \mathcal{S})|$, namely the difference between the cost in the full input and the cost in the coreset of points in $L_{\mathcal{S}}$.

This lemma is enough to conclude that the outcome of `GroupSample` is a coreset, once combined with Lemmas 6.12 and 6.14. To see the end of the proof, one can jump directly to the proof of Lemma 6.4 (in Section 6.3.7) and use use Lemma 6.15 instead of Lemma 6.20. This would give a coreset of size $\tilde{O}\left(k\varepsilon^{-2-z}\right)$, instead of $\tilde{O}\left(k\varepsilon^{-\max(2,z)}\right)$.

6.3.6 Bounding Interesting $I_{\ell,\mathcal{S}}$: Improved Analysis

The shortcoming of the previous estimator is its huge variance, with dependency in ε^{-z}. We present an alternate estimator with small variance, allowing in turn to increase the success probability of the algorithm.

As for the previous estimator, we only need to focus on some interesting clusters $L_{\mathcal{S}}$, namely clusters that do not have any point in a huge $I_{\ell,\mathcal{S}}$ and satisfy Eq. (6.5), important enough to be recalled here: all clusters in $L_{\mathcal{S}}$ verify

$$\nexists p \in \tilde{C} \mid \text{cost}(p, \mathcal{S}) \geq \left(\frac{8z}{\varepsilon}\right)^z \cdot \text{cost}(p, \mathcal{A}). \tag{6.6}$$

Designing a Good Estimator: Reducing the Variance

Our first observation is that we can estimate the cost of points in $I_{\ell,\mathcal{S}} \cap L_{\mathcal{S}}$, for each ℓ independently, instead of estimating directly the cost of $L_{\mathcal{S}}$ as in previous section. For them, we will use the following estimator:

▶ **Definition 6.16.** Let G be a group of points, and $\tilde{C}_i$ be the clusters induced by a solution $\mathcal{A}$ on G. For a given set of interesting clusters $L_{\mathcal{S}}$, we let

$$E_{\ell,\mathcal{S}}(L_{\mathcal{S}}) := \sum_{\tilde{C}_i \in L_{\mathcal{S}}} \sum_{p \in \tilde{C}_i \cap I_{\ell,\mathcal{S}} \cap \Omega} f(p)(\text{cost}(p, \mathcal{S}) - \text{cost}(q_{i,\mathcal{S}}, \mathcal{S})), \tag{6.7}$$

where $q_{i,\mathcal{S}} = \operatorname{argmin}_{p \in \tilde{C}_i} \text{cost}(p, \mathcal{S})$. ◀

$E_{\ell,\mathcal{S}}(L_{\mathcal{S}})$ can be expressed differently:

$$E_{\ell,\mathcal{S}}(L_{\mathcal{S}}) = \sum_{\tilde{C}_i \in L_{\mathcal{S}}} \sum_{p \in \tilde{C}_i \cap I_{\ell,\mathcal{S}} \cap \Omega} f(p)(\text{cost}(p,\mathcal{S}) - \text{cost}(q_{i,\mathcal{S}},\mathcal{S}))$$
$$= \sum_{p \in I_{\ell,\mathcal{S}} \cap L_{\mathcal{S}} \cap \Omega} f(p)\text{cost}(p,\mathcal{S}) - F_{\ell,\mathcal{S}}(L_{\mathcal{S}}), \tag{6.8}$$
$$\text{with } F_{\ell,\mathcal{S}}(L_{\mathcal{S}}) := \sum_{\tilde{C}_i \in L_{\mathcal{S}}} \sum_{p \in \tilde{C}_i \cap I_{\ell,\mathcal{S}} \cap \Omega} f(p)\text{cost}(q_{i,\mathcal{S}},\mathcal{S})$$

$F_{\ell,\mathcal{S}}(L_{\mathcal{S}})$ is a random variable whose value depends on the randomly sampled points Ω (we will discuss $F_{\ell,\mathcal{S}}(L_{\mathcal{S}})$ in more detail later).

Note that the expectation of $E_{\ell,\mathcal{S}}(L_{\mathcal{S}})$ is

$$\mathbb{E}\left[E_{\ell,\mathcal{S}}(L_{\mathcal{S}})\right] = \sum_{p \in I_{\ell,\mathcal{S}} \cap L_{\mathcal{S}}} \frac{\delta\text{cost}(\tilde{C}_i,\mathcal{A})}{|\tilde{C}_i|\text{cost}(G,\mathcal{A})} \cdot f(p)\text{cost}(p,S) - \mathbb{E}[F_{\ell,\mathcal{S}}(L_{\mathcal{S}})]$$
$$= \sum_{p \in I_{\ell,\mathcal{S}} \cap L_{\mathcal{S}}} \frac{\delta\text{cost}(\tilde{C}_i,\mathcal{A})}{|\tilde{C}_i|\text{cost}(G,\mathcal{A})} \cdot \frac{|\tilde{C}_i|\text{cost}(G,\mathcal{A})}{\delta\text{cost}(\tilde{C}_i,\mathcal{A})} \cdot \text{cost}(p,S) - \mathbb{E}[F_{\ell,\mathcal{S}}(L_{\mathcal{S}})]$$
$$= \text{cost}(I_{\ell,\mathcal{S}} \cap L_{\mathcal{S}}, S) - \mathbb{E}[F_{\ell,\mathcal{S}}(L_{\mathcal{S}})],$$

Now instead of attempting to show directly concentration of all $\text{cost}(I_{\ell,\mathcal{S}} \cap L_{\mathcal{S}} \cap \Omega, \mathcal{S})$, we will instead show that:

1. $E_{\ell,\mathcal{S}}(L_{\mathcal{S}})$ is concentrated for all ℓ and $\mathcal{S}$, and
2. $\sum_\ell F_{\ell,\mathcal{S}}(L_{\mathcal{S}})$ is concentrated around its expectation.

The reason for decoupling the two arguments is that $E_{\ell,\mathcal{S}}(L_{\mathcal{S}})$ has a very small variance, for which few samples are sufficient: each term of the sum has magnitude $\text{cost}(p,\mathcal{S}) - \text{cost}(q_{i,\mathcal{S}},\mathcal{S})$ instead of simply $\text{cost}(p,\mathcal{S})$. This difference is crucial to our analysis. Furthermore, event $\mathcal{E}$ from Lemma 6.13 easily leads to a concentration bound on $F_{\mathcal{S}}(L_{\mathcal{S}}) = \sum_\ell F_{\ell,\mathcal{S}}(L_{\mathcal{S}})$.

To establish the gain in variance obtained by subtracting $\text{cost}(q_{i,\mathcal{S}},\mathcal{S})$, we have the following lemma.

▶ **Lemma 6.17.** Let G be a group of points, and $\mathcal{S}$ be an arbitrary solution and $\tilde{C}_i$ be a cluster induced $\mathcal{A}$ on G where all points have same cost, up to a factor 2. Denote by $q_{i,\mathcal{S}} = \operatorname{argmin}_{p \in \tilde{C}_i} \text{cost}(p,\mathcal{S})$. Then for every interesting range with $\ell \geq \log \varepsilon/2$ and every point $p \in \tilde{C}_i \cap I_{\ell,\mathcal{S}}$,

$$w_p := \frac{\text{cost}(p,\mathcal{S}) - \text{cost}(q_{i,\mathcal{S}},\mathcal{S})}{cost(q_{i,\mathcal{S}},\mathcal{A})} \in \left[0, 2^{\ell(1-1/z)} \cdot 2^{O(z \log z)}\right] \quad ◀$$

Proof. Let $w_p = \frac{\text{cost}(p,\mathcal{S}) - \text{cost}(q_{i,\mathcal{S}},\mathcal{S})}{\text{cost}(q_{i,\mathcal{S}},\mathcal{A})}$. By choice of $q_{i,\mathcal{S}}$, $w_p \geq 0$, so we consider the upper bound.

We first show useful inequalities, relating the different solutions. Since $p \in I_{\ell,\mathcal{S}}$, we have:

$$\begin{aligned}\text{cost}(q_{i,\mathcal{S}},\mathcal{S}) &\leq \text{cost}(p,\mathcal{S}) \leq 2^{\ell+1}\text{cost}(p,\mathcal{A})\\ &\leq 2^{\ell+2}\text{cost}(q_{i,\mathcal{S}},\mathcal{A}),\end{aligned}$$

where the last inequality holds since p and $q_{i,\mathcal{S}}$ are in the same cluster and have up to a factor 2 the same cost. We also have that $\text{cost}(p,q_{i,\mathcal{S}}) \leq 2^{z-1}(\text{cost}(p,\mathcal{A}) + \text{cost}(q_{i,\mathcal{S}},\mathcal{A})) \leq 3\cdot 2^{z-1}\text{cost}(q_{i,\mathcal{S}},\mathcal{A})$.

Now, using Lemma 1.2, for any $\alpha \leq 1$,

$$\text{cost}(p,\mathcal{S}) \leq (1+\alpha/z)^{z-1}\text{cost}(q_{i,\mathcal{S}},\mathcal{S}) + \left(1+\frac{z}{\alpha}\right)^{z-1}\text{cost}(p,q_{i,\mathcal{S}})$$

which after rearranging implies

$$\begin{aligned}\text{cost}(p,\mathcal{S}) - \text{cost}(q_{i,\mathcal{S}},\mathcal{S}) &\leq 2\alpha\cdot\text{cost}(q_{i,\mathcal{S}},\mathcal{S}) + \left(\frac{2z}{\alpha}\right)^{z-1}\text{cost}(p,q_{i,\mathcal{S}})\\ &\leq \alpha\cdot 2^{\ell+3}\cdot\text{cost}(q_{i,\mathcal{S}},\mathcal{A}) + \left(\frac{2z}{\alpha}\right)^{z-1}\cdot 3\cdot 2^{z-1}\text{cost}(q_{i,\mathcal{S}},\mathcal{A})\\ &\leq 2^{z+1}\cdot\left(\alpha\cdot 2^{\ell+3} + \left(\frac{2z}{\alpha}\right)^{z-1}\right)\cdot\text{cost}(q_{i,\mathcal{S}},\mathcal{A}).\end{aligned}$$

We optimize the final term with respect to α, which leads to $\alpha = 2^{-\frac{\ell}{z}}$ (ignoring constants that depend on z) and hence an upper bound of

$$\text{cost}(p,\mathcal{S}) - \text{cost}(q_{i,\mathcal{S}},\mathcal{S}) \leq 2^{O(z\log z)}2^{\ell(1-1/z)}\cdot\text{cost}(q_{i,\mathcal{S}},\mathcal{A}).$$

□

Concentration of the Estimator $E_{\ell,\mathcal{S}}(L_{\mathcal{S}})$

First, we show that every estimator $E_{\ell,\mathcal{S}(L_{\mathcal{S}})}$ is tightly concentrated. This follows the lines of the proof of Lemma 6.15, incorporating carefully the result of Lemma 6.17.

▶ **Lemma 6.18.** Let G be a group of points, and $\mathcal{A}$ be a solution. Consider an arbitrary solution $\mathcal{S}$. Then for any set of interesting clusters $L_{\mathcal{S}}$ induced by $\mathcal{A}$ on G, and any estimator $E_{\ell,\mathcal{S}}(L_{\mathcal{S}})$ with $\ell \leq z\log 4z/\varepsilon$, it holds that:

$$|E_{\ell,\mathcal{S}}(L_{\mathcal{S}}) - \mathbb{E}[E_{\ell,\mathcal{S}}(L_{\mathcal{S}})]| \leq \frac{\varepsilon}{z\log z/\varepsilon}\cdot(\text{cost}(G,\mathcal{A}) + \text{cost}(I_{\ell,\mathcal{S}},\mathcal{S}))\,,$$

with probability at least

$$1 - \exp\left(-2^{O(z\log z)}\cdot\frac{\min(\varepsilon^2,\varepsilon^z)}{\log^2 1/\varepsilon}\cdot\delta\right).$$

◀

Proof. In order to simplify the notations, we drop mention of $L_{\mathcal{S}}$ and define $E_{\ell,\mathcal{S}} = E_{\ell,\mathcal{S}}(L_{\mathcal{S}})$.

Lemma 6.17 allows to write slightly differently $E_{\ell,\mathcal{S}}$:

$$E_{\ell,\mathcal{S}} = \sum_{\tilde{C}_i \in L_{\mathcal{S}}} \sum_{p \in \tilde{C}_i \cap I_{\ell,\mathcal{S}} \cap \Omega} f(p) \cdot w_p \text{cost}(q_{i,\mathcal{S}}, \mathcal{S}),$$

with all the weights w_p are in $[0, 2^{\ell(1-1/z)} \cdot 2^{O(z \log z)}]$.

We can also write $E_{\ell,\mathcal{S}}$ as a sum of independent random variables: $E_{\ell,\mathcal{S}} = \sum_{j=1}^{\delta} X_j$, where $X_j = f(\Omega_j) \cdot w_{\Omega_j} \text{cost}(q_{i,\mathcal{S}}, \mathcal{A})$ when the j-th sampled point of G is $\Omega_j \in \tilde{C}_i \cap I_{\ell,\mathcal{S}} \cap L_{\mathcal{S}}$ and $X_j = 0$ when $\Omega_i \notin I_{\ell,\mathcal{S}} \cap L_{\mathcal{S}}$. Recall that, due to Fact 6.11, the probability that the j-th sampled point is p, where $p \in \tilde{C}_i$ satisfies $\mathbb{P}[\Omega_j = p] = \frac{\text{cost}(\tilde{C}_i, \mathcal{A})}{|\tilde{C}_i| \cdot \text{cost}(G, \mathcal{A})} \leq \frac{2\text{cost}(p, \mathcal{A})}{\text{cost}(G, \mathcal{A})}$. From the same fact, $f(p) \leq \frac{2\text{cost}(G, \mathcal{A})}{\delta \text{cost}(p, \mathcal{A})}$.

We will rely on Bernstein's inequality (Theorem 6.10). To do this, we need an upper bound on the variance of $E_{\ell,\mathcal{S}}$, as well as an almost sure upper bound M on every sample. We first bound $\mathbb{E}[X_j^2]$: in the second line, we use that Ω_j consists of a single point to move the square inside the sum.

$$\begin{aligned}
\mathbb{E}[X_j^2] &= \mathbb{E}\left[\left(f(\Omega_j)\text{cost}(\Omega_j, \mathcal{A}) \cdot w_{\Omega_j,\mathcal{S}}\right)^2\right] \\
&= \sum_{p \in I_{\ell,\mathcal{S}} \cap L_{\mathcal{S}}} \left(f(p)\text{cost}(p, \mathcal{A}) \cdot w_{p,\mathcal{S}}\right)^2 \cdot \Pr\left[\Omega_i = p\right] \\
&\leq \sum_{p \in I_{\ell,\mathcal{S}}} \left(\frac{2\text{cost}(G, \mathcal{A})}{\delta \text{cost}(p, \mathcal{A})} \cdot \text{cost}(p, \mathcal{A}) \cdot w_{p,\mathcal{S}}\right)^2 \cdot \Pr\left[\Omega_i = p\right] \\
&\leq \sum_{p \in I_{\ell,\mathcal{S}}} 2^{2\ell(1-1/z)} \cdot 2^{O(z \log z)} \cdot \frac{\text{cost}^2(G, \mathcal{A})}{\delta^2} \cdot \frac{\text{cost}(p, \mathcal{A})}{\text{cost}(G, \mathcal{A})} \\
&\leq \sum_{p \in I_{\ell,\mathcal{S}}} 2^{2\ell(1-1/z)} \cdot 2^{O(z \log z)} \cdot \frac{\text{cost}(G, \mathcal{A})}{\delta^2} \cdot \text{cost}(p, \mathcal{A}),
\end{aligned}$$

where the fourth line follows from using Lemma 6.17 to replace the value of $w_{p,\mathcal{S}}$.

To bound $\sum_{p \in I_{\ell,\mathcal{S}}} \text{cost}(p, \mathcal{A})$, we need to deal with the cases $z = 1$ (i.e. k-median) and $z \geq 2$ (k-means and higher powers) separately. For the former, we have $2^{2\ell(1-1/1)} = 1$, so we can use $\sum_{p \in I_{\ell,\mathcal{S}}} \text{cost}(p, \mathcal{A}) \leq \text{cost}(G, \mathcal{A})$ as an upper bound. For the latter, we use $\sum_{p \in I_{\ell,\mathcal{S}}} 2^{\ell} \cdot \text{cost}(p, \mathcal{A}) \leq \text{cost}(I_{\ell,\mathcal{S}}, \mathcal{S})$ as an upper bound. Combining this with $\text{Var}[X_i] \leq \text{E}[X_i^2]$, we obtain for $z = 1$:

$$\text{Var}[X_i] \leq \frac{\text{cost}(G, \mathcal{A})}{\delta^2} \cdot 2^{O(z \log z)} \cdot \text{cost}(G, \mathcal{A}), \tag{6.9}$$

and for $z > 1$:

$$\text{Var}[X_i] \leq \frac{\text{cost}(G, \mathcal{A})}{\delta^2} \cdot 2^{O(z \log z)} 2^{\ell(1-2/z)} \text{cost}(I_{\ell,\mathcal{S}}, \mathcal{S}). \tag{6.10}$$

The almost sure upper bound (for which no case distinction is required) can be derived similarly , using $X_i \leq \sup \frac{2\text{cost}(G,\mathcal{A})}{\delta \text{cost}(p,\mathcal{A})} \cdot \text{cost}(p,\mathcal{A}) \cdot w_{p,\mathcal{S}}$:

$$\begin{aligned} X_i \quad \leq \quad M \quad &:= \quad 2^{\ell(1-1/z)} \cdot 2^{O(z \log z)} \cdot \frac{\text{cost}(G,\mathcal{A})}{\delta} \\ &\leq \quad \frac{z}{\varepsilon} \cdot 2^{\ell(1-2/z)} \cdot 2^{O(z \log z)} \cdot \frac{\text{cost}(G,\mathcal{A})}{\delta}, \end{aligned} \tag{6.11}$$

where the inequality holds due to $\ell \leq z \log(4z/\varepsilon)$. Applying Bernstein's inequality with Equations 6.9, 6.10, and 6.11, we then have

$$\begin{aligned} &\mathbb{P}\left[|E_{\ell,\mathcal{S}} - \mathbb{E}[E_{\ell,\mathcal{S}}]| \leq \frac{\varepsilon}{z \log z/\varepsilon} \cdot (\text{cost}(G,\mathcal{A}) + \text{cost}(I_{\ell,\mathcal{S}},\mathcal{S}))\right] \\ \leq \quad &\exp\left(-\frac{\frac{\varepsilon^2}{z^2 \log^2 z/\varepsilon} \cdot (\text{cost}(G,\mathcal{A}) + \text{cost}(I_{\ell,\mathcal{S}},\mathcal{S}))^2}{2\sum_{i=1}^{\delta} \text{Var}[X_i] + \frac{1}{3}M \cdot \frac{\varepsilon}{z \log z/\varepsilon} \cdot (\text{cost}(G,\mathcal{A}) + \text{cost}(I_{\ell,\mathcal{S}},\mathcal{S}))}\right) \\ \leq \quad &\exp\left(-\frac{\frac{\varepsilon^2}{z^2 \log^2 z/\varepsilon} \cdot \delta}{2^{O(z \log z)} \cdot \begin{cases} 1 & \text{if } z = 1 \\ 2^{\ell(1-2/z)} & \text{if } z \geq 2 \end{cases}}\right) \end{aligned}$$

For $z = 1$ this becomes $\exp\left(-\frac{\varepsilon^2 \cdot \delta}{2^{O(z \log z)} \log^2 1/\varepsilon}\right)$. For $z = 2$, we have $2^{\ell(1-2/z)} = 1$, so the same bound as for $z = 1$. For $z > 2$, we use $\ell \leq z \log 4z/\varepsilon$, which implies $\varepsilon^2 \cdot 2^{-\ell(1-2/z)} \geq \varepsilon^{2+z-z2/z} \cdot 2^{-O(z \log z)} = \varepsilon^z \cdot 2^{-O(z \log z)}$. This yields our final desired bound of

$$\exp\left(-\frac{\min(\varepsilon^2, \varepsilon^z)}{2^{O(z \log z)} \log^2 1/\varepsilon} \cdot \delta\right).$$

□

Concentration of $F_{\ell,\mathcal{S}}(L_{\mathcal{S}})$

We now turn our attention to bounding the random variable $F_{\ell,\mathcal{S}}(L_{\mathcal{S}})$. It turns out that bounding

$$F_{\ell,\mathcal{S}}(L_{\mathcal{S}}) = \sum_{\tilde{C}_i \in L_{\mathcal{S}}} \sum_{p \in \tilde{C}_i \cap \Omega \cap I_{\ell,\mathcal{S}}} \text{cost}(q_{i,\mathcal{S}},\mathcal{S}) \cdot \frac{|\tilde{C}_i| \cdot \text{cost}(G,\mathcal{A})}{\delta \text{cost}(\tilde{C}_i,\mathcal{A})}$$

is rather hard, and in fact no easier than bounding $\text{cost}(I_{\ell,\mathcal{S}} \cap \Omega, \mathcal{S})$. Fortunately, this is not necessary, as it turns out that we can merely bound the sum of $F_{\ell,\mathcal{S}}(L_{\mathcal{S}})$. We consider the random variable defined as follows:

$$F_{\mathcal{S}}(L_{\mathcal{S}}) = \sum_{\ell \leq z \log(4z/\varepsilon)} F_{\ell,\mathcal{S}}(L_{\mathcal{S}})$$

with expectation

$$\mathbb{E}[F_{\mathcal{S}}(L_{\mathcal{S}})] = \sum_{\tilde{C}_i \in L_{\mathcal{S}}} \sum_{p \in \tilde{C}_i} \text{cost}(q_{i,\mathcal{S}},\mathcal{S}) \cdot \frac{|\tilde{C}_i| \cdot \text{cost}(G,\mathcal{A})}{\delta \text{cost}(\tilde{C}_i,\mathcal{A})} \cdot \Pr[p \in \Omega]$$

Showing that $F_{\mathcal{S}}(L_{\mathcal{S}})$ is concentrated is now an almost direct consequence of event $\mathcal{E}$ from Lemma 6.13, which says that $\sum_{p\in\tilde{C}_i\cap\Omega} \frac{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}{\delta\text{cost}(\tilde{C}_i,\mathcal{A})} = (1\pm\varepsilon)|\tilde{C}_i|$.

▶ **Lemma 6.19.** Let G be a group of points, and $\mathcal{A}$ be a solution. Conditioned on event $\mathcal{E}$, we have for all solutions $\mathcal{S}$ and all sets of interesting clusters $L_{\mathcal{S}}$ induced by $\mathcal{A}$ on G:

$$|F_{\mathcal{S}}(L_{\mathcal{S}}) - \mathbb{E}[F_{\mathcal{S}}(L_{\mathcal{S}})]| \leq \varepsilon\cdot\text{cost}(G,\mathcal{S}).$$ ◀

Proof. Given a solution $\mathcal{S}$ and any set of interesting clusters $L_{\mathcal{S}}$ induced by $\mathcal{A}$ on G, we have

$$\mathbb{E}[F_{\mathcal{S}}(L_{\mathcal{S}})] = \sum_{\tilde{C}_i\in L_{\mathcal{S}}}\sum_{p\in\tilde{C}_i}\text{cost}(q_{i,\mathcal{S}},\mathcal{S})\cdot\frac{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}{\delta\text{cost}(\tilde{C}_i,\mathcal{A})}\Pr[p\in\Omega] = \sum_{\tilde{C}_i\in L_{\mathcal{S}}}|\tilde{C}_i|\cdot\text{cost}(q_{i,\mathcal{S}},\mathcal{S}).$$

Event $\mathcal{E}$ ensures that the mass of each cluster is preserved in the coreset, i.e., that $\sum_{p\in\tilde{C}_i\cap\Omega}\frac{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}{\delta\text{cost}(\tilde{C}_i,\mathcal{A})} = (1\pm\varepsilon)\cdot|\tilde{C}_i|$, for every cluster $\tilde{C}_i\in L_{\mathcal{S}}$. Hence

$$F_{\mathcal{S}}(L_{\mathcal{S}}) = \sum_{\tilde{C}_i\in L_{\mathcal{S}}}\sum_{p\in\tilde{C}_i\cap\Omega}\text{cost}(q_{i,\mathcal{S}},\mathcal{S})\cdot\frac{|\tilde{C}_i|\cdot\text{cost}(G,\mathcal{A})}{\delta\text{cost}(\tilde{C}_i,\mathcal{A})} = (1\pm\varepsilon)\cdot\mathbb{E}[F_{\mathcal{S}}(L_{\mathcal{S}})].$$

Now finally observe that since $q_{i,\mathcal{S}}$ was always the point of $\tilde{C}_i$ whose cost in $\mathcal{S}$ is the smallest, we have $\mathbb{E}[F_S(L_{\mathcal{S}})] \leq \text{cost}(L_{\mathcal{S}},\mathcal{S}) \leq \text{cost}(G,\mathcal{S})$. □

6.3.7 Combining Them All

We can now show that the sample Ω indeed verifies Lemma 6.4. To do that, we naturally follow the structure of previous lemmas, and decompose

$$\left|\text{cost}(G,\mathcal{S}) - \sum_{p\in\Omega} f(p)\cdot\text{cost}(p,\mathcal{S})\right|$$

into terms for which we can apply Lemmas 6.12, 6.14, 6.18, and 6.19.

First, we note that the probability of success of Lemma 6.18 is too small to take a union-bound over its success for all $\mathcal{S}$. To cope with that issue, we use the approximate centroid set, in order to relate $E_{\ell,\mathcal{S}}(L_{\mathcal{S}})$ to $E_{\ell,\tilde{\mathcal{S}}}(L_{\mathcal{S}})$, where $\tilde{\mathcal{S}}$ comes from a small set on which union-bounding is possible.

▶ **Lemma 6.20.** Let G be a group of points, and $\mathcal{A}$ be a solution. Let $\mathbb{C}$ be

an $\mathcal{A}$-approximate centroid set, as in Definition 5.2. It holds with probability

$$1 - \exp\left(k \log |\mathbb{C}| - 2^{O(z \log z)} \cdot \frac{\min(\varepsilon^2, \varepsilon^z)}{\log^2 1/\varepsilon} \cdot \delta\right)$$

that, for all solution $\tilde{\mathcal{S}} \in \mathbb{C}^k$ and any set of interesting clusters $L_{\tilde{\mathcal{S}}}$ induced by $\mathcal{A}$ on G:

$$\left|\text{cost}(L_{\tilde{\mathcal{S}}}, \tilde{\mathcal{S}}) - \text{cost}(\Omega \cap L_{\tilde{\mathcal{S}}}, \tilde{\mathcal{S}})\right| \leq \varepsilon \left(\text{cost}(G, \mathcal{A}) + \text{cost}(L_{\tilde{\mathcal{S}}}, \tilde{\mathcal{S}})\right). \blacktriangleleft$$

Proof. Taking a union-bound over the success of Lemma 6.18 for all possible $\tilde{\mathcal{S}} \in \mathbb{C}^k$, all choice of interesting clusters $L_{\tilde{\mathcal{S}}}$ and all ℓ such that $\log(\varepsilon/2) \leq \ell \leq z \log(4z/\varepsilon)$, it holds with probability $1 - \exp(k \log |\mathbb{C}|) \exp\left(-2^{O(z \log z)} \cdot \frac{\min(\varepsilon^2, \varepsilon^z)}{\log^2 1/\varepsilon} \cdot \delta\right)$ that, for every $\tilde{\mathcal{S}} \in \mathbb{C}^k, L_{\tilde{\mathcal{S}}}$ and ℓ,

$$|E_{\ell,\tilde{\mathcal{S}}}(L_{\tilde{\mathcal{S}}}) - \mathbb{E}[E_{\ell,\tilde{\mathcal{S}}}(L_{\tilde{\mathcal{S}}})]| \leq \frac{\varepsilon}{z \log z/\varepsilon} \cdot \left(\text{cost}(G, \mathcal{A}) + \text{cost}(I_{\ell,\tilde{\mathcal{S}}}, \tilde{\mathcal{S}})\right) \tag{6.12}$$

For simplicity, we drop again the mention of $L_{\tilde{\mathcal{S}}}$ and write $E_{\ell,\tilde{\mathcal{S}}} = E_{\ell,\tilde{\mathcal{S}}}(L_{\tilde{\mathcal{S}}})$, $F_{\tilde{\mathcal{S}}} = F_{\tilde{\mathcal{S}}}(L_{\tilde{\mathcal{S}}})$. We now condition on that event, together with event $\mathcal{E}$. We write:

$$\begin{aligned}
&\left|\sum_{p \in L_{\tilde{\mathcal{S}}}} \text{cost}(p, \tilde{\mathcal{S}}) - \sum_{p \in L_{\tilde{\mathcal{S}}} \cap \Omega} f(p) \cdot \text{cost}(p, \tilde{\mathcal{S}})\right| \\
= &\left|\sum_{p \in L_{\tilde{\mathcal{S}}}} \text{cost}(p, \tilde{\mathcal{S}}) - \mathbb{E}[F_{\tilde{\mathcal{S}}}] + \mathbb{E}[F_{\tilde{\mathcal{S}}}] - F_{\tilde{\mathcal{S}}} + F_{\tilde{\mathcal{S}}} - \sum_{p \in L_{\tilde{\mathcal{S}}} \cap \Omega} f(p) \cdot \text{cost}(p, \tilde{\mathcal{S}})\right| \\
\leq &\left|\sum_{p \in L_{\tilde{\mathcal{S}}}} \text{cost}(p, \tilde{\mathcal{S}}) - \mathbb{E}[F_{\tilde{\mathcal{S}}}] + F_{\tilde{\mathcal{S}}} - \sum_{p \in L_{\tilde{\mathcal{S}}} \cap \Omega} f(p) \cdot \text{cost}(p, \tilde{\mathcal{S}})\right| + |\mathbb{E}[F_{\tilde{\mathcal{S}}}] - F_{\tilde{\mathcal{S}}}| \\
\leq &\sum_{\ell < \log \varepsilon/2} \left|\sum_{p \in I_{\ell,\tilde{\mathcal{S}}} \cap L_{\tilde{\mathcal{S}}}} \text{cost}(p, \tilde{\mathcal{S}}) - \mathbb{E}[F_{\ell,\tilde{\mathcal{S}}}] + F_{\ell,\tilde{\mathcal{S}}} - \sum_{p \in I_{\ell,\tilde{\mathcal{S}}} \cap L_{\tilde{\mathcal{S}}} \cap \Omega} f(p) \cdot \text{cost}(p, \tilde{\mathcal{S}})\right| \qquad (6.13) \\
&+ \sum_{\ell = \log \varepsilon/2}^{z \log z/4\varepsilon} \left|\sum_{p \in I_{\ell,\tilde{\mathcal{S}}} \cap L_{\tilde{\mathcal{S}}}} \text{cost}(p, \tilde{\mathcal{S}}) - \mathbb{E}[F_{\ell,\tilde{\mathcal{S}}}] + F_{\ell,\tilde{\mathcal{S}}} - \sum_{p \in I_{\ell,\tilde{\mathcal{S}}} \cap L_{\tilde{\mathcal{S}}} \cap \Omega} f(p) \cdot \text{cost}(p, \tilde{\mathcal{S}})\right| \qquad (6.14) \\
&+ |\mathbb{E}[F_{\tilde{\mathcal{S}}}] - F_{\tilde{\mathcal{S}}}|
\end{aligned}$$

We note that Equation 6.14 is $\sum_{\ell=\log \varepsilon/2}^{z \log z/4\varepsilon} |E_{\ell,\tilde{\mathcal{S}}} - \mathbb{E}[E_{\ell,\tilde{\mathcal{S}}}]|$ and can be directly bounded using Equation 6.12. To bound tiny points of Equation 6.13, we combine Lemma 6.12

with the observation that $F_{\ell,\tilde{\mathcal{S}}} \leq \sum_{p \in I_{\ell,\tilde{\mathcal{S}}} \cap \Omega} f(p)\text{cost}(p,\tilde{\mathcal{S}})$. This gives:

$$\begin{aligned}
& \sum_{\ell < \log \varepsilon/2} \left| \sum_{p \in I_{\ell,\tilde{\mathcal{S}}} \cap L_{\tilde{\mathcal{S}}}} \text{cost}(p,\tilde{\mathcal{S}}) - \mathbb{E}[F_{\ell,\tilde{\mathcal{S}}}] + F_{\ell,\tilde{\mathcal{S}}} - \sum_{p \in I_{\ell,\tilde{\mathcal{S}}} \cap \Omega} f(p) \cdot \text{cost}(p,\tilde{\mathcal{S}}) \right| \\
\leq & \sum_{\ell < \log \varepsilon/2} \left(\sum_{p \in I_{\ell,\tilde{\mathcal{S}}}} \text{cost}(p,\tilde{\mathcal{S}}) + \mathbb{E}[F_{\ell,\tilde{\mathcal{S}}}] + F_{\ell,\tilde{\mathcal{S}}} + \sum_{p \in I_{\ell,\tilde{\mathcal{S}}} \cap \Omega} f(p) \cdot \text{cost}(p,\tilde{\mathcal{S}}) \right) \\
\leq & \; 2 \sum_{\ell < \log \varepsilon/2} \left(\sum_{p \in I_{\ell,\tilde{\mathcal{S}}}} \text{cost}(p,\tilde{\mathcal{S}}) + \sum_{p \in I_{\ell,\tilde{\mathcal{S}}} \cap \Omega} f(p) \cdot \text{cost}(p,\tilde{\mathcal{S}}) \right) \\
\leq & \; 4\varepsilon \text{cost}(G,\mathcal{A}),
\end{aligned}$$

where the last equation uses Lemma 6.12. Plugging this result into the previous inequality, we have:

$$\begin{aligned}
& \left| \sum_{p \in L_{\tilde{\mathcal{S}}}} \text{cost}(p,\tilde{\mathcal{S}}) - \sum_{p \in L_{\tilde{\mathcal{S}}} \cap \Omega} f(p) \cdot \text{cost}(p,\tilde{\mathcal{S}}) \right| \\
\leq & \; 4\varepsilon \text{cost}(G,\mathcal{A}) + \sum_{\ell = \log \varepsilon/2}^{z \log z/4\varepsilon} \left| \mathbb{E}[E_{\ell,\tilde{\mathcal{S}}}] - E_{\ell,\tilde{\mathcal{S}}} \right| + |\mathbb{E}[F_{\tilde{\mathcal{S}}}] - F_{\tilde{\mathcal{S}}}| \\
\leq & \; 4\varepsilon \text{cost}(G,\mathcal{A}) + \sum_{\ell = \log \varepsilon/2}^{z \log z/4\varepsilon} \frac{\varepsilon}{z \log z/\varepsilon} \cdot \Big(\text{cost}(G,\mathcal{A}) + \text{cost}(I_{\ell,\tilde{\mathcal{S}}}, \tilde{\mathcal{S}}) \Big) + |\mathbb{E}[F_{\tilde{\mathcal{S}}}] - F_{\tilde{\mathcal{S}}}| \\
\leq & \; 4\varepsilon \text{cost}(G,\mathcal{A}) + (z \log(z/4\varepsilon) - \log \varepsilon/2) \cdot \frac{\varepsilon}{z \log z/\varepsilon} \cdot \Big(\text{cost}(G,\mathcal{A}) + \text{cost}(L_{\tilde{\mathcal{S}}}, \tilde{\mathcal{S}}) \Big) \\
& + \varepsilon \cdot \text{cost}(G,\tilde{\mathcal{S}}) \\
\leq & \; O(\varepsilon) \cdot (\text{cost}(G,\mathcal{A}) + \text{cost}(L_{\tilde{\mathcal{S}}}, \tilde{\mathcal{S}})),
\end{aligned}$$

where the second to last inequality used Lemma 6.19.

□

From the approximate centroid set to any solution. We can now finally turn to the proof of Lemma 6.4: it combines the result we show previously for the huge type, and the use of approximate centroid set with the Lemma 6.20 for the interesting and tiny types.

Proof of Lemma 6.4. Let X, k, z, G and $\mathcal{A}$ as in the lemma statement. We condition on event $\mathcal{E}$ happening. Let $\mathcal{S}$ be a set of k points, and $\tilde{\mathcal{S}} \in \mathbb{C}^k$ that approximates best $\mathcal{S}$, as given by the definition of $\mathbb{C}$ (see Definition 5.2). This ensures that for all points p with $\text{dist}(p,\mathcal{S}) \leq \frac{8z}{\varepsilon} \cdot \text{dist}(p,\mathcal{A})$ or $\text{dist}(p,\tilde{\mathcal{S}}) \leq \frac{8z}{\varepsilon} \cdot \text{dist}(p,\mathcal{A})$, we have $|\text{cost}(p,\mathcal{S}) - \text{cost}(p,\tilde{\mathcal{S}})| \leq \frac{\varepsilon}{z \log(z/\varepsilon)}(\text{cost}(p,\mathcal{S}) + \text{cost}(p,\mathcal{A}))$.

Our first step is to deal with points that have $\text{dist}(p,\mathcal{S}) > \frac{4z}{\varepsilon} \cdot \text{dist}(p,\mathcal{A})$, using Lemma 6.14. None of the remaining points is huge with respect to $\tilde{\mathcal{S}}$: hence, they all are in interesting clusters with respect to $\tilde{\mathcal{S}}$. Let $L_{\tilde{\mathcal{S}}}$ be this set of cluster: it can be handled with Lemma 6.20. The remaining of the proof formalizes the argument.

Let $H_{\mathcal{S}}$ be the set of all clusters that are intersecting with some $I_{\ell,\mathcal{S}}$ with $\ell > z\log(4z/\varepsilon)$. We also denote $H_{\mathcal{S}}$ the points contained in those clusters. We decompose the cost difference as follows:

$$\left|\text{cost}(G,\mathcal{S}) - \sum_{p\in\Omega\cap G} f(p)\cdot\text{cost}(p,\mathcal{S})\right| \leq \left|\sum_{p\in G\setminus H_{\mathcal{S}}} \text{cost}(p,\mathcal{S}) - \sum_{p\in(G\setminus H_{\mathcal{S}})\cap\Omega} f(p)\cdot\text{cost}(p,\mathcal{S})\right| \quad (6.15)$$

$$+ \left|\sum_{p\in H_{\mathcal{S}}} \text{cost}(p,\mathcal{S}) - \sum_{p\in H_{\mathcal{S}}\cap\Omega} f(p)\cdot\text{cost}(p,\mathcal{S})\right| \quad (6.16)$$

Since we condition on event $\mathcal{E}$, the term 6.16 is $O(\varepsilon)\cdot(\text{cost}(G,\mathcal{A})+\text{cost}(G,\mathcal{S}))$, using Lemma 6.14. Now we take a closer look at term 6.15. By definition of $\tilde{\mathcal{S}}$, it holds for all points $p \in G\setminus H_{\mathcal{S}}$ that $|\text{cost}(p,\mathcal{S}) - \text{cost}(p,\tilde{\mathcal{S}})| \leq \varepsilon(\text{cost}(p,\mathcal{S})+\text{cost}(p,\mathcal{A}))$. Therefore:

$$\begin{aligned}\left|\sum_{p\in G\setminus H_{\mathcal{S}}} \text{cost}(p,\mathcal{S}) \;-\; \sum_{p\in(G\setminus H_{\mathcal{S}})\cap\Omega} f(p)\cdot\text{cost}(p,\mathcal{S})\right| & \\ \leq \left|\sum_{p\in G\setminus H_{\mathcal{S}}} \text{cost}(p,\tilde{\mathcal{S}}) - \sum_{p\in(G\setminus H_{\mathcal{S}})\cap\Omega} f(p)\cdot\text{cost}(p,\tilde{\mathcal{S}})\right| & \\ + \;\varepsilon\left(\text{cost}(G,\mathcal{S})+\text{cost}(G,\mathcal{A})+\text{cost}(\Omega,\mathcal{S})+\text{cost}(\Omega,\mathcal{A})\right). &\end{aligned}$$

This allows us to focus on bounding the cost difference to solution $\tilde{\mathcal{S}}$ instead of $\mathcal{S}$.

For the remaining points in $G\setminus H_{\mathcal{S}}$, we aim at using Lemma 6.20: for that, we show that $L_{\tilde{\mathcal{S}}} := G\setminus H_{\mathcal{S}}$ contains only interesting clusters with respect to $\tilde{\mathcal{S}}$. Indeed, for any $p \in L_{\tilde{\mathcal{S}}}$, we have $|\text{cost}(p,\mathcal{S}) - \text{cost}(p,\tilde{\mathcal{S}})| \leq \frac{\varepsilon}{z\log z/\varepsilon}(\text{cost}(p,\mathcal{S})+\text{cost}(p,\mathcal{A}))$ by definition of $\tilde{\mathcal{S}}$. Hence,

$$\begin{aligned}\text{cost}(p,\tilde{\mathcal{S}}) &\leq \text{cost}(p,\mathcal{S}) + \frac{\varepsilon}{z\log z/\varepsilon}(\text{cost}(p,\mathcal{S})+\text{cost}(p,\mathcal{A})) \\ &\leq \left((1+\varepsilon)\left(\frac{4\varepsilon}{z}\right)^z + \varepsilon\right)\text{cost}(p,\mathcal{A}) \\ &\leq \left(\frac{8\varepsilon}{z}\right)^z \text{cost}(p,\mathcal{A}),\end{aligned}$$

and p is indeed not huge with respect to $\tilde{\mathcal{S}}$. Therefore, we can apply Lemma 6.20 to

get:

$$\left|\sum_{p\in G\setminus H_{\mathcal{S}}}\mathrm{cost}(p,\tilde{\mathcal{S}})-\sum_{p\in(G\setminus H_{\mathcal{S}})\cap\Omega}f(p)\cdot\mathrm{cost}(p,\tilde{\mathcal{S}})\right|=\left|\sum_{p\in L_{\tilde{\mathcal{S}}}}\mathrm{cost}(p,\tilde{\mathcal{S}})-\sum_{p\in L_{\tilde{\mathcal{S}}}\cap\Omega}f(p)\cdot\mathrm{cost}(p,\tilde{\mathcal{S}})\right|$$
$$\leq\varepsilon(\mathrm{cost}(G,\mathcal{A})+\mathrm{cost}(L_{\tilde{\mathcal{S}}},\tilde{\mathcal{S}}))$$
$$=O(\varepsilon)(\mathrm{cost}(L_{\tilde{\mathcal{S}}},\mathcal{S})+\mathrm{cost}(G,\mathcal{A}))$$

Combining all the equations yields

$$|\mathrm{cost}(G,\mathcal{S})-\mathrm{cost}(\Omega,\mathcal{S})|\leq O(\varepsilon)\cdot(\mathrm{cost}(G,\mathcal{A})+\mathrm{cost}(G,\mathcal{S})+\mathrm{cost}(\Omega,\mathcal{A})+\mathrm{cost}(\Omega,\mathcal{S}))\,.$$

To conclude the proof, it only remains to remove the term $\mathrm{cost}(\Omega,\mathcal{A})+\mathrm{cost}(\Omega,\mathcal{S})$ from the right-hand-side. Applying this inequality for $\mathcal{S}=\mathcal{A}$ and using $\mathrm{cost}(\Omega,\mathcal{A})\leq\mathrm{cost}(G,\mathcal{A})+|\mathrm{cost}(G,\mathcal{A})-\mathrm{cost}(\Omega,\mathcal{A})|$ yields first

$$\mathrm{cost}(\Omega,\mathcal{A})=O(1)\cdot\mathrm{cost}(G,\mathcal{A}).$$

Similarly, we can use $\mathrm{cost}(\Omega,\mathcal{S})\leq\mathrm{cost}(G,\mathcal{S})+|\mathrm{cost}(G,\mathcal{S})-\mathrm{cost}(\Omega,\mathcal{S})|$ to get

$$\mathrm{cost}(\Omega,\mathcal{S})=O(1)\cdot\big(\mathrm{cost}(G,\mathcal{S})+\mathrm{cost}(G,\mathcal{A})\big).$$

Hence, we finally conclude:

$$|\mathrm{cost}(G,\mathcal{S})-\mathrm{cost}(\Omega,\mathcal{S})|\leq O(\varepsilon)\cdot(\mathrm{cost}(G,\mathcal{A})+\mathrm{cost}(G,\mathcal{S}))\,.$$

The probability now follows from taking a union-bound over the failure probability of Lemma 6.13 and Lemma 6.20. Specifically

$$1-\exp\left(k\log|\mathbb{C}|-2^{O(z\log z)}\cdot\frac{\min(\varepsilon^2,\varepsilon^z)}{\log^2 1/\varepsilon}\cdot\delta\right)-k\cdot z^2\log^2(z/\varepsilon)\exp\left(-O(1)\frac{\varepsilon^2}{k}\delta\right)$$

In a given cluster $\tilde{C}_i$ induced by $\mathcal{A}$ on G, the complexity of the algorithm is $O(|\tilde{C}_i|)$: it is both the cost of computing the scaling factor $f(p)$ for all $p\in\tilde{C}_i$, and the cost of sampling δ points using reservoir sampling [159]. Hence, the cost of this algorithm for all clusters is $O(|G|)$. □

6.4 Dealing with the Few Far points: Sampling from Outer Rings

In this section we prove Lemma 6.5, that we recall here:

▶ **Lemma 6.5.** Let (X, dist) be a metric space, k, z be two positive integers, P be a set of clients and $\mathcal{A}$ be a $c_{\mathcal{A}}$-approximate solution solution to (k, z)-clustering on P.
Let G be either a group G_b^O or $G_{\max}^O$. Suppose moreover that there is a $\mathcal{A}$-approximate centroid set $\mathbb{C}$ for (k, z)-clustering on G .
Then, there exists an algorithm `SensitivitySample` running in time $O(|G|)$ that constructs a set Ω of size δ such that it holds with probability $1 - \exp\left(k \log |\mathbb{C}| - 2^{O(z \log z)} \cdot \frac{\varepsilon^2}{\log^2 1/\varepsilon} \cdot \delta\right)$ that, for all sets $\mathcal{S}$ of k centers:

$$|\text{cost}(G, \mathcal{S}) - \text{cost}(\Omega, \mathcal{S})| = \frac{\varepsilon}{z \log z/\varepsilon} \cdot (\text{cost}(\mathcal{S}) + \text{cost}(\mathcal{A})) .$$ ◀

Recall that the `SensitivitySample` procedure merely picks δ points p with probability $\frac{\text{cost}(p,\mathcal{A})}{\text{cost}(G,\mathcal{A})}$. Each of the δ sampled points has a weight $\frac{\text{cost}(G,\mathcal{A})}{\delta \cdot \text{cost}(p,\mathcal{A})}$. The procedure runs in time $O(|G|)$.

The main steps of the proof are as follows.

- First, we consider the cost of the points in G such that $\text{cost}(p, \mathcal{S})$ is at most $4^z \cdot \text{cost}(p, \mathcal{A})$. For this case, we can (almost) directly apply Bernstein's inequality as in the previous section.
- Second, we consider the cost of the points in G such that $\text{cost}(p, \mathcal{S}) > 4^z \cdot \text{cost}(p, \mathcal{A})$. Denote this set by $G_{far,\mathcal{S}}$. For these points, we can afford to replace their cost in $\mathcal{S}$ with the distance to the closest center $c \in \mathcal{A}$ plus the distance from c to the closest center in $\mathcal{S}$. The latter part can be charged to the remaining points of the cluster from the original dataset (i.e., not restricted to group G) which are in much larger number and already paying a similar value in $\mathcal{S}$.

We first analyse the points not in $G_{far,\mathcal{S}}$. For that, we will go through the approximate centroid set $\mathbb{C}$ to afford a union-bound: we show the following lemma.

▶ **Lemma 6.21.** Let $\tilde{\mathcal{S}} \in \mathbb{C}^k$, and define $G_{close,\tilde{\mathcal{S}}}$ to be the set of points of G such that $\text{cost}(p, \tilde{\mathcal{S}}) \leq 5^z \cdot \text{cost}(p, \mathcal{A})$. It holds with probability

$$1 - \exp\left(-2^{-O(z)} \left(\frac{\varepsilon}{\log 1/\varepsilon}\right)^2 \delta\right)$$

that

$$|\text{cost}(G_{close,\tilde{\mathcal{S}}}, \tilde{\mathcal{S}}) - \text{cost}(\Omega \cap G_{close,\tilde{\mathcal{S}}}, \tilde{\mathcal{S}})| \leq \frac{\varepsilon}{z \log z/\varepsilon} \left(\text{cost}(G, \mathcal{A}) + \text{cost}(G_{close,\tilde{\mathcal{S}}}, \tilde{\mathcal{S}})\right)$$

◀

Proof. We aim to use Bernstein's Inequality. Let $E_{close,\tilde{\mathcal{S}}} = \sum_{i=1}^{\delta} X_i$, where $X_i = \frac{\text{cost}(G,\mathcal{A})}{\delta \cdot \text{cost}(p,\mathcal{A})} \cdot \text{cost}(p, \tilde{\mathcal{S}})$ if the i-th sampled point is $p \in G_{close,\tilde{\mathcal{S}}}$ and $X_i = 0$ the i-th

sampled point is $p \notin G_{close,\tilde{\mathcal{S}}}$. Recall that the probability that p is the i-th sampled point is $\frac{\text{cost}(p,\mathcal{A})}{\text{cost}(G,\mathcal{A})}$. We consider the second moment $\mathrm{E}[X_i^2]$:

$$\begin{aligned}
E[X_i^2] &= \sum_{p \in G_{close,\tilde{\mathcal{S}}}} \left(\frac{\text{cost}(G,\mathcal{A})}{\delta \cdot \text{cost}(p,\mathcal{A})} \cdot \text{cost}(p,\tilde{\mathcal{S}}) \right)^2 \cdot \mathbb{P}[p \in \Omega] \\
&= \text{cost}(G,\mathcal{A}) \cdot \sum_{p \in G_{close,\tilde{\mathcal{S}}}} \frac{\text{cost}(p,\tilde{\mathcal{S}})}{\delta^2 \cdot \text{cost}(p,\mathcal{A})} \cdot \text{cost}(p,\tilde{\mathcal{S}}) \\
&\leq \text{cost}(G,\mathcal{A}) \cdot \sum_{p \in G_{close,\tilde{\mathcal{S}}}} \frac{5^z}{\delta^2} \cdot \text{cost}(p,\tilde{\mathcal{S}}) \\
&\leq \frac{5^z}{\delta^2} \cdot \text{cost}(G,\mathcal{A}) \cdot \text{cost}(G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}})
\end{aligned}$$

Furthermore, we have the following upper bound for the maximum value any of the X_i:

$$X_i \leq M := \max_{p \in G_{close,\tilde{\mathcal{S}}}} \frac{\text{cost}(G,\mathcal{A})}{\delta \cdot \text{cost}(p,\mathcal{A})} \cdot \text{cost}(p,\tilde{\mathcal{S}}) \leq \frac{5^z}{\delta} \cdot \text{cost}(G,\mathcal{A}). \tag{6.17}$$

Combining both bounds with Bernstein's inequality now yields

$$\begin{aligned}
&\mathbb{P}[|E_{close,\tilde{\mathcal{S}}} - \mathbb{E}[E_{close,\tilde{\mathcal{S}}}]| \leq \frac{\varepsilon}{z \log z/\varepsilon} \cdot \Big(\text{cost}(G,\mathcal{A}) + \text{cost}(G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}})\Big)] \\
\leq\ & \exp\left(- \frac{\left(\frac{\varepsilon}{z \log z/\varepsilon}\right)^2 \cdot \Big(\text{cost}(G,\mathcal{A}) + \text{cost}(G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}})\Big)^2}{2\sum_{i=1}^{\delta} Var[X_i] + \frac{1}{3}M \cdot \varepsilon \cdot \Big(\text{cost}(G,\mathcal{A}) + \text{cost}(G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}})\Big)} \right) \\
\leq\ & \exp\left(- \frac{\left(\frac{\varepsilon}{z \log z/\varepsilon}\right)^2 \cdot \delta \cdot \Big(\text{cost}(G,\mathcal{A}) + \text{cost}(G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}})\Big)^2}{24^z \cdot \text{cost}(G,\mathcal{A}) \cdot \text{cost}(G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}}) + 4^z \cdot \text{cost}(G,\mathcal{A}) \cdot \varepsilon \cdot \Big(\text{cost}(G,\mathcal{A}) + \text{cost}(G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}})\Big)} \right) \\
\leq\ & \exp\left(-2^{-O(z)} \cdot \left(\frac{\varepsilon}{z \log z/\varepsilon}\right)^2 \cdot \delta \right)
\end{aligned}$$

Noting that $\text{cost}(\Omega \cap G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}}) = \mathbb{E}[E_{close,\tilde{\mathcal{S}}}]$, concludes: we have with probability $1 - \exp\left(-2^{-O(z)} \cdot \left(\frac{\varepsilon}{\log 1/\varepsilon}\right)^2 \cdot \delta \right)$ that:

$$|\text{cost}(G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}}) - \text{cost}(\Omega \cap G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}})| \leq \frac{\varepsilon}{z \log z/\varepsilon} \cdot \Big(\text{cost}(G,\mathcal{A}) + \text{cost}(G_{close,\tilde{\mathcal{S}}},\tilde{\mathcal{S}})\Big)$$

□

Now we turn our attention to $G_{far,\mathcal{S}}$. For this, we analyse the following event $\mathcal{E}_{far}$, similar to $\mathcal{E}$: For all cluster C of solution $\mathcal{A}$ such that $C \cap G \neq \emptyset$

$$\sum_{p \in C \cap G \cap \Omega} \frac{\text{cost}(G,\mathcal{A})}{\delta \cdot \text{cost}(p,\mathcal{A})} \text{cost}(p,\mathcal{A}) = (1 \pm \varepsilon) \cdot \text{cost}(C \cap G,\mathcal{A})$$

▶ **Lemma 6.22.** Event $\mathcal{E}_{far}$ happens with probability at least

$$1 - k \exp\left(\frac{\varepsilon^2}{6k} \cdot \delta\right). \qquad \blacktriangleleft$$

Proof. We aim to use Bernstein's Inequality. Let $E_C = \sum_{i=1}^{\delta} X_i$, where $X_i = \frac{\text{cost}(G,\mathcal{A})}{\delta \cdot \text{cost}(p,\mathcal{A})} \cdot \text{cost}(p,\mathcal{A})$ if the i-th sampled point $p \in C$ and $X_i = 0$ the i-th sampled point $p \notin C$. Recall that the probability that the i-th sampled point is p is $\frac{\text{cost}(p,\mathcal{A})}{\text{cost}(G,\mathcal{A})}$. We consider the second moment $\mathrm{E}[X_i^2]$:

$$\begin{aligned} E[X_i^2] &= \sum_{p \in C \cap G} \left(\frac{\text{cost}(G,\mathcal{A})}{\delta \cdot \text{cost}(p,\mathcal{A})} \cdot \text{cost}(p,\mathcal{A})\right)^2 \cdot \mathbb{P}[p \text{ is the } i\text{-th sampled point}] \\ &= \frac{\text{cost}(G,\mathcal{A})}{\delta^2} \cdot \sum_{p \in C \cap G} \text{cost}(p,\mathcal{A}) \\ &= \frac{\text{cost}(G,\mathcal{A})}{\delta^2} \text{cost}(C \cap G, \mathcal{A}) \\ &\leq \frac{2k}{\delta^2} \cdot \text{cost}^2(C \cap G, \mathcal{A}) \end{aligned}$$

where the final inequality follows since every cluster has cost at least half the average. Indeed, either the group considered is $G^O_{\max}$, and then any cluster verifies $\text{cost}(C \cap G) \geq \frac{1}{k}\text{cost}(R^{\mathcal{A}}_O, \mathcal{A}) \geq \frac{1}{k}\text{cost}(G^O_{\max}, \mathcal{A})$, or all the clusters in G^O_b have an equal cost, up to a factor of 2 – hence none cost less than half of the average.

Furthermore, we have by the same argument the following upper bound for the maximum value any of the X_i:

$$X_i \leq M := \max_{p \in C \cap G} \frac{\text{cost}(G,\mathcal{A})}{\delta \cdot \text{cost}(p,\mathcal{A})} \cdot \text{cost}(p,\mathcal{A}) \leq \frac{2k}{\delta} \cdot \text{cost}(C \cap G, \mathcal{A}).$$

Combining both bounds with Bernstein's inequality now yields

$$\begin{aligned} &\mathbb{P}[|\text{cost}(C \cap G \cap \Omega, \mathcal{A}) - \text{cost}(C \cap G, \mathcal{A})| \leq \varepsilon \cdot \text{cost}(C \cap G, \mathcal{A})] \\ \leq\ & \exp\left(-\frac{\varepsilon^2 \cdot \text{cost}^2(C \cap G, \mathcal{A})}{2\sum_{i=1}^{\delta} Var[X_i] + \frac{1}{3}M \cdot \varepsilon \cdot \text{cost}(C \cap G, \mathcal{A})}\right) \leq \exp\left(-\frac{\varepsilon^2}{6k} \cdot \delta\right) \end{aligned}$$

Reformulating, we now have

$$\sum_{p \in C \cap G \cap \Omega} \frac{\text{cost}(G,\mathcal{A})}{\delta \cdot \text{cost}(p,\mathcal{A})} \text{cost}(p,\mathcal{A}) = (1 \pm \varepsilon) \cdot \text{cost}(C \cap G, \mathcal{A})$$

□

▶ **Lemma 6.23.** Let (X, dist) be a metric space, k, z be two positive integers. Suppose G is either a group G_b^O or $G_{\max}^O$. Let $G_{far,\mathcal{S}} \subset G$ be the set of all clients such that $\text{cost}(p,\mathcal{S}) > 4^z \cdot \text{cost}(p,\mathcal{A})$. Condition on event $\mathcal{E}_{far}$.
Then, the set Ω of size δ constructed by `SensitivitySample` verifies the following. It holds for all sets $\mathcal{S}$ of k centers that:

$$\text{cost}(G_{far,\mathcal{S}},\mathcal{S}) + \text{cost}(\Omega \cap G_{far,\mathcal{S}},\mathcal{S}) \leq \frac{2\varepsilon}{z\log z/\varepsilon} \cdot \text{cost}(\mathcal{S}). \quad \blacktriangleleft$$

Proof. Our aim will be to show that $\max\left(\text{cost}(G_{far,\mathcal{S}},\mathcal{S}), \text{cost}(\Omega \cap G_{far,\mathcal{S}},\mathcal{S})\right) \leq \frac{\varepsilon}{z\log z/\varepsilon} \cdot \text{cost}(\mathcal{S})$. It is key here that we compare to the cost of the full input in $\mathcal{S}$, and not simply the cost of the group G.

First, we fix a cluster $C \in \mathcal{A}$, and show that the total contribution of points of $C \cap G_{far,\mathcal{S}}$ is very cheap compared to $\text{cost}(C,\mathcal{S})$, i.e. that $\text{cost}(G_{far,\mathcal{S}} \cap C,\mathcal{S}) \leq \frac{\varepsilon}{z\log z/\varepsilon} \cdot \text{cost}(C,\mathcal{S})$.

For this, fix a point $p \in G_{far,\mathcal{S}} \cap C$, and let c be the center of cluster C.

Let C_{close} be the points of C with cost at most $\left(\frac{z}{\varepsilon}\right)^z \cdot \frac{\text{cost}(C,\mathcal{A})}{|C|}$. Due to Markov's inequality, most of C's points are in C_{close}: $|C_{close}| \geq (1-\varepsilon/z)\cdot|C|$.

Using that the point p is both in the outer ring of C and in $G_{far,\mathcal{S}}$, we can lower bound the distance from c to $\mathcal{S}$ as follows. Triangle inequality and $\text{cost}(p,\mathcal{S}) > 4^z \cdot \text{cost}(p,c)$, yield $\text{dist}(c,\mathcal{S}) \geq \text{dist}(p,\mathcal{S}) - \text{dist}(p,c) \geq 4\text{dist}(p,c) - \text{dist}(p,c) \geq 3\text{dist}(p,c)$. Since p is from an outer group, it verifies $\text{cost}(p,c) \geq \left(\frac{z}{\varepsilon}\right)^{2z} \cdot \frac{\text{cost}(C,c)}{|C|}$. Combining those two observations yields: $\text{cost}(c,\mathcal{S}) \geq 3^z \text{cost}(p,\mathcal{A}) \geq 3^z \cdot \left(\frac{z}{\varepsilon}\right)^{2z} \cdot \frac{\text{cost}(C,c)}{|C|}$.

Using this and Lemma 1.2, we now have for any $q \in C_{close}$:

$$\begin{aligned}
\text{cost}(c,\mathcal{S}) &\leq (1+\varepsilon/(2z))^{z-1} \cdot \text{cost}(q,\mathcal{S}) + \left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1} \cdot \text{cost}(q,c) \\
&\leq (1+\varepsilon)\text{cost}(q,\mathcal{S}) + \left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1} \cdot \left(\frac{z}{\varepsilon}\right)^{z} \cdot \frac{\text{cost}(C,c)}{|C|} \\
&\leq (1+\varepsilon)\text{cost}(q,\mathcal{S}) + 3^{z-1} \cdot \left(\frac{z}{\varepsilon}\right)^{2z-1} \cdot \frac{\text{cost}(C,c)}{|C|} \\
&\leq (1+\varepsilon)\text{cost}(q,\mathcal{S}) + \frac{\varepsilon}{3z} \cdot \text{cost}(c,\mathcal{S}) \\
\Rightarrow \text{cost}(q,\mathcal{S}) &\geq \frac{1-\varepsilon}{1+\varepsilon} \cdot \text{cost}(c,\mathcal{S}) \\
\Rightarrow \text{cost}(C,\mathcal{S}) &\geq \text{cost}(C_{close},\mathcal{S}) \geq |C_{close}| \cdot \frac{1-\varepsilon}{1+\varepsilon} \cdot cost(c,\mathcal{S}). \qquad (6.18)
\end{aligned}$$

Using additionally that $|C_{close}| \geq (1-\frac{\varepsilon}{z})\cdot|C|$ and $\text{cost}(c,\mathcal{S}) \geq 3^z \cdot \left(\frac{z}{\varepsilon}\right)^{2z} \cdot \frac{\text{cost}(C,c)}{|C|}$, we get:

$$\text{cost}(C,\mathcal{S}) \geq |C_{close}| \cdot \frac{1-\varepsilon}{1+\varepsilon} \cdot 3^z \cdot \left(\frac{z}{\varepsilon}\right)^{2z} \cdot \frac{\text{cost}(C,\mathcal{A})}{|C|} \geq 3^z \cdot \left(\frac{z}{\varepsilon}\right)^{2z-1} \cdot \text{cost}(C,\mathcal{A}). \quad (6.19)$$

We are now equipped to show the first part of the lemma, namely $\text{cost}(G_{far,\mathcal{S}},\mathcal{S}) \leq \frac{\varepsilon}{z\log z/\varepsilon}\cdot\text{cost}(\mathcal{S})$.

Since $G\cap C$ contains only points from the outer ring of C, with distance at least $(z/\varepsilon)^2$ times the average, Markov's inequality implies that $|G\cap C| \leq \left(\frac{\varepsilon}{z}\right)^2\cdot|C|$. Hence, $|G_{far,\mathcal{S}}\cap C| \leq \frac{1}{1-\varepsilon/z}\cdot\left(\frac{\varepsilon}{z}\right)^2\cdot|C_{close}|$. This yields

$$\begin{aligned}
&\text{cost}(G_{far,\mathcal{S}}\cap C,\mathcal{S}) = \sum_{p\in G_{far,\mathcal{S}}\cap C}\text{cost}(p,\mathcal{S})\\
(Lem.1.2)\quad &\leq \sum_{p\in G_{far,\mathcal{S}}\cap C}(1+\varepsilon/2z)^{z-1}\text{cost}(c,\mathcal{S}) + \left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1}\cdot\text{cost}(p,c)\\
&\leq |G_{far,\mathcal{S}}\cap C|\cdot(1+\varepsilon)\cdot\text{cost}(c,\mathcal{S})\\
&\quad + \left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1}\cdot\text{cost}(G_{far,\mathcal{S}}\cap C,\mathcal{A}) \qquad (6.20)\\
&\leq \frac{1+\varepsilon}{1-\varepsilon/z}\cdot\left(\frac{\varepsilon}{z}\right)^2\cdot|C_{close}|\text{cost}(c,\mathcal{S}) + \left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1}\text{cost}(G_{far,\mathcal{S}}\cap C,\mathcal{A})\\
(Eq.\ 6.18)\quad &\leq \frac{(1+\varepsilon)^2}{(1-\varepsilon)^2}\cdot\left(\frac{\varepsilon}{z}\right)^2\cdot\text{cost}(C,\mathcal{S}) + \left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1}\cdot\text{cost}(G_{far,\mathcal{S}}\cap C,\mathcal{A})\\
(Eq.\ 6.19)\quad &\leq \frac{(1+\varepsilon)^2}{(1-\varepsilon)^2}\cdot\left(\frac{\varepsilon}{z}\right)^2\cdot\text{cost}(C,\mathcal{S})\\
&\quad + \left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1}\cdot\frac{1}{3^z}\cdot\left(\frac{\varepsilon}{z}\right)^{2z-1}\cdot\text{cost}(G_{far,\mathcal{S}}\cap C,\mathcal{S}) \qquad (6.21)\\
&\leq \frac{\varepsilon}{z\log z/\varepsilon}\cdot\text{cost}(C,\mathcal{S}) \qquad (6.22)
\end{aligned}$$

Summing this up over all clusters C, we therefore have

$$\text{cost}(G_{far,\mathcal{S}},\mathcal{S}) \leq \frac{\varepsilon}{z\log z/\varepsilon}\cdot\text{cost}(\mathcal{S}) \qquad (6.23)$$

What is left to show is that, in the coreset, the weighted cost of the points in $G_{far,\mathcal{S}}\cap\Omega$ can be bounded similarly. For that, we use event $\mathcal{E}_{far}$ to show that $\sum_{p\in G_{far,\mathcal{S}}\cap C\cap\Omega}\frac{\text{cost}(G,\mathcal{A}_0)}{\text{cost}(p,\mathcal{A}_0)} \approx |G_{far,\mathcal{S}}\cap C|$

In particular, event $\mathcal{E}_{far}$ implies that with probability $1-k'\cdot\exp\left(-O(1)\cdot\frac{\varepsilon^2}{k'}\cdot\delta\right)$ for all clusters C induced by $\mathcal{A}$

$$\begin{aligned}
\sum_{p\in C\cap G\cap\Omega}\frac{\text{cost}(G,\mathcal{A})}{\delta\cdot\text{cost}(p,\mathcal{A})}\cdot\left(\frac{2z}{\varepsilon}\right)^{2z}\cdot\frac{\text{cost}(C,\mathcal{A})}{|C|} &\leq \sum_{p\in C\cap G\cap\Omega}\frac{\text{cost}(G,\mathcal{A})}{\delta\cdot\text{cost}(p,\mathcal{A})}\text{cost}(p,\mathcal{A})\\
&\leq (1+\varepsilon)\cdot\text{cost}(C\cap G,\mathcal{A})
\end{aligned}$$

$$\Rightarrow \sum_{p\in C\cap G\cap\Omega} \frac{\text{cost}(G,\mathcal{A})}{\delta\cdot\text{cost}(p,\mathcal{A})} \leq (1+\varepsilon)\cdot\left(\frac{\varepsilon}{2z}\right)^{2z}\cdot|C|\frac{\text{cost}(C\cap G,\mathcal{A})}{\text{cost}(C,\mathcal{A})} \tag{6.24}$$

$$\leq (1+\varepsilon)\cdot\left(\frac{\varepsilon}{2z}\right)^{2z}\cdot|C| \tag{6.25}$$

Therefore, we have

$$\begin{aligned}
&\text{cost}(G_{far,\mathcal{S}}\cap\Omega\cap C,\mathcal{S})\\
=\ &\sum_{p\in G_{far,\mathcal{S}}\cap C}\frac{\text{cost}(G,\mathcal{A})}{\delta\cdot\text{cost}(p,\mathcal{A})}\cdot\text{cost}(p,\mathcal{S})\\
(\textit{Lem. 1.2})\ \leq\ &\sum_{p\in G_{far,\mathcal{S}}\cap\Omega\cap C}\frac{\text{cost}(G,\mathcal{A})}{\delta\cdot\text{cost}(p,\mathcal{A})}\cdot\left(\left(1+\frac{\varepsilon}{2z}\right)^{z-1}\text{cost}(c,\mathcal{S})\right.\\
&\qquad\left.+\left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1}\cdot\text{cost}(p,c)\right)\\
\leq\ &(1+\varepsilon)\cdot\text{cost}(c,\mathcal{S})\cdot\sum_{p\in G_{far,\mathcal{S}}\cap\Omega\cap C}\frac{\text{cost}(G,\mathcal{A})}{\delta\cdot\text{cost}(p,\mathcal{A})}\\
(\mathcal{E}_{far})\qquad &+\left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1}\cdot(1+\varepsilon)\cdot\text{cost}(C\cap G,\mathcal{A})\\
(\textit{Eq. 6.25})\ \leq\ &(1+\varepsilon)^2\text{cost}(c,\mathcal{S})\cdot\left(\frac{\varepsilon}{2z}\right)^{2z}\cdot|C|+\left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1}\cdot(1+\varepsilon)\cdot\text{cost}(C\cap G,\mathcal{A})\\
\leq\ &(1+\varepsilon)^2\cdot\left(\frac{\varepsilon}{2z}\right)^{2z}\cdot|C|\cdot\text{cost}(c,\mathcal{S})+\left(\frac{2z+\varepsilon}{\varepsilon}\right)^{z-1}\cdot\text{cost}(C,\mathcal{A}) \qquad (6.26)\\
\leq\ &\frac{\varepsilon}{z\log z/\varepsilon}\cdot\text{cost}(C,\mathcal{S})
\end{aligned}$$

where the steps following Equation 6.26 are identical to those used to derive Equation 6.22 from Equation 6.20. Again, summing over all clusters now yields

$$\text{cost}(G_{far,\mathcal{S}}\cap\Omega,\mathcal{S})\leq\frac{\varepsilon}{z\log z/\varepsilon}\cdot\text{cost}(\mathcal{S}),$$

which yields the claim. □

Combining far and close to show Lemma 6.5 The overall proof follows from those lemmas.

Proof of Lemma 6.5. First, we condition on event $\mathcal{E}_{far}$, and on the success of Lemma 6.21 for all solution in $\mathbb{C}^k$. This happens with probability

$$1-k\exp\left(\frac{\varepsilon^2}{k}\cdot\delta\right)-\exp\left(k\log|\mathbb{C}|-2^{-O(z)}\left(\frac{\varepsilon}{\log 1/\varepsilon}\right)^2\delta\right).$$

Let $\mathcal{S}$ be a solution, and $\tilde{\mathcal{S}}$ its corresponding solution in $\mathbb{C}^k$. We break the cost of $\mathcal{S}$ into two parts: points with $\text{cost}(p, tilde\mathcal{S}) \leq 5^z \cdot \text{cost}(p, \mathcal{A})$, on which we can apply Lemma 6.21, on the others, on which we will apply Lemma 6.23.

From Lemma 6.21, we directly get

$$|\text{cost}(G_{close,\tilde{\mathcal{S}}}, \tilde{\mathcal{S}}) - \text{cost}(\Omega \cap G_{close,\tilde{\mathcal{S}}}, \tilde{\mathcal{S}})| \leq \frac{\varepsilon}{z \log z/\varepsilon} \cdot \Big(\text{cost}(G, \mathcal{A}) + \text{cost}(G, \tilde{\mathcal{S}})\Big).$$

Since any point in $G_{close,\tilde{\mathcal{S}}}$ verifies $|\text{cost}(p, \mathcal{S}) - \text{cost}(p, \tilde{\mathcal{S}})| \leq \frac{\varepsilon}{z \log z/\varepsilon}(\text{cost}(p, \mathcal{S}) + \text{cost}(p, \mathcal{A}))$ we can relate this to $\text{cost}(G_{close,\tilde{\mathcal{S}}}, \mathcal{S})$ as follows. First, this implies $\text{cost}(G_{close,\tilde{\mathcal{S}}}, \tilde{\mathcal{S}}) \leq (1+\varepsilon)\text{cost}(G_{close,\tilde{\mathcal{S}}}, \mathcal{S}) + \varepsilon\text{cost}(G, \mathcal{A})$. Hence:

$$\begin{aligned}
& |\text{cost}(G_{close,\tilde{\mathcal{S}}}, \mathcal{S}) - \text{cost}(\Omega \cap G_{close,\tilde{\mathcal{S}}}, \mathcal{S})| \\
\leq\ & |\text{cost}(G_{close,\tilde{\mathcal{S}}}, \tilde{\mathcal{S}}) - \text{cost}(\Omega \cap G_{close,\tilde{\mathcal{S}}}, \tilde{\mathcal{S}})| \\
& \quad + \frac{\varepsilon}{z \log z/\varepsilon}(\text{cost}(G_{close,\tilde{\mathcal{S}}}, \mathcal{S}) + \text{cost}(G_{close,\tilde{\mathcal{S}}}, \mathcal{A})) \\
& \quad + \frac{\varepsilon}{z \log z/\varepsilon}(\text{cost}(G_{close,\tilde{\mathcal{S}}} \cap \Omega, \mathcal{S}) + \text{cost}(G_{close,\tilde{\mathcal{S}}} \cap \Omega, \mathcal{A})) \\
\leq\ & \frac{O(\varepsilon)}{z \log z/\varepsilon}(\text{cost}(G, \mathcal{A}) + \text{cost}(G, \mathcal{S})) \\
& \quad + \frac{\varepsilon}{z \log z/\varepsilon}(\text{cost}(G_{close,\tilde{\mathcal{S}}} \cap \Omega, \mathcal{S}) + \text{cost}(G_{close,\tilde{\mathcal{S}}} \cap \Omega, \mathcal{A}))
\end{aligned}$$

We now deal with the other far points. For this, note that $G \setminus G_{close,\tilde{\mathcal{S}}} \subseteq G_{far,\mathcal{S}}$. Indeed, any point $p \in G \setminus G_{far,\mathcal{S}}$ has its cost preserved by $\tilde{\mathcal{S}}$, and therefore verifies

$$\begin{aligned}
\text{cost}(p, \tilde{\mathcal{S}}) &\leq (1 + \frac{\varepsilon}{z \log z/\varepsilon})\text{cost}(p, \mathcal{S}) + \frac{\varepsilon}{z \log z/\varepsilon}\text{cost}(p, \mathcal{A}) \\
&\leq (1+\varepsilon) \cdot 4^z \text{cost}(p, \mathcal{A}) + \varepsilon\text{cost}(p, \mathcal{A}) \leq 5^z \text{cost}(p, \mathcal{A}).
\end{aligned}$$

Consequently, $G \setminus G_{far,\mathcal{S}} \subseteq G_{close,\tilde{\mathcal{S}}}$, which implies $G \setminus G_{close,\tilde{\mathcal{S}}} \subseteq G_{far,\mathcal{S}}$. Hence, we can use Lemma 6.23:

$$\begin{aligned}
& |\text{cost}(G \setminus G_{close,\tilde{\mathcal{S}}}, \mathcal{S}) - \text{cost}(\Omega \cap (G \setminus G_{close,\tilde{\mathcal{S}}}), \mathcal{S})| \\
\leq\ & \text{cost}(G \setminus G_{close,\tilde{\mathcal{S}}}, \mathcal{S}) + \text{cost}(\Omega \cap (G \setminus G_{close,\tilde{\mathcal{S}}}), \mathcal{S}) \\
\leq\ & \text{cost}(G_{far,\mathcal{S}}, \mathcal{S}) + \text{cost}(\Omega \cap G_{far,\mathcal{S}}, \mathcal{S}) \\
\leq\ & \frac{\varepsilon}{z \log z/\varepsilon} \cdot \text{cost}(\mathcal{S}).
\end{aligned}$$

Hence, adding the two inequalities gives that

$$\begin{aligned}
& |\text{cost}(G, \mathcal{S}) - \text{cost}(\Omega \cap G, \mathcal{S})| \\
\leq & \frac{O(\varepsilon)}{z \log z/\varepsilon} \cdot \text{cost}(\mathcal{S}) + \frac{\varepsilon}{z \log z/\varepsilon}(\text{cost}(G \cap \Omega, \mathcal{S}) + \text{cost}(G \cap \Omega, \mathcal{A})).
\end{aligned}$$

To remove the terms depending on Ω from the right hand side, one can proceed as in the end of Lemma 6.4, applying grossly the previous inequality to get $\text{cost}(G \cap \Omega, \mathcal{S}) = O(1)\text{cost}(\mathcal{S})$ and $\text{cost}(G \cap \Omega, \mathcal{A}) = O(1)\text{cost}(\mathcal{A})$. This concludes the theorem:

$$|\text{cost}(G, \mathcal{S}) - \text{cost}(\Omega \cap G, \mathcal{S})| \leq \frac{O(\varepsilon)}{z \log z/\varepsilon} \cdot (\text{cost}(G, \mathcal{S}) + \text{cost}(G, \mathcal{A})).$$

□

This concludes the coreset construction for the outer groups.

6.5 Partitioning into Well Structured Groups: Proof of Lemma 6.7

In this section, we show that the outcome of the partitioning step satisfies Lemma 6.7, which concludes the list of lemmas needed for Theorem 5.3. The main idea behind that lemma is that the cost of the discarded points can be easily charged onto the other points, either because they individually have a negligible cost (as for points in the inner ring) or because their group has negligible cost (as for $G_{j,min}$ and G^O_{min}). We recall the lemma for convenience:

▶ **Lemma 6.7.** Let (X, dist) be a metric space with a set of clients P, k, z be two positive integers, and $\varepsilon \in \mathbb{R}^*_+$. For every solution $\mathcal{S}$, it holds that

$$|\text{cost}(\mathcal{D}, \mathcal{S}) - \text{cost}(P_1, \mathcal{S})| = O(\varepsilon)\text{cost}(\mathcal{S}),$$

where $\mathcal{D}$ and P_1 are defined in Step 2 of the algorithm. ◀

Recall that the *inner ring* $R_I(C)$ (resp. *outer ring* $R_O(C)$) of a cluster C consists of the points of C with cost at most $(\varepsilon/z)^{2z}\,\Delta_C$ (resp. at least $(z/\varepsilon)^{2z}\,\Delta_C$). The *main ring* $R_M(C)$ consist of all the other points of C.

Recall also that $\mathcal{D}$ contains all points that are either in some inner ring, in some group $G_{j,min}$ or in G^O_{min}. P_1 contains the centers of $\mathcal{A}$, weighted by the number of points from $\mathcal{D}$ in their clusters.

To prove Lemma 6.7, we treat separately the inner ring and the groups $G_{j,min}$ and G^O_{min} in the next two lemmas. For all those lemmas, we fix a metric space, a set of clients P, two positive integers k and z, and $\varepsilon \in \mathbb{R}^*_+$. We also fix $\mathcal{A}$, a solution to (k, z)-clustering on P with cost $\text{cost}(\mathcal{A}) \leq c_{\mathcal{A}}\text{cost}(\text{OPT})$.

▶ **Lemma 6.24.** For any solution $\mathcal{S}$ and any cluster C with center c of $\mathcal{A}$,

$$\big|\text{cost}(R_I(C), \mathcal{S}) - |R_I(C)| \cdot \text{cost}(c, \mathcal{S})\big| \leq \varepsilon(\text{cost}(C, \mathcal{A}) + \text{cost}(R_I(C), \mathcal{S})).$$ ◀

Proof. Let C be a cluster induced by $\mathcal{A}$ with center c, and p be a point in the inner ring $R_I(C)$. We start by bounding $|\text{cost}(p,\mathcal{S}) - \text{cost}(c,\mathcal{S})|$.

Using Lemma 1.2, we get

$$|\text{cost}(p,\mathcal{S}) - \text{cost}(c,\mathcal{S})| \leq \varepsilon \cdot \text{cost}(p,\mathcal{S}) + (1+2z/\varepsilon)^{z-1} \cdot \text{cost}(c,p).$$

Since p is from the inner ring of its cluster, $\text{cost}(c,p) \leq \left(\frac{\varepsilon}{z}\right)^{2z} \Delta_C$, hence $(1 + 2z/\varepsilon)^{z-1}\text{cost}(c,p) \leq (2+\varepsilon)^{z-1} \cdot (\varepsilon/z)^{z+1} \cdot \Delta_C \leq \varepsilon\Delta_C$, for small enough ε.

Summing this over all points of the inner ring yields

$$\begin{aligned}
|\text{cost}(R_I(C),\mathcal{S}) - |R_I(C)| \cdot \text{cost}(c,\mathcal{S})| &\leq \sum_{p \in R_I(C)} |\text{cost}(p,\mathcal{S}) - \text{cost}(c,\mathcal{S})| \\
&\leq \sum_{p \in R_I(C)} \varepsilon\text{cost}(p,\mathcal{S}) + \varepsilon\Delta_C \\
&\leq \varepsilon\text{cost}(R_I(C),\mathcal{S}) + \varepsilon|R_I(C)|\Delta_C \\
&\leq \varepsilon\text{cost}(R_I(C),\mathcal{S}) + \varepsilon\text{cost}(C,\mathcal{A})
\end{aligned}$$

This implies

$$|\text{cost}(R_I(C),\mathcal{S}) - |R_I(C)| \cdot \text{cost}(c,\mathcal{S})| \leq \varepsilon(\text{cost}(C,\mathcal{A}) + \text{cost}(R_I(C),\mathcal{S})).$$

□

▶ **Lemma 6.25.** For any solution $\mathcal{S}$ and any j,

$$\left|\text{cost}(G_{j,min},\mathcal{S}) - \sum_{i=1}^{k} |C_i \cap G_{j,min}| \cdot \text{cost}(c_i,\mathcal{S})\right| \leq \varepsilon \cdot (\text{cost}(R_j,\mathcal{S}) + \text{cost}(R_j,\mathcal{A})).$$

Moreover, for any solution $\mathcal{S}$,

$$\left|\text{cost}(G^O_{min},\mathcal{S}) - \sum_{i=1}^{k} |C_i \cap G^O_{min}| \cdot \text{cost}(c_i,\mathcal{S})\right| \leq \varepsilon \cdot (\text{cost}(\mathcal{S}) + \text{cost}(\mathcal{A})).$$ ◀

Proof. Using Lemma 1.2, for a point p in cluster C_i

$$|\text{cost}(c_i,\mathcal{S}) - \text{cost}(p,\mathcal{S})| \leq \varepsilon\text{cost}(p,\mathcal{S}) + \left(1 + \frac{2z}{\varepsilon}\right)^{z-1} \text{cost}(p,c_i).$$

Let G be a group, either $G_{j,min}$ or G^O_{min}. Summing for all cluster C_i and all $p \in G \cap C_i$,

we now get

$$\begin{aligned}
&\left|\sum_{i=1}^{k} |C_i \cap G| \cdot \mathrm{cost}(c_i, \mathcal{S}) - \mathrm{cost}(G, \mathcal{S})\right| \\
\leq\ & \varepsilon \cdot \mathrm{cost}(G, \mathcal{S}) + \sum_{i=1}^{k} \sum_{p \in G \cap C_i} \left(1 + \frac{2z}{\varepsilon}\right)^{z-1} \mathrm{cost}(p, \mathcal{A}) \\
\leq\ & \varepsilon \cdot \mathrm{cost}(G, \mathcal{S}) + \sum_{i=1}^{k} \left(\frac{3z}{\varepsilon}\right)^{z-1} \mathrm{cost}(C_i \cap G, \mathcal{A}) \\
\leq\ & \varepsilon \cdot \mathrm{cost}(G, \mathcal{S}) + \left(\frac{3z}{\varepsilon}\right)^{z-1} \mathrm{cost}(G, \mathcal{A})
\end{aligned}$$

Now, either $G = G_{j,min}$ for some j, and $\mathrm{cost}(G, \mathcal{A}) \leq \left(\frac{\varepsilon}{4z}\right)^z \cdot \mathrm{cost}(R_j, \mathcal{A})$; or $G = G^O_{min}$, and $\mathrm{cost}(G, \mathcal{A}) \leq \left(\frac{\varepsilon}{4z}\right)^z \cdot \mathrm{cost}(R_O(\mathcal{A}), \mathcal{A}) \leq \left(\frac{\varepsilon}{4z}\right)^z \cdot \mathrm{cost}(\mathcal{A})$.

In both cases, the lemma follows. □

The proof of Lemma 6.7 simply combines those lemmas.

Proof of Lemma 6.7. We decompose $|\mathrm{cost}(\mathcal{D}, \mathcal{S}) - \mathrm{cost}(P_1, \mathcal{S})|$ into terms corresponding to the previous lemmas:

$$\begin{aligned}
|\mathrm{cost}(\mathcal{D}, \mathcal{S}) - \mathrm{cost}(P_1, \mathcal{S})| \leq & \sum_{i=1}^{k} |\mathrm{cost}(R_I(C_i), \mathcal{S}) - |R_I(C_i)| \mathrm{cost}(c_i, \mathcal{S})| \\
& + \sum_{j=2z\log(\varepsilon/z)}^{2z\log(z/\varepsilon)} \left|\mathrm{cost}(G_{j,min}, \mathcal{S}) - \sum_{i=1}^{k} |C_i \cap G_{j,min}| \mathrm{cost}(c_i, \mathcal{S})\right| \\
& + \left|\mathrm{cost}(G^O_{min}, \mathcal{S}) - \sum_{i=1}^{k} |C_i \cap G^O_{min}| \mathrm{cost}(c_i, \mathcal{S})\right| \\
\leq & \sum_{i=1}^{k} \varepsilon(\mathrm{cost}(C_i, \mathcal{A}) + \mathrm{cost}(R_I(C_i), \mathcal{S})) \\
& + 2\varepsilon \mathrm{cost}(\mathcal{S}) + 2\varepsilon \mathrm{cost}(\mathcal{A}) + \varepsilon(\mathrm{cost}(\mathcal{S}) + \mathrm{cost}(\mathcal{A})) \\
\leq & 8\varepsilon c_{\mathcal{A}} \mathrm{cost}(\mathcal{S}),
\end{aligned}$$

where the second inequality uses Lemmas 6.24 and Lemma 6.25. □

6.6 A Coreset of Size $k^2\varepsilon^{-2}$

In this section, we show how to trade a factor ε^{-z} for a factor k in the coreset size.

▶ **Lemma 6.26.** Let (X, dist) be a metric space, P be a set of points, k, z two positive integers and $\mathcal{A}$ a set of $O(k)$ centers such that each for each cluster with center c induced by $\mathcal{A}$, all points of the cluster are at distance between $\left(\frac{\varepsilon}{z}\right)^2 \Delta_C$ and $\left(\frac{z}{\varepsilon}\right)^2 \Delta_C$, for some Δ_C.
Suppose there exists an $\mathcal{A}$-approximate centroid set $\mathbb{C}$ for (k, z)-clustering on P.
Then, there exists an algorithm running in time $O(|P|)$ that constructs a set Ω of size $k \cdot 2^{O(z)} \frac{\log^3(1/\varepsilon)}{\varepsilon^2} (k \log |\mathbb{C}| + \log(1/\pi))$ such that, with probability $1 - \pi$, for any set $\mathcal{S}$ of k centers,

$$|\text{cost}(\mathcal{S}) - \text{cost}(\Omega, \mathcal{S})| = O(\varepsilon)\text{cost}(\mathcal{S}).$$ ◀

We will essentially the framework of the previous parts, with a few adjustments to fit with the new target size. Suppose we initially computed a set of k centers $\mathcal{A}$. First, we can apply `SensitivitySample` to the outer rings, as Lemma 6.5 already gives a coreset size $2^{O(z \log z)} \frac{\varepsilon^2}{\log^2 1/\varepsilon} k \log |\mathbb{C}|$.

For the others, we change slightly the definition of ring, to make them a bit more precise. For every cluster $\mathcal{C}_i$ of $\mathcal{A}$, we partition the points of $\mathcal{C}_i$ into rings $R_{i,j}$ made of points with cost $[\left(\frac{\varepsilon}{z}\right)^{2z} \Delta_{C_i} \cdot 2^j, \left(\frac{\varepsilon}{z}\right)^{2z} \Delta_{C_i} \cdot 2^{j+1}]$, for $j \in \{1, \dots 4z \log(z/\varepsilon)\}$.

The algorithm is as follows: from every $R_{i,j}$, sample δ points uniformly at random (if $|R_{i,j}| \leq \delta$, simply add the whole $R_{i,j}$).

The analysis of this algorithm follows the same line as the main one. Rings are divided into tiny, interesting and huge types; tiny and huge are dealt with as in Lemmas 6.12 and 6.14, and interesting points slightly differently.

From the definition of $R_{i,j}$, we immediately get the following observation.

▶ **Fact 6.27.** For every cluster we have at most $O(z \cdot \log z/\varepsilon)$ non-empty rings in total. ◀

Given a solution $\mathcal{S}$, we consider the type $I_{i,j,\ell} \subset \mathcal{C}_i$ consisting of the points of $R_{i,j}$ with $\frac{\text{cost}(p,\mathcal{S})}{\text{cost}(p,\mathcal{A})}$ in $[2^\ell, 2^{\ell+1}]$. As before, we let $\text{cost}(I_{i,j,\ell}, \mathcal{S}) = \sum_{p \in I_{i,j,\ell}} \text{cost}(p, \mathcal{S})$ and $\text{cost}(I_{j,\ell}, \mathcal{S}) = \sum_{i=1}^{k} \text{cost}(I_{i,j,\ell}, \mathcal{S})$.

Our analysis will distinguish between three cases:

1. $\ell \leq -\log(2/\varepsilon)$, in which case we say that $I_{i,j,\ell}$ is *tiny*.
2. $-\log(2/\varepsilon) \leq \ell \leq j + \log(4z/\varepsilon)$, in which case we say $I_{i,j,\ell}$ is *interesting*.
3. $\ell \geq \log(16z/\varepsilon)$, in which case we say $I_{i,j,\ell}$ is *huge*.

We first consider the huge case. For this, an equivalent of Event $\mathcal{E}$ directly holds: by design of the algorithm, the weight of each ring is preserved. This implies that the huge groups are well approximated.

▶ **Lemma 6.28.** It holds that, for any $R_{i,j}$, and any solution $\mathcal{S}$ with at least one point $p \in R_{i,j}$ such that $\text{cost}(p,\mathcal{S}) \geq \left(\frac{4z}{\varepsilon}\right)^z \text{cost}(p,\mathcal{A})$,

$$\left|\text{cost}(R_{i,j},\mathcal{S}) - \sum_{p\in\Omega\cap R_{i,j}} \frac{|R_{i,j}|}{\delta}\cdot\text{cost}(p,\mathcal{S})\right| \leq 3\varepsilon\cdot\text{cost}(R_{i,j},\mathcal{S}).$$ ◀

Proof. Fix a ring $R_{i,j}$ and let $p \in R_{i,j}$ such that $\text{cost}(p,\mathcal{S}) \geq \left(\frac{4z}{\varepsilon}\right)^z \text{cost}(p,\mathcal{A})$. First, the weight of $R_{i,j}$ is preserved in Ω: since δ points are sampled from $R_{i,j}$, it holds that

$$\sum_{p\in\Omega\cap R_{i,j}} \frac{|R_{i,j}|}{\delta} = |R_{i,j}|$$

Now, let $\mathcal{S}$ be a solution. The condition on p implies, for any $q \in R_{i,j}$: $\text{cost}(p,q) \leq 2^z\text{cost}(p,\mathcal{A}) \leq \left(\frac{\varepsilon}{2z}\right)^z \cdot \text{cost}(p,\mathcal{S})$. By Lemma 1.2, we have therefore for any point $q \in R_{i,j}$

$$\begin{aligned} \text{cost}(p,\mathcal{S}) &\leq (1+\varepsilon/2z)^{z-1}\,\text{cost}(q,\mathcal{S}) + (1+2z/\varepsilon)^{z-1}\,\text{cost}(p,q) \\ &\leq (1+\varepsilon)\,\text{cost}(q,\mathcal{S}) + \varepsilon\cdot\text{cost}(p,\mathcal{S}) \\ \Rightarrow \text{cost}(q,\mathcal{S}) &\geq \frac{1-\varepsilon}{1+\varepsilon}\text{cost}(p,S) \geq (1-2\varepsilon)\text{cost}(p,\mathcal{S}) \end{aligned}$$

Moreover, by a similar calculation, we can also derive an upper bound of $\text{cost}(q,\mathcal{S}) \leq \text{cost}(p,\mathcal{S})\cdot(1+2\varepsilon)$. Hence, combined with $\sum_{p\in\Omega\cap R_{i,j}} \frac{|R_{i,j}|}{\delta} = |R_{i,j}|$, this is sufficient to approximate $\text{cost}(R_{i,j},\mathcal{S})$.

Therefore, the cost of $R_{i,j}$ is well approximated for any solution $\mathcal{S}$ such that there is at least one huge point. □

Next, we consider the interesting cases. The main observation here is that there are only $O(\log 1/\varepsilon)$ many rings per cluster, hence a coarser estimation using Bernstein's inequality is actually sufficient to bound the cost.

▶ **Lemma 6.29.** Consider an $R_{i,j}$ and any solution $\mathcal{S}$ such that all huge $I_{i,j,\ell}$ are empty. It holds with probability at least $1 - \log(z/\varepsilon)\exp(-\frac{\varepsilon^2}{2\cdot 16^z\log^2 z/\varepsilon}\cdot\delta)$ that, for all interesting $I_{i,j,\ell}$:

$$\left|\text{cost}(I_{i,j,\ell},\mathcal{S}) - \sum_{p\in I_{i,j,\ell}\cap\Omega}\text{cost}(p,\mathcal{S})\cdot\frac{|R_{i,j}|}{\delta}\right| \leq \frac{\varepsilon}{\log(z/\varepsilon)}\cdot\left(\text{cost}(R_{i,j},\mathcal{A}) + \text{cost}(R_{i,j},\mathcal{S})\right).$$

◀

Proof. Let $\mathcal{S}$ as in the statement. Let $R := \left(\frac{\varepsilon}{z}\right)^{2z}\Delta_C\cdot 2^j$, such that points in $R_{i,j}$ have cost to $\mathcal{A}$ between R and $2R$, and let $I_{i,j,\ell}$ be some non-empty type.

We start by bounding $|R_{i,j}| \cdot R \cdot 2^\ell$ in terms of $\text{cost}(R_{i,j}, \mathcal{S}) + \text{cost}(R_{i,j}, \mathcal{A})$: this gives an helpful upper bound for $\text{cost}(I_{i,j,\ell}, \mathcal{S})$.

Let $p \in I_{i,j,\ell}$. If $\ell \geq 3z + 1$, then we can use the triangle inequality to lower bound the cost in $\mathcal{S}$ of any point $q \in R_{i,j}$:

$$\begin{aligned}\text{cost}(q, \mathcal{S}) &\geq 2^{-z} \cdot \text{cost}(p, \mathcal{S}) - \text{cost}(p, q) \\ &\geq 2^{-z-1} \cdot 2^\ell \cdot \text{cost}(p, \mathcal{A}) - 2^z \cdot R \\ &\geq (2^{\ell-z-1} - 2^z) \cdot R \geq 2^{\ell-z-2} \cdot R.\end{aligned}$$

Hence, $|R_{i,j}| \cdot R \cdot 2^\ell \leq \text{cost}(R_{i,j}, \mathcal{S}) \cdot 2^{z+2}$.

If $\ell \leq 3z$, then $|R_{i,j}| \cdot R \cdot 2^\ell \leq |R_{i,j}| \cdot R \cdot 2^{3z} \leq \text{cost}(R_{i,j}, \mathcal{A}) \cdot 8^z$. Putting both bounds together, we have

$$|R_{i,j}| \cdot R \cdot 2^\ell \leq 8^z(\text{cost}(R_{i,j}, \mathcal{S}) + \text{cost}(R_{i,j}, \mathcal{A})) \tag{6.27}$$

Since we aim to apply Bernstein's inequality, we now require a bound on the second moment of our cost estimator. We have for a single randomly chosen point P:

$$\mathbb{E}\left[\sum_{p \in I_{i,j,\ell} \cap P} \text{cost}(p, \mathcal{S}) \cdot |R_{i,j}|\right] = \text{cost}(I_{i,j,\ell}, \mathcal{S})$$

and, since $|P| = 1$:

$$\begin{aligned}\mathbb{E}\left[\left(\sum_{p \in I_{i,j,\ell} \cap P} \text{cost}(p, \mathcal{S}) \cdot |R_{i,j}|\right)^2\right] &= \mathbb{E}\left[\sum_{p \in I_{i,j,\ell} \cap P} \text{cost}(p, \mathcal{S})^2 \cdot |R_{i,j}|^2\right] \\ &= \sum_{p \in I_{i,j,\ell}} \text{cost}(p, \mathcal{S})^2 \cdot |R_{i,j}| \\ &\leq |R_{i,j}| \cdot |I_{i,j,\ell}| \cdot 2^{2\ell+2} \cdot R^2 \\ &\leq \text{cost}(I_{i,j,\ell}, \mathcal{S}) \cdot (\text{cost}(R_{i,j}, \mathcal{S}) + \text{cost}(R_{i,j}, \mathcal{A})) \cdot 16^z \end{aligned} \tag{6.28}$$

where the final equation follows from $|I_{i,j,\ell}| \cdot 2^\ell \cdot R \leq \text{cost}(I_{i,j,\ell}, \mathcal{S})$ and using Equation 6.27.

Furthermore, by the same reasoning and again using Equation 6.27, we have the upper bound M on the (weighted) cost in $\mathcal{S}$ of every sampled point:

$$M \leq R \cdot 2^{\ell+1} \cdot |R_{i,j}| \leq (\text{cost}(R_{i,j}, \mathcal{S}) + \text{cost}(R_{i,j}, \mathcal{A})) \cdot 8^z \tag{6.29}$$

Applying Bernstein's inequality and Equations 6.28 and 6.29, we now have (with r being the number of non-empty interesting $I_{i,j,\ell}$ for the ring $R_{i,j}$):

$$\mathbb{P}\left[\left|\mathrm{cost}(I_{i,j,\ell},\mathcal{S}) - \sum_{p\in I_{i,j,\ell}\cap\Omega} \mathrm{cost}(p,\mathcal{S})\cdot\frac{|R_{i,j}|}{\delta}\right| > \frac{\varepsilon}{r}\cdot(\mathrm{cost}(R_{i,j},\mathcal{S})+\mathrm{cost}(R_{i,j},\mathcal{A}))\right]$$
$$\leq \exp\left(-\frac{\frac{\varepsilon^2\cdot\delta}{r^2}\cdot(\mathrm{cost}(R_{i,j},\mathcal{S})+\mathrm{cost}(R_{i,j},\mathcal{A}))}{\mathrm{cost}(I_{i,j,\ell},\mathcal{S})\cdot 16^z + \frac{4\varepsilon}{3r}\cdot(\mathrm{cost}(R_{i,j},\mathcal{S})+\mathrm{cost}(R_{i,j},\mathcal{A}))\cdot 8^z}\right)$$
$$\leq \exp\left(-\frac{\varepsilon^2\cdot\delta}{2r^2 16^z}\right),$$

where the last line uses $\mathrm{cost}(I_{i,j,\ell},\mathcal{S}) \leq \mathrm{cost}(R_{i,j},\mathcal{S}) \leq \mathrm{cost}(R_{i,j},\mathcal{S}) + \mathrm{cost}(R_{i,j},\mathcal{A})$.

Applying a union bound over all $r = O(\log z/\varepsilon)$ interesting sets $I_{i,j,\ell}$, we obtain the above guarantee for all $I_{i,j,\ell}$ simultaneously with probability

$$1 - r\cdot\exp\left(-\frac{\varepsilon^2\cdot\delta}{2r^2 16^z}\right).$$

□

Finally, we conclude:

Proof of Lemma 6.26. As in the proof of Lemma 6.4, we decompose $|\mathrm{cost}(P,\mathcal{S}) - \mathrm{cost}(\Omega,\mathcal{S})|$ into terms corresponding to points of tiny, interesting or huge groups. We only sketch the proof here, the details are the same as for Lemma 6.4. Let $\mathcal{S}$ be a set of k points, and $\tilde{\mathcal{S}} \in \mathbb{C}^k$ that approximates best $\mathcal{S}$, as given by the definition of $\mathbb{C}$ (see Definition 5.2). This ensures that for all points p with $\mathrm{dist}(p,\mathcal{S}) \leq \frac{8z}{\varepsilon}\cdot\mathrm{dist}(p,\mathcal{A})$ or $\mathrm{dist}(p,\tilde{\mathcal{S}}) \leq \frac{8z}{\varepsilon}\cdot\mathrm{dist}(p,\mathcal{A})$, we have $|\mathrm{cost}(p,\mathcal{S}) - \mathrm{cost}(p,\tilde{\mathcal{S}})| \leq \varepsilon(\mathrm{cost}(p,\mathcal{S}) + \mathrm{cost}(p,\mathcal{A}))$.

Our first step is to deal with points that have $\mathrm{dist}(p,\mathcal{S}) > \frac{8z}{\varepsilon}\cdot\mathrm{dist}(p,\mathcal{A})$, using Lemma 6.28. All other points have distance well approximated by $\tilde{\mathcal{S}}$. Then, we can apply Lemma 6.12 and Lemma 6.29 to the set $L_{\tilde{\mathcal{S}}}$ of remaining clusters, since all points in $L_{\tilde{\mathcal{S}}}$ have $\mathrm{cost}(p,\tilde{\mathcal{S}}) \leq \frac{4z}{\varepsilon}\cdot\mathrm{dist}(p,\mathcal{A})$, and so $\mathrm{dist}(p,\mathcal{S}) \leq \frac{8z}{\varepsilon}\cdot\mathrm{dist}(p,\mathcal{A})$ and were not removed by the previous step. This concludes the proof. □

Combining this lemma and Lemma 6.7 gives an analogous to Theorem 5.3. Now, using this lemma instead of Theorem 5.3 in all proofs of section Chapter 7 gives bound with a factor k instead of a ε^{-z}.

Chapter 7

Applications of the Framework: New Coreset Bounds for Various Metric Spaces

In this chapter, we apply the coreset framework to specific metric spaces. For each of them, we show the existence of a small approximate centroid set, and apply Theorem 5.3 to prove the existence of small coresets.

We recall the definition of a centroid set (Definition 5.2): given an instance of (k, z)-clustering and a set of centers $\mathcal{A}$, an $\mathcal{A}$-approximate centroid set $\mathbb{C}$ is a set that satisfies the following: for every solution $\mathcal{S}$, there exists $\tilde{\mathcal{S}} \in \mathbb{C}^k$ such that for all points p that verifies $\text{cost}(p, \mathcal{S}) \leq \left(\frac{8z}{\varepsilon}\right)^z \text{cost}(p, \mathcal{A})$ or $\text{cost}(p, \tilde{\mathcal{S}}) \leq \left(\frac{8z}{\varepsilon}\right)^z \text{cost}(p, \mathcal{A})$, it holds $|\text{cost}(p, \mathcal{S}) - \text{cost}(p, \tilde{\mathcal{S}})| \leq \frac{\varepsilon}{z \log(z/\varepsilon)} (\text{cost}(p, \mathcal{S}) + \text{cost}(p, \mathcal{A}))$.

Theorem 5.3 states that in case there is an $\mathcal{A}$-approximate centroid set $\mathbb{C}$, then there is a linear-time algorithm that constructs with probability $1 - \pi$ a coreset of size

$$\frac{2^{O(z \log z)} \cdot \log^4 1/\varepsilon}{\min(\varepsilon^2, \varepsilon^z)} \left(k \log |\mathbb{C}| + \log\log(1/\varepsilon) + \log(1/\pi)\right).$$

7.1 Overview of our Techniques

We are looking to compute approximate centroid sets for various metric spaces. Recall that points with $\text{dist}(p, \mathcal{S}) \leq \frac{8z}{\varepsilon} \text{dist}(p, \mathcal{A})$ are called *interesting*.

Discrete metric spaces: when the metric space consists of n points, an obvious approximate centroid set is the full metric itself. This directly yields coresets of size $\frac{2^{O(z \log z)} \cdot \log^4 1/\varepsilon}{\min(\varepsilon^2, \varepsilon^z)} (k \log n + \log\log(1/\varepsilon) + \log(1/\pi))$ for discrete metric spaces, and we

will show next chapter that it is optimal.

Metrics with doubling dimension d: $\mathbb{C}$ is simply constructed taking *nets* around each input point. A γ-net of a metric space is a set of points that are at least at distance γ from each other, and such that each point of the metric is at distance at most γ from the net. The existence of γ-nets of small size is one of the key properties of doubling metrics (see Lemma 7.4). For every point p, $\mathbb{C}$ contains an $\varepsilon\text{cost}(p,\mathcal{A})$-net of the points at distance at most $\frac{8z}{\varepsilon} \cdot \text{cost}(p,\mathcal{A})$ from p. If p is an interesting point, there is therefore a center of $\mathbb{C}$ close to its center in $\mathcal{S}$.

However, this only shows that centers from the solution $\tilde{\mathcal{S}} \in \mathbb{C}^k$ are closer than those of $\mathcal{S}$. Showing that none gets too close is a different ballgame. We will see two ways of achieving it. The first one, that we apply for the doubling, treewidth and planar case, is based on the following observation: if a center $s \in \mathcal{S}$ is replaced by a center $\tilde{s}$, that is way closer to a point p than s, then s can be discarded in the first place and be replaced by the center serving p. This is formalized in Lemma 7.1. The other way of ensuring that no center from $\tilde{\mathcal{S}}$ gets too close to a point p is based on guessing distances from points in $\mathcal{S}$ to input points. It can be applied more broadly than Lemma 7.1, but yields larger centroid sets. We will use it only for minor-excluded graphs, for which Lemma 7.1 cannot be applied.

Graphs with treewidth t: The construction of $\mathbb{C}$ is not as easy in graph metrics: we use the existence of small-size *separators*, building on ideas from Baker et al. [15]. Fix a solution $\mathcal{S}$, and suppose that all interesting points are in a region R of the graph, such that the boundary B of R is made of a constant number of vertices. Fix a center $c \in \mathcal{S}$, and suppose c is not in R. Then, to preserve the cost of interesting points, it is enough to have a center c' at the same distance to all points in the boundary B as c.

$\mathbb{C}$ is therefore constructed as follows: for a point p, its distance tuple to $B = \{b_1, ..., b_{|B|}\}$ is the tuple $(d_1, ..., d_{|B|})$, where $d_i = \text{dist}(p, b_i)$ is the distance to b_i. For every distance tuple to B, $\mathbb{C}$ contains one point having approximately that distance tuple to B.

Let $\tilde{c}$ be the point of $\mathbb{C}$ having approximately the same distance tuple to B as c: this ensures that $\forall p,\ \text{cost}(p,c) \approx \text{cost}(p,\tilde{c})$.

It is however necessary to limit the size of $\mathbb{C}$. For that, we approximate the distances to B. This can be done for interesting points p as follows: since we have $\text{dist}(p,c) \leq \varepsilon^{-1}\text{dist}(p,\mathcal{A})$, rouding the distances to their closest multiple of $\varepsilon\text{dist}(p,\mathcal{A})$ ensures that there are only $O(1/\varepsilon^2)$ possibilities, and adds an error $\varepsilon\text{cost}(p,\mathcal{A})$. We show in Section 7.3 how to make this argument formal, and how to remove the assumption that all interesting points are in the same region.

In Minor-excluded graphs: this class of graphs, that includes planar graphs, admits as well small-size shortest-path separators. A construction similar in spirit to the one for treewidth is therefore possible, as presented in Section 7.5. This builds on the work of Braverman et al. [34].

However, due to the nature of the separators – which are small sets of paths, and not

simply small sets of vertices – one cannot apply the idea of Lemma 7.1 to show that no center gets too close. Instead, we will guess the distance from input points to any point in $\mathcal{S}$, allowing to construct $\tilde{\mathcal{S}}$ with the same distances. Of course, this mere idea requires way too many guesses to have a small set $\mathbb{C}$: we see in Section 7.5 how to make it work properly.

We start the section by showing two preprocessing lemmas: the first one is Lemma 7.1, as described above. The second one allows to apply Theorem 5.3 in the case the input set is weighted, so that we can assume the input has only $\text{poly}(k, \varepsilon^{-1})$ many distinct points, by first computing a non-optimal coreset.

7.1.1 Structural Property on Solutions

We also show a structural property on solutions, that we will use in order to show the existence of small approximate centroid sets. Essentially, when replacing a center s by a center in $\mathbb{C}$ we will make an error $\varepsilon\text{cost}(q, \mathcal{A})$ for some q that we can choose: it is necessary to ensure this error is tiny compared to any $\text{cost}(p, s) + \text{cost}(p, \mathcal{A})$.

Given a point q and a center s, we say that a point p is *problematic* with respect to q and s when $\text{dist}(p, \mathcal{A}) + \text{dist}(p, s) \leq \frac{\varepsilon^2}{8z^2}(\text{dist}(q, \mathcal{A}) + \text{dist}(q, s))$. In that case, we cannot bound the error $\text{dist}(q, \mathcal{A}) + \text{dist}(q, s)$ by some quantity depending on $\text{cost}(p, s) + \text{cost}(p, \mathcal{A})$. However, we show the following:

▶ **Lemma 7.1.** Let $\mathcal{S}$ be a solution, such that any input point p verifies $\text{dist}(p, \mathcal{S}) \leq \frac{8z}{\varepsilon} \cdot \text{dist}(p, \mathcal{A})$. There exists a solution $\mathcal{S}' \subseteq \mathcal{S}$ such that

- for all p, it holds that $|\text{cost}(p, \mathcal{S}) - \text{cost}(p, \mathcal{S}')| \leq \frac{\varepsilon}{z \log z/\varepsilon}(\text{cost}(p, \mathcal{S}) + \text{cost}(p, \mathcal{A}))$, and
- for any center $s \in \mathcal{S}'$, let $q = \text{argmin}_{p:\text{dist}(p,s) \leq \frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})}\text{dist}(p, \mathcal{A}) + \text{dist}(p, s)$. There is no problematic point with respect to q and s.

◀

Proof. First, we show that in case there is a problematic point p with respect to some s and q, then we can serve the whole cluster of s by $\mathcal{S}(p)$, the point that serves p in $\mathcal{S}$. We work in this proof with particular solutions, where points are not necessarily assigned to their closest center. This simplifies the proof, but needs particular care at some moments. In particular, we will ensure that $\text{dist}(p, \mathcal{S}(p)) \leq \frac{10z}{\varepsilon} \cdot \text{dist}(p, \mathcal{A})$ is always verified. We will then remove inductively centers with problematic points to construct $\mathcal{S}'$.

Removing a center that has a problematic point.
Let $s \in \mathcal{S}$, and $q = \text{argmin}_{p:\text{dist}(p,s) \leq \frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})}\text{dist}(p, \mathcal{A}) + \text{dist}(p, s)$ as in the statement. Assume furthermore that all points verify $\text{dist}(p, \mathcal{S}) \leq \frac{10z}{\varepsilon} \cdot \text{dist}(p, \mathcal{A})$. Let p be a problematic point with respect s and q, and $\mathcal{S}(p)$ its the center serving p in $\mathcal{S}$. First,

note that since p is problematic, it must be that $\text{dist}(p, \mathcal{S}(p)) \leq \text{dist}(s, p)$: otherwise, p would verify $\text{dist}(p, s) \leq \frac{10z}{\varepsilon}\text{dist}(p, \mathcal{A})$, and the minimality of q would ensure that p is not problematic. Thus, it holds that:

$$\begin{aligned}\text{dist}(s, \mathcal{S}(p)) &\leq \text{dist}(s, p) + \text{dist}(p, \mathcal{S}(p)) \leq 2\text{dist}(s, p) \\ &\leq 2(\text{dist}(s, p) + \text{dist}(p, \mathcal{A})) \leq \frac{\varepsilon^2}{4z^2}(\text{dist}(q, \mathcal{A}) + \text{dist}(q, s)).\end{aligned}$$

Now, let p' be served by s. Using the triangle inequality, we immediately get:

$$\text{dist}(p', \mathcal{S}(p)) \leq \text{dist}(p', s) + \text{dist}(s, \mathcal{S}(p)) \leq \text{dist}(p', s) + \frac{\varepsilon^2}{4z^2}(\text{dist}(q, \mathcal{A}) + \text{dist}(q, s)).$$

Hence, removing the center s and serving s's cluster by $S(p)$ yields a cost increase of $\frac{\varepsilon^2}{4z^2}(\text{dist}(q, \mathcal{A}) + \text{dist}(q, s))$. Additionally, it holds that

$$\begin{aligned}\min_{q' : \text{dist}(q', \mathcal{S}(p)) \leq \frac{10z}{\varepsilon}\text{dist}(q', \mathcal{A})} \text{dist}(q', \mathcal{A}) + \text{dist}(q', \mathcal{S}(p)) &\leq \text{dist}(p, \mathcal{S}(p)) + \text{dist}(p, \mathcal{A}) \\ &\leq \text{dist}(s, p) + \text{dist}(p, \mathcal{A}) \\ &\leq \frac{\varepsilon^2}{8z^2}(\text{dist}(q, \mathcal{A}) + \text{dist}(q, s))\end{aligned} \tag{7.1}$$

Hence, if $\mathcal{S}(p)$ is removed as well, the additional error for points served by s will be an $\frac{\varepsilon^2}{8z^2}$-fraction of the initial error. This implies that the total error will not accumulate, as we will now see.

Constructing $\mathcal{S}'$. To construct $\mathcal{S}'$, we proceed iteratively: start with $\mathcal{S}' = \mathcal{S}$, and as long as there exists a center s that have a problematic point p with respect to it, remove s and reassign the whole cluster of s to $\mathcal{S}'(p)$, the closest point to p in the current solution. This process must end, as there is no problematic point when there is a single center.

For a point p, let $s_1, ..., s_j$ be the successive clusters it is reassigned to, with corresponding $q_1, ..., q_j$. Using Eq. (7.1), it holds that $\text{dist}(q_{i+1}, \mathcal{A}) + \text{dist}(q_{i+1}, s_{i+1}) \leq \frac{\varepsilon^2}{8z^2}(\text{dist}(q_i, \mathcal{A}) + \text{dist}(q_i, s_i))$. Hence, the distance increase for p is geometric: using that $\text{dist}(q_1, \mathcal{A}) + \text{dist}(q_1, s_1) \leq \text{dist}(p, \mathcal{A}) + \text{dist}(p, \mathcal{S})$ (which holds by minimality of q_1), we get that at any given step i it holds that

$$\begin{aligned}\text{dist}(p, s_i) &\leq \text{dist}(p, \mathcal{S}) + (\text{dist}(p, \mathcal{A}) + \text{dist}(p, \mathcal{S})) \sum_{\ell=1}^{i} \left(\frac{\varepsilon^2}{8z^2}\right)^{\ell} \\ &\leq \text{dist}(p, \mathcal{S}) + \frac{\varepsilon^2}{4z^2}(\text{dist}(p, \mathcal{A}) + \text{dist}(p, \mathcal{S})) \\ &\leq \frac{8z}{\varepsilon} \cdot \text{dist}(p, \mathcal{A}) + \frac{\varepsilon^2}{4z^2} \cdot (1 + \frac{8z}{\varepsilon})\text{dist}(p, \mathcal{A}) \\ &\leq \frac{10z}{\varepsilon} \cdot \text{dist}(p, \mathcal{A}),\end{aligned} \tag{7.2}$$

as promised in order to remove centers: the argument of previous paragraph therefore holds.

Last, we show that the first bullet of the lemma holds. We consider now the standard assignment: p is assigned to its closest center of $\mathcal{S}'$, instead of s_j. First, since we only removed centers, it holds that $\text{cost}(p,\mathcal{S}) \leq \text{cost}(p,\mathcal{S}')$. Second, combining Lemma 1.2 with Eq. (7.2), we have for $\varepsilon' = \frac{\varepsilon}{z \log z/\varepsilon}$:

$$\begin{aligned}
\text{cost}(p,\mathcal{S}') &\leq \text{cost}(p,s_j) \\
&\leq (1+\varepsilon')\text{cost}(p,\mathcal{S}) + \left(\frac{4z}{\varepsilon'}\right)^{z-1} \cdot \left(\frac{\varepsilon^2}{4z^2}\right)^{z} \cdot (\text{dist}(p,\mathcal{A}) + \text{dist}(p,\mathcal{S}))^z \\
&\leq (1+\varepsilon')\text{cost}(p,\mathcal{S}) + \left(\frac{\varepsilon}{z}\right)^{z} \cdot (2\log z/\varepsilon)^{z-1} \cdot (\text{cost}(p,\mathcal{A}) + \text{cost}(p,\mathcal{S})) \\
&\leq (1+\varepsilon')\text{cost}(p,\mathcal{S}) + \varepsilon'(\text{cost}(p,\mathcal{A}) + \text{cost}(p,\mathcal{S})),
\end{aligned}$$

where the last inequality holds when $(\varepsilon' 2\log^2(z/\varepsilon))^{z-1} \leq \varepsilon'$, which is true for any $\varepsilon < 1/3$. Hence, we conclude that

$$\left|\text{cost}(p,\mathcal{S}') - \text{cost}(p,\mathcal{S})\right| \leq \frac{2eps}{z\log z/\varepsilon}\left(\text{cost}(p,\mathcal{S}) + \text{cost}(p,\mathcal{A})\right).$$

Rescaling ε by 2 concludes the lemma. □

7.1.2 From Weighted to Unweighted Inputs

We start by showing a simple reduction from weighted to unweighted inputs, that will be convenient to apply recursively our constructions and remove the dependency in $|P|$, as sketched in Section 5.3. Essentially, we convert a point with weight w to w copies of the point. This is inspired by Feldman et al. [78], and yields the following corollary of Theorem 5.3:

▶ **Corollary 7.2.** Let $\varepsilon, \pi > 0$. Let (X, dist) be a metric space, P a set of clients with weights $w : P \to \mathbb{R}^+$ and two positive integers k and z. Let also $\mathcal{A}$ be a constant-factor approximation for (k,z)-clustering on P with weights. Suppose there exists a $\mathcal{A}$-approximate centroid set, denoted $\mathbb{C}$. Then, there exists an algorithm running in time $O(|P|)$ that constructs with probability at least $1-\pi$ a positively-weighted coreset of size

$$O\left(\frac{2^{O(z\log z)} \cdot \log^4 1/\varepsilon}{\min(\varepsilon^2, \varepsilon^z)}\left(k\log|\mathbb{C}| + \log\log(1/\varepsilon) + \log(1/\pi)\right)\right)$$

for the (k,z)-clustering problem on P with weights. ◀

Proof. We start by making all weights integers: let $w_{min} = \min_{p\in P} w(p)$, and $\tilde{w}(p) = \left\lfloor 2\frac{w(p)}{\varepsilon w_{min}} \right\rfloor$. This definition ensures that

$$\forall p,\ \left|w(p) - \frac{\varepsilon w_{min}}{2}\cdot\tilde{w}(p)\right| \leq \frac{\varepsilon}{2}w_{min} \leq \frac{\varepsilon}{2}w(p).$$

We denote $\tilde{P}$ the set of points P with weight $\tilde{w}$. First, we note that for any solution $\mathcal{S}$,

$$\left|\text{cost}(P,\mathcal{S}) - \varepsilon w_{min}\text{cost}(\tilde{P},\mathcal{S})\right| \leq \frac{\varepsilon}{2}\text{cost}(P,\mathcal{S}).$$

Hence, it is enough to find an $\varepsilon/2$-coreset for $\tilde{P}$, and then scale the coreset weights of the coreset points by $\varepsilon w_{min}/2$. We have that the weights in $\tilde{P}$ are integers: a weighted point can therefore be considered as multiple copies of the same points.

By the previous equation, $\mathcal{A}$ is a constant-factor approximation for $\tilde{P}$ as well. The definition of a centroid set does not depend on weights, so $\mathbb{C}$ is a $\mathcal{A}$-centroid set for $\tilde{P}$ as well. Hence, we can apply Theorem 5.3 on $\tilde{P}$ and scale the resulting coreset by $\varepsilon w_{min}/2$ to conclude the proof. □

7.2 Metrics with Bounded Doubling Dimension

We start by defining the Doubling Dimension of a metric space, and stating a key lemma.

Consider a metric space (X, dist). For a point $p \in X$ and an integer $r \geq 0$, we let $\beta(p,r) = \{x \in X \mid \text{dist}(p,x) \leq r\}$ be the *ball* around p with radius r.

▶ **Definition 7.3.** The *doubling dimension* of a metric is the smallest integer d such that any ball of radius $2r$ can be covered by 2^d balls of radius r. ◀

Notably, the Euclidean space $\mathbb{R}^d$ has doubling dimension $\Theta(d)$.

A γ-*net* of V is a set of points $X \subseteq V$ such that for all $v \in V$ there is an $x \in X$ such that $\text{dist}(v,x) \leq \gamma$, and for all $x, y \in X$ we have $\text{dist}(x,y) > \gamma$. A net is therefore a set of points not too close to each other, such that every point of the metric is close to a net point. The following lemma bounds the cardinality of a net in doubling metrics.

▶ **Lemma 7.4 (from Gupta et. al [91]).** Let (V, dist) be a metric space with doubling dimension d and, diameter D, and let X be a γ-net of V. Then $|X| \leq 2^{d \cdot \lceil \log_2(D/\gamma) \rceil}$. ◀

The goal of this section is to prove the following lemma. Combined with Theorem 5.3, it ensures the existence of small coreset in graphs with small doubling dimension.

▶ **Lemma 7.5.** Let $M = (X, \text{dist})$ be a metric space with doubling dimension d, let $P \subset X$, let k and z be positive integers and let $\varepsilon > 0$. Further, let $\mathcal{A}$ be a $c_{\mathcal{A}}$-approximate solution with at most k centers. There exists an $\mathcal{A}$-approximate

centroid set for P of size

$$|P| \cdot \left(\frac{z}{\varepsilon}\right)^{O(d)}$$ ◀

A direct corollary of that lemma is the existence of a coreset in Doubling Metrics, as it is enough to show the mere existence of a small centroid set for applying Corollary 7.2.

▶ **Corollary 7.6.** Let $M = (X, \mathrm{dist})$ be a metric space with doubling dimension d, and two positive integers k and z.
There exists an algorithm with running time $\tilde{O}(nk)$ that constructs an ε-coreset for (k, z)-clustering on $P \subseteq X$ with size

$$\frac{2^{O(z \log z)} \log^5 1/\varepsilon}{\min(\varepsilon^2, \varepsilon^z)} (kd + k \log k + \log 1/\pi) .$$ ◀

Proof. We first compute a coreset of size $\tilde{O}(k^3 d \varepsilon^{-2})$, in time $O(nk)$ using the standard sensitivity sampling algorithm [98]. Then, combining Corollary 7.2 to compute a coreset of the coreset and Lemma 7.5 yields an algorithm constructing a coreset of size

$$\frac{2^{O(z \log z)} \log^4 1/\varepsilon}{\min(\varepsilon^2, \varepsilon^z)} (kd \log 1/\varepsilon + k \log(kd/\varepsilon) + \log 1/\pi) .$$

If $\log k > d$ then $\log(kd) = O(\log k)$. If $d > \log k$ then $kd + k \log(kd) = O(kd)$, hence the claimed bound follows. □

Proof of Lemma 7.5. For each point $p \in P$, let c be the center to which p was assigned in $\mathcal{A}$. Let $\beta\left(p, \left(\frac{8z}{\varepsilon}\right) \mathrm{dist}(p, c)\right)$ be the metric ball centered around p with radius $\left(\frac{8z}{\varepsilon}\right) \cdot \mathrm{dist}(p, c)$, and let N_p be an $\left(\frac{\varepsilon}{4z}\right) \cdot \mathrm{dist}(p, \mathcal{A})$-net of that ball.

Due to Lemma 7.4, N_p has size $(\varepsilon/z)^{-O(d)}$. Additionally, let s_f be a point not in any $\beta(p, \left(\frac{10z}{\varepsilon}\right) \mathrm{dist}(p, \mathcal{A}))$, if such a point exist.

Let $\mathbb{C} := \{s_f\} \bigcup_{p \in Y} N_p$. We claim that $\mathbb{C}$ is the desired approximate centroid set.

For a candidate solution $\mathcal{S}$, apply first Lemma 7.1, so that we can assume that for any center $s \in \mathcal{S}$, and $q = \mathrm{argmin}_{p:\mathrm{dist}(p,s) \leq \frac{10z}{\varepsilon} \mathrm{dist}(p,\mathcal{A})} \mathrm{dist}(p, \mathcal{A}) + \mathrm{dist}(p, s)$, there is no problematic point with respect to q and s.

let $\tilde{\mathcal{S}}$ be the solution obtained by replacing every center $s \in \mathcal{S}$ by $\tilde{s} \in \mathbb{C}$ as follows: let $q = \mathrm{argmin}_{p:\mathrm{dist}(p,s) \leq \frac{10z}{\varepsilon} \mathrm{dist}(p,\mathcal{A})} \mathrm{dist}(p, \mathcal{A}) + \mathrm{dist}(p, s)$. Pick $\tilde{s}$ to be the closest point to s in N_q. If such a q does not exist, pick $\tilde{s} = s_f$.

Now, let p be a point such that $\mathrm{cost}(p, \mathcal{S}) \leq \left(\frac{8z}{\varepsilon}\right)^z \cdot \mathrm{cost}(p, \mathcal{A})$, let s be any center in $\mathcal{S}$ and assume that $q = \mathrm{argmin}_{p:\mathrm{dist}(p,s) \leq \frac{10z}{\varepsilon} \mathrm{dist}(p,\mathcal{A})} \mathrm{dist}(p, \mathcal{A}) + \mathrm{dist}(p, s)$ exists. Then, by construction of $\tilde{\mathcal{S}}$, there is a center $\tilde{s}$ with $\mathrm{dist}(s, \tilde{s}) \leq \left(\frac{\varepsilon}{4z}\right) \mathrm{dist}(q, \mathcal{A})$ and therefore,

using that p is not problematic with respect to s and q:

$$\begin{aligned}\text{cost}(p,\tilde{\mathcal{S}}) &\leq \text{cost}(p,\tilde{s}) \leq (1+\varepsilon)\text{cost}(p,s) + (1+z/\varepsilon)^{z-1}\text{cost}(s,\tilde{s}) \\ &\leq (1+\varepsilon)\text{cost}(p,s) + (2z/\varepsilon)^{z-1}\left(\frac{\varepsilon}{4z}\right)^z \text{cost}(q,\mathcal{A}) \\ &\leq (1+\varepsilon)\text{cost}(p,s) + \varepsilon\text{cost}(q,\mathcal{A}) \\ &\leq (1+\varepsilon)\text{cost}(p,s) + \varepsilon\text{cost}(p,\mathcal{A}).\end{aligned}$$

To show the other direction, for any point in $\tilde{\mathcal{S}}$ different than s_f, there is a center s with $\text{dist}(s,\tilde{s}) \leq \left(\frac{\varepsilon}{4z}\right)\text{dist}(q,\mathcal{A})$. Hence the previous equations apply as well, and yield $\text{cost}(p,\mathcal{S}) \leq (1+\varepsilon)\text{cost}(p,\tilde{s}) + \varepsilon\text{cost}(p,\mathcal{A})$.

When s (resp. $\tilde{s}$) being the closest point to p in $\mathcal{S}$ (resp. $\tilde{\mathcal{S}}$), the q defined must exist as by choice of p it holds that $\text{cost}(p,\mathcal{S}) \leq \text{cost}(p,\mathcal{A})$. For the same reason, $\tilde{s} \neq s_f$. Hence, using those equations with those s and $\tilde{s}$, we can conclude: for a point p such that $\text{cost}(p,\mathcal{S}) \leq \left(\frac{8z}{\varepsilon}\right)^z \cdot \text{cost}(p,\mathcal{A})$,

$$|\text{cost}(p,\mathcal{S}) - \text{cost}(p,\tilde{\mathcal{S}})| \leq \varepsilon(\text{cost}(p,\mathcal{S}) + \text{cost}(p,\mathcal{A})).$$

Rescaling ε concludes the lemma: there is an $\mathcal{A}$-approximate centroid set with size $|P|\left(\frac{z^2\log z/\varepsilon}{\varepsilon}\right)^{O(d)} = |P|\left(\frac{z}{\varepsilon}\right)^{O(d)}$.

□

7.3 Graphs with Bounded Treewidth

In this section, we show that for graphs with treewidth t, there exists a small approximate centroid set. Hence, the main framework provides an algorithm computing a small coreset. We first define the treewidth of a graph:

▶ **Definition 7.7.** A tree decomposition of a graph $G = (V, E)$ is a tree $\mathcal{T}$ where each node b (call a *bag*) is a subset of V and the following conditions hold:

- The union of bags is V,
- $\forall v \in V$, the nodes containing v in $\mathcal{T}$ form a connected subtree of $\mathcal{T}$, and
- for all edge $(u,v) \in E$, there is one bag containing u and v.

The treewidth of a graph G is the smallest integer t such that their exists a tree decomposition with maximum bag size $t+1$. ◀

▶ **Lemma 7.8.** Let $G = (V, E)$ be a graph with treewidth t, $X \subseteq V$ and $k, z > 0$. Furthermore, let $\mathcal{A}$ be a solution to (k, z)-clustering for X. Then, there exists an $\mathcal{A}$-approximate centroid set for (k, z)-clustering on X of size $\text{poly}(|X|) \left(\frac{z^2 \log z/\varepsilon}{\varepsilon}\right)^{O(t)}$. ◀

Applying this lemma with X yields the direct corollary:

▶ **Corollary 7.9.** Let $G = (V, E)$ be a graph with treewidth t, $X \subseteq V$, k and $z > 0$.
There exists an algorithm running time $\tilde{O}(nk)$ that constructs an ε-coreset for (k, z)-clustering on X, with size

$$2^{O(z \log z)} \frac{\log^5 1/\varepsilon}{\min(\varepsilon^2, \varepsilon^z)} (k \log k + kt + \log(1/\pi)) .$$ ◀

Proof. Let $X \subseteq V$. We start by computing a (k, ε)-coreset X_1 of size $O(\text{poly}(k, 1/\varepsilon, t))$, using the algorithm from [15]

We now apply our framework to X_1. Computing an approximation on X_1 takes time $\tilde{O}(|X_1|k)$, using Lemma 1.3.

Lemma 7.8 ensure the existence of an approximate centroid set for X_1 with size $\text{poly}(|X_1|) \left(\frac{z}{\varepsilon}\right)^{O(t)}$. Hence, Corollary 7.2 and the framework developed in the previous sections gives an algorithm that computes an ε-coreset of X with size

$$O\left(\frac{\log^4 1/\varepsilon}{2^{O(z \log z)} \min(\varepsilon^2, \varepsilon^z)} (k \log |X_1| + kt \log 1/\varepsilon + \log(1/\pi))\right).$$

Using that $|X_1| = O(\text{poly}(k, \varepsilon, t))$ yields a coreset of size

$$O\left(\frac{\log^5 1/\varepsilon}{2^{O(z \log z)} \min(\varepsilon^2, \varepsilon^z)} (k \log k + kt + \log(1/\pi))\right).$$

Instead of using [15], one could apply our algorithm repeatedly as in Theorem 3.1 of [34], to reduce iteratively the number of distinct point consider and to eventually get the same coreset size. The number of repetition needed to achieve that size bound is $O(\log^* n)$, where $\log^*(x)$ is the number of times log is applied to x before the result is at most 1; formally $\log^*(x) = 0$ for $x \leq 1$, and $\log^*(x) = 1 + \log^* \log x$ for $x > 1$. The complexity of this repetition is therefore $\tilde{O}(nk)$, and the success probability $1 - \pi$, as proven in [34]. □

For the proof of Lemma 7.8, we rely on the following structural lemma,[1] that shows the existence of a partition of the graph into parts that have small vertex-separators:

[1] In the statement of [15], the third item is slightly different. To recover our statement from theirs, take $P_A = A$ when $|A| = O(t)$.

▶ **Lemma 7.10 (Lemma 3.7 of [15]).** Given a graph $G = (V, E)$ of treewidth t, and $X \subseteq V$, there exists a collection $\mathcal{T}$ of subsets of V such that:

1. $\cup_{A\in\mathcal{T}} A = V$,
2. $|\mathcal{T}| = \text{poly}(|X|)$,
3. For each $A \in \mathcal{T}$, $|A \cap X| = O(t)$, and there exists $P_A \subseteq V$ with $|P_A| = O(t)$ such that there is no edge between $A \setminus P_A$ and $V \setminus (A \cup P_A)$.

◀

Our construction relies on the following simple observation. Let s be a possible center, and p be a vertex such that $\text{cost}(p, s) \leq \left(\frac{4z}{\varepsilon}\right)^z \text{cost}(p, \mathcal{A})$. Let $A \in \mathcal{T}$ such that $p \in A$. Then, either $s \in A$, or the path connecting p to s has to go through P_A.

We use this observation as follows: it would be enough to replace a center s from solution $\mathcal{S}$ by one that has approximately the same distance of all points of P_A. The main question is : how should we round the distances to P_A? The goal is to classify the potential centers into few classes, such that taking one representative per class gives an approximate centroid set. The previous observation indicates that classifying the centers according to their distances to points of P_A is enough. However, there are too many different classes: instead, we round those distances.

Ideally, this rounding would ensure that for any point p and any center s, all centers in s's class have same distance to p, up to an additive error $\varepsilon(\text{cost}(p, s) + \text{cost}(p, \mathcal{A}))$. This would mean rounding the distance from s to any point in P_A by that amount – for instance, rounding to the closest multiple of $\varepsilon(\text{cost}(p, s)+\text{cost}(p, \mathcal{A}))$. Nonetheless, this way of rounding depends on each point p: a rounding according to p may not be suited for another point q. To cope with that, we will quite naturally round distances according to the point p that minimizes $\text{cost}(p, s)+\text{cost}(p, \mathcal{A})$. Additionally, to ensure that the number of classes stays bounded, it is not enough to round to the closest multiple of $\varepsilon(\text{cost}(p, s) + \text{cost}(p, \mathcal{A}))$: we also show that distances bigger than $\frac{1}{\varepsilon}(\text{cost}(p, s)+\text{cost}(p, \mathcal{A}))$ can be trimmed down to $\frac{1}{\varepsilon}(\text{cost}(p, s)+\text{cost}(p, \mathcal{A}))$. That way, for each point of P_A there are only $1/\varepsilon^2$ many possible rounded distances.

Hence, a class is defined by a certain point p, a part A and by $|P_A| = t$ many rounded distances: in total, that makes $\text{poly}(|X|)\varepsilon^{-O(t)}$ many classes. The approximate centroid set contains one representative of each class: this would prove Lemma 7.8. We now make the argument formal, in particular to show that the error incurred by the trimming is affordable.

Proof of Lemma 7.8. Given a point $s \in V$ and a set $A \in \mathcal{T}$, we call a *distance tuple to* A $\mathbf{d}_A(s) := (\text{dist}(s, x) \mid \forall x \in X \cap A) + (\text{dist}(s, x) \mid \forall x \in P_A)$. Let $q \in X$: the rounded distance tuple of s with respect to q is $\widetilde{\mathbf{d}_{A,q}}(s)$ defined as follows:

1. For $x \in X \cap A$ or $x = q$, $\widetilde{d}(s, x)$ is the multiple of $\frac{\varepsilon}{z} \cdot \text{dist}(x, \mathcal{A})$ smaller than $\frac{10z}{\varepsilon}\text{dist}(x, \mathcal{A})$ closest to $\text{dist}(s, x)$.

2. For $y \in P_A$, $\widetilde{d}(s,y)$ is the multiple of $\frac{\varepsilon^3}{8z^3} \cdot \text{dist}(q,\mathcal{A})$ smaller than $\frac{200z^3}{\varepsilon^3}\text{dist}(q,\mathcal{A})$ closest to $\text{dist}(s,y)$.

Now, for every $A \in \mathcal{T}$, $q \in X$ and every rounded distance tuple T to A with respect to q such that $\exists s : T = \mathbf{d}_A(s)$, $\mathbb{C}$ contains one point $s \in A$ having that rounded distance tuple.

Bounding the size of $\mathbb{C}$. Fix some $A \in \mathcal{T}$, and $q \in X$. A rounded distance tuple to A is made of $O(t)$ many distances. Each of them takes its value among $\text{poly}(z/\varepsilon)$ possible numbers, due to the rounding. Hence, there are at most $\left(\frac{z}{\varepsilon}\right)^{O(t)}$ possible rounded distance tuple to A, and so at most that many points in $\mathbb{C}$. Since there are $\text{poly}(|X|)$ different choices for A and q, the total size of $\mathbb{C}$ is $\text{poly}(|X|)\left(\frac{z}{\varepsilon}\right)^{O(t)}$.

Bounding the error. We now bound the error induced by approximating a solution $\mathcal{S}$ by a solution $\tilde{\mathcal{S}} \subseteq \mathbb{C}$.

First, by applying Lemma 7.1, we can assume that for any center $s \in \mathcal{S}$, and $q = \text{argmin}_{p:\text{dist}(p,s)\leq\frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})}\text{dist}(p,\mathcal{A}) + \text{dist}(p,s)$, there is no problematic point with respect to q and s.

Let $A \in \mathcal{T}$ such that $s \in A$, and $q = \text{argmin}_{p:\text{dist}(p,s)\leq\frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})}\text{dist}(p,\mathcal{A}) + \text{dist}(p,s)$. $\tilde{s}$ is chosen to have the same rounded distance tuple to A with respect to q as s. $\tilde{\mathcal{S}}$ is the solution made of all such $\tilde{s}$, for $s \in \mathcal{S}$.

As in the proof of Lemma 7.5, we first show that points close to s have cost preserved in $\tilde{s}$. We will later show that points with large distance to s have also large distance to $\tilde{s}$, to ensure that their distance to $\tilde{\mathcal{S}}$ does not decrease.

Let $p \in X$ be an input point. By Lemma 7.1, p is not problematic with respect to s and q. Note that s is not necessarily the closest center to p. We aim at showing that $|\text{cost}(p,s) - \text{cost}(p,\tilde{s})| \leq \varepsilon(\text{cost}(p,s) + \text{cost}(p,\mathcal{A}))$.

First, when $p \notin X \cap A$, we distinguish two subcases:

- either $\text{dist}(p,s) \leq \frac{200z^3}{\varepsilon^3}\text{dist}(q,\mathcal{A})$: in that case, let $x \in p_A$ that is on the shortest path between p and s. We have $\text{dist}(s,x) \leq \frac{200z^3}{\varepsilon^3}\text{dist}(q,\mathcal{A})$, and so s and $\tilde{s}$ have the same rounded distance to x. Hence,

$$\begin{aligned}\text{dist}(p,\tilde{s}) &\leq \text{dist}(p,x) + \text{dist}(x,\tilde{s}) \leq \text{dist}(p,x) + \text{dist}(x,s) + \frac{\varepsilon^3}{8z^3}\text{dist}(q,\mathcal{A}) \\ &\leq \text{dist}(p,s) + \frac{\varepsilon^3}{8z^3} \cdot \left(\frac{8z^2}{\varepsilon^2}\right)(\text{dist}(p,\mathcal{A}) + \text{dist}(p,s)) \\ &\leq \left(1 + \frac{\varepsilon}{z}\right)\text{dist}(p,s) + \frac{\varepsilon}{z}\text{dist}(p,\mathcal{A}),\end{aligned}$$

  The first line implies that $\text{dist}(p,\tilde{s}) \leq \frac{200z^3}{\varepsilon^3}\text{dist}(q,\mathcal{A})$ as well: we can therefore repeat the argument, choosing x to be on the shortest path between p and $\tilde{s}$ instead, to show that $\text{dist}(p,s) \leq \left(1 + \frac{\varepsilon}{z}\right)\text{dist}(p,\tilde{s}) + \frac{\varepsilon}{z}\text{dist}(p,\mathcal{A})$. This implies, using Lemma 1.2, that $|\text{cost}(p,\tilde{s}) - \text{cost}(p,s)| \leq \frac{\varepsilon}{z} \cdot (\text{cost}(p,s) + \text{cost}(p,\mathcal{A}))$.

- Otherwise, $\text{dist}(p,s) > \frac{200z^3}{\varepsilon^3}\text{dist}(q,\mathcal{A})$. In that case, we can argue that $\text{dist}(s,\tilde{s})$ is negligible compared to $\text{dist}(p,s)$. Recall that $\text{dist}(q,s) \leq \frac{10z}{\varepsilon}\text{dist}(q,\mathcal{A})$.

  The rounding ensures that the distance to q is preserved: $\text{dist}(q,\tilde{s}) \leq \text{dist}(q,s) + \frac{\varepsilon}{z}\text{dist}(q,\mathcal{A})$. Hence, we get that

$$\begin{aligned}\text{dist}(s,\tilde{s}) &\leq 2\text{dist}(q,s) + \frac{\varepsilon}{z}\text{dist}(q,\mathcal{A}) \\ &\leq \left(\frac{100z^2}{\varepsilon^2} + \frac{\varepsilon}{z}\right) \cdot \text{dist}(q,\mathcal{A}) \\ &\leq \frac{200z^2}{\varepsilon^2} \cdot \frac{\varepsilon^3}{200z^3} \cdot \text{dist}(p,s) \leq \frac{\varepsilon}{z} \cdot \text{dist}(p,s).\end{aligned}$$

  Finally, using Lemma 1.2, we conclude again that $|\text{cost}(p,\tilde{s}) - \text{cost}(p,s)| \leq \varepsilon\text{cost}(p,s) + \varepsilon\text{cost}(p,\mathcal{A})$.

Now, in the other case where $p \in X \cap A$, if $\text{dist}(p,s) \leq \frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})$, then the choice of $\tilde{s}$ ensures that $|\text{dist}(\tilde{s},p) - \text{dist}(s,p)| \leq \frac{\varepsilon}{z}\text{dist}(x,\mathcal{A})$ and therefore $|\text{cost}(p,s) - \text{cost}(p,\tilde{s})| \leq \varepsilon\text{cost}(p,s) + (1+z/\varepsilon)^{z-1}\text{cost}(s,\tilde{s}) \leq \varepsilon\text{cost}(p,\mathcal{S}) + \varepsilon\text{cost}(p,\mathcal{A})$. In the last case when $\text{dist}(p,s) > \frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})$, then the rounding enforces $\text{dist}(p,\tilde{s}) = \frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})$.

Hence, in all possible cases, it holds that either $\text{dist}(p,s)$ and $\text{dist}(p,\tilde{s})$ are bigger than $\frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})$, or:

$$|\text{cost}(p,\tilde{s}) - \text{cost}(p,s)| \leq \varepsilon\text{cost}(p,s) + \varepsilon\text{cost}(p,\mathcal{A}). \tag{7.3}$$

To extend that result to the full solutions $\mathcal{S}$ and $\tilde{\mathcal{S}}$ instead of a particular center, we note that since p is interesting, $\text{dist}(p,\mathcal{S}) \leq \frac{8z}{\varepsilon}\text{dist}(p,\mathcal{A})$. Hence, we can apply Eq. (7.3) with s being the closest point to p in $\mathcal{S}$: $\text{cost}(p,\tilde{\mathcal{S}}) \leq (1+\varepsilon)\text{cost}(p,\mathcal{S}) + \varepsilon\text{cost}(p,\mathcal{A})$.

In particular, this implies that $\text{dist}(p,\tilde{s}) \leq \frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})$. Choose now $\tilde{s}$ to be the closest point to p in $\tilde{\mathcal{S}}$ and s its corresponding center in $\mathcal{S}$. Using Eq. (7.3) therefore gives:

$$\begin{aligned}&\text{cost}(p,s) \leq \text{cost}(p,\tilde{\mathcal{S}}) + \varepsilon(\text{cost}(p,\mathcal{A}) + \text{cost}(p,s)) \\ \implies\ &\text{cost}(p,s) \leq \frac{1}{1-\varepsilon}\text{cost}(p,\tilde{\mathcal{S}}) + \frac{\varepsilon}{1-\varepsilon}\text{cost}(p,\mathcal{A}) \\ &\leq (1+2\varepsilon)\text{cost}(p,\tilde{\mathcal{S}}) + 2\varepsilon\text{cost}(p,\mathcal{A}) \\ \implies\ &\text{cost}(p,\mathcal{S}) \leq (1+2\varepsilon)\text{cost}(p,\tilde{\mathcal{S}}) + 3\varepsilon\text{cost}(p,\mathcal{A}).\end{aligned}$$

Hence, combining those two inequalities yields

$$|\text{cost}(p,\mathcal{S}) - \text{cost}(p,\tilde{\mathcal{S}})| \leq \varepsilon\text{cost}(p,\mathcal{S}) + 2\varepsilon\text{cost}(p,\tilde{\mathcal{S}}) + 4\varepsilon\text{cost}(p,\mathcal{A}).$$

To remove the dependency in $\text{cost}(p,\tilde{\mathcal{S}})$ from the right hand side, one can upper bound

it with $\text{cost}(p,\mathcal{S}) + |\text{cost}(p,\mathcal{S}) - \text{cost}(p,\tilde{\mathcal{S}})|$, which yields the following:

$$\begin{aligned} |\text{cost}(p,\mathcal{S}) - \text{cost}(p,\tilde{\mathcal{S}})| &\leq \varepsilon\text{cost}(p,\mathcal{S}) + 2\varepsilon\left(\text{cost}(p,\mathcal{S}) + |\text{cost}(p,\mathcal{S}) - \text{cost}(p,\tilde{\mathcal{S}})|\right) \\ &\quad + 4\varepsilon\text{cost}(p,\mathcal{A}) \\ \iff |\text{cost}(p,\mathcal{S}) - \text{cost}(p,\tilde{\mathcal{S}})| &\leq \frac{1}{1-2\varepsilon}\left(3\varepsilon\text{cost}(p,\mathcal{S}) + 4\varepsilon\text{cost}(p,\mathcal{A})\right) \\ &\leq 9\varepsilon\text{cost}(p,\mathcal{S}) + 12\varepsilon\text{cost}(p,\mathcal{A}) \end{aligned}$$

Finally, rescaling ε concludes.

□

7.4 Planar Graphs

The goal of this section is to prove the existence of small centroid sets for planar graph, analogously to the treewidth case. This is the following lemma:

▶ **Lemma 7.11.** Let $G = (V,E)$ be an edge-weighted planar graph, a set $X \subseteq V$ and two positive integers k and z. Furthermore, let $\mathcal{A}$ be a solution of (k,z)-clustering of X.
Then, there exists an $\mathcal{A}$-approximate centroid set for (k,z)-clustering on V of size $\text{poly}(|X|) \cdot \exp\left(O(z^3\varepsilon^{-3}\log z/\varepsilon)\right)$.

◀

As for treewidth, this lemma implies the following corollary:

▶ **Corollary 7.12.** Let $G = (V,E)$ be an edge-weighted planar graph, a set $X \subseteq V$, and two positive integers k and z .
There exists an algorithm with running time $\tilde{O}(nk)$ that constructs an ε-coreset for (k,z)-clustering on X with size

$$2^{O(z\log z)}\frac{\log^5 1/\varepsilon}{\min(\varepsilon^2,\varepsilon^z)}\left(k\log^2 k + \frac{k\log k}{\varepsilon^3} + \log 1/\pi\right)$$

◀

The big picture is the same as for treewidth. As in the treewidth case, planar graph can be broken into $\text{poly}(X)$ pieces, each containing at most 2 vertices of X. The main difference is in the nature of the separators: while treewidth admit small vertex separators, the region in the planar decomposition are bounded by a few number of shortest paths instead. This makes the previous argument void: we cannot round distances to all vertices in the boundary of a region. We show how to bypass this, using the fact that separators are shortest paths: it is enough to round distances to a well-chosen subset of the paths, as we will argue in the proof.

Formally, the decomposition is as follows:

▶ **Lemma 7.13 (Lemma 4.5 of [34], see also [70]).** For every edge-weighted planar graph $G = (V, E)$ and subset $X \subseteq V$, there exists a collection of subsets of V $\Pi := \{V_i\}$ with $|\Pi| = \text{poly}(|X|)$ and $\cup V_i = V$ such that, for every $V_i \in \Pi$:

- $|V_i \cap X| = O(1)$, and
- there exists a collection of shortest paths $\mathcal{P}_i$ with $|\mathcal{P}_i| = O(1)$ such removing the vertices of all paths of $\mathcal{P}_i$ disconnects V_i from $V \setminus V_i$.

◀

As for treewidth, we proceed as follows: given the decomposition of Lemma 7.13, for any center $s \in V_i$, we identify a point q and round distances from s to $\mathcal{P}_i$ according to $\text{dist}(q, \mathcal{A})$. $\mathbb{C}$ contains one point $\tilde{s}$ with the same rounded distances as s, and we will argue that $\tilde{s}$ can replace s. As mentioned, we cannot round distances to the whole shortest-paths $\mathcal{P}_i$. Instead, we show that it is enough to round distances from s to points on the boundary of V_i that are close to q: since the boundary consists of shortest path, it is possible to discretize that set.

Proof of Lemma 7.11. Let $\Pi = \{V_i\}$ be the decomposition given by Lemma 7.13. For any V_i and any $q \in X$, we define a set of *landmarks* $\mathcal{L}_{i,q}$ as follows: for any $P \in \mathcal{P}_i$, let $\mathcal{L}_{i,q,P}$ be a $\frac{\varepsilon}{z} \cdot \text{dist}(q, \mathcal{A})$-net of $P \cap B\left(q, \frac{90z^2}{\varepsilon^2} \cdot \text{dist}(q, \mathcal{A})\right)$. Note that since P is a shortest path, the total length of $P \cap B\left(q, \frac{90z^2}{\varepsilon^2} \cdot \text{dist}(q, \mathcal{A})\right)$ is at most $\frac{180z^2}{\varepsilon^2} \cdot \text{dist}(q, \mathcal{A})$, and so the net has size at most $\frac{180z^3}{\varepsilon^3}$. We define $\mathcal{L}_{i,q} = (V_i \cap X) \cup_{P \in \mathcal{P}_i} \mathcal{L}_{i,q,P}$.

Rounding the distances to $\mathcal{L}_{i,q}$ We now describe how we round distances to landmarks, and define $\mathbb{C}$ such that for each possible distance tuple, $\mathbb{C}$ contains a point having that distance tuple. Formally, given a point $s \in V_i$ and a point $q \in X$, the distance tuple $\mathbf{d}_q(s)$ of s is defined as $\mathbf{d}_q(s) = (\text{dist}(s, x) \mid \forall\ x \in X \cap V_i) + (\text{dist}(s, y) \mid \forall y \in \mathcal{L}_{i,q}, \forall i)$. The *rounded distance tuple* $\tilde{\mathbf{d}}_q(s)$ of s is defined as follows :

- For $x \in X \cap V_i$ or $x = q$, $\tilde{d}(s, x)$ is the multiple of $\frac{\varepsilon}{z}\text{dist}(x, \mathcal{A})$ smaller than $\frac{10z}{\varepsilon}\text{dist}(x, \mathcal{A})$ closest to $\text{dist}(s, x)$.
- For $y \in \mathcal{L}_{i,q}$, $\tilde{d}(s, y)$ is the multiple of $\frac{\varepsilon}{z} \cdot \text{dist}(q, \mathcal{A})$ smaller than $\frac{90z^2}{\varepsilon^2}\text{dist}(q, \mathcal{A})$ closest to $\text{dist}(s, y)$.

The set $\mathbb{C}$ is constructed as follows: for every V_i and every q, for every rounded distance tuple $\{\tilde{\mathbf{d}}_q(p)\}$, add to $\mathbb{C}$ a point that realizes this rounded distance tuple (if such a point exists).

It remains to show both that $\mathbb{C}$ has size $\text{poly}(|X|) \exp\left(O(z^3\varepsilon^{-3} \log z/\varepsilon)\right)$, and that $\mathbb{C}$ contains good approximation of each center of any given solution.

Size analysis. For any given V_i and q, there are $\left(\frac{90z^3}{\varepsilon^3}\right)^{|\mathcal{L}_{i,q}|}$ possible rounded distances. As explained previously, $|\mathcal{L}_{i,q}| = O(z^3/\varepsilon^3)$.

There are $|V|$ choices of q, and Lemma 7.13 ensures that there are $\text{poly}(|X|)$ choices for V_i.

Hence, the total size of $\mathbb{C}$ is at most $\text{poly}(|X|) \cdot \exp\left(O(z^3\varepsilon^{-3}\log z/\varepsilon)\right)$.

Error analysis. We now show that for all solution $\mathcal{S}$, every center can be approximated by a point of $\mathbb{C}$. First, by applying Lemma 7.1, we can assume that for any center $s \in \mathcal{S}$, and $q = \text{argmin}_{p:\text{dist}(p,s)\leq\frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})}\text{dist}(p,\mathcal{A}) + \text{dist}(p,s)$, there is no problematic point with respect to q and s.

Let S be some cluster of $\mathcal{S}$, with center s. As in Lemma 7.5 and 7.8, we aim at showing how to find $\tilde{s} \in \mathbb{C}$ such that, for every $p \in X \cap S$ with $\text{dist}(p,\mathcal{S}) \leq \frac{10z}{\varepsilon} \cdot \text{dist}(p,\mathcal{A})$, we have $|\text{cost}(p,s) - \text{cost}(p,\tilde{s})| \leq 3\varepsilon\,(\text{cost}(p,s) + \text{cost}(p,\mathcal{A}))$.

For this, let V_i be a part of Π containing s, and $\mathcal{P}_i$ be the paths given by Lemma 7.13. We let $q := \text{argmin}_{p\in X:\text{dist}(p,s)\leq\frac{10z}{\varepsilon}\text{dist}(p,\mathcal{A})}\text{dist}(p,s) + \text{dist}(p,\mathcal{A})$. We define $\tilde{s}$ to be the point of $\mathbb{C}$ that has the same rounded distance tuple to $\mathcal{L}_{i,q}$ as s. Let $\tilde{\mathcal{S}}$ be the solution constructed from $\mathcal{S}$ that way. We show now that $\tilde{\mathcal{S}}$ has the required properties.

First, if $p \notin V_i$, then we show how to use that s and $\tilde{s}$ have the same rounded distances to $\mathcal{L}_{i,q}$.

- If $\text{dist}(p,s) > \frac{21z^2}{\varepsilon^2} \cdot \text{dist}(q,\mathcal{A})$, we argue that $d(s,\tilde{s})$ is negligible. The argument is exactly alike the one from Lemma 7.8, we repeat it for completeness.

  The rounding ensures that the distance to q is preserved: $\text{dist}(q,\tilde{s}) \leq \text{dist}(q,s) + \frac{\varepsilon}{z}\text{dist}(q,\mathcal{A})$, and therefore:

$$\begin{aligned}\text{dist}(s,\tilde{s}) &\leq 2\text{dist}(q,s) + \frac{\varepsilon}{z} \cdot \text{dist}(q,\mathcal{A}) \\ &\leq \left(\frac{20z}{\varepsilon} + \frac{\varepsilon}{z}\right) \cdot \text{dist}(q,\mathcal{A}) \\ &\leq \frac{21z}{\varepsilon} \cdot \frac{\varepsilon^2}{21z^2} \cdot \text{dist}(p,s) \leq \frac{\varepsilon}{z} \cdot \text{dist}(p,s).\end{aligned}$$

  Hence, using the modified triangle inequality Lemma 1.2, we can conclude: $|\text{cost}(p,\tilde{s}) - \text{cost}(p,s)| \leq \varepsilon\text{cost}(p,s) + \varepsilon\text{cost}(p,\mathcal{A})$.

- Otherwise, $\text{dist}(p,s) \leq \frac{21z^2}{\varepsilon^2} \cdot \text{dist}(q,\mathcal{A})$ and we can make use of the landmarks. Since $p \notin V_i$ the shortest-path $p \rightsquigarrow s$ and crosses $\mathcal{P}_i$ at some vertex x.

  First, it holds that $\text{dist}(x,q) \leq \text{dist}(x,s) + \text{dist}(s,q) \leq \text{dist}(p,s) + \text{dist}(s,q) \leq (\frac{10z}{\varepsilon} + \frac{8z^2}{\varepsilon^2})\text{dist}(q,\mathcal{A})$, hence x is in $P \cap B(q, \frac{90z^2}{\varepsilon^2}\text{dist}(q,\mathcal{A}))$. By choice of landmarks, this implies that there is $\ell \in \mathcal{L}_{i,q}$, with $\text{dist}(x,\ell) \leq \frac{\epsilon}{z}\text{dist}(q,\mathcal{A})$. To show that s and $\tilde{s}$ have the same distance to ℓ, it is necessary to show that s is not

too far away from ℓ:

$$\begin{aligned}\mathrm{dist}(s,\ell) &\leq \mathrm{dist}(s,x) + \frac{\epsilon}{z}\cdot\mathrm{dist}(q,\mathcal{A}) \leq \mathrm{dist}(p,s) + \frac{\epsilon}{z}\mathrm{dist}(q,\mathcal{A}) \\ &\leq \frac{21z^2}{\varepsilon^2}\cdot\mathrm{dist}(q,\mathcal{A}) + \frac{\varepsilon}{z}\cdot\mathrm{dist}(q,\mathcal{A})\end{aligned}$$

Hence, s is close enough to ℓ to ensure that $\tilde{s}$ has the same rounded distance to ℓ as s, and we get:

$$\begin{aligned}\mathrm{dist}(p,\tilde{s}) &\leq \mathrm{dist}(p,\ell) + \mathrm{dist}(\ell,\tilde{s}) \\ &\leq \mathrm{dist}(p,\ell) + \mathrm{dist}(\ell,s) + \frac{\epsilon}{z}\cdot\mathrm{dist}(q,\mathcal{A}) \\ &\leq \mathrm{dist}(p,x) + \mathrm{dist}(x,s) + 2\mathrm{dist}(x,\ell) + \frac{\epsilon}{z}\cdot\mathrm{dist}(q,\mathcal{A}) \\ &= \mathrm{dist}(p,s) + \frac{3\epsilon}{z}\cdot\mathrm{dist}(q,\mathcal{A})\end{aligned}$$

First, this ensures that $\mathrm{dist}(p,\tilde{s}) \leq \frac{8z^2}{\varepsilon^2}\cdot\mathrm{dist}(q,\mathcal{A})$, and so we can repeat the argument switching roles of s and $\tilde{s}$, to get $|\mathrm{dist}(p,\tilde{s}) - \mathrm{dist}(p,s)| \leq \frac{3\epsilon}{z}\cdot\mathrm{dist}(q,\mathcal{A})$ Using that p is not problematic with respect to q and s, we can conclude that

$$|\mathrm{dist}(p,\tilde{s}) - \mathrm{dist}(p,s)| \leq \frac{3\epsilon}{z}\cdot(\mathrm{dist}(p,\mathcal{A}) + \mathrm{dist}(p,s)).$$

In turn, using Lemma 1.2, we conclude:

$$|\mathrm{cost}(p,\tilde{s}) - \mathrm{cost}(p,s)| \leq \varepsilon\cdot(\mathrm{cost}(p,\mathcal{A}) + \mathrm{cost}(p,s)).$$

Finally, in the case where $p \in V_i$, then we get either $|\mathrm{dist}(p,\tilde{s}) - \mathrm{dist}(p,s)| \leq \frac{\varepsilon}{z}\mathrm{dist}(p,\mathcal{A})$ and we are done, or both $\mathrm{dist}(p,\tilde{s})$ and $\mathrm{dist}(p,s)$ are bigger than $\frac{8z}{\varepsilon}\mathrm{dist}(p,\mathcal{A})$.

We can now conclude, exactly as in the treewidth case: in all possible cases, it holds that either $\mathrm{dist}(p,s)$ and $\mathrm{dist}(p,\tilde{s})$ are bigger than $\frac{10z}{\varepsilon}\mathrm{dist}(p,\mathcal{A})$, or:

$$|\mathrm{cost}(p,\tilde{s}) - \mathrm{cost}(p,s)| \leq \varepsilon\mathrm{cost}(p,s) + \varepsilon\mathrm{cost}(p,\mathcal{A}). \tag{7.4}$$

To extend that result to the full solutions $\mathcal{S}$ and $\tilde{\mathcal{S}}$ instead of a particular center, we note that since p is interesting, $\mathrm{dist}(p,\mathcal{S}) \leq \frac{8z}{\varepsilon}\mathrm{dist}(p,\mathcal{A})$. Hence, we can apply Eq. (7.4) with s being the closest point to p in $\mathcal{S}$: $\mathrm{cost}(p,\tilde{\mathcal{S}}) \leq (1+\varepsilon)\mathrm{cost}(p,\mathcal{S}) + \varepsilon\mathrm{cost}(p,\mathcal{A})$.

In particular, this implies that $\mathrm{dist}(p,\tilde{s}) \leq \frac{10z}{\varepsilon}\mathrm{dist}(p,\mathcal{A})$. Chose now $\tilde{s}$ to be the closest point to p in $\tilde{\mathcal{S}}$ and s its corresponding center in $\mathcal{S}$. Using Eq. (7.4) therefore gives:

$$\begin{aligned}\mathrm{cost}(p,\mathcal{S}) &\leq (1+\varepsilon)\mathrm{cost}(p,\tilde{\mathcal{S}}) + \varepsilon\mathrm{cost}(p,\mathcal{A}) \\ &\leq (1+\varepsilon)^2\mathrm{cost}(p,\mathcal{S}) + \varepsilon(2+\varepsilon)\mathrm{cost}(p,\mathcal{A}) \\ &\leq (1+3\varepsilon)\mathrm{cost}(p,\mathcal{S}) + 3\varepsilon\mathrm{cost}(p,\mathcal{A}).\end{aligned}$$

Rescaling ε and combining the two inequality concludes. □

7.5 Minor-Excluded Graphs

A graph H is a *minor* of a graph G if it can be obtained from G by deleting edges and vertices and contracting edges.

We are interested here in families of graph excluding a fixed minor H, i.e. none of the graph in the family contains H as a minor. The graphs are weighted: we assume that for each edge, its value is equal to shortest-path distance between its two endpoints.

The goal of this section is to prove the following lemma, analogous to Lemma 7.11.

▶ **Lemma 7.14.** Let $G = (V, E)$ be an edge-weighted graph that excludes a minor of fixed size, a set $X \subseteq V$ and two positive integers k and z. Furthermore, let $\mathcal{A}$ be a solution of (k, z)-clustering of X.
Then, there exists an $\mathcal{A}$-approximate centroid set for (k, z)-clustering on V of size $\exp(O(\log^2 |X| + \log |X|/\varepsilon^4))$. ◀

As for the bounded treewidth and planar cases, this lemma implies the following corollary:

▶ **Corollary 7.15.** Let $G = (V, E)$ be an edge-weighted graph that excludes a fixed minor, and two positive integers k and z .
There exists an algorithm with running time $\tilde{O}(nk)$ that constructs an ε-coreset for (k, z)-clustering on V with size

$$2^{O(z \log z)} \frac{\log^5 1/\varepsilon}{\min(\varepsilon^2, \varepsilon^z)} \left(k \log^2 k \log(1/\varepsilon) + \frac{k \log k}{\varepsilon^4} + \log 1/\pi \right)$$ ◀

The big picture is the same as for planar graphs. Minor-free graphs have somewhat nice separators, that we can use to select centers. However, those separators are not shortest paths in the original graph, as described in the next structural lemma.

▶ **Lemma 7.16 (Lemma 4.12 in [34], from Theorem 1 in [1]).** For every edge-weighted graph $G = (V, E)$ excluding some fixed minor, and subset $X \subseteq V$, there exists a collection of subsets of V $\Upsilon := \{\Pi_i\}$ with $|\Upsilon| = \text{poly}(|X|)$ and $\cup \Pi_i = V$ such that, for every $\Pi_i \in \Upsilon$:

- $|\Pi_i \cap X| = O(1)$, and
- there exists a group of paths $\{\mathcal{P}_j^i\}_j$ with $|\cup_j \mathcal{P}_j^i| = O(\log |X|)$ such that removing the vertices of all paths of $\mathcal{P}_i$ disconnects Π_i from $V \setminus \Pi_i$, and such that paths in $\mathcal{P}_j^i$ are shortest-paths in the graph $G_j^i := G \setminus \left(\cup_{j'<j} \mathcal{P}_{j'}^i \right)$.

◀

The general sketch of the proof is as follows: we consider the boundary B of a region

Π_i, and enumerate all possible tuple of distances from a point inside the leaf to the boundary. For each tuple, we include in $\mathbb{C}$ a point realizing it. Of course, this would lead to a set $\mathbb{C}$ way too big: the boundary of each leaf consists of too many points, and there are too many distances possible. For that, we show how to discretize the boundary, and how to round distances from a point to the boundary.

Discretizing the boundary is not as easy as in the planar case, as the separating paths are not shortest paths in the original graph G. A separating path $P \in \mathcal{P}_j$, however, is a shortest path in the graph $G^i_j := G \setminus \left(\cup_{j'<j} \mathcal{P}^i_{j'}\right)$.

As in the planar case, we therefore start from the point q closest to s in the graph G^i_j. Note here that we cannot infer much on the distances in the original graph G: for this reason, we are not able to apply Lemma 7.1, and we need to present a whole different argument.

We will assume that we know $D = \text{dist}_j(q, s)$, where dist_j is the distance in the graph G^i_j. In that case, we can simply take an εD-net of $P \cap B_j(q, D)$, where $B_j(q, D)$ is the ball centered at q and of radius D in G^i_j. This net has size $O(1/\varepsilon^2)$, as P is a shortest path in G^i_j. Then, if $\tilde{s}$ has same distances to this net as s, we are able to show as in the previous cases that for any point separated from s by P, $\text{dist}(p, \tilde{s}) \lessapprox \text{dist}(p, s)$; and for any point separated from $\tilde{s}$ by P, $\text{dist}(p, s) \lessapprox \text{dist}(p, \tilde{s})$.

To estimate $\text{dist}_i(q, s)$, we proceed as follows: either $\text{dist}_i(q, s) \approx \text{dist}_i(q, q_2)$ for some $q_2 \in X$, or not. In the first case, we can pick such a q_2. In the second case, we will need to ensure that when p is such that $\text{dist}_i(p, q) \gg \text{dist}_i(q, s)$, then $\tilde{s}$ stays close to q. When p is such that $\text{dist}_i(q, p) \ll \text{dist}_i(q, s)$, then p and q are essentially located at the same spot, and we ensure that $\tilde{s}$ stays far from q.

7.5.1 Construction of the centroid set.

From Lemma 7.16, we have a decomposition into regions $\Upsilon = \{\Pi_i\}$. In this argument, we fix a region $\Pi_j \in \Upsilon$. Π_j is bounded by $O(\log |X|)$ paths $P_1, ..., P_m$ and P_i is a shortest path in some graph G_i, subgraph of G: if $P_i \in \mathcal{P}^j_\ell$, then $G_i := G^j_\ell$. We change the indexing for simplicity, and let $\Pi = \Pi_j$. We let dist_i be the distances in the graph G_i.

We consider two ways of rounding the distances. The first starts from a point $q_1 \in X$, and is useful when there is $q_2 \in X$ such that $\varepsilon \text{dist}_i(q_1, s) \leq \text{dist}_i(q_1, q_2) \leq \frac{1}{\varepsilon}\text{dist}_i(q_1, s)$.

Along each paths, we designate portals as follows. Consider a path P_i. For any pair of vertices $q_1, q_2 \in X$, let $D = \text{dist}_i(q_1, q_2) + \text{dist}(q_2, \mathcal{A})$ and let N_{i,q_1,q_2} be an $\varepsilon^2 D$-net of $P_i \cap B_i(q_1, \frac{D}{\varepsilon^2})$, where $B_i(q, \frac{D}{\varepsilon^2})$ is the ball centered at q and of radius $\frac{D}{\varepsilon^2}$ in G_i.

For each possible q_1, q_2 and any point $s \in \Pi$, we consider the following distance tuple: $(\text{dist}_i(s, n),\ \forall n \in N_{i,q_1,q_2}) \cup (\text{dist}_i(s, q_1)) \cup (\text{dist}_i(x, s),\ \forall x \in \Pi \cap X)$. We define the rounded tuple $\tilde{d}^1(q_1, q_2) := \left(\tilde{d}^1(s, n),\ \forall n \in N_{i,q_1,q_2}\right) \cup \left(\tilde{d}^1(s, q_1)\right) \cup \left(\tilde{d}^1(x, s),\ \forall x \in \Pi \cap X\right)$, where

- $\tilde{d}^1(s,n)$ is the multiple of $\varepsilon^2 D$ closest to $\min\left(\frac{3D}{\varepsilon^2}, \mathrm{dist}_i(s,n)\right)$.
- $\tilde{d}^1(s,q_1)$ is the multiple of εD closest to $\mathrm{dist}_i(s,q_1)$ and smaller than $\frac{3D}{\varepsilon}$.
- for any $x \in \Pi \cap X$, $\tilde{d}^1(x,s)$ is the closest multiple of $\varepsilon \mathrm{dist}(x,\mathcal{A})$ to $\mathrm{dist}_i(x,s)$ smaller than $\frac{1}{\varepsilon} \cdot \mathrm{dist}(x,\mathcal{A})$.

We also consider another rounding, which will be helpful when for all points , $\mathrm{dist}(p,\mathcal{A}) + \mathrm{dist}_i(q,q_1) \notin [\varepsilon \mathrm{dist}_i(q_1,s), \frac{1}{\varepsilon}\mathrm{dist}_i(q_1,s)]$.

For any q_1, q_3, and q_4 in X, $\tilde{d}^2(q_1,q_3,q_4) = \top$ when $\frac{1}{\varepsilon} \cdot (\mathrm{dist}_i(q_1,q_4) + \mathrm{dist}(q_4,\mathcal{A})) < \mathrm{dist}_i(q_1,s) < \varepsilon \cdot (\mathrm{dist}_i(q_1,q_3) + \mathrm{dist}(q_3,\mathcal{A}))$, and $\tilde{d}^2(q_1,q_3,q_4) = \bot$ otherwise. q_3 or q_4 may be unspecified. In that case, the corresponding part of the inequality is dropped.[2]

To construct $\mathbb{C}$, we proceed as follows: for any region $\Pi \in \Upsilon$ given by Lemma 7.16, and for any path P_i in the boundary of Π, select a rounding $\tilde{d}^1_i(q^i_1, q^i_2)$ or $\tilde{d}^2_i(q^i_1, q^i_3, q^i_4)$. If there is any, pick one point s achieving all those rounding distances, and add s to $\mathbb{C}$.

We will show Lemma 7.14 using this centroid set. For that, we break the proof into two parts: first, the size of $\mathbb{C}$ is the desired one; then, $\mathbb{C}$ is indeed an approximate centroid set.

7.5.2 $\mathbb{C}$ has Small Size

▶ **Lemma 7.17.** $\mathbb{C}$ constructed as previously has size $\exp\left(O(\log^2|X| + \log|X|/\varepsilon^{-4})\right)$. ◀

Proof. Fix a region Π, a path P_i on Π's boundary, and points q_1, q_2. There are $O\left(1/\varepsilon^4\right)$ points in the net N_{i,q_1,q_2}, and $O(1)$ in $\Pi \cap X$. For each of those points, there are at most $3/\varepsilon^4$ many choices of distances.

For a fixed region Π, path P_i on Π's boundary, and points q_1, q_2, q_3, there only 2 possible different $\tilde{d}^2(q_1,q_2,q_3)$.

Now, there are $\mathrm{poly}(|X|)$ many regions Π, and for each of them $O(\log|X|)$ many paths P_i. For each path, there are at most $|X|^3$ choices of q_j for it, so in total $|X|^{O(\log|X|)}$ possible choices. Each choice gives rise to $O(\log|X|) \cdot O\left(1/\varepsilon^4\right)$ many net points, each having at most $3/\varepsilon^4$ many choices of distances.

So, in total, there are

$$|X|^{O(\log|X|)} \cdot (1/\varepsilon)^{O\left(\log|X|/\varepsilon^4\right)}$$

[2] When q_3 is unspecified, $\tilde{d}^2(q_1,q_3,q_4) = \top$ when $\frac{1}{\varepsilon} \cdot (\mathrm{dist}_i(q_1,q_4) + \mathrm{dist}(q_4,\mathcal{A})) < \mathrm{dist}_i(q_1,s)$, and $\tilde{d}^2(q_1,q_3,q_4) = \bot$ otherwise. When q_4 is unspecified, $\tilde{d}^2(q_1,q_3,q_4) = \top$ when $\mathrm{dist}_i(q_1,s) < \varepsilon \cdot (\mathrm{dist}_i(q_1,q_3) + \mathrm{dist}(q_3,\mathcal{A}))$, and $\tilde{d}^2(q_1,q_3,q_4) = \bot$ otherwise.

many choices of rounded distances tuples. That upper bounds the size of $\mathbb{C}$, as there is at most one point per rounded distance tuple. □

7.5.3 $\mathbb{C}$ is an Approximate Centroid Set

Construction of solution $\tilde{\mathcal{S}}$. Now, for a point $s \in \mathcal{S}$, we construct $\tilde{s}$ as follows, using the rounded distance tuples. Let Π be a region of Υ that contains s. For each path P_i in the boundary of Π, we define the tuple $\tilde{d}_i$ as follows. Let q_1^i be the point minimizing $\text{dist}_i(p, s)$. Now, we distinguish two cases:

- either there is some q_2^i such that $\varepsilon\text{dist}_i(q_1^i, s) \leq \text{dist}_i(q_1^i, q_2^i) + \text{dist}(q_2^i, \mathcal{A}) \leq \frac{1}{\varepsilon}\text{dist}_i(q_1^i, s)$. Then $\tilde{d}_i$ is the tuple $\tilde{d}^1(q_1^i, q_2^i)$.
- or there exists points q with $\text{dist}_i(q_1^i, q) + \text{dist}(q, \mathcal{A}) > \frac{1}{\varepsilon}\text{dist}_i(q_1^i, s)$: let q_3^i be such a point, with smallest $\text{dist}_i(q_1^i, q_3^i) + \text{dist}(q_3^i, \mathcal{A})$ value. If there are no such points, q_3^i is unspecified.

  If there are points q with $\text{dist}_i(q_1^i, q) + \text{dist}(q, \mathcal{A}) < \varepsilon\text{dist}_i(q_1^i, s)$, then let q_4^i be the point with largest $\text{dist}_i(q_1^i, q_4^i) + \text{dist}(q_4^i, \mathcal{A})$ value. Otherwise, q_4^i is unspecified. Note that since we are not in the first case, either q_3^i are q_4^i is specified.

  Then $\tilde{d}_i$ is the tuple $\tilde{d}^2(q_1^i, q_3^i, q_4^i)$.

$\tilde{s}$ is chosen to be in $\mathbb{C} \cap \Pi$ and to have the same rounded distance tuples as s, for *all* the rounded tuples $\tilde{d}_i$. $\tilde{\mathcal{S}}$ is the union of all those $\tilde{s}$ for $s \in \mathcal{S}$.

▶ **Lemma 7.18.** Let $\mathcal{S}$ be a solution, and $s \in \mathcal{S}$. Let $\tilde{s}$ defined as previously. For any point $p \in X$, either $|\text{cost}(p, s) - \text{cost}(p, \tilde{s})| \leq \varepsilon(\text{cost}(p, s) + \text{cost}(p, \mathcal{A}))$ or both $\text{dist}(p, s)$ and $\text{dist}(p, \tilde{s})$ are bigger than $\frac{10z \cdot \text{dist}(p, \mathcal{A})}{\varepsilon}$. ◀

Proof. Fix $s \in \mathcal{S} \cap \Pi$, and let $\tilde{s}$ be its corresponding point in $\tilde{\mathcal{S}}$. Let $s_1 \in \{s, \tilde{s}\}$, and s_2 the other choice: we will show that $\text{dist}(p, s_1) \leq (1 + \varepsilon)\text{dist}(p, s_2) + \varepsilon\text{dist}(p, \mathcal{A})$. This implies that the costs verify the same inequality, which will allow us to conclude, switching the roles of s_1 and s_2.

First, in the case where $p \in \Pi \cap X$, then the rounding directly ensures that either $\text{dist}(p, s) > 1/\varepsilon \cdot \text{dist}(x, \mathcal{A})$, in which case it holds as well than $\text{dist}(p, \tilde{s}) > 1/\varepsilon \cdot \text{dist}(x, \mathcal{A})$, or $|\text{dist}(p, s) - \text{dist}(p, \tilde{s})| \leq \varepsilon\text{dist}(x, \mathcal{A}))$.

Otherwise, p is separated from s_1 by some path among $\{P_1, ..., P_m\}$. Let i be the smallest integer such that P_i intersects the shortest path between p and s_1. Our argument depends on the type of tuple $\tilde{d}_i$ chosen for s. Since s and $\tilde{s}$ have the same rounded distance tuples $\tilde{d}_1, \tilde{d}_2, ...$, they have in particular the same rounded distance $\tilde{d}_i$. Let q_1^i be the point with smallest $\text{dist}_i(q, s)$ value (importantly, the q_1^i, q_2^i, q_3^i and q_4^i appearing in the proof are defined with respect to s, not to s_1).

If we can estimate $\mathbf{dist}_i(q_1^i, s)$. In the first case, there is a q_2^i such that $\varepsilon \mathrm{dist}_i(q_1^i, s) \leq \mathrm{dist}_i(q_1^i, q_2^i) + \mathrm{dist}(q_2^i, \mathcal{A}) \leq \frac{1}{\varepsilon}\mathrm{dist}_i(q_1^i, s)$. We let $D := \mathrm{dist}_i(q_1^i, q_2^i) + \mathrm{dist}(q_2^i, \mathcal{A})$ our (rough) estimate on the distance $\mathrm{dist}_i(q_1^i, s)$.

Then, our argument goes as follows. Let x be a point in the intersection of P_i and the shortest path $s_1 \rightsquigarrow p$. We have the following properties: by choice of i, $\mathrm{dist}_i(p, s_1) = \mathrm{dist}(p, s_1)$ and $\mathrm{dist}_i(x, s_1) = \mathrm{dist}(x, s_1)$. By choice of x, $\mathrm{dist}(p, x) + \mathrm{dist}(x, s_1) = \mathrm{dist}(p, s_1)$. Last, by choice of q_1^i, $\mathrm{dist}_i(q_1^i, s) \leq \mathrm{dist}_i(p, s)$, and $D \leq \frac{\mathrm{dist}_i(p,s)}{\varepsilon}$.

- First, if $\mathrm{dist}_i(x, q_1^i) \leq \frac{D}{\varepsilon^2}$. Then there is a point n from N_{i,q_1^i,q_2^i} with $\mathrm{dist}_i(n, x) \leq \varepsilon^2 D$. Furthermore, $\mathrm{dist}_i(s_1, n) = \mathrm{dist}_i(s_2, n) \pm \varepsilon^2 D$, as $\mathrm{dist}_i(s, n) \leq \mathrm{dist}_i(s, x) + \mathrm{dist}_i(x, q_1^i) + \mathrm{dist}_i(q_1^i, s) \leq \frac{3D}{\varepsilon^2}$ and so n has same rounded distances to s_1 and s_2. Hence, we get:

$$\begin{aligned}
\mathrm{dist}_i(p, s_2) &\leq \mathrm{dist}_i(p, x) + \mathrm{dist}_i(x, n) + \mathrm{dist}_i(n, s_2) \\
&\leq \mathrm{dist}_i(p, x) + \mathrm{dist}_i(x, n) + \mathrm{dist}_i(n, s_1) + \varepsilon^2 D \\
&\leq \mathrm{dist}_i(p, x) + \mathrm{dist}_i(x, s_1) + 2\mathrm{dist}_i(x, n) + \varepsilon^2 D \\
&\leq \mathrm{dist}_i(p, s_1) + 3\varepsilon^2 D \\
&\leq \mathrm{dist}(p, s_1) + 3\varepsilon \mathrm{dist}_i(p, s).
\end{aligned}$$

  Now, two cases: either $s = s_1$ and $\mathrm{dist}_i(p, s) = \mathrm{dist}(p, s)$, and then we get $\mathrm{dist}(p, s_2) \leq (1 + 3\varepsilon)\mathrm{dist}(p, s_1)$. Or $s = s_2$, and we have $(1 - 3\varepsilon)\mathrm{dist}(p, s) \leq \mathrm{dist}(p, \tilde{s})$ which implies $\mathrm{dist}(p, s_2) \leq (1 + 6\varepsilon)\mathrm{dist}(p, s_1)$.

- Otherwise, $\mathrm{dist}_i(x, q_1^i) > \frac{D}{\varepsilon^2}$: we first show that $\mathrm{dist}(s, \tilde{s}) \leq 3\varepsilon(\mathrm{dist}_i(p, s) + \mathrm{dist}(p, \mathcal{A}))$, which will allow to conclude. It holds that $\mathrm{dist}_i(q_1^i, s) = \mathrm{dist}_i(q_1^i, \tilde{s}) \pm \varepsilon D$, as by definition of D, $\mathrm{dist}_i(q_1^i, s) \leq D/\varepsilon$. Hence,

$$\begin{aligned}
\mathrm{dist}_i(s, \tilde{s}) &\leq \mathrm{dist}_i(s, q_1^i) + \mathrm{dist}_i(\tilde{s}, q_1^i) \leq 2\mathrm{dist}_i(s, q_1^i) + \varepsilon D \\
&\leq \frac{2 + \varepsilon^2}{\varepsilon} D \leq 3\varepsilon \mathrm{dist}_i(x, q_1^i) \\
&\leq 3\varepsilon(\mathrm{dist}_i(x, s_1) + \mathrm{dist}_i(s_1, s_2) + \mathrm{dist}_i(s, q_1^i)) \\
\Rightarrow \mathrm{dist}(s, \tilde{s}) &\leq 9\varepsilon(\mathrm{dist}(p, s_1) + \mathrm{dist}_i(p, s)).
\end{aligned}$$

  Hence,

$$\begin{aligned}
\mathrm{dist}_i(p, s_2) &\leq \mathrm{dist}_i(p, s_1) + \mathrm{dist}_i(s, \tilde{s}) \\
&\leq \mathrm{dist}(p, s_1) + 9\varepsilon(\mathrm{dist}(p, s_1) + \mathrm{dist}_i(p, s))
\end{aligned}$$

  Similarly as in the previous case, either $s_1 = s$ and the right hand side is $(1 + 18\varepsilon)\mathrm{dist}(p, s_1)$, or $s_2 = s$ and we infer $\mathrm{dist}(p, s_2) \leq (1 + 27\varepsilon)\mathrm{dist}(p, s_1)$.

When we can only overestimate or underestimate $\mathbf{dist}_i(q_1^i, s)$ In the second case, q_3^i is such that $\mathrm{dist}(q_3^i, \mathcal{A}) + \mathrm{dist}_i(q_1^i, q_3^i) > \frac{1}{\varepsilon}\mathrm{dist}_i(q_1^i, s)$, and has minimal $\mathrm{dist}_i(q_1^i, q_3^i) + \mathrm{dist}(q_3^i, \mathcal{A})$ value among those. Similarly, q_4^i is the point with largest $\mathrm{dist}_i(q_1^i, q_4^i) + \mathrm{dist}(q_4^i, \mathcal{A})$ value among those verifying $\mathrm{dist}_i(q_1^i, q) + \mathrm{dist}(q, \mathcal{A}) < \varepsilon \mathrm{dist}_i(q_1^i, s)$.

By choice of q_3^i and q_4^i, it must be that

$$\frac{1}{\varepsilon} \cdot (\mathrm{dist}_i(q_1^i, q_4^i) + \mathrm{dist}(q_4^i, \mathcal{A})) < \mathrm{dist}_i(q_1^i, s) < \varepsilon \cdot (\mathrm{dist}_i(q_1^i, q_3^i) + \mathrm{dist}(q_3^i, \mathcal{A})).$$

Hence, $\tilde{d}^2(q_1^i, q_3^i, q_4^i) = \top$, and $\tilde{s}$ is chosen such that

$$\frac{1}{\varepsilon} \cdot (\mathrm{dist}_i(q_1^i, q_4^i) + \mathrm{dist}(q_4^i, \mathcal{A})) < \mathrm{dist}_i(q_1^i, \tilde{s}) < \varepsilon \cdot (\mathrm{dist}_i(q_1^i, q_3^i) + \mathrm{dist}(q_3^i, \mathcal{A})).$$

Since we are not in the first case where we can estimate $\mathrm{dist}_i(q_1^i, s)$, p verifies either $\mathrm{dist}(p, \mathcal{A}) + \mathrm{dist}_i(p, q_1^i) < \varepsilon \mathrm{dist}_i(q_1^i, s)$ or $\mathrm{dist}(p, \mathcal{A}) + \mathrm{dist}_i(p, q_1^i) > \frac{1}{\varepsilon}\mathrm{dist}_i(q_1^i, s)$.

First, if $\mathrm{dist}_i(p, q_1^i) + \mathrm{dist}(p, \mathcal{A}) > \frac{1}{\varepsilon}\mathrm{dist}_i(q_1^i, s)$. Then we have, by choice of q_3^i:

$$\begin{aligned}
\mathrm{dist}_i(s, \tilde{s}) &\leq \mathrm{dist}_i(s, q_1^i) + \mathrm{dist}_i(\tilde{s}, q_1^i) \\
&\leq 2\varepsilon(\mathrm{dist}_i(q_1^i, q_3^i) + \mathrm{dist}(q_3^i, \mathcal{A})) \\
&\leq 2\varepsilon(\mathrm{dist}_i(q_1^i, p) + \mathrm{dist}(p, \mathcal{A})) \\
&\leq 2\varepsilon(\mathrm{dist}_i(p, s) + \mathrm{dist}_i(q_1^i, s) + \mathrm{dist}(p, \mathcal{A})) \\
&\leq 4\varepsilon(\mathrm{dist}_i(p, s) + \mathrm{dist}(p, \mathcal{A}))
\end{aligned}$$

and therefore, we can conclude just as before (distinguishing whether $s = s_1$ or $s = s_2$) that

$$\mathrm{dist}(p, s_2) \leq (1 + 12\varepsilon)\mathrm{dist}(p, s_1) + 12\varepsilon \mathrm{dist}(p, \mathcal{A}).$$

Lastly, in the case where $\mathrm{dist}_i(p, q_1^i) + \mathrm{dist}(p, \mathcal{A}) < \varepsilon \mathrm{dist}_i(q_1^i, s)$, we use that $\mathrm{dist}_i(q_1^i, s_1) > \frac{1}{\varepsilon} \cdot (\mathrm{dist}_i(q_1^i, q_4^i) + \mathrm{dist}(q_4^i, \mathcal{A}))$ (as both s and $\tilde{s}$ verifies this) to get:

$$\begin{aligned}
\mathrm{dist}(p, s_1) = \mathrm{dist}_i(p, s_1) &\geq \mathrm{dist}_i(q_1^i, s_1) - \mathrm{dist}_i(p, q_1^i) \\
&\geq \frac{1}{\varepsilon} \cdot (\mathrm{dist}_i(q_1^i, q_4^i) + \mathrm{dist}(q_4^i, \mathcal{A})) - \mathrm{dist}_i(p, q_1^i) \\
&\geq \frac{1}{\varepsilon} \cdot (\mathrm{dist}_i(q_1^i, p) + \mathrm{dist}(p, \mathcal{A})) - \mathrm{dist}_i(p, q_1^i) \\
&\geq \frac{\mathrm{dist}(p, \mathcal{A})}{\varepsilon}.
\end{aligned}$$

Conclusion. Rescaling ε by $1/27z$, the previous inequalities gives us that either $\mathrm{dist}(p, s_1) \geq \frac{27z \cdot \mathrm{dist}(p, \mathcal{A})}{\varepsilon}$, or $\mathrm{dist}(p, s_2) \leq (1 + \varepsilon/z)\mathrm{dist}(p, s) + \varepsilon/z \cdot \mathrm{dist}(p, \mathcal{A})$. The second inequality combined with Lemma 1.2 implies that $\mathrm{cost}(p, s_2) \leq (1 + \varepsilon)\mathrm{cost}(p, s_1) + \varepsilon \cdot \mathrm{cost}(p, \mathcal{A})$. Therefore, using this result with $s_1 = s, s_2 = \tilde{s}$ and then $s_1 = \tilde{s}, s_2 = s$ shows that:

- either $\mathrm{dist}(p, s) \geq \frac{27z \cdot \mathrm{dist}(p, \mathcal{A})}{\varepsilon}$, or $\mathrm{cost}(p, \tilde{s}) \leq (1 + \varepsilon)\mathrm{cost}(p, s) + \varepsilon \cdot \mathrm{cost}(p, \mathcal{A})$
- either $\mathrm{dist}(p, \tilde{s}) \geq \frac{27z \cdot \mathrm{dist}(p, \mathcal{A})}{\varepsilon}$, or $\mathrm{cost}(p, s) \leq (1 + \varepsilon)\mathrm{cost}(p, \tilde{s}) + \varepsilon \cdot \mathrm{cost}(p, \mathcal{A})$.

Therefore, if p is such that $\text{dist}(p,s) \leq \frac{5z \cdot \text{dist}(p,\mathcal{A})}{\varepsilon}$, then $\text{dist}(p,\tilde{s}) \leq \frac{10z \cdot \text{dist}(p,\mathcal{A})}{\varepsilon}$, and reciprocally when $\text{dist}(p,\tilde{s}) \leq \frac{5z \cdot \text{dist}(p,\mathcal{A})}{\varepsilon}$, then $\text{dist}(p,s) \leq \frac{10z \cdot \text{dist}(p,\mathcal{A})}{\varepsilon}$.

Thus we conclude: either both $\text{dist}(p,\tilde{s})$ and $\text{dist}(p,s)$ are bigger than $\frac{10z \cdot \text{dist}(p,\mathcal{A})}{\varepsilon}$, and we are done. Or both are smaller than $\frac{20z \cdot \text{dist}(p,\mathcal{A})}{\varepsilon}$, and then using the previous inequalities we get:

$$|\text{cost}(p,s) - \text{cost}(p,\tilde{s})| \leq 2\varepsilon(\text{cost}(p,s) + \text{cost}(p,\tilde{s}) + \text{cost}(p,\mathcal{A}))$$

Which, using $\text{cost}(p,\tilde{s}) \leq \text{cost}(p,s) + |\text{cost}(p,s) - \text{cost}(p,\tilde{s})|$, yields

$$|\text{cost}(p,s) - \text{cost}(p,\tilde{s})| \leq 7\varepsilon(\text{cost}(p,s) + \text{cost}(p,\mathcal{A})).$$

□

Lemma 7.18 gives exactly the same guarantee as Eq. (7.3): hence, as in the proof for treewidth, we can conclude from that inequality that for any solution $\mathcal{S}$ and any interesting point p, $|\text{cost}(p,\mathcal{S}) - \text{cost}(p,\tilde{\mathcal{S}})| \leq \varepsilon(\text{cost}(p,\mathcal{S}) + \text{cost}(p,\mathcal{A}))$.

Combining the guarantees from Lemma 7.18 and Lemma 7.17 concludes the proof of Lemma 7.14.

7.6 Euclidean Spaces

Lastly, we briefly want to survey the state of the art results for eliminating the dependency on the dimension in Euclidean spaces.

In a nutshell, the frameworks by both Feldman and Langberg [76] and us only yield coresets of size $O(kd\text{poly}(\log k, \varepsilon^{-1}))$. To eliminate the dependency on the dimension, we typically have to use some form of dimension reduction.

In a landmark paper, Feldman et al. [78] showed that one can replace the dependency on d with a dependency on k/ε^2 for the k-means problem, see also Cohen et al. [48] for further improvements on this idea. Subsequently, Sohler and Woodruff [151] gave a construction for arbitrary k-clustering objectives which lead to the first existence proof of dimension independent coresets for these problems. More recently, the result was made constructive in the work of Feng, Kacham and Woodruff [80]. Huang and Vishnoi [100] showed that the mere existence of the Sohler-Woodruff construction was enough to compute coresets of size $\text{poly}(k/\varepsilon)$.

Having obtained a $\text{poly}(k/\varepsilon)$-sized coreset, one can now use a terminal embedding to replace the dependency on d by a dependency $\varepsilon^{-2}\log k/\varepsilon$. Terminal embeddings are defined as follows:

▶ **Definition 7.19 (Terminal Embeddings).** Let $\varepsilon \in (0,1)$ and let $A \subset \mathbb{R}^d$ be an arbitrary set of n points. Define the Euclidean norm of a d-dimensional vector $\|x\| = \sqrt{\sum_{i=1}^{d} x_i^2}$. Then a mapping $f : \mathbb{R}^d \to \mathbb{R}^m$ is a *terminal embedding* if

$$\forall x \in A,\ \forall y \in \mathbb{R}^d,\ (1-\varepsilon) \cdot \|x-y\| \leq \|f(x)-f(y)\| \leq (1+\varepsilon) \cdot \|x-y\|.$$

◀

Terminal embeddings were studied by [71, 127, 145], with Narayanan and Nelson [145] achieving an optimal target dimension of $O(\varepsilon^{-2} \log n)$, where n is the number of points[3].

It was first observed by Becchetti et al. [25] how terminal embeddings can be combined with the Feldman-Langberg [76] (or indeed our) framework. Specifically, given the existence of a poly(k/ε)-sized coreset, applying a terminal embedding with n being the number of distinct points in the coreset now allows us to further reduce the dimension. At the time, the only problem with such a coreset bound was k-means. The generalization to arbitrary k-clustering objectives is now immediate following the results by Huang and Vishnoi [100] and Feng et al. [80].

It should be noted that more conventional Johnson-Lindenstrauss type embeddings proposed in [25, 48, 129] do not (obviously) imply the same guarantee as terminal embeddings. We appended a short proof showing that terminal embeddings are sufficient at the end of this section. For a more in-depth discussion as to why normal Johnson-Lindenstrauss transforms may not be sufficient, we refer to Huang and Vishnoi [100].

Combining our $O(k(d+\log k) \cdot \varepsilon^{-\max(2,z)})$ bound for general Euclidean spaces with either the Huang and Vishnoi [99] or the Feng et al. [80] constructions and terminal embeddings now immediately imply the following corollary.

▶ **Corollary 7.20.** There exists a coreset of size

$$O\left(k \log k \cdot \left(\varepsilon^{-2-\max(2,z)}\right) \cdot 2^{O(z \log z)} \cdot \operatorname{polylog}(\varepsilon^{-1})\right)$$

for (k,z)-clustering in Euclidean spaces. ◀

Huang and Vishnoi further considered clustering in ℓ_p metrics for $p \in [1,2)$, i.e. non-Euclidean spaces. For this they reduced constructing a coreset for (k,z) clustering in an ℓ_p space to constructing a constructing a coreset for $(k,2z)$ clustering in Euclidean space. Plugging in our framework into their reduction then yields the following corollary:

[3] See the paper by Larsen and Nelson for a matching lower bound [116]

▶ **Corollary 7.21.** There exists a coreset of size

$$O\left(k \log k \cdot \left(\varepsilon^{-2-2z}\right) \cdot 2^{O(z \log z)} \cdot \text{polylog}(\varepsilon^{-1})\right)$$

for (k, z)-clustering in any ℓ_p space for $p \in [1, 2)$. ◀

▶ **Proposition 7.22.** Suppose we have a (possibly weighted) point set A in $\mathbb{R}^d$. Let $f : \mathbb{R}^d \to \mathbb{R}^m$ with $m \in O(\varepsilon^{-2} \cdot z^2 \log n)$ be a terminal embedding for A and let $f(A)$ be the projected point set. Then if $f(P) \subset f(A)$ is an ε-coreset for $f(A)$, $P \subset A$ is an $O(\varepsilon)$-coreset for A. Conversely, if $P \subset A$ is an ε-coreset for A, then $f(P) \subset f(A)$ is an $O(\varepsilon)$-coreset for $f(A)$ ◀

Proof. We prove the result for the first direction, the other direction is analogous. Consider an arbitrary solution S in $\mathbb{R}^d$. We first notice that for any point $p \in A$, we have

$$(1 - \varepsilon/2z)^z \cdot \text{cost}(f(p), f(\mathcal{S})) \leq (1 - \varepsilon) \cdot \text{cost}(f(p), f(\mathcal{S}))$$

and

$$(1 + \varepsilon/2z)^z \cdot \text{cost}(f(p), f(\mathcal{S})) \geq (1 + \varepsilon) \cdot \text{cost}(f(p), f(\mathcal{S}))$$

Therefore,

$$(1 - \varepsilon) \cdot \text{cost}(f(p), f(\mathcal{S})) \leq \text{cost}(p, \mathcal{S}) \leq (1 + \varepsilon) \cdot \text{cost}(f(p), f(\mathcal{S})). \tag{7.5}$$

Now suppose $f(P)$ is a coreset for $f(A)$, which means for any set of k points $f(S) \subset \mathbb{R}^m$

$$\left| \sum_{p \in f(A)} w_p \cdot \text{cost}(p, f(\mathcal{S})) - \sum_{q \in f(P)} w'_q \cdot \text{cost}(q, f(\mathcal{S})) \right| \leq \varepsilon \cdot \sum_{p \in f(A)} w_p \cdot \text{cost}(p, f(\mathcal{S})), \tag{7.6}$$

where w and w' are the weights assigned to points in $f(A)$ and $f(P)$, respectively. Let us now consider a solution S in the original d-dimensional space. Since P is a subset of A, we have by combining Equations 7.5 and 7.6

$$\begin{aligned}
& \left| \sum_{p \in A} w_p \cdot \text{cost}(p, \mathcal{S}) - \sum_{q \in P} w'_q \cdot \text{cost}(p, \mathcal{S}) \right| \\
\leq\ & \varepsilon \cdot \sum_{p \in A} w_p \cdot \text{cost}(f(p), f(\mathcal{S})) + \varepsilon \cdot \sum_{q \in P} w'_q \cdot \text{cost}(f(q), f(\mathcal{S})) \\
& + \left| \sum_{p \in A} w_p \cdot \text{cost}(f(p), f(\mathcal{S})) - \sum_{q \in P} w'_q \cdot \text{cost}(f(q), f(\mathcal{S})) \right| \\
\leq\ & 2\varepsilon \cdot \sum_{p \in A} w_p \cdot \text{cost}(f(p), f(\mathcal{S})) + \varepsilon \cdot \sum_{q \in P} w'_q \cdot \text{cost}(f(q), f(\mathcal{S})) \\
\leq\ & (3 + \varepsilon)\varepsilon \cdot \sum_{p \in A} w_p \cdot \text{cost}(f(p), f(\mathcal{S})) \\
\leq\ & (3 + 3\varepsilon)\varepsilon \cdot \sum_{p \in A} w_p \cdot \text{cost}(p, \mathcal{S}),
\end{aligned}$$

where the second inequality uses Equation 7.6 and the triangle inequality and the last inequality uses Equation 7.5. □

Chapter 8

Lower Bounds for Coreset

While the previous chapters focused on coreset construction, we now turn to impossibility result and answer Question 4 for the case of metric with bounded doubling dimension, and general discrete metric spaces. This lower bound matches (up to polylog$(1/\varepsilon)$ factors) the constructions presented previously.

8.1 Introduction

We show in this chapter a lower bound for coreset in discrete metric spaces. This lower bound is for a more general notion of coreset, that allows for having an *offset* that encodes part of the cost:

▶ **Definition 8.1.** An ε-coreset with offset for (k, z)-clustering in a metric space (X, dist) for clients P is a weighted subset Ω of P, with weights $w : \Omega \to \mathbb{R}_+$, and an offset F, such that for any set $\mathcal{S} \subset X$, $|S| = k$,

$$\left| \sum_{p \in P} \mathrm{cost}(p, \mathcal{S}) - \left(\sum_{p \in \Omega} w(p) \mathrm{cost}(p, \mathcal{S}) + F \right) \right| \leq \varepsilon \sum_{p \in P} \mathrm{cost}(p, \mathcal{S}) \qquad \blacktriangleleft$$

In the standard ε-coreset we used in the previous chapters, $F = 0$. This more general definition was introduced by Feldman et al [78], and used thereafter by [48, 151]. This is also the definition we will use in Chapter 9. In case Ω satisfies the definition with offset F, we say Ω is an ε-coreset for (k, z)-clustering using offset F. The main result of the present chapter is:

▶ **Theorem 8.2.** For any $0 < \varepsilon < 1/2$, k and $n \geq 2k\varepsilon^{-5}$, there exists a finite n point metric with a set P of $k\varepsilon^{-2}\log n$ client such that any ε-coreset for (k,z)-clustering on P consists of at least $\Omega\left(\frac{k}{\varepsilon^2}\log n\right)$ points. ◀

We start by giving a precise overview, before proving formally the lower bound.

8.1.1 Overview of the Lower Bound

The general idea behind our lower bound is to use the tight concentration and anti-concentration bounds on the sum of random variables.

We first build an instance for $k = 1$, and combines several copies of it to obtain a lower bound for any arbitrary k. Our instance for $k = 1$ has a set C of candidate centers and is such that: (1) when $|\Omega| \leq \varepsilon^{-2}\log|C|$ there exists a center with $\mathrm{cost}(\Omega, c) > (1 + 100\varepsilon)\mathrm{cost}(P, c)$, and (2): for any $|\Omega| > \varepsilon^{-2}\log|C|$ there exists a center c with $\mathrm{cost}(\Omega, c) \in (1 \pm \varepsilon)\mathrm{cost}(P, c)$.

To show the existence of such an instance, we consider a complete bipartite graph with nodes $P \cup C$ where there is an edge between each point of P and each point of C, with length 1 with probability $\pi = 1/4$ and length 2 otherwise. The set of clients is P. For simplicity, we will assume in this overview that the coreset weights are uniform. Making the idea work for non-uniform weights requires several other technical ingredients.

In that instance for $k = 1$, the cost of a solution (with a single center, c) is fully determined by $n_1(c)$, the number of length 1 edges to c. Indeed, $\mathrm{cost}(P, c) = 2(|P| - n_1(c)) + n_1(c) = 2|P| - n_1(c)$. Let us further assume that $n_1(c)$ is equal to its expectation, $\pi|P|$. For a fixed subset of points Ω, the cost of the solution for Ω with uniform weights $\frac{|P|}{|\Omega|}$ satisfies the same equation: it is $2|P| - n_1(\Omega, c) \cdot \frac{|P|}{|\Omega|}$, where $n_1(\Omega, c)$ is the number of length 1 edges from Ω to c. Note that $\mathbb{E}[n_1(\Omega, c)] = \pi|\Omega|$.

Using anti-concentration inequalities, we show that $n_1(\Omega, c) > (1 + 200\varepsilon)\mathbb{E}[n_1(\Omega, c)]$ with probability at least $\exp(-\alpha\varepsilon^2|\Omega|)$, for some constant α. When this event happens, then Ω does not preserve the cost of solution c: indeed,

$$\begin{aligned} 2|P| - n_1(\Omega, c) \cdot \frac{|P|}{|\Omega|} &< 2|P| - (1 + 200\varepsilon)\pi|\Omega| \cdot \frac{|P|}{|\Omega|} \\ &= 2|P| - \pi|P| - 200\varepsilon\pi|P| = (1 - 200\varepsilon)(2|P| - n_1(c)), \end{aligned}$$

where we assumed $n_1(c) = \pi|P|$ for the last equality.

Since the edges are drawn independently, the coreset costs for all possible centers c are independent. Hence, there exists one center with $n_1(\Omega, c) > (1 + 200\varepsilon)\pi|\Omega|$ with probability at least $1 - (1 - \exp(-\alpha\varepsilon^2|\Omega|))^{|C|}$. By doing a union-bound over all possible subsets Ω, one can show the following: with positive (close to 1) probability, for any $|\Omega| \leq \varepsilon^{-2}\log|C|$ there exists a center with $\mathrm{cost}(\Omega, c) > (1 + 100\varepsilon)\mathrm{cost}(P, c)$.

Using standard concentration inequality, one can show that with probability close to

1, for any $|\Omega| > \varepsilon^{-2} \log |C|$, there exists a center c with $\text{cost}(\Omega, c) \in (1 \pm \varepsilon)\text{cost}(P, c)$. Since the probabilities are taken on the edges randomness, those two result ensure the existence of a graph that verifies properties (1) and (2) desired for the $k = 1$ instance.

Now, the full instance is made of k distinct copies $X_1, ..., X_k$ of the $k = 1$ instance, placed at infinite distance from each other. Let P_i be the set of clients of X_i: the clients for the full instance are $\cup P_i$. Let Ω be a set of at most $1/100 \cdot k\varepsilon^{-2} \log n$ points: we show that Ω cannot be a coreset. By Markov's inequality, there are at least $99/100k$ copies that contain less than $\varepsilon^{-2} \log n$ points of Ω. We say those copies are *bad*, the others are *good*. Consider now the solution $\mathcal{S}$ defined as follows: from each X_i, take the center such that $\text{cost}(\Omega \cap P_i, c) > (1 + 100\varepsilon)\text{cost}(P_i, c)$ when X_i is bad, and the center such that $\text{cost}(\Omega \cap P_i, c) \in (1 \pm \varepsilon)\text{cost}(P_i, c)$ when X_i is good. Observe also that by construction of the instance for $k = 1$, the cost in each copy must lie in $[|P|, 2|P|]$. For that solution, we have:

$$\begin{aligned}
\text{cost}(\Omega, \mathcal{S}) &= \sum \text{cost}(\Omega \cap P_i, s_i) = \sum_{i \text{ bad}} \text{cost}(\Omega \cap P_i, s_i) + \sum_{i \text{ good}} \text{cost}(\Omega \cap P_i, s_i) \\
&> \sum_{i \text{ bad}} (1 + 100\varepsilon)\text{cost}(P_i, s_i) + \sum_{i \text{ good}} (1 - \varepsilon)\text{cost}(P_i, s_i) \\
&> \text{cost}(\mathcal{S}) + \frac{99k}{100} \cdot 100\varepsilon|P| - \frac{k}{100} \cdot \varepsilon 2|P| \\
&> \text{cost}(\mathcal{S}) + 98k\varepsilon|P| > (1 + \varepsilon)\text{cost}(\mathcal{S}).
\end{aligned}$$

Hence, any Ω with $|\Omega| \leq 1/100 \cdot k\varepsilon^{-2} \log n$ cannot be a coreset for our instance, which concludes the proof.

8.1.2 Technical lemmas

Our proof relies on Corollary 8.4, which we prove using the following result from [73].

▶ **Lemma 8.3 (Equation 2.11 in [73]).** Let $\xi_1, ..., \xi_m$ be independent centered random variables, and $\tilde{\varepsilon}$ such that

$$\forall i, k \geq 3 \ |\mathbb{E}[\xi_i^k]| \leq \frac{1}{2} k! \tilde{\varepsilon}^{k-2} \mathbb{E}[\xi_i^2].$$

Let $\sigma^2 = \sum \mathbb{E}[\xi_i^2]$, and $S_m = \sum_{i=1}^m \xi_i$.
Then, for all $0 \leq x \leq 0.1\frac{\sigma}{\tilde{\varepsilon}}$,

$$\Pr[S_m \geq x\sigma] \geq \left(1 - \Phi\left(x(1 - cx\frac{\tilde{\varepsilon}}{\sigma}\right)\right) \cdot \left(1 - c(1 + x)\frac{\tilde{\varepsilon}}{\sigma}\right),$$

where c is an absolute positive constant and Φ is the standard normal distribution function. ◀

▶ **Corollary 8.4.** Let $X_1, ..., X_m$ be independent Bernouilli random variables with expectation $p = 1/4$, $\varepsilon > 0$ and $w_1, ..., w_m$ be some positive weights, such that $\max w_i \leq \gamma \cdot \frac{\sum w_i}{\varepsilon m}$, for some γ. Let $\mu = p \cdot \sum w_i$. Then, there exists a constant β such that

$$\Pr\left[\sum w_i X_i - \mu > \varepsilon\mu\right] \geq \exp(-\frac{\beta}{\gamma^2}\varepsilon^2 mp) \qquad \blacktriangleleft$$

Proof. Define $\xi_i = w_i X_i - pw_i$. We show that the variables ξ_i verify the conditions of Lemma 8.3. They are independent, centered, and:

$$\begin{aligned}\mathbb{E}[\xi_i^2] &= p(w_i - pw_i)^2 + (1-p)(pw_i)^2 \\ &= w_i^2\left(p - 2p^2 + p^3 + p^2 - p^3\right) \\ &= w_i^2(p - p^2) \geq \frac{w_i^2}{8},\end{aligned}$$

using $p = 1/4$. For $k \geq 3$, the k-th moment verifies:

$$\left|\mathbb{E}[\xi_i^k]\right| = w_i^k \cdot \left(p \cdot (1-p)^k + (1-p) \cdot p^k\right) \leq \frac{w_i^k}{8} = w_i^{k-2}\mathbb{E}[\xi_i^2],$$

where the last line uses that the function $k \to p \cdot (1-p)^k + (1-p) \cdot p^k$ is decreasing, and is equal to $15/128 \leq$ ‘$q1/8$ for $p = 1/4$ and $k = 3$. Hence ξ_i verifies the condition of Lemma 8.3 with $\tilde{\varepsilon} = \max_i w_i$. As in the lemma, we let $\sigma^2 = \sum \mathbb{E}[\xi_i^2]$. We want to apply that lemma to x of the order $\varepsilon\frac{\mu}{\sigma}$: therefore, we need to bound that quantity. Note that

$$\sigma^2 \geq \frac{p}{2}\sum w_i^2 \geq \frac{p}{2} \cdot \frac{(\sum w_i)^2}{m}, \tag{8.1}$$

and so by the assumptions of the lemma $\frac{\sigma}{\tilde{\varepsilon}} \geq \frac{\varepsilon\sqrt{mp}}{\gamma\sqrt{2}}$. Furthermore,

$$\frac{\mu}{\sigma} \leq \frac{p\sum w_i}{\sqrt{\frac{p}{2m}}\sum w_i} \leq \sqrt{2mp} \tag{8.2}$$

Now, let $x := \frac{\varepsilon}{10\gamma c\sqrt{2}} \cdot \frac{\mu}{\sigma}$. Thus, x verifies $x \leq \frac{\varepsilon}{10\gamma c\sqrt{2}} \cdot \sqrt{2pm} \leq 0.1\frac{\sigma}{c\tilde{\varepsilon}}$ and so applying Lemma 8.3 we obtain:

$$\begin{aligned}\Pr\left[\sum w_i X_i - \mu > \varepsilon\mu\right] &\geq \left(1 - \Phi\left(x(1 - cx\frac{\tilde{\varepsilon}}{\sigma})\right)\right) \cdot \left(1 - c(1+x)\frac{\tilde{\varepsilon}}{\sigma}\right) \\ &\geq (1 - \Phi(0.9x)) \cdot 0.9 \\ &= 0.9 \cdot \Pr[\mathcal{N}(0,1) \geq 0.9x] \\ &\geq 0.9 \cdot \frac{1}{2}\left(1 - \sqrt{1 - e^{-(0.9x)^2}}\right) \\ &\geq \exp(-\frac{\beta}{\gamma^2}\varepsilon^2\frac{\mu^2}{\sigma^2}) \\ &\geq \exp(-\frac{\beta}{\gamma^2}\varepsilon^2 mp),\end{aligned}$$

where β is some absolute constant, and where the last line uses Eq. (8.2). □

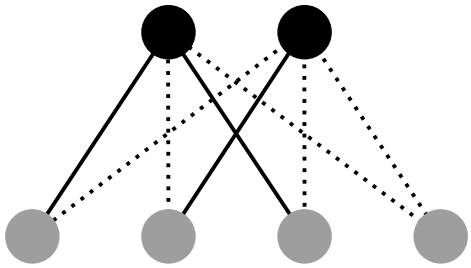


Figure 8.1: Illustration of an instance U_π. Dashed edges have length $2^{1/z}$, black ones have length 1.

8.2 A subinstance for the case $k = 1$

We now turn to proving a lower bound for the case where $k = 1$. This is going to be our building block in the next section where we generalize the result to arbitrary k. Let $\pi = 1/4$ be a parameter.

▶ **Definition 8.5.** A subinstance U_π is defined as follows. Let C be a set of *candidate centers* and P a set of of n_U *clients*. The metric on the ground set $P \cup C$ is defined according to the following probability distribution.
For each pair $(p, c) \in P \times C$,

$$\operatorname{dist}(p, c) = \begin{cases} 1 \text{ with probability } \pi \\ 2^{1/z} \text{ otherwise} \end{cases} \tag{8.3}$$

Distances between any pair of points $p, p' \in P$ or $c, c' \in C$ is set to $2^{1/z}$.
Fig. 8.1 illustrates the definition. ◀

Since any complete graph with edge length only 1 or $\ell \leq 2$ defines a metric space, it immediately follows that $(P \cup C, \operatorname{dist})$ is a metric space.

The important properties of the subinstance are summarized in the following lemma. We say that a set of weights is *ε-rounded* if all weights are multiples of ε.

▶ **Lemma 8.6.** There exists a constant η and an instance $U_\pi = (P, C, \operatorname{dist})$ with $|P| = n_U = \varepsilon^{-2} \log |C|$, $\pi \leq 1/4$ and $|C| \geq \varepsilon^{-5}$, the following holds. For any subset $\Omega \subseteq P$ with $\varepsilon/2$-rounded weights w_x being such that $\sum w_x \in (1 \pm 1/2)n_U$, we have:

1. If $|\Omega| < \varepsilon^{-2}\eta \log |C|$, there exists a center $\tilde{c} \in C$ such that

$$\sum_{x \in \Omega : \operatorname{dist}(x, \tilde{c}) = 1} w_x > (1 + 200\varepsilon)\pi n_U$$

and $|x \in P : \ \operatorname{dist}(x, \tilde{c}) = 1| \leq \lceil \pi n_U \rceil$

2. If $|\Omega| \geq \varepsilon^{-2}\eta \log |C|$, there exists a center $c^* \in C$ such that

$$\sum_{x \in \Omega : \mathrm{dist}(x,c^*)=1} w_x \geq (1-\varepsilon)\pi n_U$$

and $|x \in P : \ \mathrm{dist}(x, c^*) = 1| \leq \lceil \pi n_U \rceil$. ◀

Proof. We use the probabilistic method: we will show that, when U_π is generated according to the process defined above, the two properties of the lemma hold with some positive probability. This is enough to ensure the existence of an instance U_π verifying them.

We start by proving the first item. Fix some arbitrary subset of clients Ω of size at most $\varepsilon^{-2}\eta \log |C|$, with weight $w_x, \forall x \in \Omega$ and a candidate center $c \in C$. Let $w_1(c, \Omega) := \sum_{x \in \Omega : \mathrm{dist}(x,c)=1} w_x$ denote the (weighted) number of edges of length 1 from Ω to c. The expected value of $w_1(c, \Omega)$ over the random choice of edges is $\pi \cdot n_U$. We aim at applying Corollary 8.4 on the variable $w_1(c, \Omega)$. This cannot be done directly, as we have no control on $\max w_x$. Hence, we partition the points of Ω into five groups:

- $\Omega_1 := \{x \in \Omega : \ w_x < \varepsilon\}$
- $\Omega_2 := \{x \in \Omega : \ w_x \in [\varepsilon, 1)\}$
- $\Omega_3 := \{x \in \Omega : \ w_x \in [1, \varepsilon^{-1})\}$
- $\Omega_4 := \{x \in \Omega : \ w_x \in [\varepsilon^{-1}, 10 \log(1/\pi) \cdot \varepsilon^{-2})\}$
- $\Omega_5 := \{x \in \Omega : \ w_x \geq 20 \log(1/\pi) \cdot \varepsilon^{-2}\}$

We will show that, $\forall i \in \{2, ..., 5\}$, $w_1(c, \Omega_i)$ exceeds its expectation by a factor $(1 + 205\varepsilon)$ with large probability, and that $w_1(c, \Omega_1)$ is negligible.

First, note that since $\sum_{x \in \Omega} w_x \leq (1 + 1/2)n_U = \frac{3}{2}\varepsilon^{-2} \log |C|$, it must be that

$$|\Omega_5| \leq \frac{\log |C|}{10 \log(1/\pi)}.$$

Hence, c is connected with length 1 to all points of Ω_5 with probability $\pi^{|\Omega_5|} \geq \exp(-\log |C|/10) = |C|^{-1/10}$.

Now, on each group $\Omega_2, \Omega_3, \Omega_4$, the maximum weight cannot be more than $20 \log(1/\pi) \cdot \varepsilon^{-1}$ times the average.

For $i \in \{2, 3, 4\}$, $w_1(c, \Omega_i)$ is the sum of $m = |\Omega_i| \leq \varepsilon^{-2}\eta \log |C|$ random variables X_x, for $x \in \Omega_i$, with $X_x = 0$ with probability $(1 - \pi)$ and $X_x = w_x$ with probability $\pi = 1/4$. Due to the choice of the groups, $\max_{x \in \Omega_i} w_x \leq 10 \log(1/\pi) \frac{\sum_{y \in \Omega_i} w_y}{\varepsilon |\Omega_i|}$ Hence, Corollary 8.4 gives that:

$$\Pr\left[w_1(c,\Omega_i) \geq (1+205\varepsilon)\cdot\mathbb{E}[w_1(c,\Omega_i)]\right] > \exp(-\frac{\beta}{\log(1/\pi)^2}\varepsilon^2\pi|\Omega_i|)$$
$$\geq \exp(-\log|C|/10),$$

for some absolute constant β given by Corollary 8.4 and $\eta \leq \frac{\log(1/\pi)^2}{10\beta\pi}$.

Finally, to deal with Ω_1, we note that $\mathbb{E}[w_1(c,\Omega_1)] \leq \varepsilon\pi n_U$. Hence, $\sum_{i=2}^{5}\mathbb{E}[w_1(c,\Omega_i)] \geq \mathbb{E}[w_1(c,\Omega)] - \varepsilon\pi n_U$, and

$$\sum_{i=2}^{5} w_1(c,\Omega_i) \geq (1+205\varepsilon)\sum_{i=2}^{5}\mathbb{E}[w_1(c,\Omega_i)] \quad\Rightarrow\quad w_1(c,\Omega) \geq (1+200\varepsilon)\pi n_U.$$

Since all groups are disjoint, the variables $w_1(c,\Omega_i)$ are independents and we can combine the previous equations to get:

$$\Pr\left[\sum_{x\in\Omega:\mathrm{dist}(x,\tilde{c})=1} w_x \geq (1+200\varepsilon)\cdot\pi n_U\right] > |C|^{-4/10}.$$

Since the length of the edges are chosen independently, the probability that there exists no center $\tilde{c}$ with $\sum_{x\in\Omega:\mathrm{dist}(x,\tilde{c})=1} w_x \geq (1+200\varepsilon)\cdot\pi n_U$ is at most

$$\left(1-|C|^{-4/10}\right)^{|C|} = \exp\left(|C|\log(1-|C|^{-4/10})\right)$$
$$\leq \exp(-|C|^{6/10}).$$

And hence with probability at least $1-\exp(-|C|^{6/10})$ there is a center $\tilde{c}$ with $\sum_{x\in\Omega:\mathrm{dist}(x,\tilde{c})=1} w_x \geq (1+200\varepsilon)\cdot\pi n_U$.

To conclude the proof of the first bullet, it remains to do a union-bound over all possible weighted subset Ω. Such an Ω consists of at most n_U different points, with $\varepsilon/2$-rounded weights in $[0,(1+1/2)n_U]$. Hence, there are at most $\frac{4}{\varepsilon}n_U$ many different weights.

Therefore, there are $\left(\frac{4n_U}{\varepsilon}\right)^{n_U}$ many possible weighted subset Ω with $\varepsilon/2$-rounded weights, i.e.,

$$\exp\left(\varepsilon^{-2}\log|C|\cdot\log\left(4\varepsilon^{-3}\log|C|\right)\right).$$

We can conclude that there exists a center $\tilde{c}\in C$ with $\sum_{x\in\Omega:\mathrm{dist}(x,\tilde{c})=1} w_x \geq (1+200\varepsilon)\cdot\pi n_U$ with probability at least

$$1-\exp\left(\varepsilon^{-2}\log|C|\cdot\log\left(4\varepsilon^{-3}\log|C|\right)\right)\cdot\exp(-|C|^{6/10}) \geq \frac{99}{100}$$

by our choice of $|C| \geq \varepsilon^{-5}$, and for ε small enough. Furthermore, $\Pr[|x\in P:\ \mathrm{dist}(x,\tilde{c})=1| \leq \lceil\pi n_U\rceil] \geq 1/2$, because $|x\in P:\ \mathrm{dist}(x,\tilde{c})=1|$ follows a binomial law with median at most $\lceil\pi n_U\rceil$. This concludes the proof of the first bullet.

We now turn to the second bullet of the claim, for which the proof is a more standard application of Azuma inequality. Fix some coreset Ω of size at least $\varepsilon^{-2}\eta \log |C|$, and a center c. We have,

$$\begin{aligned}\Pr\left[w_1(c,\Omega) \notin (1\pm\varepsilon)\cdot \pi n_U\right] &\leq \exp(-2\varepsilon^2\pi^2 \frac{n_U^2}{\sum w_i^2}) \\ &\leq \exp(-2/4\cdot \pi^2\varepsilon^2) \\ &\leq \exp(-1/2\cdot \pi^2\varepsilon^2),\end{aligned}$$

where the second inequality uses $n_U^2 \geq 1/4\left(\sum w_i\right)^2 \geq 1/4 \cdot \sum w_i^2$.

Since those events are independent for different centers c, the probability that there exists no center $c \in C$ with $w_1(c,\Omega) \in (1\pm\varepsilon)\cdot\pi n_U$ is at most $\exp(-1/2\cdot\pi^2\varepsilon^2|C|)$.

Hence, a union-bound over the $\left(\frac{4n_U}{\varepsilon}\right)^{n_U}$ many possible weighted subset Ω ensures that the following holds with probability at most $1-\left(\frac{4n_U}{\varepsilon}\right)^{n_U}\cdot\exp(-1/2\cdot\pi^2\varepsilon^2|C|) \geq 99/100$: For any Ω there exists a center c with $w_1(c,\Omega) \in (1\pm\varepsilon)\cdot\pi|\Omega|$ as desired. □

8.3 Combining the subinstances

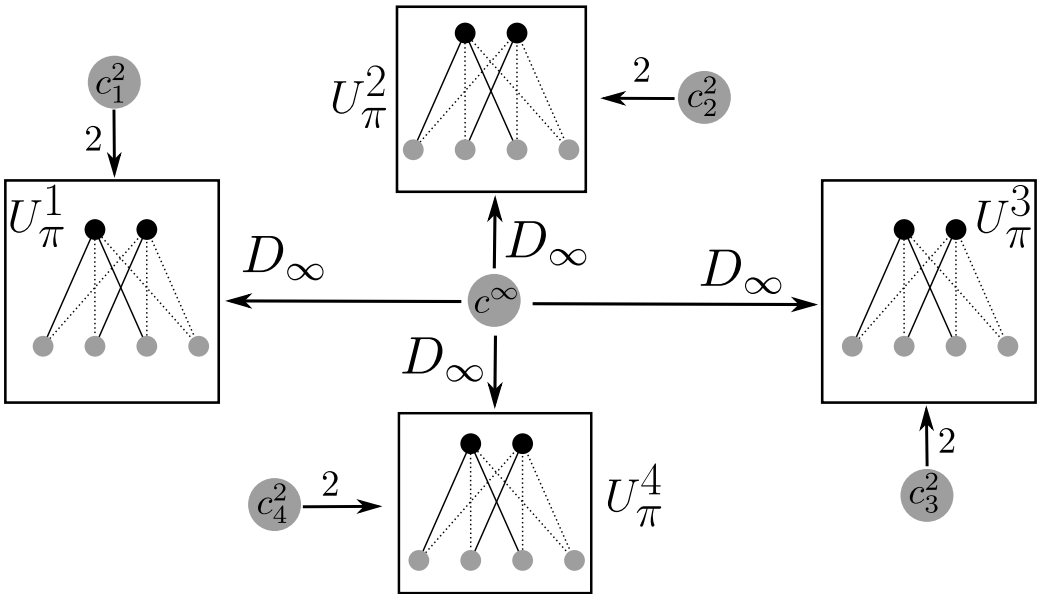


Figure 8.2: Illustration of a full instance, in the case $z = 1$. The subinstance are inside squares, and there is an edge from a node to a square when the node is linked to every point of the subinstance, with the distance written on the edge. D_∞ is set to be $\frac{n_U\cdot k}{\varepsilon}$. The node $c^{4\cdot\infty}$ is not represented.

We now conclude the proof of the lower bound for the ε-coreset with offset for (k,z)-clustering. We consider k copies of the subinstance given by Lemma 8.6, $U_\pi^1,\dots,U_\pi^k$, where the set of clients in each subinstance has size n_U and the set of candidate centers has size $|C|$, such that $|C|\geq\varepsilon^{-5}$, $n_U=10\varepsilon^{-2}\log|C|$ and $k(n_U+|C|)=n$. Using the assumption that $n\geq 2k/\varepsilon^5$, this is possible, and we have the additional property that $\log n=O(\log|C|)$.

In total, there are $k|C|$ many candidate centers, and kn_U many different clients. The subinstances are numbered from 1 to k, and connected together in a star-graph metric centered at an arbitrary point c^∞, where all points are at distance $\frac{n_U\cdot k}{\varepsilon}$ of c^∞. There is some additional candidate centers: $c^{4\cdot\infty}$, at distance $4\cdot\frac{n_U\cdot k}{\varepsilon}$ of every client, and for subinstance i there is a center c_i^2, at distance $2^{1/z}$ from every client of the subinstance. Fig. 8.2 illustrates that construction.

We can now turn to the proof of the theorem. For this, we start with three claims: The first one shows that the total weight of the coreset must be very close to the number of point in the instance. The second shows that the offset F must be negligible, and the third that the coreset weight in each subinstance is close to n_U, the number of point in a subinstance.

▶ **Claim 8.7.** If Ω is an ε-coreset with offset F for the instance, then the total weight verifies $w(\Omega)\in(1\pm 2\varepsilon)kn_U$. ◀

Proof. Consider the solution consisting only of one center placed at c^∞. Let $D_\infty=\frac{n_U\cdot k}{\varepsilon}$. This solution has cost $\mathrm{cost}(c^\infty)=kn_U\cdot D_\infty^z$, and $\mathrm{cost}(\Omega,c^\infty)=w(\Omega)\cdot D_\infty^z$. Hence,

$$F+w(\Omega)\cdot D_\infty^z\in(1\pm\varepsilon)kn_U\cdot D_\infty^z.$$

Similarly, considering the solution that places only one center at $c^{4\cdot\infty}$ gives

$$F+w(\Omega)4^zD_\infty^z\in(1\pm\varepsilon)kn_U\cdot 4^zD_\infty^z.$$

Substracting those two equations yields:

$$(4^z-1)w(\Omega)\cdot D_\infty^z\in((4^z-1)\pm(4^z+1)\varepsilon)kn_U\cdot D_\infty^z,$$

and so $w(\Omega)\in(1\pm 2\varepsilon)kn_U$. □

▶ **Claim 8.8.** If Ω is an ε-coreset with offset F for the instance, then $|F|\leq 3\varepsilon k\cdot n_U$. ◀

Proof. Consider the solution $\mathcal{S}^2=\{c_i^2,\ \forall i\}$. We have $\mathrm{cost}(\mathcal{S}^2)=2kn_U$ and $\mathrm{cost}(\Omega,\mathcal{S}^2)=2w(\Omega)\in(1\pm 2\varepsilon)\mathrm{cost}(\mathcal{S}^2)$, using Claim 8.7. Since $|F+\mathrm{cost}(\Omega,\mathcal{S}^2)-\mathrm{cost}(\mathcal{S}^2)|\leq\varepsilon\mathrm{cost}(\mathcal{S}^2)$, it must be that $|F|\leq 3\varepsilon\mathrm{cost}(\mathcal{S}^2)=3\varepsilon kn_U$.

□

▶ **Claim 8.9.** If Ω is an ε-coreset with offset F for the instance, then in every subinstance, the sum of the coreset weights is in $(1\pm 1/2)n_U$. ◀

Proof. Assume towards contradiction that, in some subinstance, say subinstance i, the coreset mass is not in $(1 \pm 1/2)n_U$, and consider a solution $\mathcal{S}$ that places one center in each subinstance but subinstance i. Suppose w.l.o.g. that the subinstance is overweighted: the coreset places a total weight larger than $3/2 \cdot n_U$ in it. The cost of the solution is a most

$$\begin{aligned}\text{cost}(\mathcal{S}) \leq & \underbrace{2(k-1)n_U}_{\text{for subinstances that contain a center}} + \underbrace{n_U \cdot \left(kn_U\varepsilon^{-1}\right)^z}_{\text{for the overweighted subinstance}} \\ & \leq (1+2\varepsilon)n_U \left(k \cdot n_U\varepsilon^{-1}\right)^z,\end{aligned}$$

while the cost in the coreset verifies (using Claim 8.8 and keeping only the cost of the overweighted subinstance):

$$\begin{aligned}F + \text{cost}(\Omega, \mathcal{S}) &> -3\varepsilon k n_U + 3/2 \cdot n_U \cdot \left(kn_U\varepsilon^{-1}\right)^z \\ &\geq (-3\varepsilon^2/n_U + 3/2)\text{`}n_U \left(kn_U\varepsilon^{-1}\right)^z \quad > (1+\varepsilon)\cdot(1+2\varepsilon)n_U \left(k \cdot n_U\varepsilon^{-1}\right)^z \\ &> (1+\varepsilon)\text{cost}(\mathcal{S}),\end{aligned}$$

hence contradicting the fact that Ω is an ε-coreset with offset F.

The proof of the case where some subinstance is underweighted is done exactly alike. □

Combining those claims allows to prove the main theorem of the chapter.

Proof of Theorem 8.2. Assume toward contradiction that there exists an ε-coreset with offset F of size smaller than $\frac{\eta}{10} \cdot k\varepsilon^{-2} \log |C|$, where η is the constant of Lemma 8.6.

First, this implies the existence of an 2ε-coreset with ε-rounded weights, simply by rounding each weight to the closest multiple of ε.

Using Claim 8.9, we can apply Lemma 8.6 on each subinstance. The total coreset size is $\frac{\eta}{10} \cdot k\varepsilon^{-2} \log |C|$: that means that there are at least $9k/10$ subinstances for which the coreset contains less than $\eta\varepsilon^{-2} \log |C|$ many different points. We refer to these subinstances as the *bad* subinstances. Using Lemma 8.6, we construct a solution $\mathcal{S}$ by taking the center given by bullet 1 for the bad subinstances, i.e.: center $\hat{c}$ as per the notation of Lemma 8.6, and bullet 2 for the others, i.e.: center c^* as per the notation of Lemma 8.6. The cost of that solution is $n_1 + 2(kn_U - n_1) = 2kn_U - n_1$, where n_1 the number of edges of length 1 from the clients to $\mathcal{S}$. Similarly, the cost of $\mathcal{S}$ for the coreset is $2 \cdot w(\Omega) - w_1(\mathcal{S}, \Omega)$, where $w(\Omega)$ is the total coreset weight and $w_1(\mathcal{S}, \Omega)$ the weighted number of length 1 edges from Ω to $\mathcal{S}$. By construction of $\mathcal{S}$, $w_1(\mathcal{S}, \Omega)$ verifies

$$w_1(\mathcal{S}, \Omega) \geq 9k/10 \cdot (1+200\varepsilon)\pi n_U + k/10 \cdot (1-\varepsilon)\pi n_U > (1+150\varepsilon)\pi \cdot kn_U$$

Furthermore, using properties of Lemma 8.6, $n_1 \leq \pi k n_U$: the cost of $\mathcal{S}$ for the full point set$P$ is large. Hence, the cost of $\mathcal{S}$ in the coreset satisfies

$$\begin{aligned}F + 2 \cdot w(\Omega) - w_1(\mathcal{S}, \Omega) &< 3\varepsilon k n_U + 2 \cdot (1+2\varepsilon)kn_U - (1+150\varepsilon)\pi \cdot kn_U \\ &\leq (2kn_U - n_1) + \varepsilon k n_U \cdot (7 - 150\pi) < (1-\varepsilon)(2kn_U - n_1),\end{aligned}$$

where the last inequality uses $\pi = 1/4$, so that $(150\pi - 7)kn_U \geq 2kn_U$. Therefore the cost of the coreset for $\mathcal{S}$ is smaller than a $(1-\varepsilon)$ factor times the cost of $\mathcal{S}$ for the points set P, a contradiction that concludes: any coreset must have size at least $\frac{\eta}{10} \cdot k\varepsilon^{-2} \log |C|$. Using the fact that $\log n = O(\log |C|)$ concludes the proof. □

A simple corollary of that proof is a lower bound for metric with bounded doubling dimension. Since any n points metric has doubling dimension $O(\log n)$, the metric constructed has doubling dimension $D = O(\log n)$, which implies the following:

▶ **Corollary 8.10.** For any ε, k, D such that $D \geq 5 \log k/\varepsilon$ there exists a graph with doubling dimension D on which any ε-coreset with offset for (k, z)-clustering must have size $\Omega\left(\frac{kD}{\varepsilon^2}\right)$. ◀

Chapter 9

Deterministic Sketches for Clustering

In this chapter, we present deterministic sketches for clustering, and put forth a different type of sketch than coreset, namely dimension reduction. This is another important tool dedicated to Euclidean spaces: the goal is to embed the input into a lower-dimensional Euclidean space, in order to avoid any form of curse of dimensionality. The main technique for that is the Johnson Lindenstrauss lemma and variants, which show in the case of (k, z)-clustering that the cost of any partition is preserved up to an $(1 \pm \varepsilon)$ factor when projecting onto a randomly chosen $O(\varepsilon^{-2} \log k)$-dimensional subspace. More precisely, the dimension reduction results are as follows. For a set of points C, denote $\phi(C)$ the optimal center. Dimension reduction seeks to preserve the cost of any partition of the input: the goal is to find a mapping $f : \mathbb{R}^d \to \mathbb{R}^{d'}$, with the following guarantee. For any partition of the input P into clusters $C_1, ..., C_k$, it holds that

$$\sum_{i=1}^{k} \sum_{p \in C_i} \text{cost}(f(p), \phi(f(C_i))) = (1 \pm \varepsilon) \sum_{i=1}^{k} \sum_{p \in C_i} \text{cost}(p, \phi(C_i)). \tag{9.1}$$

Note that this guarantee is different than the coreset one: for coresets, we seek to preserve the cost to any set of k *fixed* centers, which is not well defined in the case of dimension reduction as the space changes. The standard way of circumventing this is to preserve the cost of partitions instead. The main result for reducing dimension of clustering problems is due to Makarychev et al. [129] (see also Becchetti et al. [25]): if f is an orthogonal projection onto a random $O\left(z^4 \varepsilon^{-2} \log \frac{k}{\varepsilon \pi}\right)$-dimensional subspace, it satisfies Eq. (9.1) with probability $1 - \pi$.

In this chapter, our goal is to present deterministic sketches, in order to answer Question 5: we present deterministic constructions of both coresets and dimension reduction, that run in polynomial time for fixed ε.

As it is the case for coreset, derandomizing the dimension reduction construction using standard techniques requires time $\Omega(k^n/k!)$, as it is necessary to enumerate over all possible partitions.

To reduce the number of possible partition, a natural attempt is to go through coreset, and replace the dependency in n by one in k. However, as mentioned earlier, usual coreset do not preserve the cost of any partition. henceforth, we start by presenting deterministic construction of a particular coreset suited for dimension reduction, which preserves the cost of any partition, and not only the cost of any solution made of k-centers. This is done in Section 9.2.

Using this tool, we design a deterministic dimension reduction, in Section 9.3. Finally, once points can be projected into a low dimensional space we are able to show improved deterministic coreset constructions, in Section 9.4. This coreset construction is interesting end-of-itself: it can be implemented faster using randomization, and provides another framework to construct coreset, that allows to use VC-dimension of functions with *uniform* scale – which allows to bypass some of the technical difficulties mentioned in Section 5.3.

9.1 Introduction and Key Challenges

9.1.1 Deterministic Dimension Reduction

The result we aim for is the following:

▶ **Informal Theorem (See Theorem 9.14 for a formal statement).** For (k, z)-clustering in Euclidean space, one can reduce the dimension to $O(\varepsilon^{-O(z)} \log k)$ in deterministic $O\left(n^{\varepsilon^{-O(z)}} \cdot d\right)$ time. ◀

There currently exist only two known methods that enable deterministic dimension reduction. The first is principal component analysis and its generalizations to other norms. Applying principal component analysis to k-means requires a target dimension of $\Omega(k)$ [48] and thus fall short of the bounds we are aiming for.

The other option to derandomize dimension reduction is the conditional expectation method for derandomizing Johnson Lindenstrauss transforms by [72]. At a high level, this method fixes one by one the bits of a sketching matrix such that the number of preserved distances is maximized.

When applying this idea naively to a candidate embedding for k-clustering, it is clear that the running time is infeasible as there exist $k^n/k!$ many distinct clusterings. Thus, our goal is to prove that there exists a set N of at most $k' \in k^{\varepsilon^{-O(z)}}$ points such that preserving the distance from any input point to the points in N preserves the cost of any clustering.

For this, our key contribution is to show that for any cluster, a $(1+\varepsilon)$ approximate center lies in the convex hull of a subset of the cluster of size $\varepsilon^{-O(1)}$, that we call a

witness set. Using a careful discretization argument, we can show that it is possible to enumerate over all candidate solutions in the convex hulls of such witness sets and thereby also enumerating over all possible centers induced by any clustering. Unfortunately, there are $n^{\varepsilon^{-O(z)}}$ many possible witness sets; hence, applying Johnson-Lindenstrauss leads to a dimension reduction onto $\varepsilon^{-O(z)} \log n$ dimensions.

To remove the $\log n$ dependency, the natural idea is reducing the number of distinct point to poly(k). A coreset is the most natural idea here, however the guarantees we typically require for coresets are not strong enough for our purposes: coresets preserve the cost to any set of k centers, while our goal here is to preserve the cost of any partition. In turn, we naturally introduce the notion of *partition coreset*, which are small sets preserving cost of any partition. We show the existence of partition coreset of size $k^{\varepsilon^{-O(z)}}$; hence, we are able to reduce the dimension to $\varepsilon^{-O(z)} \log k$.

To show the existence of a partition coreset, a first attempt would be to use a k' clustering such that the cost of the k' clustering is at most an ε-fraction of the cost of an optimal k-clustering. Unfortunately, such a set must have size at least ε^{-d}. Instead, we show – and this is our main technical contribution for this part – that there exists a k' clustering such that either the cost of this clustering is very cheap, or its cost cannot decrease by adding additional centers. In the first case, the set of centers directly give a partition coreset. The second case implies essentially that the points in some cluster of the k' clustering pay the same cost in *any* possible k-clustering. Hence, we can simply replace those points by their center, and encode the cost difference in the extension coordinate.

9.1.2 Coresets

It turns out that we can also use our construction of partition coreset to obtain small coresets, to get the following informal result (in particular, we will define the "VC-dimension of balls" in Section 9.4; as a first proxy, one can think of it as an intrinsic dimension of the space, such as the Euclidean dimension, or the treewidth, or the size of an excluded minor, ...)

▶ **Informal Theorem (See Theorem 9.19).** Let (X, dist) be a metric space for which the VC-dimension of sets of k balls is D. There exist a deterministic algorithm running in deterministic time $\tilde{O}(|X| \cdot |P| \cdot k/\varepsilon) + k\,(D/\varepsilon)^{O(D)} \cdot |P|$ that constructs an ε-coreset with offset for the set P, with size $\tilde{O}_z\left(kD\varepsilon^{-5}\right)$. In the Euclidean Space $\mathbb{R}^d$, the running time is $|P|^{\varepsilon^{-O(z)}} + k\,(D/\varepsilon)^{O(D)} \cdot |P|$. ◀

The VC-dimension of set of k balls has been well studied, and there are bounds in many different metric spaces. In all of those, we can apply our theorem: in Euclidean Spaces, metric induced by a graph with bounded treewidth, excluding a minor [30], metric on curves [65]... Hence, we show deterministic coreset for those class of graphs as well. We note here that allowing randomization in our algorithm would yield polynomial time construction (more precisely, running time poly($|X|, k, 1/\varepsilon$), except

for curves.

In particular, our result simplify a lot the coreset constructions going through VC-dimension: the usual construction have to deal with balls in a *weighted* metric (more precisely, the scaled set of functions we described in Section 5.3), and as noted by Baker et al. [15], this "introduces a significant technical challenge" compared to the set of balls we work with.

Essentially, our construction is as follows. Building on the work of Chen [44], we show that constructing a coreset reduces to sampling uniformly. For this, we compute a bi-criteria approximation and partition points in exponential rings according to their cost in the bi-criteria. There are $O(k \log n)$ rings, and Chen showed that uniform sampling among them yield a coreset. Our key new ingredient here is to show how to go from $O(k \log n)$ to only $O(k)$ rings. To achieve this, we construct carefully a suited bicriteria approximation, in the same spirit as what we did for the partition coreset: we ensure that its cost cannot decrease by adding k more centers. In particular, this implies that the cost of the "far" points cannot be much different in any solution. Hence, we can replace them by copies of the center, and encode their cost in an offset F that is added to the cost of any solution in the coreset.

Next, we show that instead of sampling points, it is enough to build an ε-set-approximation[1] of a particular set system, formed by k balls. Those ε-set-approximation are heavily related to the VC-dimension: if the VC-dimension of a set system is D, then using a standard algorithm, one can compute ε-set-approximation of size $\tilde{O}(\frac{D}{\varepsilon^2})$. As we will show that such an ε-set-approximation is a coreset for a ring, this concludes our theorem.

For the particular case of Euclidean Spaces of dimension d, it is well known that the VC-dimension of that set system is bounded by $O(kd \log k)$. Hence, the size of the ε-set-approximation is merely $\tilde{O}(\frac{kd \log k}{\varepsilon^2})$.

Unfortunately, the running time of these constructions is exponential in the VC dimension, which is $k \cdot d \log k$ in the Euclidean case. To eliminate the dependency on d, we first compute a coreset of size $k' \in k^{\varepsilon^{-O(z)}}$ – using our previous results – followed by embedding it into low dimensional space via terminal embeddings. Terminal embeddings are commonly used as a coreset preserving dimension reduction, see Section 7.6 and [25, 34, 100] and the construction by Mahabadi, Makarychev, Makarychev and Razenshteyn [127] can be made deterministic. Thus, we are able to obtain coresets of size $O(k^2 \cdot \varepsilon^{O(1)})$ in time $n^{O(\varepsilon^{-2} \log 1/\varepsilon)} + \exp(\text{poly}(k, \varepsilon^{-1}))$.

9.1.3 Definitions

In this chapter, we will use matrix notations, as it is more convenient for dimension reduction. For a d-dimensional vector x, we define the ℓ_z norm $\|x\|_z = \sqrt[z]{\sum_{i=1}^{d} |x_i|^z}$. For

[1] In the VC-dimension literature, this is usually called an ε'-approximation, but we already use this terminology for the cost of solutions.

an $n \times d$ matrix A, we define $\|A\|_{z,2} = \left(\sum_{i=1}^{n} \left(\sum_{j=1}^{d} A_{i,j}^2\right)^{z/2}\right)^{1/z} = (\sum_{i=1}^{n} \|A_i\|_2^z)^{1/z}$. The special case $z = 2$ corresponds to the Frobenius norm, i.e. $\|A\|_F = \|A\|_{2,2}$.

In matrix notations, given a set of points from $\mathbb{R}^d$ represented by a $n \times d$ matrix A where each line is a point, the (k, z)-clustering objective consists of finding an $n \times d$ matrix C with at most k distinct rows minimizing $\|A - C\|_{z,2}^z$. With those notations, the i-th row of C contains the coordinates of the center serving the i-th point.

We say that an $n \times (d + 1)$ matrix A' is an extension matrix of an $n \times d$ matrix A if the first d columns of A' are equal to A. If the final column of A' is the all-zero vector, we say that A' is the 0-extension of A.

We first define the dimension reduction guarantee we aim to satisfy, similar to that of Sohler and Woodruff [151].

▶ **Definition 9.1 (Cost Preserving Sketches for Powers).** Let A be an $n \times d$ matrix corresponding to a set of n points in $\mathbb{R}^d$, and let B be an $n \times (m+1)$ matrix corresponding to a set of n points in $\mathbb{R}^m$ with an additional extension coordinate, let z be a positive integer and $\varepsilon > 0$ a precision parameter. Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a partition of the integers $\{1, \ldots, n\}$ into k sets, i.e. a k-clustering of the rows of A and B. We define the matrices $C^{A,\mathcal{C}}$ and $C^{B,\mathcal{C}}$ as follows. For a row index j such that the $j \in C_i$, the j-th row of $C^{A,\mathcal{C}}$ is equal to $c_i^A := \operatorname{argmin}_{c \in \mathbb{R}^d} \sum_{j \in C_i} \|A_j - c\|^z$. Similarly, the j-th row of $C^{B,\mathcal{C}}$ are equal to $c_i^B := \operatorname{argmin}_{\substack{c \in \mathbb{R}^{m+1} \\ c_{m+1}=0}} \sum_{j \in C_i} \|B_j - c\|^z$.
Then we say that B is an (k, z, ε)-cost preserving sketch if for every partition $\mathcal{C}$ into k parts,

$$\left| \|A - C^{A,\mathcal{C}}\|_{z,2}^z - \|B - C^{B,\mathcal{C}}\|_{z,2}^z \right| \leq \varepsilon \|A - C^{A,\mathcal{C}}\|_{z,2}^z$$

◀

This definition captures the guarantee satisfied for (k, z)-clustering given by Makarychev et al. [129], with the generalization of allowing B to have an extension. Let us briefly compare this definition to cost preserving sketches for k-means. For this, we first define clustering matrices: a clustering matrix corresponding to a clustering $C_1, ..., C_k$ is an n by k matrix X, with

$$X_{i,j} = \begin{cases} \frac{1}{\sqrt{|C_j|}} & \text{if } A_i \in C_j \\ 0 & \text{otherwise} \end{cases}.$$

A cost preserving sketch for k-means is then defined as follows:

▶ **Definition 9.2 (Cost Preserving Sketches for k-means [48]).** Let A be an n by d matrix corresponding to a set of n points in $\mathbb{R}^d$, and let B be an n by m matrix corresponding to a set of n points in $\mathbb{R}^m$. We say that B

is an $(k, 2, \varepsilon)$-cost preserving sketch with offset Δ if for every n by k clustering matrix X

$$\left|\|A - XX^TA\|_F^2 - \left(\|B - XX^TB\|_F^2 + \Delta\right)\right| \leq \varepsilon \cdot \|A - XX^TA\|_F^2. \quad \blacktriangleleft$$

It turns out that those two definitions are equivalent, for $z = 2$: we can simplify Definition 9.1 by requiring only the cost $\|A - XX^TA\|_F^2$ to be approximated, where X is n by k clustering matrix corresponding to the partitioning $C_1, ..., C_k$.

Moreover, the offset Δ used for k-means is a special case of adding an extension to B and requiring the extension coordinate of the centers being set to 0 (i.e. $c_{m+1} = 0$). Under this constraint we have for $z = 2$

$$\sum_{i=1}^{k}\sum_{j\in C_i}\|B_j - c_i^B\|^2 = \sum_{i=1}^{k}\sum_{j\in C_i}\sum_{\ell=1}^{m}\|(B_j)_\ell - (c_i^B)_\ell\|^2 + \|(B_j)_{m+1}\|^2.$$

Note that the contribution of the final coordinate is independent of the choice of centers, under the constraint $c_{m+1} = 0$. Hence, the offset Δ is equivalent to $\sum_{j=1}^{n}\|(B_j)_{m+1}\|^2$. Second, the choice of centers c_i^A and c_i^B for k-means is equivalent to the mapping obtained by using clustering matrices.

In order to construct our cost preserving sketches, we will rely on partition coresets, defined as follows:

▶ **Definition 9.3 (Partition Coresets).** An $n \times d$ matrix B is an (ε, k, z)-partition coreset of A if for any matrix C with at most k distinct rows

$$\left|\|A - C\|_{z,2}^z - \|B - C\|_{z,2}^z\right| \leq \varepsilon \cdot \|A - C\|_{z,2}^z.$$

B is an (ε, k, z)-extension partition coreset of A if there exists an extension B' of B such that for any matrix C with at most k distinct rows and 0-extension C' we have

$$\left|\|A - C\|_{z,2}^z - \|B' - C'\|_{z,2}^z\right| \leq \varepsilon \cdot \|A - C\|_{z,2}^z.$$

In both cases, we say that the size of the coreset is the number of distinct rows of B. ◀

This definition is stronger from the commonly found definition that requires the cost of an assignment $\sum_{p\in A}\min_{c\in C}\|p-c\|^z$ to be approximated for any set of k-centers C. For partition coreset, the cost of any assignment to the centers is preserved. We crucially need this fact for dimension reduction, as guarantees for cost preserving sketches are related to partitions as well.

Finally, we recall the more standard coreset, that we will work with in section 9.4. The definition of coreset we consider here allows for an *offset*, originally due to Feldman, Schmidt and Sohler [78]. For other works using this definition, see [25, 48, 151]. This is the one for which we showed lower bounds in Chapter 8.

▶ **Definition 9.4.** An (ε, k, z)-coreset with offset in a metric space (X, dist) for clients P is a weighted subset Ω of P with weights $w : \Omega \to \mathbb{R}_+$ together with a constant F such that, for any set $\mathcal{S} \subset X$, $|\mathcal{S}| = k$,

$$|\sum_{p\in P} \min_{s\in\mathcal{S}} \|p-s\|^z - \big(F + \sum_{p\in\Omega} w(p) \min_{s\in\mathcal{S}} \|p-s\|^z\big)| \leq \varepsilon \sum_{p\in P} \min_{s\in\mathcal{S}} \|p-s\|^z$$

Using notations introduced in Section 1.6, this is

$$|\text{cost}(P,\mathcal{S}) - \big(F + \text{cost}(\Omega,\mathcal{S})\big)| \leq \varepsilon \text{cost}(P,\mathcal{S}). \quad ◀$$

As opposed to the previous sketches, those coresets preserve the cost of any set of *centers*, instead of preserving the cost of any clustering. In particular, those centers may not be the optimal one for the induced partition.

9.1.4 Useful Results

A (α, β) bicriteria algorithm for clustering produces a clustering with $\beta \cdot k$ centers and cost $\alpha \cdot OPT$, where OPT denotes the cost of an optimal clustering with k centers. As we need deterministic results, we cannot rely on the constant-factor algorithm from Lemma 1.3. We use instead the following (slightly slower) procedure, from Gupta and Tangwongsan [92][2]

▶ **Theorem 9.5 (Theorem 3.2 of [92]).** Local-search gives a $(10z)^z$-approximation for (k, z)-clustering in general metric spaces. This algorithm can be implemented in time $n^{O(1)}d$. ◀

We also need an approximation ensuring a tiny error, as stated in the following lemma. An (α, β)-bicriteria approximation is a set of βk centers that have cost at most α times the optimal cost.

▶ **Lemma 9.6 (Proved in Section 9.6.).** There exist an algorithm running in time $n^{\varepsilon^{-O(z)}}$ that gives a $(1+\varepsilon, z^{O(z)} \log(1/\varepsilon)/\varepsilon)$-bicriteria approximation for (k, z)-clustering. ◀

Derandomizing Random Projections. The seminal Johnson-Lindenstrauss lemma states that any n point set in d-dimensional Euclidean space can be (linearly) embedded into a $m \in O(\varepsilon^{-2} \log n)$ dimensional space such that all pairwise distances are preserved up to a $(1 \pm \varepsilon)$ factor. Most proofs show this guarantee by sampling a

[2] Gupta and Tangwongsan showed a $5z$-approximation for the objective $(\sum \text{dist}(p,S)^z)^{1/z}$ where there are at most n possible center locations. This leads to a $(10z)^z$-approximation in our case, enforcing centers to be located at input point, as moving a center to its closest input point at most doubles the distance from a point to its closest center.

matrix from an appropriate distribution. We will use the following derandomization based on the conditional expectation method.

▶ **Theorem 9.7 ([72]).** Let $v_1, \dots v_n$ be a sequence of vectors in $\mathbb{R}^d$ and let $\varepsilon, F \in (0, 1]$. Then we can compute in deterministic time $O(nd(\log n\varepsilon^{-1})^{O(1)})$ a linear mapping $S : \mathbb{R}^d \to \mathbb{R}^m$ where $m \in O(\varepsilon^{-2} \log 1/F)$ such that

$$(1 - \varepsilon) \cdot \|v_i\|_2 \leq \|v_i S\|_2 \leq (1 + \varepsilon)\|v_i\|_2$$

for at least a $(1 - F)$ fraction of the i's. ◀

Note that setting $F < 1/n$ implies that the guarantee holds for all vectors. Also, we note that if we are given n points rather than n vectors and wish to preserve the pairwise distances, the running time has a quadratic dependency on n as there are $\binom{n}{2}$ many distance vectors. Since the running time is dominated by computing a bicriteria approximation, we will omit it from the statement of our theorems.

9.2 Deterministic Coresets and Partition Coresets for (k, z)-Clustering

9.2.1 Construction of the Partition Coreset

The main goal of this section is to prove the following theorem:

▶ **Theorem 9.8.** Let A be a set of points in $\mathbb{R}^d$. We can compute in deterministic time $O(n^{\varepsilon^{-O(z)}})$ an (ε, k, z)-extension partition coreset of A with at most $k^{\varepsilon^{-O(z)}}$ distinct rows. This coreset is also an (ε, k, z)-coreset with offset of A. ◀

The key structural lemma we use in this case is as follows. Essentially, it states that if the cost of clustering to a single center cannot be decreased by adding k centers, then the center forms a partition coreset.

▶ **Lemma 9.9.** Let A be a set of points in $\mathbb{R}^d$ and let m be a (not necessarily optimal) center for A. Define the $n \times d$ matrix M to contain m in every row and let M' be the extension of M with $M_{i,d+1} = \|A_i - m\|$. Suppose that for all set of k centers $\mathcal{S}$,

$$\text{cost}(A, \{m\}) - \text{cost}(A, \mathcal{S}) < \alpha \cdot \text{cost}(A, \{m\}),$$

for $\alpha \leq \frac{\varepsilon^{z+6}}{401408 \cdot 2^{3z} \cdot z^{z+6}}$.

Then for any matrix C that contains exactly k distinct rows and its 0-extension C', we have

$$\left| \|A - C\|_{z,2}^z - \|M' - C'\|_{z,2}^z \right| \leq \varepsilon \cdot \|A - C\|_{z,2}^z. \quad \blacktriangleleft$$

In other terms, the guaranty we have on matrix C is that

$$\left| \sum_{p \in A} \|p - C_p\|^z - \sum_{p \in A} (\|p - C_p\|^2 + \|m - C_p\|^2)^{z/2} \right| \leq \varepsilon \sum_{p \in A} \|p - C_p\|^z.$$

Before proving this lemma, we will show why it implies the theorem.

Proof of Theorem 9.8. We will use the following algorithm. In the following, let β, and γ be constants that will be specified later. One can think of $\beta, \gamma = \varepsilon^{-O(z)}$.

- Compute a $2^{O(z)}$-approximation for (k, z)-clustering with clusters $C_1, \dots C_k$ and respective centers $c_1, ..., c_k$.
- Recursively compute a $(1 + \beta, f(\beta))$-approximation $\mathcal{S}$ for every cluster C_i, using Lemma 9.6. With that algorithm, $f(\beta) = z^{O(z)} \log(1/\beta)/\beta$.
- If $\text{cost}(C_i, \{c_i\}) - \text{cost}(C_i, \mathcal{S}) \leq \beta \cdot \text{cost}(C_i, \{c_i\})$, break and use the $|C_i| \times d$ matrix $M_j = c_i$ with extension $(\text{cost}((C_i)_j, \{c_i\}))$ as a partition coreset for C_i.
- If C_i is at depth γ, break and use the $|C_i| \times (d+1)$ matrix $M_j = c_i$ with extension 0 as a partition coreset for C_i.

We first argue the correctness of this algorithm, then that it can be implemented in the desired (deterministic) time.

For correctness, we start by considering the first termination criterion, when $\text{cost}(C_i, \{c_i\}) - \text{cost}(C_i, \mathcal{S}) \leq \beta \cdot \text{cost}(C_i, \{c_i\})$. We use for this Lemma 9.9, setting $\beta = \frac{1}{2} \cdot \frac{\varepsilon^{z+6}}{401408 \cdot 2^{3z} \cdot z^{z+6}}$ (i.e.: equal to $\alpha/2$ with α being specified in Lemma 9.9). By choice of $\mathcal{S}$, it holds that for any set of k centers K, $\text{cost}(C_i, \mathcal{S}) \leq (1 + \beta)\text{cost}(C_i, K)$, and therefore:

$$\begin{aligned}
\text{cost}(C_i, \{c_i\}) - \text{cost}(C_i, K) &\leq \text{cost}(C_i, \{c_i\}) - \frac{1}{1+\beta} \cdot \text{cost}(C_i, \mathcal{S}) \\
&= \frac{1}{1+\beta} (\text{cost}(C_i, \{c_i\}) - \text{cost}(C_i, \mathcal{S})) \\
&\quad + \frac{\beta}{1+\beta} \cdot \text{cost}(C_i, \{c_i\}) \\
&\leq \frac{\beta}{1+\beta} \cdot \text{cost}(C_i, \{c_i\}) \\
&\quad + \frac{\beta}{1+\beta} \cdot \text{cost}(C_i, \{c_i\}) \\
\implies \text{cost}(C_i, \{c_i\}) - \text{cost}(C_i, K) &\leq 2\beta \cdot \text{cost}(C_i, \{c_i\}).
\end{aligned}$$

Using Lemma 9.9, this implies that M with its extension is indeed an (ε, k, z)-extension partition coreset for the points in C_i, and that c_i with weight $|C_i|$ is an (ε, k, z)-coreset with offset for C_i (where the offset is 0).

Finally, we consider the other termination criteria, namely when we reach depth γ. Let Γ be the union of all clusters at this level. In every step of the recursion, the cost decreased by at least a factor $1-\beta$ – as otherwise the recursion stops. Hence the cost of union of clusters at depth γ is at most $(1-\beta)^\gamma$ times the initial cost, which is $2^{O(z)} \cdot (1-\beta)^\gamma \operatorname{OPT}$, where OPT is the optimal cost for (k,z)-clustering. We choose γ in such a way that $2^{O(z)} \cdot (1-\beta)^\gamma \leq \left(\frac{\varepsilon}{8z}\right)^z$, which holds if $\gamma = \Omega\left(\frac{z \log\left(\frac{\varepsilon}{8z^2}\right)}{\log(1/(1-\beta))}\right)$. With $\beta = O_z\left(\varepsilon^{z+6}\right)$, we can therefore chose a $\gamma \in \tilde{O}_z\left(\varepsilon^{-z-6}\right)$ as desired. For each point $p \in \Gamma$, let c_p be the representing center in the partition coreset. Now we consider a set of at most k centers K, and denote K_p the center to which $p \in \Gamma$ is assigned. We have

$$
\begin{aligned}
& \left| \sum_{p \in \Gamma} \|p - K_p\|^z - \|c_p - K_p\|^z \right| \\
(\textit{Lem. 9.24}) \quad & \leq \sum_{p \in \Gamma} \varepsilon/2 \cdot \|p - K_p\|^z + \left(\frac{4z+\varepsilon}{\varepsilon}\right)^{z-1} \cdot \|p - c_p\|^z \\
(\text{Cost of } \Gamma) \quad & \leq \varepsilon/2 \cdot \|A - K\|_{z,2}^z + \left(\frac{4z+\varepsilon}{\varepsilon}\right)^{z-1} \cdot \left(\frac{\varepsilon}{8z}\right)^z \cdot \|A - K\|_{z,2}^z \\
& \leq \varepsilon/2 \|A - K\|_{z,2}^z + \varepsilon/2 \cdot \|A - K\|_{z,2}^z \leq \varepsilon \cdot \|A - K\|_{z,2}^z,
\end{aligned}
$$

which implies that the entire construction is an ε-extension partition coreset. The size of that coreset is the number of clusters computed by the algorithm, which is at most $k^{\varepsilon^{-O(z)}}$ by an easy induction: at each level, a cluster is divided into $k \cdot f(\beta) = k \cdot \varepsilon^{-z-O(1)}$ subclusters, and there are $\gamma \in \varepsilon^{-O(z)}$ many levels.

Note that this also implies that the multiset of all $\{c_p, p \in \Gamma\}$ is an (ε, k, z)-coreset with offset for Γ (again, with offset 0). The number of distinct points in that set is also the number of clusters computed by the algorithm, namely $k^{\varepsilon^{-O(z)}}$.

We now consider the running time. The initial $2^{O(z)}$-approximation can be derived using the approximation algorithm of Theorem 9.5. For the bicriteria approximation, we can use the algorithm of Lemma 9.6, that gives $f(\beta) = z^{O(z)} \log(1/\beta)/\beta$ in time $n^{\varepsilon^{-O(z)}}$. This bicriteria algorithm is used once per cluster of each level, hence at most $(kf(\beta))^\gamma = (kz/\varepsilon^z)^{O(\gamma)} = (kz/\varepsilon^z)^{\varepsilon^{-O(z)}}$ many times before terminating, and therefore a total complexity of $n^{\varepsilon^{-O(z)}}$. □

The remaining part of this section will now be devoted to proving Lemma 9.9.

9.2.2 Proof of the Structural Lemma 9.9

We first require a bit of notation. Let $\{c_1, \dots c_k\}$ be the distinct rows of C. Let $A^{c_i} = \{A_j \in A \mid C_j = c_i\}$, i.e. A^{c_i} are the subset of rows of A that are mapped to the same row in C with coordinates c_i. Equivalently, A^{c_i} is the set of points that are assigned to the center c_i in the assignment induced by C. For every A^{c_i}, we denote by Q^{c_i} the (multiset of) projections of the points in A^{c_i} onto the line through m and c_i; i.e. for $p \in A^{c_i}$, we have $q_p \in Q^{c_i}$ and by the Pythagorean theorem

$$\|p - c_i\| = \sqrt{\|p - q_p\|^2 + \|q_p - c_i\|^2}.$$

Our goal is to show that, when clustering to the single point m is near-optimal (as described in the statement of Lemma 9.9), then for "most" of the points $p \in A$ one of the following conditions hold.

Cond. 1 $\|p - m\| \geq \left(\frac{4z}{\varepsilon}\right) \cdot \|m - c_i\|$ or $\|p - m\| \leq \left(\frac{\varepsilon}{4z}\right) \cdot \|m - c_i\|$ (i.e. the conditions of Lemma 9.10 below are met) or

Cond. 2 $\|q_p - m\| \leq \varepsilon/(7z) \cdot \|p - m\|$ (i.e. the conditions of Lemma 9.11 below are met).

The following two lemmas show that if we can assume that one of these two cases hold, then the row of the extension matrix M' corresponding to the point p is a good proxy for $\|p - c_i\|^z$.

▶ **Lemma 9.10.** If $\|p - m\| \geq \left(\frac{4z}{\varepsilon}\right) \cdot \|m - c_i\|$ or $\|p - m\| \leq \left(\frac{\varepsilon}{4z}\right) \cdot \|m - c_i\|$ (i.e., when condition 1 is met) then

$$\left| \left(\|p - m\|^2 + \|m - c_i\|^2\right)^{z/2} - \|p - c_i\|^z \right| \leq \varepsilon \cdot \left(\|p - m\|^z + \|p - c_i\|^z\right).$$

◀

Proof. We first bound $\|p - c_i\|^z$ in terms of $\max(\|p - m\|^z, \|m - c_i\|^z)$. Using Condition 1, we have $\min(\|p - m\|, \|m - c_i\|) \leq \varepsilon/(4z) \cdot \max(\|p - m\|, \|m - c_i\|)$, and therefore

$$\begin{aligned} &\max(\|p - m\|, \|m - c_i\|) - \min(\|p - m\|, \|m - c_i\|) \\ &\qquad \geq (1 - \varepsilon/(4z)) \cdot \max(\|p - m\|, \|m - c_i\|), \\ &\max(\|p - m\|, \|m - c_i\|) + \min(\|p - m\|, \|m - c_i\|) \\ &\qquad \leq (1 + \varepsilon/(4z)) \cdot \max(\|p - m\|, \|m - c_i\|) \end{aligned}$$

Using standard convex inequalities (more precisely, the Mercator series for the upper

bound and Bernoulli's inequality for the lower bound), this now implies

$$\begin{aligned}\|p-c_i\|^z &\leq (1+\varepsilon/(4z))^z \cdot \max(\|p-m\|^z, \|m-c_i\|^z)\\ &= e^{\ln(1+\varepsilon/(4z))z} \cdot \max(\|p-m\|^z, \|m-c_i\|^z)\\ &\leq e^{\varepsilon/4} \cdot \max(\|p-m\|^z, \|m-c_i\|^z)\\ &\leq e^{\ln(1+\varepsilon/2)} \cdot \max(\|p-m\|^z, \|m-c_i\|^z)\\ &= (1+\varepsilon/2) \cdot \max(\|p-m\|^z, \|m-c_i\|^z)\end{aligned}$$

and

$$\begin{aligned}\|p-c_i\|^z &\geq (1-\varepsilon/(4z))^z \cdot \max(\|p-m\|^z, \|m-c_i\|^z)\\ &\geq (1-\varepsilon/2) \cdot \max(\|p-m\|^z, \|m-c_i\|^z).\end{aligned}$$

Therefore, with the same application of convex inequalities, we obtain

$$\begin{aligned}&\left|\left(\|p-m\|^2+\|m-c_i\|^2\right)^{z/2} - \|p-c_i\|^z\right|\\ =\ &\Big|\left((1\pm\varepsilon/(4z)) \cdot \max(\|p-m\|^2, \|m-c_i\|^2)\right)^{z/2}\\ &-(1\pm\varepsilon/2)\cdot(\max(\|p-m\|, \|m-c_i\|))^z\Big|\\ =\ &\left|(1\pm\varepsilon/2)\cdot(\max(\|p-m\|, \|m-c_i\|))^z - (1\pm\varepsilon/2)\cdot\max(\|p-m\|, \|m-c_i\|))^z\right|\\ =\ &\pm\varepsilon\cdot\max(\|p-m\|^z, \|m-c_i\|^z)\\ \leq\ &\varepsilon\cdot(\|p-m\|^z+\|p-c_i\|^z)\end{aligned}$$

□

▶ **Lemma 9.11.** Let $p, m, c_i \in \mathbb{R}^d$. Denote by q_p the orthogonal projection of p onto the line through m and c_i. Then if for some $0 \leq \varepsilon \leq 1/2$,

$$\|m-q_p\| \leq \varepsilon/(7z)\cdot\|p-m\|$$

we have for any $z \geq 1$:

$$\left|\left(\|p-m\|^2+\|m-c_i\|^2\right)^{z/2} - \|p-c_i\|^z\right| \leq \varepsilon\cdot\|p-c_i\|^z. \quad ◀$$

Proof. We first require a lower bound on $\|p-c_i\|$ in terms of $\|p-m\|$. We have, using the assumption on $\|m-q_p\|$:

$$\begin{aligned}\|p-c_i\|^2 &\geq \|p-q_p\|^2 \geq \|p-m\|^2 - \|q_p-m\|^2\\ &\geq \|p-m\|^2\cdot(1-\varepsilon^2/(49z^2))\end{aligned} \tag{9.2}$$

Using Lemma 1.2, the Pythagorean theorem and convex inequality (either the Mercator series or Bernoulli's inequality) in the last line, we have

$$
\begin{aligned}
& \left| \left(\|p - m\|^2 + \|m - c_i\|^2 \right)^{z/2} - \|p - c_i\|^z \right| \\
= & \left| \left(\|p - q_p\|^2 + \|q_p - m\|^2 + \|q_p - c_i\|^2 - \|q_p - c_i\|^2 + \|m - c_i\|^2 \right)^{z/2} - \|p - c_i\|^z \right| \\
= & \left| \left(\|p - c_i\|^2 + \|q_p - m\|^2 \pm \left(\frac{\varepsilon}{4z} \cdot \|q_p - c_i\|^2 + \left(\frac{\varepsilon + 4z}{\varepsilon} \right) \|q_p - m\|^2 \right) \right)^{z/2} - \|p - c_i\|^z \right| \\
= & \left| \left(\|p - c_i\|^2 \pm \left(\frac{\varepsilon}{4z} \cdot \|q_p - c_i\|^2 + \left(\frac{2\varepsilon + 4z}{\varepsilon} \right) \|q_p - m\|^2 \right) \right)^{z/2} - \|p - c_i\|^z \right| \\
= & \left| \left(\|p - c_i\|^2 (1 \pm \varepsilon/(4z)) \pm \left(\frac{2\varepsilon + 4z}{\varepsilon} \right) \cdot \frac{\varepsilon^2}{49z^2} \cdot \|p - m\|^2 \right)^{z/2} - \|p - c_i\|^z \right| \\
= & \left| \left(\|p - c_i\|^2 (1 \pm \varepsilon/(4z)) \pm \left(\frac{2\varepsilon + 4z}{\varepsilon} \right) \cdot \frac{\varepsilon^2}{49z^2} \cdot \frac{1}{1 - \frac{\varepsilon^2}{49z^2}} \cdot \|p - c_i\|^2 \right)^{z/2} - \|p - c_i\|^z \right| \\
= & \left| \left(\|p - c_i\|^2 (1 \pm \varepsilon/(2z)) \right)^{z/2} - \|p - c_i\|^z \right| \\
= & \left| \|p - c_i\|^z (1 \pm \varepsilon) - \|p - c_i\|^z \right| \leq \varepsilon \cdot \|p - c_i\|^z,
\end{aligned}
$$

where the third to last inequality uses Section 9.2.2, and the second to last uses $\varepsilon < \frac{1}{2}$. □

We now bound the cost of the remaining points. Let D be the set of points that satisfy neither conditions, i.e. for every point $p \in D$ we have

$$\left(\frac{\varepsilon}{4z} \right) \cdot \|m - c_i\| \leq \|p - m\| \leq \left(\frac{4z}{\varepsilon} \right) \cdot \|m - c_i\| \tag{9.3}$$

and

$$\|q_p - m\| \geq \varepsilon/(7z) \cdot \|p - m\|. \tag{9.4}$$

The two conditions allow us to relate the cost of clustering the points to m and the cost of clustering the points to the candidate solution C. We will show that the contribution of the points in D to the clustering cost of m is small.

▶ **Lemma 9.12.** Let m be a point such that there exists no solution $\mathcal{S}$ with k centers and $\text{cost}(A, \{m\}) - \text{cost}(A, \mathcal{S}) \leq \alpha \cdot \text{cost}(A, \{m\})$.
Suppose C is a set of k centers and let D be the set of points that satisfy neither condition (1) nor condition (2). It holds that

$$\sum_{p \in D} \|p - m\|^z < \alpha \cdot \frac{25088 z^6}{\varepsilon^6} \cdot \sum_{p \in A} \|p - m\|^z. \quad \blacktriangleleft$$

Proof. The high level idea of the proof is to show the existence of a set of k centers T that decreases the cost of D significantly. This implies that the cost of D must be small, as otherwise the overall cost of the solution could decrease significantly.

Consider a center $c_i \in C$ and let $\ell_i = m + (m - c_i) \cdot \frac{\varepsilon^2}{28z^2}$ and $r_i = m + (m - c_i) \cdot \frac{\varepsilon^2}{28z^2}$. Let B be union of all ℓ_i and r_i. We assign each point $p \in D \cap A^{c_i}$ to its closest center $b_i \in \{\ell_i, r_i\}$. Note that due to Equations 9.3 and 9.4 $\|q_p - m\| \geq \frac{\varepsilon}{7z} \cdot \|p - m\| \geq \frac{\varepsilon^2}{28z^z} \cdot \|m - c_i\| = \|b_i - m\|$. Using the Pythagorean theorem, we then have

$$\begin{aligned}
\|p - b_i\|^z &= \left(\|p - q_p\|^2 + \|q_p - b_i\|^2\right)^{z/2} \\
&\leq \left(\|p - q_p\|^2 + \|q_p - m\|^2 - \|b_i - m\|^2\right)^{z/2} \\
&= \left(\|p - m\|^2 - \left(\frac{\varepsilon^2}{28z^2}\right)^2 \cdot \|m - c_i\|^2\right)^{z/2} \\
(Eq.\ 9.3) \quad &\leq \left(\|p - m\|^2 - \frac{\varepsilon^6}{12544z^6} \cdot \|p - m\|^2\right)^{z/2} \\
&= \|p - m\|^z \cdot (1 - \frac{\varepsilon^6}{12544z^6})
\end{aligned}$$

Summing this over all points therefore leads to a cost decrease of at least

$$\sum_{p \in D} \|p - m\|^z - \sum_{p \in D} \|p - b_i\|^z \geq \frac{\varepsilon^6}{12544z^6} \cdot \sum_{p \in D} \|p - m\|^z.$$

Since B contains $2k$ centers, the average cost decrease per center is $\frac{1}{2k} \cdot \frac{\varepsilon^6}{800z^6} \cdot \sum_{p \in D} \|p - m\|^z$. We greedily pick the k centers with maximum cost decrease. Denote this set T and denote the set of points with cost decrease by D_T. By Markov's inequality, we therefore have

$$\sum_{p \in D_T} \|p - m\|^z - \sum_{p \in D_T} \|p - b_i\|^z \geq \frac{\varepsilon^6}{25088z^6} \cdot \sum_{p \in D} \|p - m\|^z.$$

Since we assumed that no clustering exists decreasing the cost by more than $\alpha \cdot \sum_{p \in A} \|p - m\|^z$, this implies

$$\begin{aligned}
&\frac{\varepsilon^6}{25088z^6} \cdot \sum_{p \in D} \|p - m\|^z \leq \alpha \cdot \sum_{p \in A} \|p - m\|^z \\
\Rightarrow &\sum_{p \in D} \|p - m\|^z \leq \alpha \cdot \frac{25088z^6}{\varepsilon^6} \cdot \sum_{p \in A} \|p - m\|^z.
\end{aligned}$$

□

▶ **Remark 9.13.** The dependency on ε can be improved to at least ε^{-4}, using a more involved analysis to determine the cost of a cheaper clustering T. Since the subsequent analysis incurs a dependency ε^{-z}, we chose the simpler proof in favour of optimizing lower order terms in the exponent of ε. ◀

We can now conclude the proof of Lemma 9.9. We have using the Pythagorean theorem

$$\begin{aligned}
& \left| \|A-C\|_{z,2}^z - \|M'-C'\|_{z,2}^z \right| \\
= & \left| \sum_{i=1}^{k} \sum_{p \in A^{c_i}} \|p-c_i\|^z - \left(\|m-p\|^2 + \|m-c_i\|^2 \right)^{z/2} \right| \\
\leq & \left| \sum_{i=1}^{k} \sum_{p \in A^{c_i} \setminus D} \|p-c_i\|^z - \left(\|m-p\|^2 + \|m-c_i\|^2 \right)^{z/2} \right| \qquad (9.5) \\
& + \left| \sum_{i=1}^{k} \sum_{p \in A^{c_i} \cap D} \|p-c_i\|^z - \left(\|m-p\|^2 + \|m-c_i\|^2 \right)^{z/2} \right|. \qquad (9.6)
\end{aligned}$$

The term in Equation 9.5 can be bounded by $\sum_{i=1}^{k} \sum_{p \in A^{c_i} \setminus D} \varepsilon \cdot (\|p-c_i\|^z + \|p-m\|^z) \leq \varepsilon \cdot \left(\|A-C\|_{z,2}^z + \|A-M\|_{z,2}^z \right)$ using Lemmas 9.10 and 9.11. For the term in Equation 9.6, we have

$$\begin{aligned}
& \left| \sum_{i=1}^{k} \sum_{p \in A^{c_i} \cap D} \|p-c_i\|^z - \left(\|m-p\|^2 + \|m-c_i\|^2 \right)^{z/2} \right| \\
\leq & \; 2^{z+1} \sum_{i=1}^{k} \sum_{p \in A^{c_i} \cap D} \max(\|p-c_i\|^z + \|m-c_i\|^z) \\
(Eq.\ 9.3) \quad \leq & \; 2^{z+1} \sum_{p \in D} \left(1 + \left(\frac{4z}{\varepsilon} \right)^z \right) \cdot \|p-m\|^z \\
(Lem.\ 9.12) \quad \leq & \; 2^{z+2} \cdot \left(\frac{4z}{\varepsilon} \right)^z \cdot \alpha \cdot \frac{25088 z^6}{\varepsilon^6} \cdot \sum_{p \in A} \|p-m\|^z \qquad (9.7)
\end{aligned}$$

Combining, we then obtain for

$$\begin{aligned}
& \left| \|A-C\|_{z,2}^z - \|M'-C'\|_{z,2}^z \right| \\
(Lem.\ 9.10 \text{ and } 9.11) \quad \leq & \; \varepsilon \cdot \left(\|A-C\|_{z,2}^z + \|A-M\|_{z,2}^z \right) \\
(Eq.\ 9.7) \quad & + 2^{z+2} \cdot \left(\frac{4z}{\varepsilon} \right)^z \cdot \alpha \cdot \frac{25088 z^6}{\varepsilon^6} \cdot \|A-M\|_{z,2}^z \\
(\text{no cost decrease}) \quad \leq & \; 2 \cdot \left(2\varepsilon + 2^{z+2} \cdot \left(\frac{4z}{\varepsilon} \right)^z \cdot \alpha \cdot \frac{25088 z^6}{\varepsilon^6} \right) \|A-C\|_{z,2}^z
\end{aligned}$$

The overall result now follows by our choice of α and rescaling ε.

9.3 Derandomized Dimension Reduction

In this section we will use the deterministic construction of partition coresets to obtain a derandomized dimension reduction for (k,z)-clustering. The goal is to show that we

can obtain a cost preserving sketch with dimension at most $O(\varepsilon^{-O(z)} \log k)$:

▶ **Theorem 9.14.** Let A be an $n \times d$ matrix. Suppose we are given an (ε, k, z)-extension partition coreset of A with at most $T = k^{\varepsilon^{-O(z)}}$ distinct rows. Then we can compute a $(k, z, 4\varepsilon)$-cost-preserving sketch of A with $O(\varepsilon^{-O(z)} \log k)$ columns in deterministic time $O\left(n^{\varepsilon^{-O(z)}}\right)$ time. ◀

Combing this theorem with Theorem 9.8, now yields the following corollary.

▶ **Corollary 9.15.** Let A be an $n \times d$ matrix. We can compute an (k, z, ε)-cost preserving sketch with target dimension $O\left(\varepsilon^{-O(z)} \cdot \log k\right)$ in deterministic time $O\left(n^{\varepsilon^{-O(z)}}\right)$. ◀

Again, the first-thought approach would be to use the deterministic Johnson- Lindenstrauss transform of Engebretsen et al. [72] (Theorem 9.7). Unfortunately, merely preserving pairwise distances is not sufficient to preserve the cost to an optimal $(1, z)$-center, except for the special case $z = 2$. Instead, we introduce the following notion.

▶ **Definition 9.16 (Witness Sets).** Let A be a set of n points in $\mathbb{R}^d$ (with a possible extension coordinate) and let c be the optimal $(1, z)$-center (under the possible constraint that the extension coordinate is 0) and let $\Delta = \frac{\sum_{i=1}^{n} \|A_i - c\|^z}{n}$. Then a (D, R, ε)-uniform witness set is a subset $S \subset A$ such that

- A $(1 + \varepsilon)$-approximation of c (under the possible constraint that the extension coordinate is 0) is contained in the convex hull of S.
- The diameter of S is at most $D \cdot \sqrt[z]{\Delta}$.
- The size of S is at most R. ◀

The first important result is proving that the existence of witness sets of small size exist for a specific range of parameters.

▶ **Theorem 9.17.** There exists $\left(\frac{O(z)}{\varepsilon}, \varepsilon^{-O(1)} \cdot 2^{O(z)}, \varepsilon\right)$ witness sets. ◀

We give a full proof of Theorem 9.17 in Section 9.5. The existence of small witness sets is essentially a consequence of the existence of small coresets. Let us consider how this existence of small witness sets combined with Theorem 9.8 implies that we can deterministically construct a cost preserving sketch for powers.

The main idea is that the property of witness sets allows us to efficiently "discretize" all candidate solutions in the convex hull of a witness set. We will show that the dimension reduction preserves the property of a subset of points being a witness set. Furthermore, because witness sets lie in low-dimensional spaces, we can guarantee that the cost for every point in the convex hull of a witness is preserved. Specifically, we use

that linear projections preserve subspaces, and in particular the subspace spanned by the points in the convex hull of the witness set. Hence, if the cost of every candidate solution in a witness set stays the same, up to a $(1 \pm \varepsilon)$ factor, the clustering cost overall also stays the same.

Proof of Theorem 9.14. Let B be an (ε, k, z)-extension partition coreset of A, with extension B'. By definition of B', any (ε, k, z)-cost preserving sketch for B' is also one for A. We focus henceforth on B and its extension B'.

In a first step, we enumerate over all witness sets of B', under the condition that the extension coordinate of the center is 0. This boils down to enumerating over all witness sets of B: indeed, if S' is a subsest of B' such that an $(1+\varepsilon)$-approximation of c under the constraint that the coordinate extension is 0 is contained in the convex hull of S', then the $(1+\varepsilon)$-approximation is also contained in the convex hull of $S \subseteq B$ obtained from S' by removing the coordinate extension. Hence, we can perform this enumeration in brute-force way; there only exist $T = k^{\varepsilon^{-O(z)}}$ distinct entries of B, hence there also only exist $T^{\varepsilon^{-O(z)}} = k^{\varepsilon^{-O(z)}}$ witness sets.

For each witness set S, we now construct an ε-cover of the candidate solutions in the convex hull of S. First, observe that S lies in an $|S| = \varepsilon^{-O(z)}$-dimensional space. Hence, we can compute an ε'-cover of S of size at most $\left(\frac{O(\sqrt{|S|})}{\varepsilon'}\right)^{|S|}$ in deterministic polynomial time [43][3]. We set $\varepsilon' = \varepsilon/(2z)$. Moreover, let V be an orthogonal basis of the subspace spanned by S. We additionally compute a 1/2-cover of the unit S-dimensional ball spanned by V.

Then the union $\mathcal{N}$ of all covers of all witness sets has size at most

$$T^{\varepsilon^{-O(z)}} \cdot \left(\frac{O(\sqrt{|S|}) \cdot D}{\varepsilon^z}\right)^{|S|} = k^{\varepsilon^{-O(z)}}.$$

We now use Engebretsen et al. 9.7 to deterministically compute a mapping $f : \mathbb{R}^d \to \mathbb{R}^m$ with $m \in O(\varepsilon^{-O(z)} \log k)$ that embeds all pairwise distances between the points corresponding to rows of B, as well as the points in $\mathcal{N}$, up to a distortion factor $(1 \pm \varepsilon/z)$. This can be done in time $O\left(n \cdot d \cdot k^{\varepsilon^{-O(z)}} \cdot \log^{O(1)} n\right)$. Denote the n by $m \in O(\varepsilon^{-O(z)} \log k)$ dimensional sketch of B by Π, and let Π' be the extension of Π obtained by appending the final column of B'.

In order to verify correctness, we now show that for any subset C of $\{1, ..., n\}$, the cost of the optimal $(1, z)$-clustering is preserved, namely that

$$cost(B'_C) := \sum_{j \in C} \|B'_j - c^{B'}\|^z = (1 \pm \varepsilon) \sum_{j \in C} \|\Pi'_j - c^{\Pi'}\|^z,$$

where $B'_C = \{B'_j, j \in C\}$ and $c^{B'}$ is the optimal center for points B'_C, under the constraint that the extension coordinate is 0, and $c^{\Pi'}$ the equivalent for $\Pi'_C := \{\Pi'_j, j \in C\}$.

[3] Recall that a ε-cover of a set S is a set of points N such that for an $x \in S$, we have some $y \in N$ with $\|x - y\| \leq \varepsilon$.

We first prove the somewhat easier direction which states that the cost does not increase. Let S be a witness set for B'_C. Denote by c^* the point whose 0-extension is the $(1+\varepsilon)$-approximate solution to $c^{B'}$ guaranteed to lie in the convex hull of S, and let c the point whose 0-extension is the closest cover point to $(c^*, 0)$. By definition of the cover and witness set S, we have

$$\|c^* - c\| \leq \varepsilon' \cdot \sqrt[z]{\frac{\text{cost}(B_C)}{|C|}} \tag{9.8}$$

Further, let $f(c)$ be the embedding of c, and for simplicity let $\Delta_C := \frac{\text{cost}(B_C)}{|C|}$. We have, for a fixed point j:

$$\begin{aligned}
& & \|\Pi_j - f(c)\| \\
\text{(Distortion of Embedding)} & \leq & (1+\varepsilon/z) \cdot \|B_j - c\| \\
& \leq & (1+\varepsilon/z) \cdot (\|B_j - c^*\| + \|c^* - c\|) \\
(Eq.\ 9.8) & \leq & (1+\varepsilon/z) \cdot \left(\|B_j - c^*\| + \varepsilon' \cdot \sqrt[z]{\Delta_C}\right)
\end{aligned}$$

Hence, writing $f'(c)$ the 0-extension of $f(c)$, we obtain

$$\begin{aligned}
\|\Pi'_j - f'(c)\|^z &= \left(\|\Pi_j - f(c)\|^2 + {B'_j}^2\right)^{z/2} \\
&\leq \left((1+\varepsilon/z) \cdot \left(\|B_j - c^*\|^2 + \varepsilon' \cdot \sqrt[z/2]{\Delta_C}\right) + {B'_j}^2\right)^{z/2} \\
&\leq (1+\varepsilon/z)^{z/2} \cdot \left((\|B_j - c^*\|^2 + {B'_j}^2) + \varepsilon' \cdot \sqrt[z/2]{\Delta_C}\right)^{z/2} \\
&\leq (1+\varepsilon/z)^{z/2} \left((1+\varepsilon) \cdot (\|B_j - c^*\|^2 + {B'_j}^2)^{z/2} + \left(\frac{z+\varepsilon}{\varepsilon}\right)^{z/2-1} {\varepsilon'}^{z/2} \cdot \Delta_C\right) \\
&\leq (1+2\varepsilon) \cdot (\|B_j - c^*\|^2 + {B'_j}^2)^{z/2} + \varepsilon \cdot \Delta_C,
\end{aligned}$$

where the last inequality holds by choice of ε'. Summing over all $j \in C$, we get that

$$\begin{aligned}
\text{cost}(\Pi'_C) &\leq \sum_{j \in C} \|\Pi'_j - f'(c)\|^z \leq (1+2\varepsilon)\text{cost}(B'_C) + \varepsilon \text{cost}(B'_C) \\
&\leq (1 + O(\varepsilon))\text{cost}(B'_C).
\end{aligned} \tag{9.9}$$

We now turn our attention to the somewhat more involved case of proving that the cost does not decrease. The key difference is that the witness set S_Π for points $\{\Pi'_j, j \in C\}$ may be different from the witness set S, and we cannot use the previous argument verbatim. Assume that $\text{cost}(\Pi'_C) \leq \text{cost}(B'_C)$ (otherwise we are already done).

Let us consider the (D, R, ε)-witness set S_Π of Π'_C, with $D = O(z)/\varepsilon$ and $R = \varepsilon^{-O(1)}$. By our assumption on $\text{cost}(\Pi'_C)$, we know that the diameter of S_Π is at most $D \cdot \sqrt[z]{\text{cost}(B'_C)/|C|}$. We note two facts about the embedding f preserving distances between points of $\mathcal{N}$ and rows of B: it is a linear projection, hence surjective, and the image of an ε'-cover is still an ε'-cover. Let $f(s)$ be an arbitrary point in

the convex hull of S_Π in the projected space, and let $f(c)$ be its closest point in the projection of the cover.

Exactly as above, one can show that $\|\Pi_j - f(s)\| \geq (1-\varepsilon/z)\|B_j - c\| - \varepsilon' \sqrt[z]{\Delta_C}$, then that $\|\Pi'_j - f'(s)\|^z \geq (1 + -\varepsilon) \cdot (\|B_j - c^*\|^2 + {B'_j}^2)^{z/2} - \varepsilon \cdot \Delta_C$ and conclude that $\sum_{j \in C} \|\Pi'_j - f'(s)\|^z \geq (1 - O(\varepsilon))\text{cost}(B'_C)$. For completeness, we provide a thorough proof in Section 9.7 Choosing s such that $f'(s)$ is the approximate optimal center guaranteed by the definition of witness set, to get $\text{cost}(\Pi'_C) \geq (1 - O(\varepsilon))\text{cost}(B'_C)$. Combined with Eq. (9.9), this concludes the proof of the theorem: the mapping g preserves the cost of any cluster. □

9.4 Improved Deterministic Coreset Construction via Uniform VC-dimension

We show in this section how to use ε-set-approximations of a particular set system to build improved coresets deterministically.

Recall that an (ε, k, z)-coreset with offset for the (k, z)-Clustering problem in a metric space (X, dist) with set of clients P is a weighted subset Ω of P with weights $w : \Omega \to \mathbb{R}_+$ together with a constant F such that, for any set $\mathcal{S} \subset X$, $|\mathcal{S}| = k$,

$$|\text{cost}(P, \mathcal{S}) - \big(F + \text{cost}(\Omega, \mathcal{S})\big)| \leq \varepsilon \text{cost}(P, \mathcal{S}).$$

To be consistent with the coreset literature we will go back to the notation $\text{cost}(\cdot, \cdot)$ introduced in Section 1.6, instead of matrix notations. In this section, we will use notions from VC-dimension literature, defined as follows:

▶ **Definition 9.18.** A tuple $(X, \mathcal{R})$ is a *range space* when $\mathcal{R} = (R_1, ..., R_m)$ with $R_i \subseteq X$.
A set A is an ε-set-approximation for $(X, \mathcal{R})$ if

$$\forall R \in \mathcal{R}, \left| \frac{|R|}{|X|} - \frac{|R \cap A|}{|A|} \right| \leq \varepsilon.$$

The VC-dimension of a range space is the size of the largest set Y such that $\{Y \cap R,\ R \in \mathcal{R}\} = 2^{|Y|}$, meaning that every subset of Y is of the form $Y \cap R$ for some $R \in \mathcal{R}$. ◀

The main result of the section is the following theorem. We relate the size of the coreset to the VC dimension of the range space $\mathcal{B}$ formed by k balls, defined as follows: For a point $c \in X$ and $r \in \mathbb{R}_{\geq 0}$, let $H_{c,r} = \{p \in P \mid \text{dist}(p, c) \geq r\}$. For k centers $C = \{c_1, c_2, \ldots, c_k\}$, we similarly define $H_{C,r} = \{p \in P \mid \min_{c \in C} \text{dist}(p, c) \geq r\}$. We let $\mathcal{B} := \bigcup_C \bigcup_r H_{C,r}$.

▶ **Theorem 9.19.** Let (X, dist) be a metric space such that the VC dimension of $(P, \mathcal{B})$ is D. There exists a deterministic algorithm running in deterministic time $\tilde{O}(|X| \cdot |P| \cdot k/\varepsilon) + k\,(D/\varepsilon)^{O(D)} \cdot |P|$ that constructs an ε-coreset with offset for the set P, with size $2^{O(z \log z)} \cdot \frac{kD \log(D) \text{polylog}(1/\varepsilon)}{\varepsilon^5}$.
In the Euclidean Space $\mathbb{R}^d$, the running time is $|P|^{\varepsilon^{-O(z)}} \cdot d + k\,(D/\varepsilon)^{O(D)} \cdot |P|$. ◀

Application to Euclidean Spaces. Our prominent example is the Euclidean case. It is well known from the coreset literature that the set system aforementioned has VC-dimension $O(kd \log k)$ in Euclidean spaces of dimension d. This has been proven for example Lemma 1 of [12] and Corollary 34 of [78] [4]. This in particular leads to ε'-set-approximation of size $O\left(\frac{kd \log k}{\varepsilon'^2} \cdot \log(kd \log k/\varepsilon')\right)$.

In order to remove the dependency in d and match the best coreset result, we aim at using a classical terminal embedding as in Section 7.6. In particular, we will start from the coreset of size $k^{\varepsilon^{-O(z)}}$ from Theorem 9.8, and apply a deterministic terminal embedding construction on it. As we saw in Section 7.6, this allows to consider that the dimension is $O\left(\log\left(k^{\varepsilon^{-O(z)}}\right)\right)$, and yields the following corollary:

▶ **Corollary 9.20.** There exist a deterministic algorithm running in time $n^{\varepsilon^{-O(z)}} \cdot d + (1/\varepsilon)^{\varepsilon^{-O(z)}} k \log^3 k$. that constructs an (ε, k, z)-coreset of size $k^2 \log^2 k \cdot \varepsilon^{-O(z)}$ for input in the Euclidean space $\mathbb{R}^d$. ◀

Proof. To reduce the dimension, we use our coreset construction combined with a terminal embedding, as in Huang and Vishnoi [100] and Section 7.6. We sketch the proof here and refer to Section 7.6 for more details.

We start by computing a (ε, k, z)-coreset of A of size $k^{\varepsilon^{-O(z)}}$, using Theorem 9.8, in time $n^{\varepsilon^{-O(z)}}$. Then, we build a terminal embedding of that coreset onto a space with dimension $O(\varepsilon^{-O(z)} \log k)$: this can be done deterministically in time $k^{\varepsilon^{-O(z)}}$ using the algorithm from Mahabadi et al. [127].

Now, we can compute a coreset of the embedding using Theorem 9.19. Since in dimension $O(\varepsilon^{-O(z)} \log k)$ the VC-dimension of $(P, \mathcal{B})$ is at most $\varepsilon^{-O(z)} k \log^2 k$, this concludes the proof. □

Application to Minor-Excluded Graphs, and Graphs with Bounded Treewidth. Theorem 9.19 can also be applied to graphs excluding a minor H: in that case, Bousquet and Thomassé [30] showed that the VC-dimension of $(P, \mathcal{B})$

[4] The bounds in the literature are often stated for a generalization known as the weighted function space induced by k-clustering in Euclidean spaces, which is necessary for coreset constructions based on importance sampling. In this chapter, we show how to use uniform sampling instead, which allows us to use Theorem 9.21 as a black box.

is $O(|H|k \log k)$.[5] Hence applying directly Theorem 9.19 yields a coreset of size essentially $\frac{k^2|H|\log^2 k}{\varepsilon^5}$. The result is somewhat simplified compared to Corollary 7.15, at the price of losing a factor $k\varepsilon^{-1}$ (or ε^{-3}). On the other hand, the dependency in the size of the minor is greatly improved: we are not even able to specify it precisely in Corollary 7.15, although it seems to be at least doubly exponential.

Since graphs with treewidth t exclude the complete graph with t vertices, the result allows to construct coreset of size $\frac{k^2 t \log^2 k}{\varepsilon^5}$ for those. As we will see, the running time is dominated by the construction of an ε-set-approximation. Giving up on the determinism, this can be made fast using random sampling (see Chazelle [43]): this yields a running time $\tilde{O}(|X| \cdot |P| \cdot k/\varepsilon)$.

Application to Clustering of Curves. Driemel, Nusser, Phillips, and Psarros showed in [65] VC dimension bounds for clustering of curves, under the Fréchet and Hausdorff distances. The input of that problem is a set of curves, each consisting of m segments in $\mathbb{R}^d$, and the centers are restricted to be curves consisting of ℓ segments in $\mathbb{R}^d$.

They consider two types of distances for curves: the Hausdorff distance between two curves X and Y is defined as follows: let $d_{\vec{H}}(X,Y) = sup_{x\in X} \inf_{y\in Y} \|x-y\|$. The Hausforff distance between X and Y is $\max(d_{\vec{H}}(X,Y), d_{\vec{H}}(Y,X))$. The Fréchet distance is slightly more intricate: for that, we consider parametrized curves, namely a curve is a function $X : [0,1] \to \mathbb{R}^d$ whose graph is made of ℓ consecutive segments. The Fréchet distance between two parametrized curves X and Y is defined as $\min_{f:[0,1]\to[0,1]} \max_{t\in[0,1]} \|X(f(t)) - Y(t)\|$, where the function f is restricted to be continuous, non decreasing and with $f(0) = 0, f(1) = 1$.

For both those distances, Driemel et al. [65] showed that the VC dimension of $(P, \mathcal{B})$ is bounded by $O\left(kd^2\ell^2 \log(d\ell m) \log k\right)$. Hence, Theorem 9.19 shows the existence of a coreset of size $\frac{k^2 d^2 \ell^2 \log(d\ell m)\log^2 k}{\varepsilon^{O(1)}}$. This improves over [36], in particular removing the dependency in the number of input curves, although our construction is not polynomial time in that case.

A Sketch of the Construction As explained in the introduction (Section 9.1), our construction is inspired by the one of Chen [44] and of Chapter 6. As both of those result, we start by computing a constant-factor approximation, and placing rings of exponential radius around each of its center.

We first use a careful preprocessing step to reduce the number of such rings from $O(k \log n)$ as in [44] to $O(k)$. This step is different to that of Chapter 6: in that chapter, we used importance sampling in the outer rings, something that we do not know how to do deterministically. Here, we instead use a particularly structured constant-factor approximation, that allows us to simply discard the outer rings.

To build coresets in the remaining rings, both [44] and Chapter 6 use uniform sampling. Instead, we show that building an ε'-set-approximation of the set system formed

[5] Formally, they show the result for $k = 1$. Braverman et al. [32] showed that extending the result to any k loses a $k \log k$ factor.

by k balls is actually enough: this is doable deterministically, which concludes our algorithm.

The size of an ε'-set-approximation can be bounded via the VC-dimension of the set system: if the VC dimension is D, then an ε'-set-approximation has size $\tilde{O}\left(D\varepsilon^{-2}\right)$ (see more details in Theorem 9.21)

9.4.1 How to sample uniformly and deterministically

We start by explaining how to compute a set with properties similar to that of a uniform sample. While we need to define the VC-dimension for that, the only property we will need is that it is possible to compute ε-set-approximation of balls in $\mathbb{R}^d$ in time essentially $(kd/\varepsilon)^{\tilde{O}(kd)}$.

The VC-dimension of a range-space helps bounding the size of ε-set-approximation, as highlighted by the following theorem:

▶ **Theorem 9.21 (Theorem 4.5 in Chazelle [43]).** Let $(X, \mathcal{R})$ be a range space of VC-dimension d. Given any $r > 0$, an r-set-approximation for $(X, \mathcal{R})$ of size $O\left(dr^{-2}\log(d/r)\right)$ can be computed deterministically in time $d^{O(d)}\left(r^{-2}\log(d/r)\right)^d |X|$. ◀

We will compute ε-set-approximation for the set system $(P, \mathcal{B})$, where P is the set of input points and $\mathcal{B}$ is defined in the opening paragraph.

9.4.2 The algorithm

As sketched, the algorithm partitions the input points into rings, and computes then ε-set-approximation in each ring. We recall here useful definitions, similar to those of Chapter 6.

Fix a metric space $I = (X, \mathrm{dist})$, positive integers k, z and a set of clients P. For a solution $\mathcal{S}$ of (k, z)-clustering on P and a center $c \in \mathcal{S}$, c's cluster consists of all points closer to c than to any other center of $\mathcal{S}$.

Fix as well some $\varepsilon > 0$, and let $\mathcal{A}$ be any solution for (k, z)-clustering on P with k centers. Let $C_1, ..., C_k$ be the clusters induced by the centers of $\mathcal{A}$.

- the average cost of a cluster C_i is $\Delta_{C_i} = \frac{\mathrm{cost}(C_i, \mathcal{A})}{|C_i|}$
- For all i, j, the *ring* $R_{i,j}$ is the set of points $p \in C_i$ such that
$$2^j \Delta_{C_i} \leq \mathrm{cost}(p, \mathcal{A}) \leq 2^{j+1}\Delta_{C_i}.$$
- The *inner ring* $R_I(C_i)$ (resp. *outer ring* $R_O(C_i)$) of a cluster C_i consists of the points of C_i with cost at most $(\varepsilon/z)^z \Delta_{C_i}$ (resp. at least $(z/\varepsilon)^{2z} \Delta_{C_i}$), i.e.,

$R_I(C_i) := \cup_{j \le z\log(\varepsilon/z)} R_{i,j}$ and $R_O(C_i) := \cup_{j > 2z\log(z/\varepsilon)} R_{i,j}$. The *main ring* $R_M(C_i)$ consists of all the other points of C_i. For a solution $\mathcal{S}$, we let $R_I^{\mathcal{S}}$ and $R_O^{\mathcal{S}}$ be the union of inner and outer rings of the clusters induced by $\mathcal{S}$.

The final algorithm is as follows:

Input: A metric space (X, dist), a set $P \subseteq X$, $k, z > 0$, ε such that $0 < \varepsilon < 1/3$, and $\varepsilon' \le \varepsilon$.
Output: A coreset with offset. Namely, a set of points $P' \subseteq X$, a weight function $w : P' \mapsto \mathbb{R}_+$ and an offset value F such that for any set of k centers C, $\text{cost}(P, C) = (1 \pm \varepsilon)\text{cost}(P', C) + F$.

1. **Greedy seeding:**
   (a) Compute a $c_{\mathcal{A}}$-approx $\mathcal{A}$ to the (k, z)-clustering problem for P.[6]
   (b) Initialize $\mathcal{G} := \mathcal{A}$.
   (c) While: $\text{cost}(\mathcal{G}) \ge \varepsilon\text{cost}(\mathcal{A})/c_{\mathcal{A}}$ and there is a candidate center c such that $\left(1 - \frac{\varepsilon}{k \cdot c_{\mathcal{A}}}\right)\text{cost}(\mathcal{G}) \ge \text{cost}(\mathcal{G} \cup \{c\})$ do:
   $\mathcal{G} \leftarrow \mathcal{G} \cup \{c\}$.
2. If $\text{cost}(\mathcal{G}) \le \varepsilon\text{cost}(\mathcal{A})/c_{\mathcal{A}}$, then output the following coreset and stop: $\mathcal{G}$ and $w : \mathcal{G} \mapsto \mathbb{R}_+$ where $w(c)$ corresponds to the number of points served by center c in $\mathcal{G}$, and offset $F = 0$.
3. Let $F := \text{cost}(R_O(\mathcal{G}), \mathcal{G})$ be the cost of the outer points of $\mathcal{G}$.
4. Set the weights of all the centers of $\mathcal{G}$ to 0.
5. **Reducing the number of rings:**:
   For each cluster C with center c of $\mathcal{G}$, remove all the points in $R_I(C) \cup R_O(C)$ and increase the weight of c by the number of removed points. Let I_G be the instance created with those weights.
6. **Sampling from the remaining rings:** For every cluster C_i, and every j , compute an ε'-set-approximation $\Omega_{i,j}$ of the set system $(R_{i,j}, \mathcal{B})$. Weight the points of $\Omega_{i,j}$ by $\frac{R_{i,j}}{|\Omega_{i,j}|}$.
7. **Output:**
   - An instance I_G created at Step 5
   - A coreset consisting of $\mathcal{G} \cup \Omega_{i,j}$, with offset F defined at Step 3 and weights for $\mathcal{G}$ defined throughout the algorithm, weights for $\Omega_{i,j}$ defined by the sampling procedure.

If the condition $\text{cost}(\mathcal{G}) \le \varepsilon\text{cost}(\mathcal{A})/c_{\mathcal{A}}$ is met, then we say that $\mathcal{G}$ is *low-cost*; otherwise, we say it is *locally stable*.

[6] Any solution with $O(k)$ clusters and cost $O(\text{OPT})$ can be actually used for this step, in order to have a faster algorithm.

We will need to compare cost of solution in the original instance I, and in the one computed by the algorithm: cost will denote the cost in I while cost_{I_G} the one for instance I_G.

9.4.3 Proof of the Greedy Seeding

The outcome of the greedy seeding step, I_G, satisfies the following lemma:

▶ **Lemma 9.22.** Suppose $\mathcal{G}$ is locally stable. Let $F = \sum_{C \in \mathcal{G}} \sum_{p \in R_O(C)} \text{cost}(p, \mathcal{G})$ be the cost of points in outer rings. For every solution $\mathcal{S}$, it holds that

$$|\text{cost}(\mathcal{S}) - (\text{cost}_{I_G}(\mathcal{S}) + F)| \leq \varepsilon \text{cost}(\mathcal{S}).$$ ◀

Before proving Lemma 9.22, we show that the solution $\mathcal{G}$ has the following properties:

▶ **Lemma 9.23.** $\mathcal{G}$ contains at most $O(k \cdot z \cdot \log(1/\varepsilon)/\varepsilon)$ centers and, in the case where $\mathcal{G}$ is locally stable, for any solution S and any subset $Q \subseteq P$, $\text{cost}(X, \mathcal{G}) \leq \text{cost}(Q, \mathcal{S}) + \varepsilon \text{OPT}$. ◀

Proof. Note that the cost of $\mathcal{G}$ is clearly at most the cost of $\mathcal{A}$, whose cost is at most $c_{\mathcal{A}}$ times the cost of the best solution for the k-clustering problem.

Each center added by the greedy step decreases its cost by a factor $(1 - \varepsilon/c_{\mathcal{A}}k)$, and the initial solution $\mathcal{G}$ has cost $\text{cost}(\mathcal{G})$: since the algorithm stops if the cost drops below $\varepsilon \text{cost}(\mathcal{A})/c_{\mathcal{A}}$, the total number of centers added is at most $c_{\mathcal{A}} k \log(1/\varepsilon)/\varepsilon$ The total number of centers is therefore at most $c_{\mathcal{A}} k \log(1/\varepsilon)/\varepsilon$.

Note also that $\text{cost}(\mathcal{A})/c_{\mathcal{A}} \leq \text{OPT}$: hence, the inequality $(1 - \varepsilon/c_{\mathcal{A}}k)\text{cost}(\mathcal{G}) \leq \text{cost}(\mathcal{G} \cup \{c\})$ implies $\text{cost}(\mathcal{G}) \leq \text{cost}(\mathcal{G} \cup \{c\}) + \varepsilon \text{OPT}/k$. Hence, since $\mathcal{G}$ is locally stable, there is no candidate center c such that $\text{cost}(\mathcal{G}) - \text{cost}(\mathcal{G} \cup \{c\}) \geq \varepsilon \text{OPT}/k$. Let $\mathcal{S}$ be a solution, Q be a subset of clients, s be a center of the solution $\mathcal{S}$ and C_s be its cluster. Since the improvement made by adding s is at least $\sum_{p \in C_s \cap Q} \text{cost}(p, \mathcal{G}) - \text{cost}(p, \mathcal{S})$, it holds that $\sum_{p \in C_s \cap Q} \text{cost}(p, \mathcal{G}) - \text{cost}(p, \mathcal{S}) \leq \text{cost}(\mathcal{G}) - \text{cost}(\mathcal{G} \cup \{c\}) \leq \varepsilon \text{OPT}/k$. Summing over the k centers of the solution $\mathcal{S}$ yields the inequality $\text{cost}(Q, \mathcal{S}) \leq \text{cost}(Q, \mathcal{G}) + \varepsilon \text{OPT}$, concluding the lemma. □

We can now turn to the proof of Lemma 9.22:

proof of Lemma 9.22. Fix a cluster C of $\mathcal{G}$, with center c. We first want to show that $\text{cost}(C, \mathcal{S}) \leq \text{cost}_{I_G}(C, \mathcal{S}) + \text{cost}(R_O(C), \mathcal{G}) + \varepsilon \text{cost}(C, \mathcal{S})$. First, this is equivalent to $\text{cost}(R_O(C), \mathcal{S}) \leq |R_O(C)| \text{cost}(c, \mathcal{S}) + \text{cost}(R_O(C), \mathcal{G}) + \varepsilon \text{cost}(C, \mathcal{S})$, as all points in $C \setminus R_O(C)$ have same cost in the original instance and in I_G.

Using Lemma 1.2, we have

$$\text{cost}(R_O(C), \mathcal{S}) \leq (1+\varepsilon)\text{cost}(R_O(C), \mathcal{G}) + (1 + z/\varepsilon)^{z-1}|R_O(C)|\text{cost}(c, \mathcal{S}).$$

Hence, one needs to bound $|R_O(C)|\text{cost}(c, \mathcal{S})$. For that, we show that the cost of clients in $R_O(C)$ can be charged to clients of $R_I(C) \cup R_M(C)$.

First note by Markov's inequality, $|R_O(C)| \leq (\varepsilon/z)^{2z} |C|$. Hence, one can partition $R_I(C) \cup R_M(C)$ into parts of size at least $s = (|C| - |R_O(C)|)/|R_O(C)|$, and assign every part to a point in $R_O(C)$ in a one-to-one correspondence.

For such a point $p \in R_O(C)$ consider the s points $p_1, \ldots, p_s$ of the part assigned to it. We show how to charge $\text{cost}(c, \mathcal{S})$ to the cost of those points. Let $\mathcal{S}(p_j)$ be the center serving p_j in the solution $\mathcal{S}$. It holds that $\text{cost}(c, \mathcal{S}) \leq \min_j \text{cost}(c, \mathcal{S}(p_j))$. Averaging, this is at most $\frac{1}{s}\sum_{j=1}^{s} \text{cost}(c, \mathcal{S}(p_j))$ which is due to Lemma 1.2 at most $\frac{1}{s}\sum_{j=1}^{s}(1+\varepsilon)\text{cost}(p_j, \mathcal{G}) + (1 + z/\varepsilon)^{z-1}\text{cost}(p_j, \mathcal{S})$. Since $1/s \leq 2|R_O(C)|/|C| = 2\,(\varepsilon/z)^{2z}$, we conclude that (using $2\left(\frac{\varepsilon}{z}\right)(1+\varepsilon) \leq 1$)

$$\begin{aligned} |R_O(C)|\text{cost}(c, \mathcal{S}) &\leq 2\left(\frac{\varepsilon}{z}\right)^{z+1}(1+\varepsilon) \sum_{p \in R_I(C) \cup R_M(C)} \text{cost}(p, \mathcal{G}) + \text{cost}(p, \mathcal{S}) \\ &\leq \left(\frac{\varepsilon}{z}\right)^{z} (\text{cost}(C, \mathcal{G}) + \text{cost}(C, \mathcal{S})). \end{aligned} \tag{9.10}$$

We apply this inequality to bound $\text{cost}(R_O(C), \mathcal{S})$:

$$\begin{aligned} \text{cost}(R_O(C), \mathcal{S}) &\leq (1+\varepsilon)\text{cost}(R_O(C), \mathcal{G}) + (1 + z/\varepsilon)^{z-1}|R_O(C)|\text{cost}(c, \mathcal{S}) \\ &\leq \text{cost}(R_O(C), \mathcal{G}) + 2\varepsilon(\text{cost}(C, \mathcal{G}) + \text{cost}(C, \mathcal{S})) \end{aligned}$$

Hence, summed over all cluster, this yields

$$\text{cost}(R_O, \mathcal{S}) \leq \text{cost}(R_O, \mathcal{G}) + 2\varepsilon(\text{cost}(\mathcal{G}) + \text{cost}(\mathcal{S})) \tag{9.11}$$

We now turn to the other direction of the inequality. Using Lemma 9.23,

$$\begin{aligned} \text{cost}(R_O, \mathcal{G}) &\leq \text{cost}(R_O, \mathcal{S}) + \varepsilon\text{OPT} \\ &\leq \text{cost}(R_O, \mathcal{S}) + \varepsilon\text{cost}(\mathcal{S}) \end{aligned}$$

Combined with Eq. (9.10), this yields

$$\text{cost}(R_O, \mathcal{G}) + \text{cost}_{I_G}(R_O, \mathcal{S}) - \text{cost}(R_O, \mathcal{S}) \leq 2\varepsilon(\text{cost}(\mathcal{S}) + \text{cost}(\mathcal{G})). \tag{9.12}$$

Combining Eq. (9.11) and Eq. (9.12), and using that all points not in R_O have same cost in the original instance and in I_G concludes the lemma. □

9.4.4 Computing Coresets via ε-set-approximation

We first show that an ε-set-approximation of the set system $(R_{i,j}, \mathcal{B})$ is indeed a coreset for $R_{i,j}$. We will then bound the size of such an approximation. In this section, we fix the cluster C_i and a ring $R_{i,j}$.

▶ **Lemma 9.24.** Let $R_{i,j}$ be a ring, and Ω be an ε'-set-approximation for $(R_{i,j}, \mathcal{B})$ with $\varepsilon' = 20\frac{8^z\varepsilon^2}{\log(4z/\varepsilon)}$. Then Ω with uniform weights $|R_{i,j}|/|\Omega|$ is an ε-coreset for $R_{i,j}$. ◀

We use the same proof technique as in Section 6.3 to analyze the algorithm. More precisely, we define as before ranges of clients: we let $I_{i,j,\ell}$ to be the set of points from $R_{i,j}$ that pays roughly $(1+\varepsilon')^\ell$ is $\mathcal{S}$, i.e., $I_{i,j,\ell} := \{p \in R : (1+\varepsilon'')^\ell \leq \text{cost}(p, \mathcal{S}) < (1+\varepsilon'')^{\ell+1}\}$, where ε'' will be set later.

This analysis distinguishes between three cases:

1. $\ell \leq j + \log_{1+\varepsilon''} \varepsilon$, in which case we say that I_ℓ is *tiny*. The union of all tiny ranges is denoted $I_{tiny,\mathcal{S}}$.
2. $j + \log_{1+\varepsilon''} \varepsilon \leq \ell \leq j + \log_{1+\varepsilon''}(4z/\varepsilon)$, in which case we say I_ℓ is *interesting*.
3. $\ell \geq j + \log_{1+\varepsilon''}(4z/\varepsilon)$, in which case we say I_ℓ is *huge*.

The very same proof as Lemma 6.12 shows that the cost the tiny ranges is preserved. Event $\mathcal{E}$ holds trivially due to our choice of weights: hence, when the ring $R_{i,j}$ intersects a huge range, Lemma 6.14 carries over to our setting. For completeness, we provide the proof in the last section of the chapter.

Hence, the key here is to show that the remaining $O(\log(4z/\varepsilon))$ interesting ranges are preserved by the ε'-set-approximation.

▶ **Lemma 9.25.** Let $R_{i,j}$ be a ring, and $\mathcal{S}$ be a solution such that all huge I_ℓ are empty. Further, let Ω be an ε'-set-approximation for $(R, \mathcal{B})$, and uniform weights $|R|/|\Omega|$ for point in Ω. Then, if $\varepsilon'' = \varepsilon/10$ and $\varepsilon' = 20\frac{2^{3z}\varepsilon^2}{\log(4z/\varepsilon)}$, it holds that $\text{cost}(\Omega, S) = (1 \pm \varepsilon)\text{cost}(R_{i,j}, S)$. ◀

Proof. For simplicity, we drop the subscript i, j and let $R := R_{i,j}, I_\ell := I_{i,j,\ell}$. Fix some solution $\mathcal{S}$.

To show that $\text{cost}(\mathcal{S}) \approx \text{cost}(\Omega, \mathcal{S})$, we will show the following stronger property: for any ℓ,

$$|\text{cost}(I_\ell, \mathcal{S}) - \text{cost}(I_\ell \cap \Omega, \mathcal{S})| \leq \frac{\varepsilon}{\log(4z/\varepsilon)}\text{cost}(I_\ell, \mathcal{S}).$$

Hence, we fix some integer ℓ. Note that $I_\ell = H_{\mathcal{S},(1+\varepsilon'')^{\ell+1}} \setminus H_{\mathcal{S},(1+\varepsilon'')^\ell}$, where $H_{\mathcal{S},r} = \{p \in P : \text{dist}(p, \mathcal{S}) \geq r\} \in \mathcal{B}$. Since Ω is an ε'-set-approximation for $(P, \mathcal{B})$, it holds

that

$$\left|\frac{|H_{\mathcal{S},(1+\varepsilon'')^{\ell+1}}|}{|R|} - \frac{|H_{\mathcal{S},(1+\varepsilon'')^{\ell+1}} \cap \Omega|}{|\Omega|}\right| \leq \varepsilon'$$
$$\left|\frac{|H_{\mathcal{S},(1+\varepsilon'')^{\ell}}|}{|R|} - \frac{|H_{\mathcal{S},(1+\varepsilon'')^{\ell}} \cap \Omega|}{|\Omega|}\right| \leq \varepsilon'$$

Combining those two guarantees with triangle inequality ensure that

$$\left|\frac{|I_\ell|}{|R|} - \frac{|I_\ell \cap \Omega|}{|\Omega|}\right| \leq \left|\frac{|H_{\mathcal{S},(1+\varepsilon'')^{\ell+1}}|}{|R|} - \frac{|H_{\mathcal{S},(1+\varepsilon'')^{\ell+1}} \cap \Omega|}{|\Omega|}\right| + \left|\frac{|H_{\mathcal{S},(1+\varepsilon'')^{\ell}}|}{|R|} - \frac{|H_{\mathcal{S},(1+\varepsilon'')^{\ell}} \cap \Omega|}{|\Omega|}\right| \leq 2\varepsilon'.$$

Furthermore, by definition of I_ℓ, $|I_\ell|\cdot(1+\varepsilon'')^\ell \leq \text{cost}(I_\ell, \mathcal{S}) < |I_\ell|\cdot(1+\varepsilon'')^{\ell+1}$. Similarly, due to the weighting of points in Ω, we have $(1+\varepsilon'')^\ell \frac{|R|\cdot|I_\ell \cap R|}{|\Omega|} \leq \text{cost}(I_\ell \cap \Omega, \mathcal{S})$ Combining those equations, we get

$$\begin{aligned}
\text{cost}(I_\ell, \mathcal{S}) &\leq |I_\ell| \cdot (1+\varepsilon'')^{\ell+1} \\
&\leq |I_\ell \cap \Omega| \cdot (1+\varepsilon'')^{\ell+1} \cdot \frac{|R|}{|\Omega|} + 2\varepsilon' \cdot |R|(1+\varepsilon'')^{\ell+1} \\
&\leq (1+\varepsilon'')\text{cost}(I_\ell \cap \Omega, \mathcal{S}) + 2\varepsilon' \cdot |R|(1+\varepsilon'')^{\ell+1}.
\end{aligned}$$

To bound further the right hand side, we let $q \in I_\ell$, so that $(1+\varepsilon'')^\ell \leq \text{cost}(q, \mathcal{S})$:

$$\begin{aligned}
|R|(1+\varepsilon'')^\ell &\leq \sum_{p \in R} \text{cost}(q, \mathcal{S}) \leq \sum_{p \in R} 2^{z-1}\text{cost}(p, q) + 2^{z-1}\text{cost}(p, \mathcal{S}) \\
&\leq 2^{z-1}\text{cost}(R, \mathcal{S}) + 2^{z-1} \sum_{p \in R} 2^{z-1} \left(\text{cost}(p, \mathcal{G}) + \text{cost}(q, \mathcal{G})\right) \\
&\leq 2^z \text{cost}(R, \mathcal{S}) + 2^{3z-1}\text{cost}(R, \mathcal{G}),
\end{aligned}$$

where the last line uses $\text{cost}(q, \mathcal{G}) \leq 2^z \text{cost}(p, \mathcal{G})$, since p and q are in the same ring.

Combined with the previous inequality, this yields $\text{cost}(I_\ell, \mathcal{S}) \leq (1+\varepsilon'')\text{cost}(I_\ell \cap \Omega, \mathcal{S}) + 2^{3z}\varepsilon' \cdot (\text{cost}(R, \mathcal{S}) + \text{cost}(R, \mathcal{G}))$.

Similarly, we get

$$\text{cost}(I_\ell \cap \Omega, \mathcal{S}) \leq (1+\varepsilon'')\text{cost}(I_\ell, \mathcal{S}) + 2^{3z}\varepsilon'(\text{cost}(R, \mathcal{S}) + \text{cost}(R, \mathcal{G})).$$

Summing that over the $\log_{1+\varepsilon''}(4z/\varepsilon)$ interesting ranges and combining with Lemma 9.29 gives that:

$$\begin{aligned}
&|\text{cost}(I_\ell, \mathcal{S}) - \text{cost}(I_\ell \cap \Omega, \mathcal{S})| \\
\leq\ &\log_{1+\varepsilon''}(4z/\varepsilon) \cdot 2^{3z}\varepsilon' \cdot (\text{cost}(R, \mathcal{S}) + \text{cost}(R, \mathcal{G})) + \varepsilon''\text{cost}(R \cap \Omega, \mathcal{S}) + \varepsilon''\text{cost}(R, \mathcal{S}).
\end{aligned}$$

Hence, we have:

$$\begin{aligned}
|\mathrm{cost}(I_\ell, \mathcal{S}) - \mathrm{cost}(I_\ell \cap \Omega, \mathcal{S})| &\leq \log_{1+\varepsilon''}(4z/\varepsilon) \cdot 2^{3z}\varepsilon' \cdot (\mathrm{cost}(R, \mathcal{S}) + \mathrm{cost}(R, \mathcal{G})) \\
&\quad + \varepsilon'' \mathrm{cost}(R \cap \Omega, \mathcal{S}) + \varepsilon'' \mathrm{cost}(R, \mathcal{S}) \\
&\leq \log_{1+\varepsilon''}(4z/\varepsilon) \cdot 2^{3z}\varepsilon' \cdot (\mathrm{cost}(R, \mathcal{S}) + \mathrm{cost}(R, \mathcal{G})) \\
&\quad + 2\varepsilon'' \mathrm{cost}(R, \mathcal{S}) + \varepsilon'' |\mathrm{cost}(R, \mathcal{S}) - \mathrm{cost}(R \cap \Omega, \mathcal{S})| \\
&\leq \left(\log_{1+\varepsilon''}(4z/\varepsilon) \cdot 2^{3z}\varepsilon' + 2\varepsilon''\right) \cdot (\mathrm{cost}(R, \mathcal{S}) + \mathrm{cost}(R, \mathcal{G})) \\
&\quad + \varepsilon'' |\mathrm{cost}(R, \mathcal{S}) - \mathrm{cost}(R \cap \Omega, \mathcal{S})|,
\end{aligned}$$

which is equivalent to

$$|\mathrm{cost}(R, \mathcal{S}) - \mathrm{cost}(R \cap \Omega, \mathcal{S})| \leq \frac{\log_{1+\varepsilon''}(4z/\varepsilon) \cdot 2^{3z}\varepsilon' + 2\varepsilon''}{(1-\varepsilon'')} (\mathrm{cost}(R, \mathcal{S}) + \mathrm{cost}(R, \mathcal{G}).$$

Chosing ε'' such that $\frac{2\varepsilon''}{1-\varepsilon''} \leq \frac{\varepsilon}{2}$ (e.g. $\varepsilon'' = \varepsilon/10$) and ε' such that $\frac{\log_{1+\varepsilon''}(4z/\varepsilon) \cdot 2^{3z}\varepsilon'}{1-\varepsilon''} \leq \frac{\varepsilon}{2}$ (e.g. $\varepsilon' = 20\frac{2^{3z}\varepsilon^2}{\log(4z/\varepsilon)}$) concludes the lemma. □

Combining the results for tiny, interesting and huge ranges concludes the proof of Lemma 9.24 and shows that $\Omega_{i,j}$ is an ε-coreset of $R_{i,j}$. Combined with Lemma 9.22, this concludes the coreset guarantee of Theorem 9.19. The size of the coreset for a given ring is $O\left(2^{O(z \log z)} D\varepsilon^{-4} \log(D/\varepsilon)\right)$ from Theorem 9.21, and there are $O\left(2^{O(z \log z)} k\varepsilon^{-1} \mathrm{polylog}(1/\varepsilon)\right)$ many rings. Hence, the total size of the coreset constructed is $O\left(2^{O(z \log z)} \cdot kD\varepsilon^{-k} \log(D) \mathrm{polylog}(1/\varepsilon)\right)$ It only remains to bound the time complexity of the algorithm.

9.4.5 Complexity Analysis

Two steps dominate the running time of the algorithm: the greedy seeding, and the computation of the ε'-set-approximation. For the latter, a direct combination of Theorem 9.21 shows that all the ε'-set-approximation can be computed in time $k(D/\varepsilon)^{-O(d)} \cdot |X|$.

Hence, it only remains to bound the running time of the greedy seeding.

In the discrete metric case, this is straightforward: each iteration costs $O(|X| \cdot |P| \cdot k)$, and there are $\tilde{O}(k/\varepsilon)$ many of them (see Lemma 9.23).

The Euclidean case requires more work, as it is not possible to enumerate over all candidate center. However, as shown in Section 9.6, this can be performed in time $n^{\varepsilon^{-O(z)}}$. We defer the proof to that section.

9.5 Witness Sets

The goal of that section is to prove the existence of small witness sets, namely Theorem 9.17.

For that, we rely on the existence of small coreset for $(1, z)$-clustering, that we showed in Corollary 7.20: we restate the result here for convenience.

▶ **Lemma 9.26.** [See Corollary 7.20] Let A be a set of n points in $\mathbb{R}^d$. There exists a set Ω_A of size $\tilde{O}(\varepsilon^{-2}2^{O(z)})$ and weights $w : \Omega_A \to \mathbb{R}_+$ such that, for every possible center c,

$$\left|\sum_{i=1}^{n} \|A_i - c\|^z - \sum_{i\in\Omega} w(i)\|A - c\|^z\right| \leq \varepsilon \sum_{i=1}^{n} \|A_i - c\|^z. \qquad \blacktriangleleft$$

We use those coresets as follow. Let $a \in A$ be a center inducing a 2^z-approximation for $(1, z)$-clustering on A, and let Δ_a be the average cost for that solution (note that $\Delta = \frac{\sum_{i=1}^{n} \|A-c\|^z}{n} \leq 2^z\Delta_a$). We use Lemma 9.26 to show that their exists a coreset Ω_A where all points are at distance at most $\frac{z}{\varepsilon} \cdot (\Delta_a)^{1/z}$ of a. This is enough to conclude the existence of $(\frac{2z}{\varepsilon}, \varepsilon^{-O(1)} \cdot 2^{O(z)}, \varepsilon)$-uniform witness set: the optimal center for Ω_A must lie in the convex hull of Ω_A, and the diameter of Ω_A is at most $\frac{2z}{\varepsilon}\Delta$.

▶ **Lemma 9.27.** Let A be a set of n points in $\mathbb{R}^d$, and $a \in A$ be a center inducing a 2^z-approximation for $(1, z)$-clustering on A.
There exist a set Ω_A of size $O(\varepsilon^{-2z})$ and weights $w : \Omega_A \to \mathbb{R}_+$ such that: every point of Ω_A is at distance at most $\frac{2z}{\varepsilon}\Delta_a^{1/z}$ of a, and for every possible center c,

$$\left|\sum_{i=1}^{n} \|A_i - c\|^z - \sum_{i\in\Omega_A} w(i)\|A - c\|^z\right| \leq \varepsilon \sum_{i=1}^{n} \|A_i - c\|^z. \qquad \blacktriangleleft$$

Proof. Let O_A be the set of points at distance more than $\left(\frac{z}{\varepsilon}\right)^2 \Delta_a^{1/z}$ of a. Let p be any point at distance $\left(\frac{z}{\varepsilon}\right)^2 \Delta_a^{1/z}$ of a[7]. We start by showing that, for all center c,

$$\left|\mathrm{cost}(A, c) - \left(\mathrm{cost}(A \setminus O_A, c) + \frac{\mathrm{cost}(O_A, a)}{\mathrm{cost}(p, a)}\mathrm{cost}(p, c)\right)\right| \leq O(\varepsilon)\left(\mathrm{cost}(A, c) + \mathrm{cost}(A, a)\right). \tag{9.13}$$

Proving this inequality would conclude the lemma. Indeed, since $2^{-z}\mathrm{cost}(A, a) \leq \mathrm{cost}(A, c)$, the right hand side can be upper-bounded by $O(\varepsilon 2^z)\mathrm{cost}(A, c)$. Hence, adding the point p with weight $w(p) := \frac{\mathrm{cost}(O_A,a)}{\mathrm{cost}(p,a)}$ to an ε-coreset Ω_A^1 (with weights w) of $A \setminus O_A$ provided by Lemma 9.26 gives a $O(\varepsilon 2^z)$-coreset for A. Rescaling ε by 2^z concludes.

[7] this distance is somewhat arbitrarily picked in order to simplify the proof.

To show Equation 9.13, we fix a center c. We start by bounding $|O_A|\text{cost}(a,c)$. For that, we show that the cost of clients in O_A can be charged to clients of $A \setminus O_A$. First note that by averaging, $|O_A| \leq (\varepsilon/z)^{2z} |A|$ – otherwise $\text{cost}(O_A, a) \geq |O_A| \left(\frac{z}{\varepsilon}\right)^{2z} \Delta_a > |A|\Delta_a = \text{cost}(A,a)$. Hence, one can partition $A \setminus O_A$ into parts of size at least $s = (|A| - |O_A|)/|O_A|$, and assign every part to a point in O_A in a one-to-one correspondence.

For such a point $p \in O_A$ consider the s points $p_1, \ldots, p_s$ of the part assigned to it. We show how to charge $\text{cost}(a,c)$ to the cost of those points. By the modified triangle inequality, for all i $\text{cost}(a,c) \leq (1+\varepsilon)\text{cost}(a,p_i) + (1+z/\varepsilon)^{z-1}\text{cost}(p_i,c)$. Taking an average over all i gives that $\text{cost}(a,c)$ is at most $\frac{1}{s}\sum_{j=1}^{s} \varepsilon\text{cost}(p_j,a) + (1+z/\varepsilon)^{z-1}\text{cost}(p_j,c)$. Since $1/s \leq 2|O_A|/|A| = 2\,(\varepsilon/z)^{2z}$, we conclude that (using $2\left(\frac{\varepsilon}{z}\right)(1+\varepsilon) \leq 1$)

$$\begin{aligned} |O_A|\text{cost}(a,c) &\leq 2\left(\frac{\varepsilon}{z}\right)^{z+1}(1+\varepsilon)\sum_{p\in A\setminus O_A} \text{cost}(p,a) + \text{cost}(p,c) \\ &\leq \left(\frac{\varepsilon}{z}\right)^{z}(\text{cost}(A,a) + \text{cost}(A,c)) \end{aligned} \tag{9.14}$$

We now show how to use this inequality to show Equation 9.13, and start by showing $\frac{\text{cost}(O_A,a)}{\text{cost}(p,a)}\text{cost}(p,c) \leq (1+O(\varepsilon))\text{cost}(O_A,c) + O(\varepsilon)\text{cost}(A,c)$.

We proceed as follows: we decompose the left-hand-side using Lemma 1.2. Let p be the point chosen to represent O_A:

$$\frac{\text{cost}(O_A,a)}{\text{cost}(p,a)}\text{cost}(p,c) \leq (1+\varepsilon)\text{cost}(O_A,a) + (1+z/\varepsilon)^{z-1}\left(\frac{\text{cost}(O_A,a)}{\text{cost}(p,a)}\right)\text{cost}(a,c)$$

We bound separately the two terms. First,

$$\begin{aligned} \text{cost}(O_A,a) &\leq (1+\varepsilon)\text{cost}(O_A,c) + (1+z/\varepsilon)^{z-1}|O_A|\text{cost}(a,c) \\ &\leq (1+\varepsilon)\text{cost}(O_A,c) + (1+z/\varepsilon)^{z-1}\left(\frac{\varepsilon}{z}\right)^{z}(\text{cost}(A,a)+\text{cost}(A,c)) \\ &\leq (1+\varepsilon)\text{cost}(O_A,c) + \varepsilon(\text{cost}(A,a)+\text{cost}(A,c) \\ &\leq \text{cost}(O_A,c) + O(\varepsilon)(\text{cost}(A,a)+\text{cost}(A,c). \end{aligned}$$

Similarly, using $\text{cost}(p,a) = \left(\frac{z}{\varepsilon}\right)^{2z}\frac{\text{cost}(A,a)}{|A|} \geq \left(\frac{z}{\varepsilon}\right)^{2z}\frac{\text{cost}(O_A,a)}{|A|}$:

$$\begin{aligned} (1+\frac{z}{\varepsilon})^{z-1}\left(\frac{\text{cost}(O_A,a)}{\text{cost}(p,a)}\right)\text{cost}(a,c) &\leq (1+\frac{z}{\varepsilon})^{z-1}\cdot\left(\frac{\varepsilon}{z}\right)^{2z}|A|\text{cost}(a,c) \\ &\leq \left(\frac{\varepsilon}{z}\right)^{z}\left((1+\varepsilon)\text{cost}(A,c) + (1+\frac{z}{\varepsilon})^{z-1}\text{cost}(A,a)\right) \\ &= O(\varepsilon)(\text{cost}(A,c)+\text{cost}(A,a)). \end{aligned} \tag{*}$$

Hence,

$$\frac{\text{cost}(O_A,a)}{\text{cost}(p,a)}\text{cost}(p,c) \leq \text{cost}(O_A,c) + O(\varepsilon)\left(\text{cost}(A,c)+\text{cost}(A,a)\right). \tag{1}$$

We now turn to the other direction, namely

$$\operatorname{cost}(O_A, c) \leq (1+\varepsilon)\frac{\operatorname{cost}(O_A, a)}{\operatorname{cost}(p, a)}\operatorname{cost}(p, c) + O(\varepsilon)\left(\operatorname{cost}(A, c) + \operatorname{cost}(A, a)\right).$$

We write

$$\operatorname{cost}(O_A, c) \leq (1+\varepsilon)\operatorname{cost}(O_A, a) + (1+z/\varepsilon)^{z-1}|O_A|\operatorname{cost}(a, c),$$

and bound here as well the two terms separately. First we have, using (*)

$$\begin{aligned}
\operatorname{cost}(O_A, a) &= \frac{\operatorname{cost}(O_A, a)}{\operatorname{cost}(p, a)}\operatorname{cost}(p, a) \\
&\leq (1+\varepsilon)\frac{\operatorname{cost}(O_A, a)}{\operatorname{cost}(p, a)}\operatorname{cost}(p, c) + \left(\frac{z}{\varepsilon}\right)^{z-1}\frac{\operatorname{cost}(O_A, a)}{\operatorname{cost}(p, a)}\operatorname{cost}(a, c) \\
&\leq (1+\varepsilon)\frac{\operatorname{cost}(O_A, a)}{\operatorname{cost}(p, a)}\operatorname{cost}(p, c) + O(\varepsilon)\left(\operatorname{cost}(A, c) + \operatorname{cost}(A, a)\right)
\end{aligned}$$

The second term is bounded by Equation 9.14:

$$(1+z/\varepsilon)^{z-1}|O_A|\operatorname{cost}(a, c) \leq \varepsilon\left(\operatorname{cost}(A, c) + \operatorname{cost}(A, a)\right).$$

Hence,

$$\operatorname{cost}(O_A, c) \leq (1+\varepsilon)\frac{\operatorname{cost}(O_A, a)}{\operatorname{cost}(p, a)}\operatorname{cost}(p, c) + O(\varepsilon)\operatorname{cost}(A, c). \tag{2}$$

Combining equations 1 and 2 yields:

$$\left|\operatorname{cost}(O_A, c) - \frac{\operatorname{cost}(O_A, a)}{\operatorname{cost}(p, a)}\operatorname{cost}(p, c)\right| \leq O(\varepsilon)\left(\frac{\operatorname{cost}(O_A, a)}{\operatorname{cost}(p, a)}\operatorname{cost}(p, c) + \operatorname{cost}(A, c)\right).$$

Using $\frac{\operatorname{cost}(O_A,a)}{\operatorname{cost}(p,a)}\operatorname{cost}(p, c) \leq \operatorname{cost}(O_A, c) + \left|\operatorname{cost}(O_A, c) - \frac{\operatorname{cost}(O_A,a)}{\operatorname{cost}(p,a)}\operatorname{cost}(p, c)\right|$ finally concludes the proof of Equation 9.13, and hence the lemma. □

9.6 Computing Approximate Solutions

In this section, we show how to compute a $(1+\varepsilon, O(1/\varepsilon))$-approximation for (k, z)-Clustering in time $n^{\varepsilon^{-O(z)}}$. The algorithm is the following.

- Start computing an α-approximation S_0 to (k, z)-clustering on A.
- As long as there exists a center c such that $\operatorname{cost}(S_i \cup \{c\}) \leq (1 - \frac{\varepsilon}{\alpha k})\operatorname{cost}(S_i)$, do $S_{i+1} \leftarrow S_i \cup \{c\}$ and $i \leftarrow i+1$.
- Stop this procedure when there is no such center c, or $\operatorname{cost}(S_i) \leq \frac{\varepsilon}{\alpha}\operatorname{cost}(S_0)$, and let S be the final solution.

▶ **Lemma 9.28.** S is a $(1+\varepsilon, \alpha\log(1/\varepsilon)/\varepsilon+1)$-bicriteria approximation for (k,z)-clustering on A. ◀

Proof. Each center added decreases the cost by a factor $(1-\varepsilon/\alpha k)$, and the initial solution S_0 has cost $\text{cost}(S_0)$: since the algorithm stops if the cost drops below $\varepsilon\text{cost}(S_0)/\alpha$, the total number of centers added is at most $\alpha k\log(1/\varepsilon)/\varepsilon$. The total number of centers is therefore at most $\alpha k\log(1/\varepsilon)/\varepsilon + k$.

In the case where the procedure stops because $\text{cost}(S) \leq \frac{\varepsilon}{\alpha}\text{cost}(S_0)$, then S has cost at most ε times the optimal cost, since S_0 is an α-approximation. In that case, we are done.

In the other case, we note that $\text{cost}(S_0)/\alpha \leq \text{OPT}$: hence, the inequality $(1-\varepsilon/\alpha k)\text{cost}(S) \leq \text{cost}(S\cup\{c\})$ holds for any candidate center c and implies $\text{cost}(S) \leq \text{cost}(S\cup\{c\}) + \varepsilon\text{OPT}/k$. In particular, for any subset X of input points, it must be that $\text{cost}(X,S) \leq \text{cost}(X, S\cup\{c\}) + \varepsilon\text{OPT}/k$, as $\text{cost}(P\setminus X, S) \geq \text{cost}(P\setminus X, \cup\{c\})$. Summing over the k centers of the optimal solution OPT (with X being their respective clusters) yields the inequality $\text{cost}(S) - \text{cost}(S\cup\text{OPT}) \leq \varepsilon\text{OPT}$, hence $\text{cost}(S) \leq \text{cost}(S\cup\text{OPT}) + \varepsilon\text{OPT}$ and S is a $(1+\varepsilon)$-approximation of OPT. This concludes the lemma. □

It remains to show how to implement that greedy procedure fast. For that, we discretize the set of candidate centers, to be able to check all of them. For that, we enumerate all possible witness set S and take an $\varepsilon^{O(z)}\cdot\text{diam}(S)$-net of the convex hull of S. Let $\mathbb{C}$ be the union of those nets.

Since there exists a $(\frac{O(z)}{\varepsilon}, \varepsilon^{-O(z)}, \varepsilon)$ witness set (Theorem 9.17), there are $n^{\varepsilon^{-O(z)}}$ many different set S. For each of them, the size of the net is $\varepsilon^{O(z)|S|} = \varepsilon^{\varepsilon^{-O(z)}}$, since the convex hull of S lies in a space of dimension $|S|$.

Now, the existence of a witness set for every optimal cluster of the optimal solution ensure that, for all cluster, there exists a point in $\mathbb{C}$ giving a $(1+\varepsilon)$-approximation in that cluster. Hence, there exists a $(1+\varepsilon)$-approximation in $\mathbb{C}^k$. Using this solution in place of the optimal one in the final equation of Lemma 9.28 ensures that the solution S given by the greedy algorithm restricted to pick centers in $\mathbb{C}$ yields a $(1+\varepsilon)$-approximation.

Using Theorem 9.5 for computing the constant-factor approximation gives $\alpha = z^{O(z)}$: the solution S is thus a $(1+\varepsilon, z^{O(z)}\log(1/\varepsilon)/\varepsilon)$-bicriteria approximation. The running time of this greedy bicriteria algorithm is $n^{\varepsilon^{-O(z)}}\cdot\varepsilon^{\varepsilon^{-O(z)}} = n^{\varepsilon^{-O(z)}}$.

9.7 Omitted Proofs

▶ **Lemma 9.29 (Equivalent of Lemma 6.12).** It holds that

$$\max\left(\sum_{p\in I_{tiny,\mathcal{S}}}\text{cost}(p,\mathcal{S}),\sum_{p\in I_{tiny,\mathcal{S}}\cap\Omega}\frac{|R_{i,j}|}{\delta}\text{cost}(p,\mathcal{S})\right)\leq\varepsilon\cdot\text{cost}(R_{i,j},\mathcal{G}).$$

◀

Proof. By definition of $I_{tiny,\mathcal{S}}$, $\sum_{p\in I_{tiny,\mathcal{S}}}\text{cost}(p,\mathcal{S})\leq\sum_{p\in I_{tiny,\mathcal{S}}}\frac{\varepsilon}{2}\cdot\text{cost}(p,\mathcal{G})\leq\frac{\varepsilon}{2}\cdot\text{cost}(R_{i,j},\mathcal{G})$. Similarly, we have for the other term

$$\begin{aligned}\sum_{p\in I_{tiny,\mathcal{S}}\cap\Omega}\frac{|R_{i,j}|}{\delta}\cdot\text{cost}(p,\mathcal{S})&\leq\sum_{p\in I_{tiny,\mathcal{S}}\cap\Omega}\frac{|R_{i,j}|}{\delta}\frac{\varepsilon}{2}\cdot\text{cost}(p,\mathcal{G})\\&\leq\varepsilon\cdot\frac{|R_{i,j}|}{\delta}\sum_{p\in C_i\cap I_{tiny,\mathcal{S}}\cap\Omega}\frac{2^z\text{cost}(R_{i,j},\mathcal{G})}{|R_{i,j}|}\\&\leq\varepsilon\cdot\frac{|I_{tiny,\mathcal{S}}\cap\Omega|}{\delta}\text{cost}(R_{i,j},\mathcal{G})\leq\varepsilon\cdot\text{cost}(G,\mathcal{G}).\end{aligned}$$

where we used that since each point of $R_{i,j}$ have same cost up to a factor 2^z, it holds that $\forall p\in R_{i,j},\text{cost}(p,\mathcal{G})\leq 2^z\frac{\text{cost}(R_{i,j},\mathcal{G})}{|R_{i,j}|}$. The last inequality uses that Ω contains δ points. □

▶ **Lemma 9.30 (Equivalent of Lemma 6.28).** It holds that, for any $R_{i,j}$ and for all solutions S with at least one non-empty huge group $I_{i,j,\ell}$

$$\left|\text{cost}(R_{i,j},\mathcal{S})-\sum_{p\in\Omega\cap R_{i,j}}\frac{|R_{i,j}|}{\delta}\cdot\text{cost}(p,\mathcal{S})\right|\leq 3\varepsilon\cdot\text{cost}(R_{i,j},\mathcal{S}).$$

◀

Proof. Fix a ring $R_{i,j}$ and let $I_{i,j,\ell}$ be a huge group. First, the weight of $R_{i,j}$ is preserved in Ω: since the set Ω has size δ, it holds that

$$\sum_{p\in\Omega\cap R_{i,j}}\frac{|R_{i,j}|}{\delta}=|R_{i,j}|$$

Now, let $\mathcal{S}$ be a solution, and $p\in I_{i,j,\ell}$ with $I_{i,j,\ell}$ being huge. This implies, for any $q\in R_{i,j}$: $\text{cost}(p,q)\leq(2\cdot\varepsilon\cdot 2^{j+1})^z\leq 4^z\cdot\varepsilon^z\cdot 2^{(\ell-\log(4z/\varepsilon))z}\leq(\varepsilon/z)^z\cdot\text{cost}(p,\mathcal{S})$. By

Lemma 1.2, we have therefore for any point $q \in R_{i,j}$

$$\begin{aligned} \operatorname{cost}(p,\mathcal{S}) &\leq (1+\varepsilon/z)^{z-1}\operatorname{cost}(q,\mathcal{S}) + (1+z/\varepsilon)^{z-1}\operatorname{cost}(p,q) \\ &\leq (1+\varepsilon)\operatorname{cost}(q,\mathcal{S}) + \varepsilon\cdot\operatorname{cost}(p,\mathcal{S}) \\ \Rightarrow \operatorname{cost}(q,\mathcal{S}) &\geq \frac{1-\varepsilon}{1+\varepsilon}\operatorname{cost}(p,S) \geq (1-2\varepsilon)\operatorname{cost}(p,\mathcal{S}) \end{aligned}$$

Moreover, by a similar calculation, we can also derive an upper bound of $\operatorname{cost}(q,\mathcal{S}) \leq \operatorname{cost}(p,\mathcal{S})\cdot(1+2\varepsilon)$. Hence, combined with $\sum_{p\in\Omega\cap R_{i,j}} \frac{|R_{i,j}|}{\delta} = |R_{i,j}|$, this is sufficient to approximate $\operatorname{cost}(R_{i,j},\mathcal{S})$.

Therefore, the cost of $R_{i,j}$ is well approximated for any solution $\mathcal{S}$ such that there is a non-empty huge group $I_{i,j,\ell}$. □

We finally complete the proof of Theorem 9.14.

Proof of Theorem 9.14 completed. Recall the settings: S_Π is a (D,R,ε)-witness set of Π'_C, with $D = O(z)/\varepsilon$ and $R = \varepsilon^{-O(1)}$. By our assumption $\operatorname{cost}(\Pi'_C) \leq \operatorname{cost}(B'_C)$, we know that the diameter of S_Π is at most $D\cdot\sqrt[z]{\operatorname{cost}(B'_C)/|C|}$. $f(s)$ is an arbitrary point in the convex hull of S_Π in the projected space, and $f(c)$ is its closest point in the projection of the cover.

Our goal here is to show that $\sum_{j\in C}\|\Pi'_j - f'(s)\|^z \geq (1-O(\varepsilon))\operatorname{cost}(B'_C)$.

We first require an upper bound on $\|f(c)-f(s)\|$. By definition of the cover, this is at most

$$\|f(c)-f(s)\| \leq \varepsilon'\cdot\sqrt[z]{\operatorname{cost}(B'_C)/|C|} = \varepsilon'\Delta_C. \tag{9.15}$$

Therefore, using triangle inequality, the distortion of embedding f and Section 9.7:

$$\begin{aligned} \|\Pi_j - f(s)\| &\geq \|\Pi_j - f(c)\| - \|f(c)-f(s)\| \\ &\geq (1-\varepsilon/z)\|B_j - c\| - \varepsilon'\sqrt[z]{\Delta_C}. \end{aligned}$$

Writing again $f'(x)$ for the 0-extension of x, we have:

$$\begin{aligned} \|\Pi'_j - f'(s)\|^z &= \left(\|\Pi_j - f(s)\|^2 + {B'_j}^2\right)^{z/2} \\ &\geq \left((1-\varepsilon/z)\cdot\left(\|B_j-c\|^2 - \varepsilon'\cdot\sqrt[z/2]{\Delta_C}\right) + {B'_j}^2\right)^{z/2} \\ &\geq (1-\varepsilon/z)^{z/2}\cdot\left((\|B_j-c^*\|^2 + {B'_j}^2) - \varepsilon'\cdot\sqrt[z/2]{\Delta_C}\right)^{z/2} \\ &\geq (1-\varepsilon/z)^{z/2}\left((1-\varepsilon)\cdot(\|B_j-c^*\|^2 + {B'_j}^2)^{z/2} - \left(\frac{z+\varepsilon}{\varepsilon}\right)^{z/2-1}\varepsilon'^{z/2}\cdot\Delta_C\right) \\ \text{(choice of } \varepsilon') &\geq (1-2\varepsilon)\cdot(\|B_j-c^*\|^2 + {B'_j}^2)^{z/2} - \varepsilon\cdot\Delta_C \end{aligned}$$

Hence, summing over all points in cluster C, we get

$$\sum_{j\in C}\|\Pi'_j - f'(s)\|^z \geq (1-O(\varepsilon))\operatorname{cost}(B'_C).$$

□

Chapter 10

Sublinear Algorithms for Power Mean in Euclidean Spaces

10.1 Introduction

In this chapter, we leverage the coreset construction previously introduced in order to get *sublinear* algorithms to compute some precise statistics on a dataset. By sublinear, we mean that we design algorithms that do not need to read the entire dataset in order to produce an accurate solution.

Except for trivial problems, deterministic time sublinear algorithms do not exist. Our primary tool in designing sublinear algorithms is thus the following basic approach:

- Take a uniform sample of the input.
- Run an algorithm on the sample.

Hence, the performance of a sublinear algorithm is often measured in terms of its *query complexity*, i.e. the number of samples required such that we can extract a high quality solution in the second step above that generalizes to the entire input. Sublinear algorithms have close ties to questions in learning theory and estimation theory, where we are similarly interested in the tradeoff between quality and sample size.

We notice a great resemblance between he sublinear approach and the coreset construction presented in the previous chapter. In both cases, one attempts to sample a summary from the input data to then be able to solve the problem by only looking at the summary. The key difference, of course, is that the coreset sampling distribution is not uniform.

In this chapter, we show how to use the coreset construction of Section 9.4 in order to estimate the optimal solution of the $(1, z)$-clustering problem, that we dub *z-power means*, in high dimensional Euclidean spaces. Specifically, given an arbitrary set of

points P, we wish to determine the number of uniform queries Ω such that we can extract a z-power mean m with

$$\text{cost}(m) := \sum_{p \in P} \|p - m\|^z \leq (1+\varepsilon) \cdot \min_{\mu} \sum_{p \in P} \|p - \mu\|^z,$$

where $\|p\|$ denotes the Euclidean norm of a vector p.

The problem of estimating the z-power mean falls into the category of problems that seek to efficiently estimate the parameters of a distribution. This is one of the fundamental problem in data analysis: for example, given a distribution $\mathcal{D}$, how many samples do we need to estimate the mean? Even such a simple and basic question has surprisingly involved answers and are still subject to ongoing research ([125, 118]).

The z-power mean problem captures a number of important problems in computational geometry and multivariate statistics. For example, for $z = 1$, this corresponds to the Fermat-Weber problem also known as the geometric median. For $z = 2$, the problem is to determine the mean or centroid of the data set. Letting $z \to \infty$, we have the *Minimum Enclosing Ball* (MEB), where one needs to find the Euclidean sphere of smallest radius containing all input points.

For $z > 2$, the problem is not as well studied, but it still has many applications. First, higher powers allow us to interpolate between $z = 2$ and $z \to \infty$, which is interesting as the latter admits no sublinear algorithms[1]. Skewness (a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean) and kurtosis (a measure of the "tailedness" of the probability distribution) are the centralized moments with respect to the three and the four norms and are frequently used in statistics. The z-power mean is a way of estimating these values for multivariate distributions.

Another application is when dealing with non-Euclidean distances, such as the Hamming metric, coreset constructions for powers of z can be reduced to coreset constructions for powers $2z$. So for example if we want the mean in Hamming space, we can reduce it to the $z = 4$ case in squared Euclidean spaces [100].

These problems are convex and thus can be approximated in the near-linear time efficiently via convex optimization techniques. However, aside from the mean ($z = 2$), doing so in a sublinear setting is challenging and to the best of our knowledge, only the mean and the geometric median ($z = 1$) are currently known to admit nearly linear time algorithms.

Our main result is:

> ▶ **Theorem 10.1.** There exists an algorithm that, with query complexity $O\left(\varepsilon^{-z-5} \cdot \text{polylog}(\varepsilon^{-1}) \log^2 1/\delta\right)$, computes a $(1+\varepsilon)$ approximate solution to the high dimensional z-power mean problem with probability at least $1-\delta$. ◀

Our algorithm is based on coreset constructions. We showed in Chapter 7 the existence

[1] To see this, we place $n-1$ points at 0 and one point with probability $1/2$ at 1 and with probability $1/2$ at 0. Any $2-\varepsilon$ approximation can distinguish between the two cases, but this clearly requires us to query $\Omega(n)$ points.

of coresets of size $\tilde{O}(\varepsilon^{-4} \cdot 2^{O(z)})$ for $(1, z)$-clustering. Comparing with the bounds in Theorem 10.1, one may question whether the exponential dependency in z is necessary for computing an approximation, as it is not for computing a coreset. Unfortunately, we show that the exponential dependency in the power is indeed necessary even in a single dimension:

▶ **Theorem 10.2.** For any $\varepsilon > 0$ and z, any algorithm that computes with probability more than $4/5$ a $(1 + \varepsilon)$-approximation for a one-dimensional z-power mean has query complexity $\Omega(\varepsilon^{-z+1})$. ◀

Hence, up to constants in the exponent, our sublinear algorithm is tight. Moreover, the algorithm is very simple to implement and performs well empirically.

10.1.1 Techniques

While stochastic gradient descent has been used for a variety of center-based problems ([47, 49]), it is difficult to apply it for higher powers. Indeed, Cohen et al. [49] remark in their paper that even for the mean[2] ($z = 2$) their analysis does not work as the objective function is neither Lipschitz, nor strictly convex.

Hence, one needs to use new tools. A natural starting point is to use techniques from coresets, as they allow us to preserve most of the relevant information, using a substantially smaller number of points. Unfortunately, coresets have a drawback: the sampling distributions used to construct coresets is non-uniform and therefore difficult to use in a sublinear setting. Thankfully, we showed in previous chapter how to design coresets from uniform sampling: given a sufficiently good initial solution $\mathfrak{a}$, one can partition the points into rings exponentially increasing radii such that the points cost the same, up to constant factors. Thereafter, taking a uniform sample of size $\tilde{O}(\varepsilon^{-4})$ from each ring produces a coreset. Since there are at most $O(\log n)$ rings in the worst case, this yields a coreset of size $\tilde{O}(\varepsilon^{-4} \cdot \log n)$.

To realize these ideas in a sublinear setting, we are now faced with a number of challenges. First, rings that are particularly far from $\mathfrak{a}$ may contain very few points. This makes is difficult for a sublinear algorithm to access them. Second, partitioning the points into rings depends on the cost of $\mathfrak{a}$, and it is very simple to construct examples where estimating the cost of an optimal power center requires $\Omega(n)$ many queries. Finally, this very brief analysis loses a factor $\log n$, which we aim to avoid.

We improve and extend this framework as follows. We show that it is sufficient to only consider $O(\log \varepsilon^{-1})$ many rings, which, in of itself, already removes the dependency on $\log n$. Moreover, we show that it is possible to simply ignore any ring containing too few points, i.e. any ring with less than $\varepsilon^{z+O(1)} \cdot n$ points may be discarded. The intuition is that, while rings with few point may contribute significantly to the cost, these points do not influence the position of the optimal center by much. Thus, using

[2] For the special case of the mean, Inaba et al. [102] observed that $O(\varepsilon^{-2})$ samples are nevertheless sufficient.

a number of carefully chosen pruning steps, we show how to reduce the problem of obtaining a sublinear algorithm, as well as obtaining coresets, to sampling from few selected rings containing many points.

10.1.2 Experimental Evaluation

While we can prove that the algorithm can compute a good solution in constant time for every constant ε and z, even for moderately small ε (e.g. $\varepsilon = 1/2$) the sampling complexity becomes quite large for even small values of z, as indeed our lower bound shows is necessarily the case. Our experiments therefore aim at evaluating the performance of the sublinear algorithm on realistic, not necessarily worst case data sets.

As baseline algorithm, we implemented a simple version of a batched gradient descent. Since all considered objectives are convex, we can expect such an algorithm to find a good solution in a reasonable time. The sublinear algorithm runs Algorithm 9 before calling the batched gradient descent. Note that our point is not to find a good solution, but to compare a given algorithm ran on the full dataset or simply on the small set of points generated by Algorithm 9.

We selected two data sets from the UCI repository [66], both of which are under the Creative Commons license. The first data set is the 3D Road Network data set from [111]. It consists of elevation information with the attributes longitude, latitude and altitude. The total number of points is 434,874. For this data set, we considered all powers from $z = 3$ to 7. The second data set is the USCensus data set, consisting of the records from a 1990 census. The total size of the data set was 2,458,285 samples, each with 68 attributes. For this data set, we considered the powers $z = 3, 4, 5$.

The results essentially confirmed that the sublinear algorithm succeeded in finding a good candidate solution in a fraction of the time as batch gradient descent for essentially all considered problems. We give more details in Section 10.7.

10.1.3 Related Work

Sublinear Approximation for Clustering: A number of sublinear algorithm are known for clustering problems. For k-Median, under the constraint that the input space has a small diameter, a constant factor approximation is known [60]. [26] proposed a different set of conditions under which a sublinear algorithm for k-median and k-means exists. Other approximations, with different constraint are also known: for instance, [137] give an algorithm achieving a $O(1)$-approximation in time $\text{poly}(k/\varepsilon)$ for discrete metrics, when each cluster has size $\Omega(n\varepsilon/k)$. For the 1-median problem, this assumption is always satisfied, and their algorithm gives a constant factor approximation in constant running time. The algorithm by [49] produces a $(1+\varepsilon)$-approximation in time $\min(nd\log^3(n/\varepsilon), d/\varepsilon^2)$ for Euclidean spaces of dimension d. [63, 64] and [47]

showed how to obtain sublinear algorithms for the minimum enclosing ball problem assuming that either the algorithm is allowed to drop a fraction of the points, or with an additive error. For the unconstrained version of the problem, no sublinear time algorithm is possible. For the k-means problem, [11] showed how to approximate the k-means++ algorithm in sublinear time. To our knowledge, no algorithm is known for higher distance powers z.

10.1.4 Preliminaries

As always, for any set P and candidate solution c, we define $\text{cost}(P, c) = \sum_{p \in P} \|p - c\|^z$. If the set is clear from context, we simply write $\text{cost}(c)$.

To show that our sampled set verifies nice properties, we will rely on the notion of VC-dimension. We defined it already in Definition 9.18, but we briefly recall the definition here. Note however that we will never need to manipulate VC dimension directly: we will only need Lemma 10.4, and a few classical bounds on the VC dimension of range spaces.

▶ **Definition 10.3.** Let X be a ground set, and $\mathcal{R} \subset \mathcal{P}(X)$. We say that $(X, \mathcal{R})$ is a *range space*.
The VC-dimension of a range space $(X, \mathcal{R})$ is the largest d such that, for some $S \subseteq X$ with $|S| = d$, $|\{R \cap S \mid R \in \mathcal{R}\}| = 2^d$. ◀

In this chapter, we will need a notion slightly more precise than the ε-set-approximation defined in Chapter 9:

▶ **Lemma 10.4 ([121]).** Given a range space $(X, \mathcal{R})$ with VC-dimension d, constants $\varepsilon, \delta, \eta$, and a uniform sample $S \subset X$ of size at least $O(\frac{1}{\eta \cdot \varepsilon^{-2}}(d \log 1/\eta + \log 1/\delta))$, we have for all ranges $R \in \mathcal{R}$ with $|X \cap R| \geq \eta \cdot |X|$

$$\left| \frac{|X \cap R|}{|X|} - \frac{|S \cap R|}{|S|} \right| \leq \varepsilon \cdot \frac{|X \cap R|}{|X|}$$

and for all ranges $R \in \mathcal{R}$ with $|X \cap R| \leq \eta \cdot |X|$

$$\left| \frac{|X \cap R|}{|X|} - \frac{|S \cap R|}{|S|} \right| \leq \varepsilon \cdot \eta$$

with probability at least $1 - \delta$. ◀

This lemma gives stronger guarantees than ε-set-approximation, in particular for sets with $|X \cap R| \leq \eta |X|$: an ε-set-approximation of size $\tilde{O}(d/\varepsilon^2)$ preserves their size up to an additive ε, compared to $\varepsilon\eta$ here. To get the same guarantee using Theorem 9.21, one would need a sample size $\tilde{O}\left(d/(\varepsilon\eta)^2\right)$, instead of $\tilde{O}\left(d/(\varepsilon^2\eta)\right)$ using Lemma 10.4.

The only range spaces we will consider are induced by Euclidean rings centered around

a few fixed point.

For instance, consider the range space induced by Euclidean rings centered around a single point. The VC dimension induced of this range space is 3, which seems to be a well known fact, although we could not find a reference. For completeness, we add here a brief proof.

Let us consider the range space induced by Euclidean rings centered around a single point p. A range R induced by ring of radius (r_1, r_2) is the set of all points at distance between r_1 and r_2 from p, i.e. $R = \{q \in X \mid \|p - q\| \in [r_1, r_2]\}$. We show that for any set of three points, we cannot generate all possible dichotomies, i.e. for any point set S of size 3 we have $|\{R \cap S \mid R \in \mathcal{R}\}| < 8$. If two points have the same distance from p, then it is not possible to define a range that contains one point and not the other. If all points have different distance from p, it is not possible to define a range that contains the furthest point and the closest one, without the third point. Hence, the VC dimension of that range space is at most 3.

Instead of using our common definition of coreset, we will relax it as follows.

▶ **Definition 10.5.** Let P be a set of points in $\mathbb{R}^d$. We say that Ω is a weak ε-coreset if for some $\alpha \in [0, 1]$ any point satisfying $\sum_{p\in\Omega} w_p \cdot \|p - c'\|^z \leq (1 + \alpha \cdot \varepsilon)\operatorname{argmin}_{c\in\mathbb{R}^d} \sum_{p\in\Omega} w_p \cdot \|p - c\|^z$ also satisfies $\sum_{p\in P} \|p - c'\|^z \leq (1 + \varepsilon)\operatorname{argmin}_{c\in\mathbb{R}^d} \sum_{p\in P} \|p - c\|^z$. ◀

The difference between the two notions is that ε-coresets give a guarantee for all candidate centers, whereas the weak coreset guarantee only applies for solutions close to optimum. For our sublinear application, we will be satisfied with a weak coreset guarantee only: this is enough to extract a $(1 + \varepsilon)$-approximation. To highlight the difference between the two notions, we call ε-coreset *strong coresets*.

10.2 Sublinear Algorithm and Analysis

We first describe an algorithm that succeeds with constant probability. This probability can be amplified (non-trivially) using independent repetition, as we will see in Section 10.5. In the following we will use parameters n_{ring} and η that depends on ε which we will specify later, and δ such that the target success probability is $1 - \delta$. α is the approximation ratio of $\mathfrak{a}$. We let P be the set of input points.

Our goal is to show that the outcome Ω of Algorithm 9 satisfies a weak coreset guarantee. Specifically, we will show that Ω has the property that for all points $P' \subseteq P$ that are not too far from $\mathfrak{a}$ (as defined in Lemma 10.7), the weight function defined in Algorithm 9 is such that Ω is a strong coreset for P'. We then show that a strong coreset for P' is also weak coreset for P, i.e. we preserve the cost of all points in P',

Algorithm 9 Sublinear Algorithm for Power Means

1. Sample a random point $\mathfrak{a}$.
2. Sample a set Ω of $T = O(\alpha\varepsilon^{-z-5} \cdot \text{polylog}(\varepsilon^{-1}\delta^{-1}))$ points uniformly at random.
3. Compute the maximum distance Δ such that there exist $2/3 \cdot \varepsilon \cdot \eta \cdot |\Omega|$ points with distance at least Δ from $\mathfrak{a}$. Discard all points at distance greater than Δ.
4. Define rings R_i such that $R_i \cap \Omega$ contains all the points at distance $(\Delta \cdot 2^{-i}, \Delta \cdot 2^{-i+1}]$ from $\mathfrak{a}$, with $i = \{1, \dots, n_{\text{ring}}\}$.
5. If $|R_i \cap \Omega| < \varepsilon\eta \cdot T$, remove all points in $R_i \cap \Omega$ from Ω.
6. Define $\hat{R}_i = n \cdot \frac{|R_i \cap \Omega|}{|\Omega|}$. Weight the points in $R_i \cap \Omega$ by $\frac{\hat{R}_i}{|R_i \cap \Omega|} = \frac{n}{|\Omega|}$.
7. Solve the problem on the (weighted) set Ω.

and we preserve the optimum for P.

Our key lemma is an equivalent to Lemma 9.24: in each ring that contain enough points, the sample Ω is indeed a coreset with some good probability. Removing the far points allows us to call that result on only few rings: in turn, the probability of success of all the calls is high. We start by showing that key lemma – we will see later that the assumptions on $|\Omega|$ are satisfied.

▶ **Lemma 10.6.** Let Ω be the output of Algorithm 9 such that $|\Omega| \geq \eta^{-1} \cdot \varepsilon^{-5} \cdot \log(n_{\text{ring}}/\delta)$, and R_i be a ring such that $|R_i \cap \Omega| \geq \varepsilon\eta|\Omega|$. Then, with probability $1 - \delta/n_{\text{ring}}$, it holds that $\Omega \cap R_i$ is a strong coreset for R_i: namely,

$$\left| \sum_{p \in R_i} \|p - c\|^z - \frac{|R_i|}{|\Omega \cap R_i|} \sum_{p \in \Omega \cap R_i} \|p - c\|^z \right| \leq \varepsilon \cdot \left(\sum_{p \in R_i} \|p - c\|^z + \|p - \mathfrak{a}\|^z \right)$$

◀

Before proving the lemma, note that the weights $\frac{|R_i|}{|\Omega \cap R_i|}$ cannot be directly computed in sublinear time, as we do not know $|R_i|$. We will show later that the weights $\frac{\hat{R}_i}{|\Omega \cap R_i|}$ used instead in the algorithm are a good enough approximation to them, and that the points in $\Omega \cap R_i$ with weights $\frac{\hat{R}_i}{|\Omega \cap R_i|}$ still form a coreset.

Proof of Lemma 10.6. By assumption of the lemma, we have that $|R_i \cap \Omega| > \varepsilon^{-4} \log(n_{\text{ring}}/\delta)$.

As the algorithm only uses uniform sampling, projecting points or not does not change the output's distribution. Hence, we may assume that the points are in a $O(\varepsilon^{-2})$-dimensional space, using results from Section 7.6. Let $\mathcal{B}$ be the set of balls in that space. That way, standard results (for instance Appendix B of [12]) ensures the set system $(R_i, \mathcal{B})$ has VC-dimension at most $O(\varepsilon^{-2})$.

Hence, by Lemma 10.4, $\Omega \cap R_i$ is an ε-set-approximation for R_i with probability $1 - \delta/n_{\text{ring}}$: applying Lemma 9.24 directly concludes. □

Pruning Lemmas We give a brief explanation of the parameters: α is the approximation factor of the initial solution, n_{ring} is such that points at distance closer than $2^{-n_{\text{ring}}} d_{\text{far}}$ can be merged to the center (see Lemma 10.9), and η is such that rings with less than an $\varepsilon\eta$-fraction of the points can be discarded (see Lemma 10.8)

Before turning to the pruning lemmas, we explain briefly what is at stake. What Lemma 10.7 essentially says is that points with $\|p - \mathfrak{a}\|^z > d_{\text{far}} \cdot 4^z \cdot \frac{2\alpha \cdot \text{OPT}}{n}$ do not really influence the position of the optimal center. Unfortunately, we are not given a knowledge of OPT a priori: It is therefore not possible to merely discard those points. This is why step 3 of Algorithm 9 computes a value Δ that seeks to estimate $d_{\text{far}} \cdot 4^z \cdot \frac{2\alpha \cdot \text{OPT}}{n}$.

The number of points at distance more than $d_{\text{far}} \cdot 4^z \cdot \frac{2\alpha \cdot \text{OPT}}{n}$ from $\mathfrak{a}$ is at most n/d_{far}. Combining this observation with Lemma 10.4 ensures step 3 indeed discards all those points. It may however discard slightly more: we show nonetheless in Lemma 10.8 that we can safely discard a fraction of the input, without changing neither the position of the optimal center. This fact is also helpful to deal with rings that do not contain enough points to be well enough represented by the uniform sample.

Finally, to reduce still the number of rings, we show in 10.9 that all points that are very close to $\mathfrak{a}$ can also be discarded. We will combine those results to prove Theorem 10.1.

▶ **Lemma 10.7.** Suppose we are given an α-approximate center $\mathfrak{a}$. Let $B(\mathfrak{a}, r)$ be the ball centered at $\mathfrak{a}$ with radius $r = 4 \cdot \left(\frac{2\alpha \cdot \text{OPT}}{n}\right)^{1/z}$. Then the following two statements hold.

1. Any α-approximate center is in $B(\mathfrak{a}, r)$.
2. For any two points $c, c' \in B(\mathfrak{a}, r)$ and for any point p with $\|p - \mathfrak{a}\|^z > d_{\text{far}} \cdot r^z$ with $d_{\text{far}} = \varepsilon^{-z} \cdot (12z)^z$, we have $\|p - c\|^z \leq (1 + \varepsilon) \cdot \|p - c'\|^z$.

◀

Proof. For the first claim we consider a point c not in $B(\mathfrak{a}, r)$ and show that c cannot be an α-approximate center.

The average cost of the points when using $\mathfrak{a}$ as a center is $\alpha \cdot \frac{\text{OPT}}{n}$. Hence, by Markov's inequality, at least half of the points of P lie in $B(\mathfrak{a}, r/4)$. Furthermore, by choice of c and the triangle inequality, we have $\|p - c\| > 2 \cdot \|p - \mathfrak{a}\|$ for any point $p \in B(\mathfrak{a}, r/4)$. Hence, the cost of clustering all the points in $P \cap B(\mathfrak{a}, r/4)$ to c is at least $n/2 \cdot (2 \cdot r)^z \geq \alpha \cdot \text{OPT}$.

For the second claim, let $c, c' \in B(\mathfrak{a}, r)$ and p with $\|p - \mathfrak{a}\|^z \geq d_{\text{far}} \cdot r^z$. We first note that $\|p - c'\| \geq \|p - \mathfrak{a}\| - \|\mathfrak{a} - c'\| \geq d_{\text{far}}^{1/z} \cdot r - 2r = (d_{\text{far}}^{1/z} - 2)r$, which yields the inequality

$$r \leq \|p - c'\| \cdot \frac{1}{d_{\text{far}}^{1/z} - 2} \tag{10.1}$$

We then have

$$\begin{aligned}
& \|p-c\|^z \\
(\textit{Lem. 1.2}) \quad & \leq (1+\varepsilon/2z)^{z-1}\|p-c'\|^z + \left(\frac{\varepsilon+2z}{\varepsilon}\right)^{z-1} \cdot \|c-c'\|^z \\
& \leq (1+\varepsilon/2)\cdot\|p-c'\|^z + \left(\frac{3z}{\varepsilon}\right)^{z-1}(2r)^z \\
(\textit{Eq. 10.1}) \quad & \leq (1+\varepsilon/2)\cdot\|p-c'\|^z + \left(\frac{3z}{\varepsilon}\right)^{z-1} 2^z \cdot \|p-c'\|^z \left(\frac{1}{d_{\text{far}}^{1/z}-2}\right)^z \\
& \leq (1+\varepsilon/2)\cdot\|p-c'\|^z + \left(\frac{3z}{\varepsilon}\right)^{z-1} 4^z \cdot \|p-c'\|^z \cdot d_{\text{far}}^{-1} \\
(\text{Choice of } d_{\text{far}}) \quad & \leq (1+\varepsilon/2)\cdot\|p-c'\|^z + \varepsilon/2 \cdot \|p-c'\|^z \\
& \leq (1+\varepsilon)\cdot\|p-c'\|^z
\end{aligned}$$

□

To allow the removal of a small fraction of points, we have the following lemma.

▶ **Lemma 10.8.** Suppose that $\mathfrak{a}$ that is an α-approximate solution. Let $p \in P$ with $\|p-\mathfrak{a}\| \leq 4(d_{\text{far}} \cdot \frac{\alpha \cdot \text{OPT}}{n})^{1/z}$, where d_{far} is given by Lemma 10.7. Then, for any candidate solution c

$$\|p-c\|^z \leq 8^{z+1} \cdot d_{\text{far}} \cdot \alpha \cdot \frac{\sum_{p\in P}\|p-c\|^z}{n}.$$ ◀

Proof. We first require a bound on $\|\mathfrak{a}-c\|^z$. Using Lemma 1.2, we have

$$\begin{aligned}
n \cdot \|\mathfrak{a}-c\|^z & \leq 2^z \sum_{p\in P}\|p-\mathfrak{a}\|^z + \|p-c\|^z \leq \alpha \cdot 2^{z+1} \cdot \sum_{p\in P}\|p-c\|^z \\
\Rightarrow \|\mathfrak{a}-c\|^z & \leq \alpha \cdot 2^{z+1} \cdot \frac{\sum_{p\in P}\|p-c\|^z}{n} \qquad (10.2)
\end{aligned}$$

Therefore with another application of Lemma 1.2

$$\begin{aligned}
\|p-c\|^z & \leq 2^z \cdot (\|p-\mathfrak{a}\|^z + \|\mathfrak{a}-c\|^z) \\
(\textit{Eq. 10.2}) \quad & \leq 2^z \cdot \left(4^z d_{\text{far}} \cdot \frac{\alpha \cdot \text{OPT}}{n} + \alpha \cdot 2^{z+1} \cdot \frac{\sum_{p\in P}\|p-c\|^z}{n}\right) \\
& \leq \alpha \cdot d_{\text{far}} \cdot 8^{z+1} \cdot \frac{\sum_{p\in P}\|p-c\|^z}{n}.
\end{aligned}$$

□

Finally, the following lemma deals with the points that are very close to $\mathfrak{a}$.

▶ **Lemma 10.9.** Suppose that $\mathfrak{a}$ is an α-approximate solution. Let $P_{near} \subset P$ be a set of points with cost at most $(\varepsilon/(\alpha 5z))^z \cdot \frac{\mathrm{OPT}}{n}$. Let $\hat{P}$ such that $\hat{P} \in (1 \pm \varepsilon)|P_{near}|$. Then for any candidate solution c we have

$$\left| \hat{P} \cdot \|\mathfrak{a} - c\|^z - \sum_{p \in P_{near}} \|p - c\|^z \right| \leq \varepsilon/\alpha \cdot \left(\sum_{p \in P_{near}} \|p - c\|^z + \mathrm{OPT} \right). \blacktriangleleft$$

Proof. We first prove the result for $\hat{P} = |P_{near}|$, the claim for an estimation of $|P_{near}|$ is a simple corollary. We have, using Lemma 1.2:

$$\begin{aligned}
\left| \sum_{p \in P_{near}} (\|p - c\|^z - \|\mathfrak{a} - c\|^z) \right| &\leq \sum_{p \in P_{near}} |\|p - c\|^z - \|\mathfrak{a} - c\|^z| \\
&\leq \sum_{p \in P_{near}} \left(\frac{\varepsilon}{2\alpha} \cdot \|p - c\|^z + \left(\frac{\alpha 5z}{\varepsilon} \right)^{z-1} \|p - \mathfrak{a}\|^z \right) \\
&\leq \sum_{p \in P_{near}} \left(\frac{\varepsilon}{2\alpha} \cdot \|p - c\|^z + \left(\frac{\alpha 5z}{\varepsilon} \right)^{z-1} \left(\frac{\varepsilon}{\alpha 5z} \right)^z \frac{\mathrm{OPT}}{n} \right) \\
&\leq \varepsilon/\alpha \cdot \left(\sum_{p \in P_{near}} \|p - c\|^z + \mathrm{OPT} \right)
\end{aligned}$$

For an approximation to $|P_{near}|$ we now merely add an additional additive error $\varepsilon \cdot \sum_{p \in P_{near}} \|p - c\|^z \leq \varepsilon(\varepsilon/(5\alpha z))^z \mathrm{OPT}$ to the difference of the two terms. □

Finally, the next lemma is key to implement those idea in sublinear time: it shows that the sample Ω can be used to estimate the number of points in any ring.

▶ **Lemma 10.10.** Let $\mathfrak{a}$ be a point that is an α-approximation and let Ω be a uniform sample consisting of $O\left(\alpha \cdot \eta^{-1} \cdot \varepsilon^{-5} \mathrm{polylog}(\varepsilon^{-1} \cdot \delta^{-1})\right)$ points. Then with probability at least $1 - \delta$ for all rings R_i,

$$\begin{aligned}
|R_i \cap P| - \varepsilon \cdot \max(n \cdot \eta, |R_i \cap P|) \leq \frac{|R_i \cap \Omega|}{|\Omega|} \cdot n \\
\leq |R_i \cap P| + \varepsilon \cdot \max(n \cdot \eta,\ |R_i \cap P|).
\end{aligned}$$

Furthermore, let Δ as in the algorithm, i.e., such that $\frac{2}{3} \cdot \varepsilon \cdot \eta \cdot |\Omega| \leq \left| \Omega \setminus (B(\mathfrak{a}, \Delta) \cap \Omega) \right|$. Then $\Delta < \left(3(\varepsilon\eta)^{-1} \cdot \frac{\alpha \cdot \mathrm{OPT}}{n}\right)^{1/z}$, and there are at most $\frac{5}{3} \cdot \varepsilon\eta n$ points in P at distance more than Δ from $\mathfrak{a}$. ◀

Proof. We consider the range space induced by rings centered around $\mathfrak{a}$. This range space has VC dimension at most 3, as explained previously. Hence for our choice of $|\Omega|$, Lemma 10.4 ensures that we have approximated the cardinality of all rings up to

the additive error $\varepsilon \cdot \max(\eta \cdot n, |R_i \cap P|)$ with probability at least $1 - \delta$, which proves the first and last claims.

For the second claim, let Δ as in the algorithm. By Lemma 10.4, we have $\big|\Omega \setminus (B(\mathfrak{a}, \Delta) \cap \Omega)\big| \cdot \frac{n}{|\Omega|} \leq (1+\varepsilon)\big|P \setminus (B(\mathfrak{a}, \Delta) \cap P)\big|$. Hence, we have

$$\big|P \setminus (B(\mathfrak{a}, \Delta) \cap P)\big| \geq \frac{2}{3} \cdot \varepsilon \cdot \eta \cdot |\Omega| \cdot \frac{n}{|\Omega|(1+\varepsilon)} \geq \frac{\varepsilon \cdot \eta \cdot n}{3}$$

Using Markov's inequality, we now know that the number of points with cost $3(\varepsilon\eta)^{-1} \cdot \frac{\alpha \cdot \mathrm{OPT}}{n}$ is at most $\frac{\varepsilon \cdot \eta \cdot n}{3}$. This implies $\Delta \leq \left(3(\varepsilon\eta)^{-1} \cdot \frac{\alpha \cdot \mathrm{OPT}}{n}\right)^{1/z}$. □

10.3 Proof of Theorem 10.1

We first conclude the proof of Theorem 10.1, with only a success probability of 9/10. This somewhat low probability is due to the computation of a constant-factor approximation, and we will show in Section 10.5 how to boost this probability.

Proof. We start by specifying our parameters: the approximation is set to be $\alpha = 20^z$. To prune the far points, we set $d_{\mathrm{far}} = (12z/\varepsilon)^z$, as in Lemma 10.7. Finally, we pick n_{ring} and η such that $\eta = \frac{1}{8^{z+1}\alpha \cdot n_{\mathrm{ring}} \cdot d_{\mathrm{far}}}$ and $6 \cdot 2^{-zn_{\mathrm{ring}}+z} \leq \varepsilon$. This is possible for $n_{\mathrm{ring}} = O_z(\log(1/\varepsilon))$ and $\eta \in O_z(\varepsilon^z/\log(1/\varepsilon))$. With those choices, assumptions of Lemma 10.6 are satisfied: $eta^{-1} \cdot \varepsilon^{-5} \cdot \log(n_{\mathrm{ring}}/\delta) = O\left(\alpha\varepsilon^{-z-5} \cdot \mathrm{polylog}(\varepsilon^{-1}\delta^{-1})\right)$.

Let c_{OPT} be the optimal $(1, z)$-center. We start by showing that the initial sampled point $\mathfrak{a}$ is an α-approximation with constant probability. Indeed, there are at most $n/10$ points with cost more than $10\frac{\mathrm{OPT}}{n}$: hence, it holds with probability at least 9/10 that $\|\mathfrak{a} - c_{\mathrm{OPT}}\|^z \leq 10 \cdot \frac{OPT}{n}$. In that case,

$$\begin{aligned}\sum_{p \in P} \|p - \mathfrak{a}\|^z &\leq \sum_{p \in P} 2^z(\|p - c_{\mathrm{OPT}}\|^z + \|c_{\mathrm{OPT}} - \mathfrak{a}\|^z) \\ &\leq 2^z \cdot OPT + 2^z \cdot 10\mathrm{OPT} \leq 20^z\mathrm{OPT},\end{aligned}$$

which shows that $\mathfrak{a}$ is an α-approximation.

Let Ω be the set of points sampled and pruned by the algorithm. We wish to show that any $(1+\varepsilon)$-approximation for Ω is also one for P. First, note that Lemma 10.7 shows that it is enough to compute an approximate solution for the set P' consisting of points that are at distance less than $4\left(\frac{2\alpha d_{\mathrm{far}}\mathrm{OPT}}{n}\right)^{1/z}$ for $\mathfrak{a}$.

Recall that the ring R_i consists of points at distance $(2^{-i}\Delta, 2^{-i+1}\Delta]$ from $\mathfrak{a}$. Additionally, we group all remaining points in a single ring, and define $R_{n_{\mathrm{ring}}}$ to be the set of points at distance less than $2^{-n_{\mathrm{ring}}}\Delta$ from $\mathfrak{a}$, i.e., $R_{n_{\mathrm{ring}}} := \{p \in \Omega : \|p - \mathfrak{a}\| \leq 2^{-n_{\mathrm{ring}}}\Delta\}$

We denote as in the algorithm $\hat{R}_i$ either $\frac{n \cdot |R_i \cap \Omega|}{|\Omega|}$, if $|R_i \cap \Omega| \geq \varepsilon \cdot \eta \cdot |\Omega|$ (which is a a $(1 \pm \varepsilon)$-estimate of the size of $|R_i|$), or we set $\hat{R}_i = 0$ if $|R_i \cap \Omega| < \varepsilon \cdot \eta \cdot |\Omega|$.

We show aim at showing that for any candidate solution c', we have

$$\left| \sum_{p \in P'} \|p - c'\|^z - \left(\sum_{i \leq n_{\text{ring}}-1} \frac{\hat{R}_i}{|R_i \cap \Omega|} \sum_{p \in R_i \cap \Omega} \|p - c'\|^z + \frac{\widehat{R_{n_{\text{ring}}}}}{|R_{n_{\text{ring}}} \cap \Omega|} \|\mathfrak{a} - c'\|^z \right) \right| \leq \varepsilon \cdot \sum_{p \in P'} \|p - c'\|^z. \tag{10.3}$$

Hence, computing a $(1+\varepsilon)$-approximate solution on the set Ω will give $(1+\varepsilon)$-approximate solution for P', which is also one for P following Lemma 10.7. We will formalize this after showing Eq. (10.3).

To prove Eq. (10.3), we proceed ring by ring.

Close points. First, for the points very close with $i = n_{\text{ring}}$. In the case where $\widehat{R_{n_{\text{ring}}}} = 0$, the number of points in $P' \cap R_{n_{\text{ring}}}$ is at most $2\varepsilon\eta n$. Furthermore, the upper bound on Δ from Lemma 10.10 ensures that the cost of points in R_i is at most $\left(2^{-n_{\text{ring}}+1}\Delta\right)^z \leq 2^{-zn_{\text{ring}}+z} \frac{3}{\varepsilon\eta} \frac{\text{OPT}}{n}$. Using the choice of n_{ring}, we conclude that $\sum_{p \in P' \cap R_{n_{\text{ring}}}} \|p - c'\|^z \leq 6 \cdot 2^{-zn_{\text{ring}}+z} \text{OPT} \leq \varepsilon \text{OPT}$.

In the other case where $\widehat{R_{n_{\text{ring}}}} \neq 0$, we can use Lemma 10.9 to bound

$$\left| \sum_{p \in P' \cap R_{n_{\text{ring}}}} \|p - c'\|^z - \hat{R_{n_{\text{ring}}}} \|\mathfrak{a} - c'\|^z \right| \leq \varepsilon/\alpha \cdot \left(\sum_{p \in P_{near}} \|p - c'\|^z + \text{OPT} \right) \leq \varepsilon \cdot \sum_{p \in P} \|p - c'\|^z. \tag{10.4}$$

Far points. Having dealt with the points close to $\mathfrak{a}$, we now deal with those far away. Since P' results in the pruning of P, there is no point in P' at distance more than $4(d_{\text{far}} \cdot \frac{\alpha \cdot OPT}{n})^{1/z}$ from $\mathfrak{a}$. On the other hand, in Ω, the points further away than Δ from $\mathfrak{a}$ are pruned away. Lemma 10.10 ensures that $\Delta < \left(3(\varepsilon\eta)^{-1} \cdot \frac{\alpha \cdot \text{OPT}}{n}\right)^{1/z} \leq 4(d_{\text{far}} \cdot \frac{\alpha \cdot OPT}{n})^{1/z}$. Hence, the contribution of points in $P \setminus P'$ to Equation 10.3 is zero.

We now deal with points moderately far away, namely point at distance between Δ and $4(d_{\text{far}} \cdot \frac{\alpha \cdot \text{OPT}}{n})^{1/z}$ from $\mathfrak{a}$. Since they are pruned away, contribution is zero in Ω. Furthermore, by choice of Δ and Lemma 10.10, we note there are at most $2\varepsilon \cdot \eta \cdot n$ of those. Therefore, their contribution in P' is at most

$$2\varepsilon \cdot \eta \cdot n \cdot 4^z d_{\text{far}} \cdot \frac{\alpha \cdot OPT}{n} \leq 2\varepsilon \text{OPT}. \tag{10.5}$$

Middle points. Finally, we turn our attention to the rings R_i with $i \in \{1, ..., n_{\text{ring}} -$

1}. In the case where $\hat{R}_i = 0$, we have, due to Lemma 10.8:

$$\begin{aligned}\sum_{p\in R_i\cap P} \|p-c\|^z &\le |R_i\cap P|\cdot 8^{z+1}\cdot\alpha\cdot d_{\text{far}}\cdot\frac{\left(\sum_{p\in P}\|p-c\|^z\right)}{n}\\ &\le \varepsilon\eta\cdot 8^{z+1}\cdot\alpha\cdot d_{\text{far}}\cdot\sum_{p\in P}\|p-c\|^z\\ &\le \frac{\varepsilon}{n_{\text{ring}}}\cdot\sum_{p\in P}\|p-c\|^z\end{aligned}$$

Summing over all rings R_{cheap} that have $\hat{R}_i = 0$, we therefore obtain:

$$\left|\sum_{p\in R_{cheap}\cap P}\|p-c\|^z - \frac{\hat{R}_i}{|R_i\cap\Omega|}\sum_{p\in R_i\cap\Omega}\|p-c\|^z\right| \le \varepsilon\sum_{p\in P}\|p-c\|^z. \tag{10.6}$$

Last, Lemma 10.6 ensures that with probability $1-\delta$, for any rings with $|R_i\cap\Omega| > \varepsilon\eta|\Omega|$, with $i\in\{1,...,n_{\text{ring}}-1\}$,

$$\left|\sum_{p\in R_i}\|p-c\|^z - \frac{|R_i|}{|R_i\cap\Omega|}\sum_{p\in R_i\cap\Omega}\|p-c\|^z\right| \le \varepsilon\left(\sum_{p\in R_i}\|p-c\|^z + \|p-\mathfrak{a}\|^z\right)$$

Using that $\hat{R}_i = (1\pm\varepsilon)|R_i|$, we get from the previous equation that

$$\begin{aligned}&\left|\sum_{p\in R_i}\|p-c\|^z - \frac{|\hat{R}_i|}{|R_i\cap\Omega|}\sum_{p\in R_i\cap\Omega}\|p-c\|^z\right|\\ \le\ &\left|\sum_{p\in R_i}\|p-c\|^z - \frac{|R_i|}{|R_i\cap\Omega|}\sum_{p\in R_i\cap\Omega}\|p-c\|^z\right| + \varepsilon\frac{|R_i|}{|R_i\cap\Omega|}\sum_{p\in R_i\cap\Omega}\|p-c\|^z\\ \le\ &(1+\varepsilon)\left|\sum_{p\in R_i}\|p-c\|^z - \frac{|R_i|}{|R_i\cap\Omega|}\sum_{p\in R_i\cap\Omega}\|p-c\|^z\right| + \varepsilon\sum_{p\in R_i}\|p-c\|^z\\ \le\ &2\varepsilon\left(\sum_{p\in R_i}\|p-c\|^z + \|p-\mathfrak{a}\|^z\right)\end{aligned} \tag{10.7}$$

Wrapping up. Summing up Equations 10.4, 10.5, 9 and 10.7 yields a total error of $O(\varepsilon)\cdot\sum_{p\in P'}\|p-c\|^z + \|p-c'\|^z \in O(\varepsilon)\sum_{p\in P'}\|p-c'\|^z$, which concludes the proof of Eq. (10.3).

Consequently, any solution c' has the same cost on P' and on Ω, up to an additive $O(\varepsilon)\sum_{p\in P'}\|p-c'\|^z$. Since we know by Lemma 10.7 that the optimal solution c_{OPT} is in the ball $\beta(\mathfrak{a},r)$, any $(1+\varepsilon)$-approximation c for Ω restricted to lie inside that ball verifies

$$\left|\sum_{p\in P'}\|p-c\|^z - \sum_{p\in P'}\|p-c_{\text{OPT}}\|^z\right| \le O(\varepsilon)\text{OPT}.$$

Using the second claim of Lemma 10.7, we finally get that

$$\left| \sum_{p \in P} \|p - c\|^z - \sum_{p \in P} \|p - c_{\mathrm{OPT}}\|^z \right| \leq O(\varepsilon)\mathrm{OPT},$$

which concludes: to get a $(1+\varepsilon)$-approximation, one can merely compute a $(1+\varepsilon)$-approximate solution on Ω, with weights $\frac{\hat{R}_i}{|R_i \cap \Omega|}$ on points of R_i.

We note here that, conditioned on $\mathfrak{a}$ being a constant-factor approximation, the algorithm works with probability $1 - \delta$. Hence, the bottleneck for boosting success probability is the computation of a constant factor approximation. □

10.4 A Brief Note on the MPC Algorithm From Section 4.4.2

Algorithm 5 is very similar to Algorithm 9. A huge difference however is that we are given a center and a cost to cluster to that center. Hence, the close and far points can be pruned without the need of estimating Δ as in Algorithm 9. In turn, there are $n_{\mathrm{ring}} = O(\log n)$ rings, instead of a constant number – this can be removed, but we use it to simplify the algorithm. Another difference is that the initial solution is a $O(\mathrm{polylog} n)$-approximation: hence, the size of the sample Ω must be of that order. The parameters we chose for Algorithm 5 are actually the same as in Algorithm 9: as we are looking for constant approximation for 1-median only, we can set $\varepsilon = 1/2$, $z = 1$, $\alpha = \mathrm{polylog}(n)$. Then, t is the size of Ω from Lemma 10.10, which is $\mathrm{polylog}(n)$, d_{close} is $1/10$ (to match Lemma 10.9), and r_{small} is $\varepsilon\eta$ – which is $O(1/\log(n))$.

Furthermore, in Algorithm 9, the weights are uniform, and can be discarded in order to compute a solution, as in Algorithm 5.

Then, the proof of Theorem 10.1 show that Algorithm 5 indeed computes a $O(1)$-approximation to the median, using the parameters specified above.

10.5 Probability Amplification

While the aforementioned algorithm is guaranteed to produce a $(1+\varepsilon)$-approximation with constant probability, amplifying this is non-trivial. Indeed, when running the algorithm multiple times, it is not clear how to distinguish a successful run from an unsuccessful one. The main issue in amplifying the probability lies in the initial solution $\mathfrak{a}$, as any invocation of Lemma 10.4 allows us to control the failure probability of

the remaining part of the algorithm. The simplest way to achieve a success probability $1-\delta$ is to condition on $\|\mathfrak{a}-c\|^z \leq \delta \cdot \frac{\mathrm{OPT}}{n}$. Unfortunately, this makes ε dependent on δ, which significantly increases the sampling complexity.

Instead, we use the following algorithm. We sample $m \in O(\log 1/\delta)$ points $\mathfrak{a}_1, \ldots \mathfrak{a}_m$ uniformly at random. For each point, we additionally sample $O(\varepsilon^{-2}(\log 1/\varepsilon + \log 1/\delta))$ points $\Omega_{\mathfrak{a}_i}$. We claim that the point $\mathfrak{a}_i$ with smallest estimated median distance to it is a 15^z approximation.

▶ **Lemma 10.11.** Given query access to P, we can identify with probability $1-\delta$ a 15^z-approximate solution using $O(\varepsilon^{-2}(\log 1/\varepsilon + \log 1/\delta)\log 1/\delta)$ samples. ◀

To achieve an overall success probability of $1-\delta$, we only need to sample from non-cheap rings. Thereafter, a higher probability bound can be obtained by in Lemma 10.6 by simply increasing the sample size by constant factor – following the application of Lemma 10.4.

Proof. Let c be the optimal center.

With probability at least $1/2$, a random point $\mathfrak{a}_i$ satisfies

$$\|\mathfrak{a}_i - c\|^z \leq 2^z \frac{\mathrm{OPT}}{n}.$$

Furthermore,

$$\sum_{p\in P} \|p-\mathfrak{a}_i\|^z \leq \sum_{p\in P} 2^{z-1}\cdot(\|p-c\|^z + \|\mathfrak{a}_i - c\|^z) \leq 2^z \cdot \mathrm{OPT}.$$

Therefore, when sampling $\log 1/\delta$ points, we will have sampled a 2^z approximate solution with probability at least $1-\delta$.

Now since the range space induced by unit Euclidean balls centered around $\mathfrak{a}$ has VC dimension 3, we can estimate the number of points for any given radius up to an additive error of $n/10$ with Lemma 10.4. Hence, with probability $1-\delta$, for every 2^z-approximate solution $\mathfrak{a}_i$, the estimated number of points in $B(\mathfrak{a}_i, 2\left(4\frac{\mathrm{OPT}}{n}\right)^{1/z})$ will be at least $3n/4 - n/10$, as there are at least $3n/4$ points in that ball. In particular, the estimated median distance to $\mathfrak{a}_i$ is at most $2\left(4\frac{\mathrm{OPT}}{n}\right)^{1/z}$.

Conversely, if some point $\mathfrak{a}_i$ is not 15^z approximate, we can show that the estimated

number of points in $B(\mathfrak{a}_i, 2\left(4\frac{\mathrm{OPT}}{n}\right)^{1/z})$ is small. Indeed:

$$\begin{aligned}
&\sum_p \|p-\mathfrak{a}_i\|^z > 15^z \sum_p \|p-c\|^z \\
\Rightarrow\quad &\left(\sum_p \|p-\mathfrak{a}_i\|^z\right)^{1/z} > 15\left(\sum_p \|p-c\|^z\right)^{1/z} \\
\Rightarrow\quad &\left(\sum_p \|c-\mathfrak{a}_i\|^z\right)^{1/z} > 14\left(\sum_p \|p-c\|^z\right)^{1/z} \\
\Rightarrow\quad &\|c-\mathfrak{a}_i\| > 14\left(\frac{\mathrm{OPT}}{n}\right)^{1/z}
\end{aligned}$$

Hence, any point at distance at most $10\left(\frac{\mathrm{OPT}}{n}\right)^{1/z}$ from $\mathfrak{a}_i$ is at distance more than $\left(4\frac{\mathrm{OPT}}{n}\right)^{1/z}$ of c. Since at least $\frac{3n}{4}$ points lie in $B(c, \left(4\frac{\mathrm{OPT}}{n}\right)^{1/z})$, the median distance to $\mathfrak{a}_i$ is therefore at least $10\left(\frac{\mathrm{OPT}}{n}\right)^{1/z} \geq 2\left(5\frac{\mathrm{OPT}}{n}\right)^{1/z}$. Hence, the estimated median distance is strictly more than $2\left(4\frac{\mathrm{OPT}}{n}\right)^{1/z}$ with probability $1-\delta$.

Conditioned on having a 2^z approximate solution in our sample, the point $\mathfrak{a}_i$ with smallest estimated median distance to it is a 15^z approximation. □

10.6 Lower bound

The goal of that section is to prove Theorem 10.2, i.e., that any $(1+\varepsilon)$-approximation algorithm must sample ε^{-z+1} points. For that, we analyze a very simple example on the line, where many points are located at 0 and a tiny fraction located at 1 concentrates all the cost.

Proof of Theorem 10.2. Consider the instance $\mathcal{I}$ on the 1-dimensional line where n points are located at 0 and $\varepsilon^{z-1}n$ points are located at 1. Intuitively, we show that any approximation algorithm on $\mathcal{I}$ must sample at least a point at 1, and so must sample at least $\varepsilon^{-z+1}n$ points.

For simplicity, we rescale the instance so that $n=1$. The optimal solution is $\mathrm{OPT}_{\mathcal{I}} = \inf x^z + \varepsilon^{z-1}(1-x)^z$, and the optimal center is such that the derivative of the objective function is zero:

$$\frac{\partial}{\partial x}\left(x^z + \varepsilon^{z-1}(1-x)^z\right) = (z-1)\left(x^{z-1} - (\varepsilon-\varepsilon x)^{z-1}\right)$$

so the optimal value is for x_{OPT} such that $(z-1)\left(x_{\mathrm{OPT}}^{z-1} - (\varepsilon-\varepsilon x_{\mathrm{OPT}})^{z-1}\right) = 0$, which

is $x_{\mathrm{OPT}} = \frac{\varepsilon}{\varepsilon+1}$. Hence,

$$\begin{aligned}\mathrm{OPT} &= \left(\frac{\varepsilon}{\varepsilon+1}\right)^z + \varepsilon^{z-1}\left(1-\frac{\varepsilon}{\varepsilon+1}\right)^z \\ &= \frac{\varepsilon^{z-1}}{(\varepsilon+1)^z}(\varepsilon+1) = \left(\frac{\epsilon}{1+\varepsilon}\right)^{z-1}.\end{aligned}$$

Since the cost of the solution having a center at 0 is ε^{z-1}, is it bigger than $(1+\varepsilon)$OPT: indeed,

$$(1+\varepsilon)\left(\frac{\epsilon}{1+\varepsilon}\right)^{z-1} < \varepsilon^{z-1}. \tag{10.8}$$

Now, consider the instance $\mathcal{I}$ and the instance $\mathcal{I}'$ that has n points located at 0. let $\mathcal{A}$ be an algorithm that, with probability more than 4/5, computes a $(1+\varepsilon)$-approximation for $(1,z)$-clustering.

Assume by contradiction that $\mathcal{A}$ samples less than $\varepsilon^{-z+1}/10$ points. Let X be the random variable counting the number of points located at 1 in that sample: we have $\Pr[X>0] \leq \mathbb{E}[X] \leq 1/10$. So with probability at least 9/10, $\mathcal{A}$ samples only point located at 0: even when that event occures, $\mathcal{A}$ must output a center at a position different than 0 (following Equation 10.8) with some probability p.

Since $\mathcal{A}$ succeeds with probability 4/5 and $X=0$ with probability at least 9/10, we must have have $\frac{9}{1}0p + \frac{1}{1}0 \geq 4/5$, and so $p \geq \frac{7}{9}$.

Hence, when $\mathcal{A}$ samples only points located at 0, it must output a center different from 0 with probability at least 7/9. In particular, on instance $\mathcal{I}'$, $\mathcal{A}$ fails with probability at least 7/9, a contradiction.

So, any algorithm that computes a $(1+\varepsilon)$-approximation for $(1,z)$-clustering with probability more than 4/5 must sample more than $\varepsilon^{-z+1}/10$ points. □

10.7 Experiments

Implementation: We used a variant of Algorithm 9 which now describe. Instead of specifying a desired accuracy, the algorithm has access to m samples picked uniformly at random from the data set. As an α-approximate solution q, the algorithm merely selects a random point. The number of rings considered is set to $n_{\mathrm{ring}} = 20$.

We also estimate $\frac{OPT}{n}$, by sampling another point q' and using $\|q-q'\|^z$ as a (coarse) estimate. We then apply the pruning procedures. Our algorithm chooses $\{100, 200, \ldots, 1000\}$ samples. For each sample size, we repeated the algorithm 10 times and outputted the best center we could find.

Since the objective function is convex, we use a (simple) stochastic gradient descent on both the sample and the full data set to compute a desired center. We iterated

over the data set a total of 10 times. In every iteration, we partitioned the data into random chunks of size $\min(m, 2000)$, and used chunk to perform a gradient step. We did not attempt to optimize the stochastic gradient descent; as our focus is less on solving the problem in the fastest way possible and more on showcasing how the sublinear algorithm can be used to potentially speed up any baseline algorithm.

The algorithms were coded in Python and run on a Intel Core i7-8665U processor with four 2 GHz cores and 32 GByte RAM.

Results Tables with exact figures are given below. Here, we report and interpret the results.

On the Road Network data set, all samples sizes found a nearly optimal solution in at least one of the 10 repetitions, with the largest deviation from the optimum of 4% occurring for 500 samples and the $z = 4$ problem. In addition, the sublinear algorithms all required only a very small amount of time compared to the baseline optimal solution (e.g. a factor of at least 400 quicker for the largest sample size). What is notable is that starting with $z = 5$, the variance in cost of any given sample size increased significantly. Since this occurred regardless of sample size, we attribute this effect to quality of the seeding solution (q in Algorithm 9). The approximation factor of q directly impacts the quality of the subsequent coreset construction, meaning that even with large sample sizes, the algorithm has difficulty to recover. This means that good seeding solution q, for example using Lemma 10.11 is essential.

Processing the USCensus data set was in its entirety was time consuming, running for more than 90 minutes. Constructing the coreset and optimizing it never took more than 14 seconds, however the algorithm did not compute near optimal solutions as was the case for Road Networks data set. For $z = 3$ and $z = 5$ the approximation was still rather small, and tightly concentrated. For an unclear reasons, there exists a larger gap at $z = 4$. While a gap of that magnitude is consistent with the lower bound, the data set does not seem to have a structure similar to said lower bound.

Samples	$z = 3$			$z = 4$			$z = 5$		
	Cost		Time	Cost		Time	Cost		Time
	Min	Avg	Avg	Min	Avg	Avg	Min	Avg	Avg
100	1,306	1,310	6,03	1,907	1,910	7,03	1,527	1,560	7,22
200	1,306	1,311	6,03	1,907	1,911	7,04	1,518	1,554	7,81
300	1,305	1,310	6,30	1,907	1,908	7,59	1,523	1,560	8,65
400	1,305	1,310	6,48	1,907	1,909	7,74	1,521	1,544	9,06
500	1,306	1,309	6,74	1,906	1,908	7,92	1,512	1,547	9,84
600	1,305	1,308	7,06	1,907	1,908	8,21	1,516	1,553	10,39
700	1,307	1,309	7,20	1,907	1,908	8,35	1,516	1,545	11,14
800	1,306	1,309	7,53	1,907	1,908	8,82	1,528	1,547	11,86
900	1,306	1,310	7,60	1,907	1,909	8,98	1,523	1,553	12,43
1k	1,307	1,312	8,04	1,906	1,909	9,6	1,526	1,550	13,26
OPT	1,125	-	5544	1,296	-	5586	1,499	-	5934

Figure 10.1: Overview of cost and running time for the sublinear algorithm on the USCensus data set. Costs scaled by a factor 10^{12} for $z = 3$, 10^{14} for $z = 4$ and 10^{16} for $z = 5$. The variance was extremely small for all values (cost and running time), as indicated by the small gaps between minimum and average. We therefore omitted it from the table. The largest variance (relative to the squared cost) we encountered was for $z = 5$ and 600 samples, where it was still below 0.0005. Running time is given in seconds. The running for the sampling algorithms only considers the time required to sample the points, prune the data set, and run the optimization, i.e. the time required to evaluate the computed solution on the entire data set (which vastly exceeds the given time bounds) is not included.

Samples	$z=3$ Cost			Time	$z=4$ Cost			Time	$z=5$ Cost			Time
	Min	Avg	Var/Avg^2	Avg	Min	Avg	Var/Avg^2	Avg	Min	Avg	Var/Avg^2	Avg
100	4,90	6,67	0,08	0,62	1,79	2,83	0,12	0,62	7,47	10,13	0,35	0,61
200	4,80	6,36	0,05	1,16	1,86	3,85	0,24	1,29	7,37	8,29	0,30	1,16
300	4,79	7,40	0,21	1,74	1,78	2,48	0,16	1,76	7,39	12,44	0,15	1,76
400	4,86	6,89	0,11	2,27	1,79	2,82	0,13	2,37	7,37	12,44	0,51	2,41
500	4,95	7,80	0,04	2,88	1,84	2,88	0,47	2,90	7,39	11,11	0,61	2,89
600	4,84	10,35	0,47	3,42	1,78	2,91	0,76	3,46	7,55	21,52	0,23	3,48
700	4,79	6,67	0,22	3,97	1,80	3,12	0,31	4,10	7,37	16,22	0,60	4,06
800	4,79	6,92	0,19	4,60	1,78	6,20	0,67	4,62	7,38	15,89	0,57	4,71
900	4,79	9,27	0,58	5,12	1,78	3,43	0,69	5,23	7,37	25,93	0,20	5,26
1k	4,81	10,97	0,30	5,65	1,78	4,91	0,62	5,90	7,47	29,68	0,41	5,81
OPT	4,79	-	-	871	1,78	-	-	875	7,37	-	-	877

Samples	$z=6$ Cost			Time	$z=7$ Cost			Time
	Min	Avg	Var/Avg^2	Avg	Min	Avg	Var/Avg^2	Avg
100	3,40	5,38	0,20	0,63	1,59	2,28	0,32	0,61
200	3,32	6,22	0,41	1,20	1,60	2,72	0,46	1,19
300	3,51	11,47	0,91	1,77	1,59	6,20	1,21	1,82
400	3,35	6,54	0,71	2,33	1,59	8,70	4,12	2,36
500	3,54	8,22	1,43	2,89	1,61	2,42	0,33	2,92
600	3,32	6,33	0,85	3,47	1,61	14,01	2,25	3,50
700	3,32	7,86	0,93	4,06	1,60	6,53	2,87	4,10
800	3,35	11,10	2,25	4,63	1,61	5,06	0,76	4,70
900	3,31	8,46	0,29	5,21	1,59	7,49	0,54	5,28
1k	3,34	7,89	0,57	5,98	1,59	2,35	0,16	5,78
OPT	3,31	-	-	885	1,59	-	-	882

Figure 10.2: Overview of cost and running time for the sublinear algorithm on the Road Networks data set. Costs scaled by a factor 10^9 for $z = 3$, 10^{11} for $z = 4$ and $z = 5$, 10^{14} for $z = 6$ and 10^{16} for $z = 7$. The variance was extremely small for running times, so we omit it. Running time is given in seconds. The running time for the sampling algorithms only considers the time required to sample the points, prune the data set, and run the optimization, i.e. the time required to evaluate the computed solution on the entire data set is not included.

Conclusion and Open Questions

During this thesis, we studied the (k, z)-Clustering problem under several aspects.

In the first part, we answered our initial Question 1 from the introduction: we presented an algorithm that runs in near linear time, and computes a very good approximation to the optimum in spaces with bounded doubling dimension. For this, we developed techniques amenable to constrained version of the problem: we showed how to use them to build a *differentially private* algorithm, that has provable guarantee and is very efficient in practice.

In particular, we presented and used the notion of *badly cut* vertices for clustering problems. This allowed us to improve the standard quadtree and split-tree based approach in two ways: first, we reduce the number of portals necessary to a constant, and second, the approach can now work with squared distances. This raises the following (somewhat informal) questions:

> ▶ **Open Question 1.** Could we use the notion of badly cut for other problems? For instance, could we use it to get *linear* time approximation schemes for Traveling Salesperson Problem, Steiner Tree or Steiner Forest? ◀

This would require a new way of dealing with the badly cut vertices, at it would be necessary to connect them nicely with the solution – the equivalent of opening the badly cut centers that we used in this thesis. There already exist near-linear time algorithms for TSP from Gottlieb [89], and for Steiner Tree and Steiner Forest from Bartal and Gottlieb [21]. Studying those through a notion of badly-cut may provide an alternative view on those problems, where split trees are the key tool – as in Euclidean spaces.

In the experiments of Chapter 4, we saw that the quadtree based approached worked surprisingly well. Instead of losing the huge polylog$n$ factor that is promised by the theory, our algorithm is really competitive with the best (non-private) approach. The algorithm is obtained by embedding the input into a quadtree. The worst-case analysis of the distortion of that embedding is the source of the polylog$n$ factor appearing in our theoretical approximation guarantee, that we do not observe in practice. Therefore, the quadtree seems to perform way better in real world instances than in the worst-case, which yields the following question:

▶ **Open Question 2.** Is there a *beyond worst case* analysis of distortion of the quadtree? ◀

In the second part of the thesis, we provided several answers to Question 2, and presented ways of reducing the size of the input without changing much the structure of the problem – namely, all solution has cost preserved by our reductions. We present two new ways of constructing coreset: the first and main one, in Chapter 6, allows to construct coreset with near-optimal dependency in k, and in some cases in ε^{-1} as well. Our construction applies in all spaces that admits small approximate centroid set (see 5.2), a geometric notion that express how much the set of candidate solutions can be discretized. The second construction, presented in Chapter 9, allows to construct coresets with a slightly suboptimal size. This construction can be used deterministically, and works in all spaces where the set of balls centered on k distinct points has bounded VC-dimension. This simplifies a lot the previous construction described in Section 5.3: it is not necessary anymore to bound the *scaled* VC-dimension.

Those two constructions are very related: they extend the work of Chen [44], constructing coreset on a structured subset of the input (a ring, or a group of rings) instead of directly dealing with the full input, as in the work of Feldman and Langberg [76]. This may be the key for future improved coreset construction: find even more structured subset of the input, on which it is easier to sample a coreset.

As a second step, we show that the size of the coreset constructed in Chapter 7 is optimal in doubling metrics and general discrete spaces. This optimality shows essentially that the key part for constructing coreset is using concentration inequality for sums of independent random variables, such as Hoeffding's inequality or its generalizations. Indeed, Hoeffding's inequality is key in our coreset construction, and the lower bound relies precisely on its optimality.

However, this optimality does not carry over to other spaces: in Euclidean spaces, we presented a coreset of size $\widetilde{O}\left(k\varepsilon^{-4}\right)$ for k-median and k-means. With Vincent Cohen-Addad, Kasper Green Larsen and Chris Schwiegelshohn, we showed in [A8] a different tradeoff between k and ε^{-1}, namely coresets of size $\widetilde{O}\left(k^2\varepsilon^{-2}\right)$, and we complemented this result with a lower bound of $\Omega\left(k\varepsilon^{-2}\right)$. Put together, those result seems to show that the best coreset size should be $\widetilde{\Theta}\left(k\varepsilon^{-2}\right)$.

▶ **Open Question 3.** In Euclidean spaces, is it possible to construct coresets of size $\widetilde{O}\left(k\varepsilon^{-2}\right)$? ◀

Answering this question would not only nicely close the problem of constructing coreset in Euclidean space, but also extend the link between coreset constructions and concentration inequalities: can we reduce Euclidean coreset to such inequalities, or is the dimension playing a crucial role preventing us from reaching the same dependency in ε as in doubling and general metrics?

The techniques presented in this thesis (in Chapter 6 and Chapter 9) fall short of answering the question for the following reason. They proceed as follows: first construct a coreset of size dependent on the dimension, and then use dimension reduction techniques. It seems very unlikely to get coreset of size below $\Omega\left(k\varepsilon^{-4}\right)$ that way: one should loose an ε^{-2} factor in the dimension reduction – as the dimension reduction lemmas are tight – and another one in the coreset – as otherwise, it would likely contradict the doubling dimension lower bound. Hence, it is probably necessary to proceed in one shot, and to use carefully the dimension reduction during the coreset construction (or at least, during the proof, as it is not necessary for the algorithm to perform a dimension reduction).

The other coreset construction we present in Chapter 9 gives a different method for constructing coreset. While the algorithm is efficient if allowing randomization, its deterministic implementation has an exponential dependency in k and ε^{-1}. This is related to the construction of ε-nets and ε-set-approximation from the VC-dimension literature, for which no construction faster than the one we use (i.e., with a subexponential dependency in the VC dimension) is yet imaginable. We thus ask:

▶ **Open Question 4.** Is it possible to show that any deterministic coreset construction has running time exponential in k and ε^{-1}? ◀

In the particular Euclidean case, the complexity has an additional $n^{\varepsilon^{-O(z)}}$ dependency, due to the mere computation of a bicriteria solution, namely a solution with cost $(1+\varepsilon)$ time the optimal but using $O(k)$ centers. It seems this complexity is due to our poor understanding of the possible center's location in high dimensional Euclidean spaces. For instance, using a greedy algorithm that iteratively places a center at the best location would give an $(1+\varepsilon)$-approximation with $O(k\log(1/\varepsilon))$ many centers – but we actually do not know how to implement this simple algorithm fast. This raises the question:

▶ **Open Question 5.** In Euclidean space, given a current solution $\mathcal{S}$, is it possible to find efficiently the best possible location of an additional new center? ◀

Answering this question would probably help in other settings. For instance, under the differential privacy constraint, one of the key part of current algorithms (ours from Chapter 4, but also [17] and [87]) is to compute a set of candidate center, privately. Understanding better the possible location of centers would therefore probably impact our knowledge on differentially private clustering.

In the final chapter of this thesis, we studied algorithm for computing the power mean of a dataset, namely the optimal center for $(1, z)$-clustering. This is motivated by modeling questions: when is the median, or the mean, useful to analyze a dataset? The same interrogations also apply for the power mean, k-median, k-means and (k, z)-clustering: when are those objectives helpful to recover a ground-truth clustering of the input dataset? We provided in the introduction tentative explanations, but those objectives are not always perfectly suited. We therefore conclude this thesis with the broad question:

▶ **Open Question 6.** Is it possible to characterize precisely instances where the optimal (k, z)-clustering corresponds to a "natural" clustering of the dataset? ◀

Bibliography

[1] Ittai Abraham and Cyril Gavoille. "Object location using path separators". In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*. Ed. by Eric Ruppert and Dahlia Malkhi. ACM, 2006, pp. 188–197. DOI: 10.1145/1146381.1146411. URL: https://doi.org/10.1145/1146381.1146411.

[2] Manu Agarwal, Ragesh Jaiswal, and Arindam Pal. "k-means++ under Approximation Stability". In: *Theoretical Computer Science* 588 (2015), pp. 37–51.

[3] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. "Adaptive Sampling for k-Means Clustering". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*. Ed. by Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim. Vol. 5687. Lecture Notes in Computer Science. Springer, 2009, pp. 15–28. DOI: 10.1007/978-3-642-03685-9_2. URL: https://doi.org/10.1007/978-3-642-03685-9_2.

[4] Thomas Dybdahl Ahle. "Optimal Las Vegas Locality Sensitive Data Structures". In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. Ed. by Chris Umans. IEEE Computer Society, 2017, pp. 938–949. DOI: 10.1109/FOCS.2017.91. URL: https://doi.org/10.1109/FOCS.2017.91.

[5] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. "Better guarantees for k-means and euclidean k-median by primal-dual algorithms". In: *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*. Ieee. 2017, pp. 61–72.

[6] Alexandr Andoni, Zhao Song, Clifford Stein, Zhengyu Wang, and Peilin Zhong. "Parallel graph connectivity in log diameter rounds". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2018, pp. 674–685.

[7] Sanjeev Arora. "Nearly linear time approximation schemes for Euclidean TSP and other geometric problems". In: *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*. IEEE. 1997, pp. 554–563.

[8] Sanjeev Arora. "Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems". In: *Journal of the ACM (JACM)* 45.5 (1998), pp. 753–782.

[9] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. "Approximation Schemes for Euclidean K-medians and Related Problems". In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. STOC '98. Dallas, Texas, USA: ACM, 1998, pp. 106–113. ISBN: 0-89791-962-9. DOI: 10.1145/276698.276718. URL: http://doi.acm.org/10.1145/276698.276718.

[10] David Arthur and Sergei Vassilvitskii. "k-means++: the advantages of careful seeding". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*. 2007, pp. 1027–1035. URL: http://dl.acm.org/citation.cfm?id=1283383.1283494.

[11] Olivier Bachem, Mario Lucic, S Hamed Hassani, and Andreas Krause. "Approximate k-means++ in sublinear time". In: *Thirtieth AAAI conference on artificial intelligence*. 2016.

[12] Olivier Bachem, Mario Lucic, S. Hamed Hassani, and Andreas Krause. "Uniform Deviation Bounds for k-Means Clustering". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 283–291. URL: http://proceedings.mlr.press/v70/bachem17a.html.

[13] Olivier Bachem, Mario Lucic, and Silvio Lattanzi. "One-shot Coresets: The Case of k-Clustering". In: *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*. 2018, pp. 784–792. URL: http://proceedings.mlr.press/v84/bachem18a.html.

[14] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. "Scalable K-Means++". In: *Proc. VLDB Endow.* 5.7 (2012), pp. 622–633. DOI: 10.14778/2180912.2180915. URL: http://vldb.org/pvldb/vol5/p622_bahmanbahmani_vldb2012.pdf.

[15] Daniel Baker, Vladimir Braverman, Lingxiao Huang, Shaofeng H. C. Jiang, Robert Krauthgamer, and Xuan Wu. *Coresets for Clustering in Graphs of Bounded Treewidth*. 2020.

[16] Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. *Code of the algorithm described in Differentially Private Clustering in High-Dimensional Euclidean Spaces*. 2017. URL: https://github.com/mouwenlong/dp-clustering-icml17.

[17] Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. "Differentially Private Clustering in High-Dimensional Euclidean Spaces". In: *Proceedings of the 34th International Conference on Machine Learning, ICML*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 322–331.

[18] Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. "Distributed k-means and k-median clustering on general communication topologies". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. 2013, pp. 1995–2003. URL: http://papers.nips.cc/paper/5096-distributed-k-means-and-k-median-clustering-on-general-topologies.

[19] Pierre Baldi, Peter Sadowski, and Daniel Whiteson. "Searching for exotic particles in high-energy physics with deep learning". In: *Nature communications* 5.1 (2014), pp. 1–9.

[20] Yair Bartal and Lee-Ad Gottlieb. "A Linear Time Approximation Scheme for Euclidean TSP". In: *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*. 2013.

[21] Yair Bartal and Lee-Ad Gottlieb. "Near-linear time approximation schemes for Steiner tree and forest in low-dimensional spaces". In: *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM, 2021, pp. 1028–1041. DOI: 10.1145/3406325.3451063.

[22] Yair Bartal, Lee-Ad Gottlieb, Tsvi Kopelowitz, Moshe Lewenstein, and Liam Roditty. "Fast, precise and dynamic distance queries". In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics. 2011, pp. 840–853.

[23] MohammadHossein Bateni, Erik D. Demaine, MohammadTaghi Hajiaghayi, and Dániel Marx. "A PTAS for planar group Steiner tree via spanner bootstrapping and prize collecting". In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. ACM, 2016, pp. 570–583. DOI: 10.1145/2897518.2897549. URL: https://doi.org/10.1145/2897518.2897549.

[24] Paul Beame, Paraschos Koutris, and Dan Suciu. "Communication steps for parallel query processing". In: *Journal of the ACM (JACM)* 64.6 (2017), pp. 1–58.

[25] Luca Becchetti, Marc Bury, Vincent Cohen-Addad, Fabrizio Grandoni, and Chris Schwiegelshohn. "Oblivious dimension reduction for k-means: beyond subspaces and the Johnson-Lindenstrauss lemma". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. 2019, pp. 1039–1050. DOI: 10.1145/3313276.3316318. URL: https://doi.org/10.1145/3313276.3316318.

[26] Shai Ben-David. "A framework for statistical clustering with constant time approximation algorithms for K-median and K-means clustering". In: *Mach. Learn.* 66.2-3 (2007), pp. 243–257. DOI: 10.1007/s10994-006-0587-3. URL: https://doi.org/10.1007/s10994-006-0587-3.

[27] Rajen Bhatt and Abhinav Dhall. 2009. URL: https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation.

[28] Jock A Blackard and Denis J Dean. "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables". In: *Computers and electronics in agriculture* 24.3 (1999), pp. 131–151.

[29] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. "Learnability and the Vapnik-Chervonenkis dimension". In: *J. ACM* 36.4 (1989), pp. 929–965. DOI: 10.1145/76359.76371. URL: https://doi.org/10.1145/76359.76371.

[30] Nicolas Bousquet and Stéphan Thomassé. "VC-dimension and Erdős–Pósa property". In: *Discrete Mathematics* 338.12 (2015), pp. 2302–2317.

[31] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. "Near Optimal Column-Based Matrix Reconstruction". In: *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. Ed. by Rafail Ostrovsky. IEEE Computer Society, 2011, pp. 305–314.

[32] Vladimir Braverman, Dan Feldman, Harry Lang, and Daniela Rus. "Streaming Coreset Constructions for M-Estimators". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*. 2019, 62:1–62:15. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2019.62. URL: https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2019.62.

[33] Vladimir Braverman, Gereon Frahling, Harry Lang, Christian Sohler, and Lin F. Yang. "Clustering High Dimensional Dynamic Data Streams". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 576–585. URL: http://proceedings.mlr.press/v70/braverman17a.html.

[34] Vladimir Braverman, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Xuan Wu. "Coresets for Clustering in Excluded-minor Graphs and Beyond". In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*. Ed. by Dániel Marx. Consulted on arxiv on May 2022. SIAM, 2021, pp. 2679–2696. URL: https://doi.org/10.1137/1.9781611976465.159.

[35] Vladimir Braverman, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Xuan Wu. "Coresets for Ordered Weighted Clustering". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. 2019, pp. 744–753. URL: http://proceedings.mlr.press/v97/braverman19a.html.

[36] Maike Buchin and Dennis Rohde. "Coresets for (k, l)-Median Clustering under the Fréchet Distance". In: (2021).

[37] Jaroslaw Byrka, Thomas Pensyl, Bartosz Rybicki, Srinivasan Aravind, and Khoa Trinh. "An Improved Approximation for k-median, and Positive Correlation in Budgeted Optimization". In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*. 2015, pp. 737–756.

[38] Matteo Ceccarello, Andrea Pietracaprina, and Geppino Pucci. "Fast coreset-based diversity maximization under matroid constraints". In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 2018, pp. 81–89.

[39] TH Hubert Chan, Shuguang Hu, and Shaofeng H-C Jiang. "A PTAS for the steiner forest problem in doubling metrics". In: *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*. IEEE. 2016, pp. 810–819.

[40] Alisa Chang, Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. "Locally Private k-Means in One Round". In: *CoRR* abs/2104.09734 (2021). URL: https://arxiv.org/abs/2104.09734.

[41] Moses Charikar, Samir Khuller, David M. Mount, and Giri Narasimhan. "Algorithms for facility location problems with outliers". In: *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*. 2001, pp. 642–651.

[42] Moses S Charikar. "Similarity estimation techniques from rounding algorithms". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002, pp. 380–388.

[43] Bernard Chazelle. *The discrepancy method - randomness and complexity*. Cambridge University Press, 2001. ISBN: 978-0-521-00357-5.

[44] Ke Chen. "On Coresets for k-Median and k-Means Clustering in Metric and Euclidean Spaces and Their Applications". In: *SIAM J. Comput.* 39.3 (2009), pp. 923–947.

[45] Davin Choo, Christoph Grunau, Julian Portmann, and Václav Rozhon. "k-means++: few more steps yield constant approximation". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1909–1917.

[46] Marek Chrobak, Claire Kenyon, and Neal E. Young. "The reverse greedy algorithm for the metric k-median problem". In: *Inf. Process. Lett.* 97.2 (2006), pp. 68–72.

[47] Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff. "Sublinear optimization for machine learning". In: *J. ACM* 59.5 (2012), 23:1–23:49. DOI: 10.1145/2371656.2371658. URL: https://doi.org/10.1145/2371656.2371658.

[48] Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. "Dimensionality Reduction for k-Means Clustering and Low Rank Approximation". In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 2015, pp. 163–172.

[49] Michael B. Cohen, Yin Tat Lee, Gary L. Miller, Jakub Pachocki, and Aaron Sidford. "Geometric median in nearly linear time". In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. 2016, pp. 9–21. DOI: 10.1145/2897518.2897647. URL: http://doi.acm.org/10.1145/2897518.2897647.

[50] Vincent Cohen-Addad. "A Fast Approximation Scheme for Low-dimensional K-means". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '18. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2018, pp. 430–440. ISBN: 978-1-6119-7503-1. URL: http://dl.acm.org/citation.cfm?id=3174304.3175298.

[51] Vincent Cohen-Addad. "Approximation Schemes for Capacitated Clustering in Doubling Metrics". In: *CoRR* abs/1812.07721 (2018). URL: http://arxiv.org/abs/1812.07721.

[52] Vincent Cohen-Addad, Hossein Esfandiari, Vahab S. Mirrokni, and Shyam Narayanan. "Improved Approximations for Euclidean k-means and k-median, via Nested Quasi-Independent Sets". In: (2022).

[53] Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. "Local Search Yields Approximation Schemes for k-Means and k-Median in Euclidean and Minor-Free Metrics". In: *SIAM J. Comput.* 48.2 (2019), pp. 644–667. DOI: 10.1137/17M112717X. URL: https://doi.org/10.1137/17M112717X.

[54] Vincent Cohen-Addad, Silvio Lattanzi, Ashkan Norouzi-Fard, Christian Sohler, and Ola Svensson. "Fast and Accurate k-means++ via Rejection Sampling". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020.

[55] Vincent Cohen-Addad and Jason Li. "On the Fixed-Parameter Tractability of Capacitated Clustering". In: *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*. 2019, 41:1–41:14. URL: https://doi.org/10.4230/LIPIcs.ICALP.2019.41.

[56] Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. "Johnson Coverage Hypothesis: Inapproximability of k-means and k-median in ℓ_p-metrics". In: *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*. Ed. by Joseph (Seffi) Naor and Niv Buchbinder. SIAM, 2022, pp. 1493–1530. DOI: 10.1137/1.9781611977073.63. URL: https://doi.org/10.1137/1.9781611977073.63.

[57] Vincent Cohen-Addad and Chris Schwiegelshohn. "On the Local Structure of Stable Clustering Instances". In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. 2017, pp. 49–60. DOI: 10.1109/FOCS.2017.14. URL: https://doi.org/10.1109/FOCS.2017.14.

[58] Richard Cole and Lee-Ad Gottlieb. "Searching dynamic point sets in spaces with bounded doubling dimension". In: *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. ACM. 2006, pp. 574–583.

[59] Mónika Csikós, Nabil H. Mustafa, and Andrey Kupavskii. "Tight Lower Bounds on the VC-dimension of Geometric Set Systems". In: *J. Mach. Learn. Res.* 20 (2019), 81:1–81:8. URL: http://jmlr.org/papers/v20/18-719.html.

[60] Artur Czumaj and Christian Sohler. "Sublinear-time approximation algorithms for clustering via random sampling". In: *Random Structures & Algorithms* 30.1-2 (2007), pp. 226–256.

[61] Sanjoy Dasgupta and Yoav Freund. "Random projection trees for vector quantization". In: *IEEE Transactions on Information Theory* 55.7 (2009), pp. 3229–3242.

[62] Jeffrey Dean and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pp. 107–113.

[63] Hu Ding. "A Sub-Linear Time Framework for Geometric Optimization with Outliers in High Dimensions". In: *28th Annual European Symposium on Algorithms, ESA 2020, September 7-9, 2020, Pisa, Italy (Virtual Conference)*. 2020, 38:1–38:21. DOI: 10.4230/LIPIcs.ESA.2020.38. URL: https://doi.org/10.4230/LIPIcs.ESA.2020.38.

[64] Hu Ding. "Stability Yields Sublinear Time Algorithms for Geometric Optimization in Machine Learning". In: *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*. Ed. by Petra Mutzel, Rasmus Pagh, and Grzegorz Herman. Vol. 204. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 38:1–38:19.

[65] Anne Driemel, André Nusser, Jeff M Phillips, and Ioannis Psarros. "The VC dimension of metric balls under Fréchet and Hausdorff distances". In: *Discrete & Computational Geometry* 66.4 (2021), pp. 1351–1381.

[66] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: http://archive.ics.uci.edu/ml.

[67] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography*. Ed. by Shai Halevi and Tal Rabin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284.

[68] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: *Found. Trends Theor. Comput. Sci.* 9.3-4 (2014), pp. 211–407. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042.

[69] David Eisenstat and Dana Angluin. "The VC dimension of k-fold union". In: *Inf. Process. Lett.* 101.5 (2007), pp. 181–184. DOI: 10.1016/j.ipl.2006.10.004. URL: https://doi.org/10.1016/j.ipl.2006.10.004.

[70] David Eisenstat, Philip N. Klein, and Claire Mathieu. "Approximating k-center in planar graphs". In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. Ed. by Chandra Chekuri. SIAM, 2014, pp. 617–627. DOI: 10.1137/1.9781611973402.47. URL: https://doi.org/10.1137/1.9781611973402.47.

[71] Michael Elkin, Arnold Filtser, and Ofer Neiman. "Terminal embeddings". In: *Theor. Comput. Sci.* 697 (2017), pp. 1–36. DOI: 10.1016/j.tcs.2017.06.021. URL: https://doi.org/10.1016/j.tcs.2017.06.021.

[72] Lars Engebretsen, Piotr Indyk, and Ryan O'Donnell. "Derandomized dimensionality reduction with applications". In: *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*. 2002, pp. 705–712. URL: http://dl.acm.org/citation.cfm?id=545381.545476.

[73] Xiequan Fan, Ion Grama, and Quansheng Liu. "Sharp large deviation results for sums of independent random variables". In: *Science China Mathematics* 58.9 (2015), pp. 1939–1958.

[74] Tomás Feder and Daniel Greene. "Optimal algorithms for approximate clustering". In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*. ACM. 1988, pp. 434–444.

[75] Dan Feldman. *Introduction to Core-sets: an Updated Survey*. 2020. DOI: 10.1002/widm.1335. URL: https://doi.org/10.1002/widm.1335.

[76] Dan Feldman and Michael Langberg. "A unified framework for approximating and clustering data". In: *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*. 2011, pp. 569–578.

[77] Dan Feldman, Morteza Monemizadeh, and Christian Sohler. "A PTAS for k-means clustering based on weak coresets". In: *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*. 2007, pp. 11–18.

[78] Dan Feldman, Melanie Schmidt, and Christian Sohler. "Turning Big Data Into Tiny Data: Constant-Size Coresets for k-Means, PCA, and Projective Clustering". In: *SIAM J. Comput.* 49.3 (2020), pp. 601–657. DOI: 10.1137/18M1209854. URL: https://doi.org/10.1137/18M1209854.

[79] Vitaly Feldman, Konstantin Kakaes, Katrina Ligett, Kobbi Nissim, Aleksandra Slavkovic, and Adam Smith. *Differential Privacy: Issues for Policymakers*. https://simons.berkeley.edu/news/differential-privacy-issues-policymakers, consulted on 04/27/2022. 2020.

[80] Zhili Feng, Praneeth Kacham, and David P. Woodruff. "Strong Coresets for Subspace Approximation and k-Median in Nearly Linear Time". In: *CoRR* abs/1912.12003 (2019). arXiv: 1912.12003. URL: http://arxiv.org/abs/1912.12003.

[81] Hendrik Fichtenberger, Marc Gillé, Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. "BICO: BIRCH Meets Coresets for k-Means Clustering". In: *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*. 2013, pp. 481–492.

[82] Gereon Frahling and Christian Sohler. "Coresets in dynamic geometric data streams". In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*. 2005, pp. 209–217.

[83] Pasi Fränti and Sami Sieranoja. *K-means properties on six clustering benchmark datasets*. 2018. URL: http://cs.uef.fi/sipu/datasets/.

[84] Gouvernement Français. *Industrie du futur: Moderniser notre outil industriel*, https://www.economie.gouv.fr/files/files/PDF/industrie-du-futur_dp.pdf, consulted on 04/26/2022. 2015.

[85] Zachary Friggstad, Kamyar Khodamoradi, Mohsen Rezapour, and Mohammad R Salavatipour. "Approximation schemes for clustering with outliers". In: *ACM Transactions on Algorithms (TALG)* 15.2 (2019), p. 26.

[86] Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. "Local Search Yields a PTAS for k-Means in Doubling Metrics". In: *SIAM J. Comput.* 48.2 (2019), pp. 452–480. DOI: 10.1137/17M1127181. URL: https://doi.org/10.1137/17M1127181.

[87] Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. "Differentially Private Clustering: Tight Approximation Ratios". In: *Advances in Neural Information Processing Systems*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020.

[88] Michael T Goodrich, Nodari Sitchinava, and Qin Zhang. "Sorting, searching, and simulation in the mapreduce framework". In: *International Symposium on Algorithms and Computation*. Springer. 2011, pp. 374–383.

[89] Lee-Ad Gottlieb. "A Light Metric Spanner". In: *Symposium on Foundations of Computer Science, FOCS*. 2015.

[90] Sudipto Guha and Samir Khuller. "Greedy Strikes Back: Improved Facility Location Algorithms". In: *J. Algorithms* 31.1 (1999), pp. 228–248. DOI: 10.1006/jagm.1998.0993. URL: http://dx.doi.org/10.1006/jagm.1998.0993.

[91] Anupam Gupta, Robert Krauthgamer, and James R. Lee. "Bounded Geometries, Fractals, and Low-Distortion Embeddings". In: *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '03. 2003.

[92] Anupam Gupta and Kanat Tangwongsan. "Simpler Analyses of Local Search Algorithms for Facility Location". In: *CoRR* abs/0809.2554 (2008). URL: http://arxiv.org/abs/0809.2554.

[93] Pierre Hansen and Nenad Mladenovic. "J-M_{EANS}: a new local search heuristic for minimum sum of squares clustering". In: *Pattern Recognition* 34.2 (2001), pp. 405–413. DOI: 10.1016/S0031-3203(99)00216-2. URL: http://dx.doi.org/10.1016/S0031-3203(99)00216-2.

[94] Sariel Har-Peled. *Geometric approximation algorithms*. 173. American Mathematical Soc., 2011.

[95] Sariel Har-Peled and Akash Kushal. "Smaller Coresets for k-Median and k-Means Clustering". In: *Discrete & Computational Geometry* 37.1 (2007), pp. 3–19.

[96] Sariel Har-Peled and Soham Mazumdar. "On coresets for k-means and k-median clustering". In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*. 2004, pp. 291–300.

[97] Sariel Har-Peled and Manor Mendel. "Fast construction of nets in low-dimensional metrics and their applications". In: *SIAM Journal on Computing* 35.5 (2006), pp. 1148–1184.

[98] Lingxiao Huang, Shaofeng H.-C. Jiang, Jian Li, and Xuan Wu. "Epsilon-Coresets for Clustering (with Outliers) in Doubling Metrics". In: *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. 2018, pp. 814–825. DOI: 10.1109/FOCS.2018.00082. URL: https://doi.org/10.1109/FOCS.2018.00082.

[99] Lingxiao Huang, Shaofeng H.-C. Jiang, and Nisheeth K. Vishnoi. "Coresets for Clustering with Fairness Constraints". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*. 2019, pp. 7587–7598. URL: http://papers.nips.cc/paper/8976-coresets-for-clustering-with-fairness-constraints.

[100] Lingxiao Huang and Nisheeth K. Vishnoi. "Coresets for clustering in Euclidean spaces: importance sampling is nearly optimal". In: *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*. Ed. by Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy. ACM, 2020, pp. 1416–1429. DOI: 10.1145/3357713.3384296. URL: https://doi.org/10.1145/3357713.3384296.

[101] Jonathan Huggins, Trevor Campbell, and Tamara Broderick. "Coresets for scalable Bayesian logistic regression". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4080–4088.

[102] Mary Inaba, Naoki Katoh, and Hiroshi Imai. "Applications of Weighted Voronoi Diagrams and Randomization to Variance-Based k-Clustering (Extended Abstract)". In: *Proceedings of the Tenth Annual Symposium on Computational Geometry, Stony Brook, New York, USA, June 6-8, 1994*. 1994, pp. 332–339. DOI: 10.1145/177424.178042. URL: https://doi.org/10.1145/177424.178042.

[103] Piotr Indyk, Sepideh Mahabadi, Shayan Oveis Gharan, and Alireza Rezaei. "Composable Core-sets for Determinant Maximization Problems via Spectral Spanners". In: *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*. Ed. by Shuchi Chawla. SIAM, 2020, pp. 1675–1694. DOI: 10.1137/1.9781611975994.103. URL: https://doi.org/10.1137/1.9781611975994.103.

[104] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S. Mirrokni. "Composable core-sets for diversity and coverage maximization". In: *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*. Ed. by Richard Hull and Martin Grohe. ACM, 2014, pp. 100–108. DOI: 10.1145/2594538.2594560. URL: https://doi.org/10.1145/2594538.2594560.

[105] INSEE. *Moyenne : définition*. Institut National de la Statistique et des Études Économiques, https://www.insee.fr/fr/metadonnees/definition/c1970, consulted on 04/25/2022.

[106] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, and Dennis Fetterly. "Dryad: distributed data-parallel programs from sequential building blocks". In: *ACM SIGOPS operating systems review*. Vol. 41. 3. ACM. 2007, pp. 59–72.

[107] Roger Iyengar, Joseph P. Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. "Towards Practical Differentially Private Convex Optimization". In: *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*. IEEE, 2019, pp. 299–316.

[108] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V Vazirani. "Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP". In: *Journal of the ACM (JACM)* 50.6 (2003), pp. 795–824.

[109] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. "A local search approximation algorithm for k-means clustering". In: *Computational Geometry* 28.2-3 (2004), pp. 89–112.

[110] Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. "A model of computation for MapReduce". In: *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM. 2010, pp. 938–948.

[111] Manohar Kaul, Bin Yang, and Christian S. Jensen. "Building Accurate 3D Spatial Networks to Enable Next Generation Intelligent Transportation Systems". In: *2013 IEEE 14th International Conference on Mobile Data Management, Milan, Italy, June 3-6, 2013 - Volume 1*. 2013, pp. 137–146. DOI: 10.1109/MDM.2013.24. URL: https://doi.org/10.1109/MDM.2013.24.

[112] Stavros G Kolliopoulos and Satish Rao. "A nearly linear-time approximation scheme for the Euclidean k-median problem". In: *SIAM Journal on Computing* 37.3 (2007), pp. 757–782.

[113] Ravishankar Krishnaswamy, Shi Li, and Sai Sandeep. "Constant approximation for k-median and k-means with outliers via iterative rounding". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. ACM. 2018, pp. 646–659.

[114] Amit Kumar and Ravindran Kannan. "Clustering with Spectral Norm and the k-Means Algorithm". In: *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. 2010, pp. 299–308. DOI: 10.1109/FOCS.2010.35. URL: http://dx.doi.org/10.1109/FOCS.2010.35.

[115] Michael Langberg and Leonard J. Schulman. "Universal ε-approximators for Integrals". In: *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*. 2010, pp. 598–607.

[116] Kasper Green Larsen and Jelani Nelson. "Optimality of the Johnson-Lindenstrauss Lemma". In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. 2017, pp. 633–638. DOI: 10.1109/FOCS.2017.64. URL: https://doi.org/10.1109/FOCS.2017.64.

[117] Silvio Lattanzi and Christian Sohler. "A better k-means++ algorithm via local search". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3662–3671.

[118] Jasper C. H. Lee and Paul Valiant. "Optimal Sub-Gaussian Mean Estimation in $\mathbb{R}$". In: *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. IEEE, 2021, pp. 672–683. URL: https://doi.org/10.1109/FOCS52979.2021.00071.

[119] Shi Li. "A 1.488 approximation algorithm for the uncapacitated facility location problem". In: *Inf. Comput.* 222 (2013), pp. 45–58.

[120] Yi Li and Philip M. Long. "Learnability and the doubling dimension". In: *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*. 2006, pp. 889–896. URL: http://papers.nips.cc/paper/3068-learnability-and-the-doubling-dimension.

[121] Yi Li, Philip M. Long, and Aravind Srinivasan. "Improved Bounds on the Sample Complexity of Learning". In: *J. Comput. Syst. Sci.* 62.3 (2001), pp. 516–527. DOI: 10.1006/jcss.2000.1741. URL: https://doi.org/10.1006/jcss.2000.1741.

[122] Chen Liao and Shiyan Hu. "Polynomial time approximation schemes for minimum disk cover problems". In: *Journal of combinatorial optimization* 20.4 (2010), pp. 399–412.

[123] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137.

[124] Zhigang Lu and Hong Shen. "Differentially Private k-Means Clustering with Guaranteed Convergence". In: *CoRR* abs/2002.01043 (2020). URL: https://arxiv.org/abs/2002.01043.

[125] Gábor Lugosi and Shahar Mendelson. "Mean Estimation and Regression Under Heavy-Tailed Distributions: A Survey". In: *Found. Comput. Math.* 19.5 (2019), pp. 1145–1190. DOI: 10.1007/s10208-019-09427-x. URL: https://doi.org/10.1007/s10208-019-09427-x.

[126] Alaa Maalouf, Ibrahim Jubran, and Dan Feldman. "Fast and accurate least-mean-squares solvers". In: *Advances in Neural Information Processing Systems*. 2019, pp. 8307–8318.

[127] Sepideh Mahabadi, Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. "Nonlinear dimension reduction via outer Bi-Lipschitz extensions". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*. 2018, pp. 1088–1101. DOI: 10.1145/3188745.3188828. URL: http://doi.acm.org/10.1145/3188745.3188828.

[128] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi R. Varadarajan. "The planar k-means problem is NP-hard". In: *Theor. Comput. Sci.* 442 (2012), pp. 13–21.

[129] Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. "Performance of Johnson-Lindenstrauss transform for k-means and k-medians clustering". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. 2019, pp. 1027–1038. DOI: 10.1145/3313276.3316350. URL: https://doi.org/10.1145/3313276.3316350.

[130] Konstantin Makarychev, Aravind Reddy, and Liren Shan. "Improved Guarantees for k-means++ and k-means++ Parallel". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/ba304f3809ed31d0ad97b5a2b5df2a39-Abstract.html.

[131] Yair Marom and Dan Feldman. "k-Means Clustering of Lines for Big Data". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 12797–12806. URL: https://proceedings.neurips.cc/paper/2019/hash/6084e82a08cb979cf75ae28aed37ecd4-Abstract.html.

[132] Dániel Marx and Michał Pilipczuk. "Optimal parameterized algorithms for planar facility location problems using Voronoi diagrams". In: *Algorithms-ESA 2015*. Springer, 2015, pp. 865–877.

[133] Shigeru Masuyama, Toshihide Ibaraki, and Toshiharu Hasegawa. "The computational complexity of the m-center problems on the plane". In: *IEICE TRANSACTIONS (1976-1990)* 64.2 (1981), pp. 57–64.

[134] Jirí Matousek. "On Approximate Geometric k-Clustering". In: *Discrete & Computational Geometry* 24.1 (2000), pp. 61–84. DOI: 10.1007/s004540010019. URL: http://dx.doi.org/10.1007/s004540010019.

[135] Nimrod Megiddo and Kenneth J Supowit. "On the complexity of some common geometric location problems". In: *SIAM journal on computing* 13.1 (1984), pp. 182–196.

[136] Ramgopal R. Mettu and C. Greg Plaxton. "Optimal Time Bounds for Approximate Clustering". In: *Mach. Learn.* 56.1-3 (2004), pp. 35–60. DOI: 10.1023/B:MACH.0000033114.18632.e0. URL: https://doi.org/10.1023/B:MACH.0000033114.18632.e0.

[137] Adam Meyerson, Liadan O'callaghan, and Serge Plotkin. "A k-median algorithm with running time independent of data size". In: *Machine Learning* 56.1 (2004), pp. 61–87.

[138] Glenn W Milligan. "An examination of the effect of six types of error perturbation on fifteen clustering algorithms". In: *psychometrika* 45.3 (1980), pp. 325–342.

[139] Joseph SB Mitchell. "Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems". In: *SIAM Journal on computing* 28.4 (1999), pp. 1298–1309.

[140] Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. "GUPT: privacy preserving data analysis made easy". In: *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 2012, pp. 349–360.

[141] Alejandro Molina, Alexander Munteanu, and Kristian Kersting. "Core Dependency Networks". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, 2018, pp. 3820–3827. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16847.

[142] Le Monde. *Pour Emmanuel Macron, l'intelligence artificielle est aussi "une révolution politique"*. https://www.lemonde.fr/pixels/article/2018/03/31/pour-emmanuel-macron-l-intelligence-artificielle-est-aussi-une-revolution-politique_5279161_4408996.html, consulted on 04/26/2022.

[143] Alexander Munteanu and Chris Schwiegelshohn. "Coresets-Methods and History: A Theoreticians Design Pattern for Approximation and Streaming Algorithms". In: *Künstliche Intell.* 32.1 (2018), pp. 37–53. DOI: 10.1007/s13218-017-0519-3. URL: https://doi.org/10.1007/s13218-017-0519-3.

[144] Alexander Munteanu, Chris Schwiegelshohn, Christian Sohler, and David P. Woodruff. "On Coresets for Logistic Regression". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 6562–6571.

[145] Shyam Narayanan and Jelani Nelson. "Optimal terminal dimensionality reduction in Euclidean space". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. Ed. by Moses Charikar and Edith Cohen. ACM, 2019, pp. 1064–1069. DOI: 10.1145/3313276.3316307. URL: https://doi.org/10.1145/3313276.3316307.

[146] Huy L. Nguyen, Anamay Chaturvedi, and Eric Z. Xu. "Differentially Private k-Means via Exponential Mechanism and Max Cover". In: (2021), pp. 9101–9108. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17099.

[147] Richard Nock, Raphaël Canyasse, Roksana Boreli, and Frank Nielsen. "k-variates++: more pluses in the k-means++". In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. Ed. by Maria-Florina Balcan and Kilian Q. Weinberger. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 145–154.

[148] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. "The effectiveness of Lloyd-type methods for the k-means problem". In: *J. ACM* 59.6 (2012), p. 28. DOI: 10.1145/2395116.2395117. URL: http://doi.acm.org/10.1145/2395116.2395117.

[149] Tamás Sarlós. "Improved Approximation Algorithms for Large Matrices via Random Projections". In: *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 2006, pp. 143–152.

[150] Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. "Fair Coresets and Streaming Algorithms for Fair k-means". In: *Approximation and Online Algorithms - 17th International Workshop, WAOA 2019, Munich, Germany, September 12-13, 2019, Revised Selected Papers*. 2019, pp. 232–251. DOI: 10.1007/978-3-030-39479-0_16. URL: https://doi.org/10.1007/978-3-030-39479-0_16.

[151] Christian Sohler and David P. Woodruff. "Strong Coresets for k-Median and Subspace Approximation: Goodbye Dimension". In: *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. 2018, pp. 802–813. DOI: 10.1109/FOCS.2018.00081. URL: https://doi.org/10.1109/FOCS.2018.00081.

[152] Hugo Steinhaus. "Sur la division des corps matériels en parties". In: *Bull. Acad. Polon. Sci. Cl. III.* (1956), 801–804.

[153] Uri Stemmer and Haim Kaplan. "Differentially Private k-Means with Constant Multiplicative Error". In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett. 2018, pp. 5436–5446. URL: https://proceedings.neurips.cc/paper/2018/hash/32b991e5d77ad140559ffb95522992d0-Abstract.html.

[154] Michel Talagrand et al. "Majorizing measures: the generic chaining". In: *The Annals of Probability* 24.3 (1996), pp. 1049–1103.

[155] K. Talwar. "Bypassing the embedding: algorithms for low dimensional metrics". In: *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM. 2004, pp. 281–290. DOI: 10.1145/1007352.1007399.

[156] Mikkel Thorup. "Quick k-median, k-center, and facility location for sparse graphs". In: *SIAM Journal on Computing* 34.2 (2005), pp. 405–432.

[157] Kasturi Varadarajan and Xin Xiao. "On the Sensitivity of Shape Fitting Problems". In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012)*. Ed. by Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan. Vol. 18. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 486–497. ISBN: 978-3-939897-47-7. DOI: 10.4230/LIPIcs.FSTTCS.2012.486. URL: https://drops.dagstuhl.de/opus/volltexte/2012/3883.

[158] Andrea Vattani. "K-means requires exponentially many iterations even in the plane". In: *Discrete & Computational Geometry* 45.4 (2011), pp. 596–616.

[159] Jeffrey Scott Vitter. "Random Sampling with a Reservoir". In: *ACM Trans. Math. Softw.* 11.1 (1985), pp. 37–57. DOI: 10.1145/3147.3165. URL: https://doi.org/10.1145/3147.3165.

[160] Dennis Wei. "A Constant-Factor Bi-Criteria Approximation Guarantee for k-means++". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. 2016, pp. 604–612.

[161] Tom White. *Hadoop: The definitive guide.* " O'Reilly Media, Inc.", 2012.

[162] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. "Spark: Cluster computing with working sets." In: *HotCloud* 10.10-10 (2010), p. 95.