UNIVERSITE PARIS.DIDEROT (Paris 7) SORBONNE PARIS CITE

École doctorale 386: Sciences mathématiques de Paris Centre

THÈSE PRÉPARÉE AU LABORATOIRE D'INFORMATIQUE DE L'ÉCOLE NORMALE SUPÉRIEURE

THÈSE DE DOCTORAT

Mention Informatique

présentée et soutenue publiquement
le 15 septembre 2016
en vue de l'obtention du grade de

Docteur en sciences

par

VINCENT COHEN-ADDAD

From Practice to Theory

Approximation schemes for clustering and network design

Thèse dirigée par Mme Claire MATHIEU

Membres du jury:

M. Philippe JACQUET (Directeur de recherche, Nokia Bell Labs, France)	examinateur
M. Zhentao LI (Maître de conférence, ENS, France)	examinateur
Mme. Claire MATHIEU (Directrice de recherche, CNRS, ENS, France)	directrice de thèse
M. Yuval RABANI (Professor, The Hebrew University of Jerusalem, Israël)	rapporteur
M. András SEBŐ (Directeur de recherche, CNRS, G-SCOP, France)	rapporteur
M. Ola SVENSSON (Assistant professor, EPFL, Suisse)	examinateur
M. Stéphan THOMASSÉ (Professeur, ENS Lyon, France)	examinateur

To my Mother, my Father, Grand-Ma and Papi – my sources of inspiration.

Contents

Acknowledgments

Several people have made the last three years an intense, rewarding and exciting experience. I am extremely grateful to my advisors Claire Mathieu and Zhentao Li. Claire and Zhentao have been – and continue to be – very inspiring for my future academic life and research. Thank you Claire and Zhentao for your patience, your support, giving me your insight into theoretical computer science. Thank you Claire for having given me the opportunity to do my Ph.D under your guidance and so many opportunities to learn and to discover research during those three years.

I thank András Sebő and Yuval Rabani for having accepted to review this manuscript and to be on my thesis jury. Thank you very much for the interest you had in my work, your time and patience. I am also extremely grateful to Philippe Jacquet, Ola Svensson and Stéphan Thomassé for having accepted to be on my thesis jury; I am very honored.

It was a great pleasure to spend three years in the Talgo group at ENS. I was very happy to work with and learn from Éric Colin de Verdière. Thank you Éric for sharing your knowledge on embedded graphs, and providing me with your advises for my future academic life. I thank all the people in the Talgo group for having made every day spent at ENS very exciting. Thank you Frederik, Hang, and Victor!

I thank Amos Fiat for having introduced me to algorithmic game theory, to the ways of sailing, and with the excellent restaurants in Tel-Aviv. I will keep a great memory of my two visits at Tel-Aviv and hope to make many more. Those visits would not have been the same without Alon Eden and Orit Raz – Thanks to both of you for sharing great ideas, talking about our experiences in research and showing me the bars of Tel-Aviv, I hope we can meet again soon! Many thanks to Michal Feldman for her patience and collaboration.

I also had the great chance to benefit from the wisdom and advises of Philip N. Klein. Thank you Phil for your patience and sharing your knowledge on algorithms and planar graphs. I hope I'll have the chance to learn from you again in the future.

I thank Christian Sohler for hosting me in TU Dortmund and guiding me through clustering problems in dynamic data streams. I had a very rewarding experience in the group. I was also very lucky to work with Chris Schwiegelshohn in Dortmund. Chris is a great team mate; thanks Chris for sharing your view and your expertise!

I thank Evripidis Bampis, Christoph Dürr, and Ioannis Millis for collaborating with me at the beginning of those three years and sharing their knowledge on scheduling problems.

I thank Nabil Mustafa for sharing his knowledge on computational geometry and his advices.

I thank Daniel Král for his fruitful collaboration.

My academic life would not have been the same without the support and trust of all my professors at ENS Lyon. Thank you Éric Thierry and Daniel Hirschkoff for your training and for having given me the opportunity to join ENS Lyon.

I thank Michel Habib and Fabien de Montgolfier for advising me during my master thesis.

I also thank Arnaud de Mesmay for being my big brother in research. Thanks Arnaud for giving me advises, sharing your experience and knowledge, and cheering me up.

I thank Varun Kanade for sharing his views on research and theoretical computer science, for introducing me to learning theory and machine learning. Thank you Varun for your great support.

This thesis has also been heavily supported by several (more or less extra-academic) friends and family members. Thank you Lucca and Marie-Charlotte for all this time spent together! Thank you Simon for all the evenings we spent talking about life, research, and art. I also thank Bruno, Thomas, and Mathieu for all the discussions about research and life we had during those three years. Thank you Jean-Florent for sharing all your journeys (in research but not only).

I also thank my friends of always, Antoine, Basile, Manu, Nicolas, and Sébastien. Thank you for everything we lived together during the last 10 years (actually 23 for some of you).

Thank you very much Sophie for your strong support, cheering me up and recalling me that research is a strict subset of life.

Finally, thank you to all my family for taking care of me for so long and making me reach that point. I thank very much my father and my mother, Jacques, Jade, Sylvie, Reinhard, Charlotte, and Elisabeth for having closely supported me and having listened to me during those three years. Thank you maman, papa for having inspired me and made me as I am.

General Introduction

How to solve *this* problem? How fast can it be solved? Answering those questions rigorously for a computational problem often entails the design and analysis of an *algorithm* – a step-by-step method that, applied to an *instance* of the problem, produces a *solution* to that instance. In 2013, Frey and Osborne [87] estimated that about 47 percent of the total U.S. employment could be concerned by job automation in roughly two decades, showing that algorithms are taking a preponderant role in our lives. Thus, proceeding naïvely by, for example, designing algorithms that achieve good enough performances on a benchmark or in a very specific context is often unsatisfactory when not risky. How can one rely on an algorithm whose efficiency and correctness are not established for solving critical tasks?

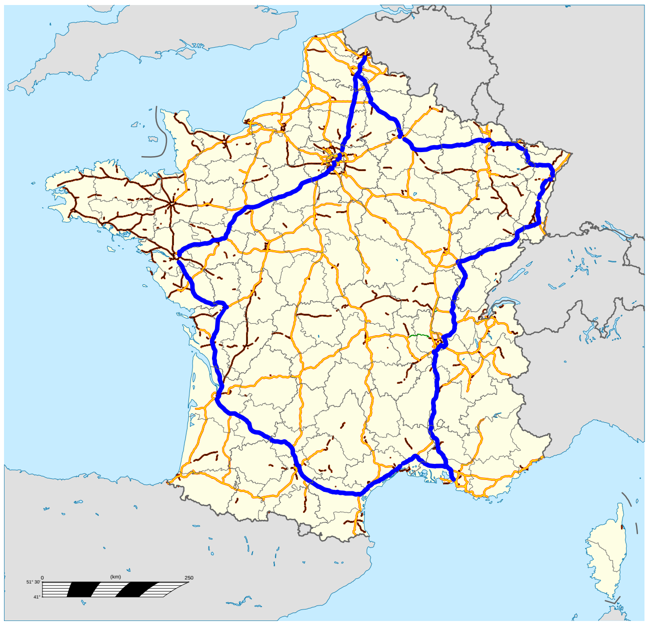
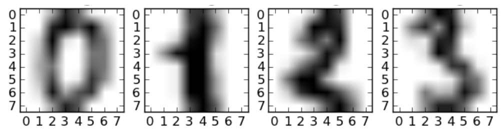


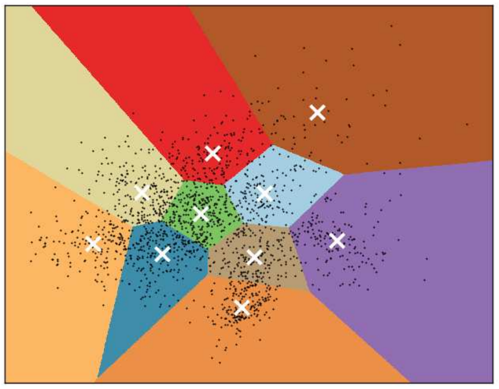
Figure 1: A (traveling salesman) *tour* through several french cities using the highway network.

A systematic approach consists in proceeding at a more abstract level by: (1) *distilling* a problem faced in practice by proposing a model (2) *designing and analyzing* an algorithm for the problem in that model and (3) *applying* this algorithm to practical instances.

However, the distilling step sometimes introduces a gap between theory and practice. If the model is too general it induces a two-fold gap: on the one hand no guarantee on the algorithms that are efficient in practice can be proven, on the other hand the best theoretical algorithms turn out to be noncompetitive in practice. This gap has been observed in several contexts, in particular for clustering and network design problems.



(a) Three 8px-by-8px images of handwritten digits.



(b) A *clustering* of a 2-dimensional embedding of a dataset of handwritten digits images. Clusters (areas of the same color) contain data points corresponding to the images of the same digit.

Figure 2: 8px-by-8px images can be seen as a set of points lying in a 64-dimensional space by reading each pixel as a coordinate. For images of handwritten digits, it is possible to project the dataset to a 2-dimensional space while preserving some of the underlying structure of the data (using principal component analysis).

The distilling gap: illustrative examples. Given a set of points in a metric space, a clustering is a partition of the points into *clusters* according to a measure of proximity. There is a tremendous number of contexts in which clustering problems occur. For example, in machine learning and data sciences, it is standard to measure the similarity of two data points by a distance function (see Figure 2). Hence, a clustering of the input points provides a fundamental information: points in the same cluster have common features.

Now suppose that we are asked to build a few warehouses to serve a set of shops. Here, we would like to optimize the distance from each shop to its closest warehouse. Thus, we again look for a clustering of the shops: each cluster contains points that are close to one another and therefore, that could be served by a the same warehouses (see Figure 3).

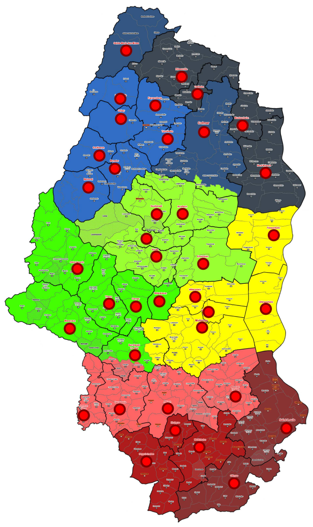


Figure 3: Location and district of activity of the fire stations in the french département du Haut-Rhin. Each red point corresponds to a fire station. Bold boundaries represent the districts of activity of the fire stations. Each district of activity can be seen as a cluster: all the points – the dwellings – are close to the same fire station.

This illustrates the heterogeneity of the contexts in which clustering problems arise. However, at a more abstract level, this historically resulted in the general k-means problem[1] which *makes no distinction* between those different contexts (see Definition A.1 for a general version).

This lends clustering an easy-in-practice, hard-in-theory character. On the one hand, various heuristics have been designed to solve specific instances. Those heuristics achieve good experimental results although no performance guarantees have been established. On the other hand, theoretical results showed that the problem is NP-hard[2], preventing the community from deriving theoretical bounds that match experimental observations and thus, preventing a formal explanation of the success of the heuristics.

Another symbolic example is the famous traveling salesman problem (TSP). Given a set of points in a metric space, this problem asks for a tour of the points of minimum length (see Definition A.4). This is one of the most famous problems in combinatorial optimization because of its impressive number of applications and because its study has led to the development of new algorithmic techniques over the years (see also the popular science book of Cook [61]).

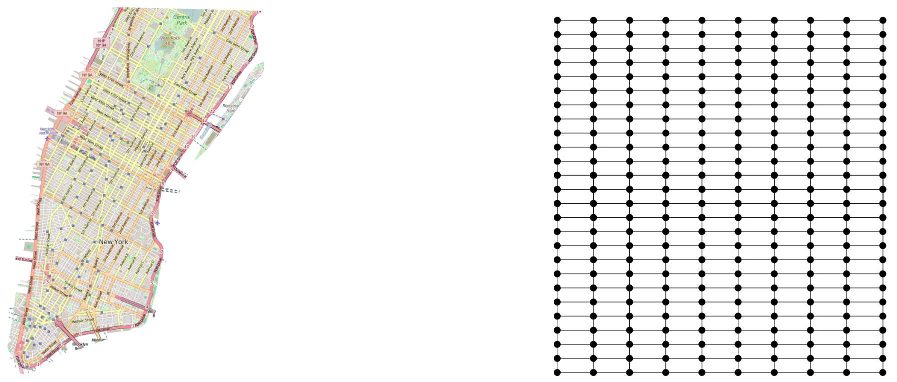
In 2000, David Johnson et al. organized the 8th DIMACS implementation Challenge on TSP [70]. The goal was to thoroughly compare the different heuristics people have

[1]Or slight variations like k-median and k-center.

[2]Assuming P $\neq$ NP, there cannot exist an algorithm returning an optimal solution to the problem in a number of operations that is polynomial in the size of the input.

developed through the years using random and real-world Euclidean instances. They reviewed more than 30 heuristics on thousands of instances. Quite surprisingly the best known theoretical algorithms for Euclidean TSP – for which Sanjeev Arora and Joseph S. B. Mitchell received the Gödel price in 2010 – is absent from the challenge. Johnson and McGeoch [109] explain that despite the strong theoretical guarantees, it is likely that those algorithms are not competitive on real-world instances.

From practice to theory. The attempts to bridge the gap between theory and practice resulted in various characterizations of real-world instances (see for example [25, 36, 151]).



(a) Map of Manhattan, its grid-like road network is a planar graph.

(b) A grid planar graph with 250 vertices.

Figure 4: The road network of Manhattan on the left can be modeled by grid graph (similar to the one on the right).

However, those approaches mainly resulted in either a class of instances for which the theoretical analysis still yield bounds that are very far from what is experienced in practice or ad-hoc algorithms that leverage the properties of a class of instances but that are not used in practice.

We proceed in the reverse order. We focus on an algorithm that is widely used in practice for hard combinatorial optimization problems, *local search* (Algorithm 1), and investigate in which contexts this algorithm performs well.

This algorithm proceeds almost naïvely. Initially, it starts with an arbitrary solution S to the problem. Then, while there exists a *neighboring* solution S' – a solution obtained via a slight modification of S – of smaller cost, it replaces S by S' and repeats.

Algorithm 1 The Local Search algorithm.

```
1: Input: An instance of an optimization problem.
2: Parameter: A positive integer $s$ for the neighborhood size.
3: $S \leftarrow$ arbitrary solution to the problem
4: while there is a solution $S'$: $|S \backslash S'| + |S' \backslash S| \leqslant s$ and cost$(S') <$ cost$(S)$
5:  do
6:      $S \leftarrow S'$
7: Output: $S$
```

This algorithm can be adapted to any combinatorial optimization problem. For example, in the case of the k-means problem a solution is defined by a set of *centers*[1]. Thus, a neighboring solution for local search is obtained from the current solution by removing a few of the current centers and adding back a few different centers.

It can also be easily adapted to TSP. Let us view a tour as a sequence of *edges*[2]. Given a current tour T, a neighboring tour is obtained by removing a constant number of edges – this breaks T into a constant number of *paths* – and reconnecting the paths in the best possible way.

This makes local search very appealing: it is easy to implement, easy to run in parallel, and easy to tune. Indeed, the space of neighboring solutions can be explored in parallel and the number of modifications allowed to define a neighboring solution, the *neighborhood size*, allows some trade-off between the quality of the solution and the running time. It has received a considerable amount of attention over the years, from both practical and theoretical points of view (see the book of Aarts and Lenstra [2]).

Yet, the theoretical analysis of this algorithm for clustering problems and TSP is very unsatisfactory. In 2004, Arya et al. [16] and Kanungo et al. [112] showed strong theoretical guarantees but still quite far from the performances experienced in practice. Similarly, the theoretical understanding of local search for TSP does not fit the experimental results obtained by Johnson et al. during the DIMACS Challenge.

Therefore, the first question we tackle is the following.

Is it possible to prove better performance guarantees for local search on real-world instances?

[1]The centers *induce* the clustering: each input point is *assigned* to its closest center, thus defining a partition.

[2]For example, the tour that starts from a city A goes to a city B, then to a city C and goes back to city A can be encoded by the following sequence of pairs, called *edges*: (city A, city B), (city B, city C), (city C, city A).

Contributions of this thesis.

We answer the above question by proving that local search achieves nearly-optimal performances in a variety of types of instances stemming from real-world applications. This leads local search to be the best known theoretical algorithm for several problems.

Several practical instances of both k-means[1] (see Figure 2) and TSP consists of points lying in a 2- or 3-dimensional Euclidean space (see Figure 5).

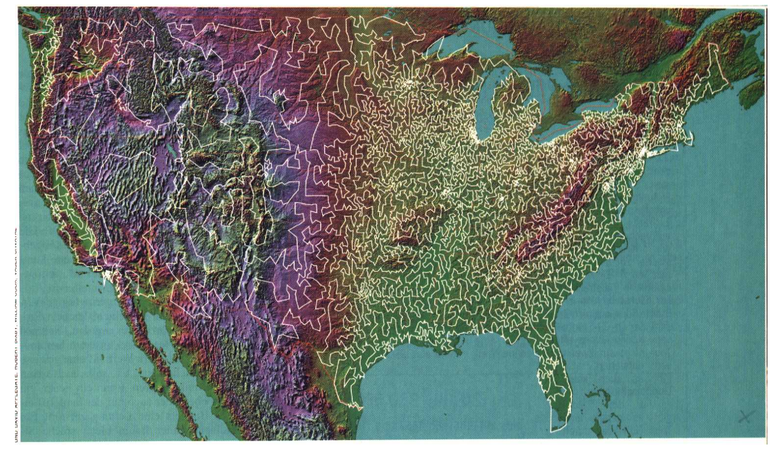


Figure 5: A tour through 13,509 cities of the U.S.A. The instance was created by embedding the cities in $\mathbb{R}^2$. It turns out that the real distances between the cities are very close to their Euclidean distances in the embedding.

However, even for this type of instances, both TSP and k-means are NP-hard (see [137, 140] for example). Thus, the best result one can hope for is a polynomial-time approximation scheme (PTAS) – loosely speaking, an algorithm that approximates the optimal solution within a $1 + \varepsilon$ factor in polynomial time[2] for any constant ε.

First, we observe that local search for TSP could return a solution of cost at least two times the optimal cost in the worst-case, very far from the results observed by Johnson et al. [70] (a factor less than 1.05 roughly). Therefore, we turn to the case of random instances[3] and show that local search is a PTAS in this setting (Theorem 7.1).

For any d-dimensional random instance of TSP, local search with neighborhood of size $1/\varepsilon^{O(d)}$ returns a solution of cost at most $(1 + \varepsilon)$ times the cost of the optimal solution with high probability.

[1]For ease of exposition, we describe the results for the well-known k-means problem. Our results apply for other clustering objectives and to a more general problem called the *k-clustering* problem (Definition A.1)

[2]For ε fixed.

[3]For a formal definition, see Chapter 8.

Giving the first PTAS for k-means for low-dimensional instances is a central problem that has received a lot of attention during the past 15 years (see [19] for example). In this thesis, we solve this problem by showing that local search is a PTAS for d-dimensional instances of the k-means problem (Theorem 1.3).

For any d-dimensional instance of the k-means problem, local search with neighborhood of size $1/\varepsilon^{O(d)}$ returns a solution of cost at most $(1 + \varepsilon)$ times the cost of the optimal solution.

Another significant part of the real-world instances of the k-means problem consists of road networks. Thus, we turn to instances that consist of a planar graph[1] (see Figure 4). Researchers have been trying to find a PTAS for planar instances of k-means since at least 2001 (see [6, 77] for example). We solve this problem by showing that local search is a PTAS for planar instances of the k-means problem (Theorem 1.2).

For any instance of the k-means problem that consists of a planar graph, local search with neighborhood of size $1/\varepsilon^{O(1)}$ returns a solution of cost at most $(1 + \varepsilon)$ times the cost of the optimal solution.

Finally, as shown in Figure 2, clustering data points is a useful tool in data sciences. We then consider various characterizations of real-world instances ([17, 36, 126]). Each characterization is motivated by a particular type of instances stemming from data science. For each of them, an ad-hoc algorithm leveraging the properties induced by the characterization has been designed. We identify three main characterizations of real-world instances: *distribution stability* (Definition 1.4), *perturbation-resilience* (Definition 1.6), and *separability* (Definition 1.9). We show that for all those families of instances local search achieves strong theoretical guarantees (Theorems 1.8, 1.5 and 1.10).

For any instance of the k-means problem that satisfies the stability or separability conditions, local search with neighborhood size $1/\varepsilon^{O(1)}$ returns a solution of cost at most $(1 + \varepsilon)$ times the cost of the optimal solution.
For any instance of the k-means problem that is $(3 + \varepsilon)$-perturbation-resilient, local search with neighborhood size $1/\varepsilon^{O(1)}$ returns the optimal clustering.

Those results make a concrete step towards the understanding of the success of local search for clustering problems and TSP. Indeed, they show that the algorithms used in practice are nearly-optimal *for any* real-world instance of those types. Thus, we may wonder what properties makes local search efficient and which contexts are conductive to this approach. This leads us to our second question.

What structural properties of the instances lead local search to be efficient?

[1]More generally a graph drawn from a minor-closed family of graphs

Clustering (Definition A.1)	Best known approximation guarantee	
	Prev.	**New (by local search)**
d-dimensional Euclidean instances	$1+\varepsilon$ (k-median [11, 123]) 9 (k-means [112])	$1+\varepsilon$
H-minor free graphs (includes planar graphs)	$2.675+\varepsilon$ (k-median [129, 46]) 9 (k-means [112])	$1+\varepsilon$
$\alpha\sqrt{k}$-separability (Def. 1.9)	$1+\alpha$ ([20])	$1+\varepsilon$
β-distribution-stability (Def. 1.4)	$1+\varepsilon$ ([17])	$1+\varepsilon$
γ-perturbation-resilience (Def. 1.6)	Opt if $\gamma \geqslant 1+\sqrt{2}$ ([27])	Opt for $\gamma > 3$

Table 1: Summary of our results on local search for clustering. There is no PTAS in general metrics for this problem unless P = NP. The previous best known approximation for k-means was 9 for general metrics (even in the cases of low dimensional Euclidean space and planar graphs). For k-median, there was already a PTAS in low dimensional Euclidean space. Nothing better than the general case was known for planar graphs. For the main characterizations of real-world instances, we show that local search is better than the state-of-the-art algorithms for the separability condition, matches the performances of the best algorithm for the distribution-stability condition and is a little bit worse than the state-of-the-art algorithm for perturbation-resilient instances.

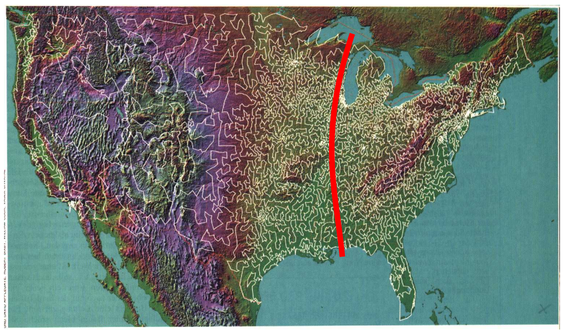
The separation property. The proofs of Theorems 1.3, 1.2, and 7.1 rely on a common key ingredient: the existence of cheap *separators* in the instances. For the problems we consider, both Euclidean and planar instances feature a property that resembles the isoperimetric inequality.

Indeed, each instance contains a set of input elements whose removal yields two sub-instances of the same problem that (1) can be processed independently and (2) whose solutions can be combined at small cost to form a solution to the initial instance.

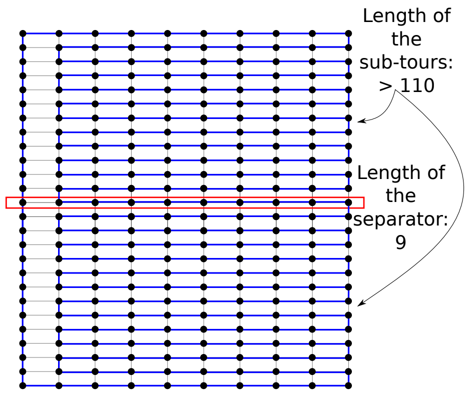
For example, any planar graph with n vertices contains a set of at most $2\sqrt{n}$ vertices whose removal splits the graph into two parts of roughly equal size (see Figure 6).

The existence of cheap separators often implies strong theoretical results (see for example [134]) by providing a decomposition of the instance into small pieces whose solutions can be combined at small cost. Intuitively, this allows to apply very basic techniques such as divide-and-conquer or dynamic programming. It turns out that it is also a very efficient tool to analyze the performance of local search.

An oblivious use of the separation property. Local search uses cheap separators in an *oblivious* manner. As we have seen, local search proceeds naively by enumerating neighboring solutions; it does not require to compute a cheap separator or to decompose the graph. Yet, the analysis shows that the existence of cheap separators implies that any local



(a) The red line splits the instance into two parts. Observe that the length of the tour in each part is much bigger than the length of the red line. Thus, if the traveling salesperson makes a detour from Chicago to New-Orleans and back, its tour is not much longer. Therefore, we can compute the optimal tours on the west and east coasts and combine them by adding, at most, the red line. In this case, the resulting tour will not be much more expensive.



(b) A grid planar graph with 250 vertices and a set of $10 < \sqrt{250}$ vertices that splits the graph into two disconnected parts. The existence of a separator set of at most $\sqrt{n}$ vertices in every planar graph with n vertices was proved by Lipton and Tarjan in 79 [133].

Figure 6: Examples of the separation property and its algorithmic implications.

optimum is close to a global optimum.

Our analysis of local search relies on the existence of a decomposition of the instance into small *regions* that can be dealt with separately. In a nutshell, local search iteratively optimizes a solution by refining its current solution to make it optimal in one of the regions. Since the regions are "small", the modification is minor and so, this solution is indeed a solution neighboring the current solution. Since the regions are (almost) independent from one another this allows local search to end up with a nearly optimal solution.

More structured separators. This leads us to the study of separators and their limitations. Separators are the cornerstone of three central algorithmic paradigms for combinatorial optimization in planar graphs. Those three techniques ([24, 66, 117]) jointly yield most approximation schemes known for planar graphs.

However, we identify a variety of network design problems, for which no PTAS is known and for which those approaches fail. Arguably, the failure lies in the lack of structure of the separator theorems proved so far. For example, when dealing with a network design problem whose solution has to be a connected network, a *good* separator often needs to be both connected and of small length so that combining two solutions can be done by simply adding the separator to the solutions.

We then show how to compute a more structured separator tailored to network design problems (Theorem 7.4). From this we deduce a new framework that yields the first PTAS for several problems on planar graphs. We summarize our results in table 7.1.

Problem	Best known approximation guarantee	
	Prev. (for general graphs)	**New**
(Edge-weights) Tree Cover	2 [149]	$1+\varepsilon$
(Edge-weights) Tour Cover	3 [124]	$1+\varepsilon$
(Vertex-weights) Connected Dominating Set	$O(\log(n))$ [96]	$1+\varepsilon$
(Vertex-weights) Maximum Leaf Spanning Tree	$O(\log(n))$ [96]	$1+\varepsilon$
(Vertex-weights) Connected Vertex Cover	$O(\log(n))$ [90]	$1+\varepsilon$
(Vertex-weights) Feedback Vertex Set	2 [22]	$1+\varepsilon$

Table 2: Summary of our results. None of the problems admit a PTAS in general graphs unless P = NP and the approximation ratios of the Weighted Dominating Set, the Vertex-Weighted Connected Vertex Cover and the Vertex-Weighted Connected Dominating Set problems are $\Omega(\log(n))$ for general graphs assuming P $\neq$ NP. All the problems are NP-hard in planar graphs. Previous to our work, polynomial-time approximation schemes were known [66] for the unweighted versions of these problems in planar graphs. For each of the weighted versions, the best approximation known before our work was the approximation for general graphs. We obtain PTASs for bounded-genus weighted graphs, except for feedback vertex set, where the algorithm is restricted to weighted planar graphs.

Back to practice. Finally, we run several experiments to analyze the practical performances of local search for the clustering. We compare the local search presented in this chapter to Lloyd's famous algorithm for clustering. We show that for several relevant cases, our algorithm outperforms Lloyd's algorithm and converges to a nearly-optimal solution much faster. We also experimentally study for which contexts our model, the k-means problem, is relevant. Indeed, in some practical cases, the "natural clustering" of the instance induces a "bad" solution for the k-means problem. In those particular cases, the local search solution yields a clustering that is very far from the natural one.

Those experimental observations together with our theoretical results raise several open problem that we summarize in the last parts of this thesis.

Organization of this thesis. Chapter A introduces formal definitions of the problems we study in this thesis and provides some background on the separation property.

Part I is devoted to the clustering problems. We start by giving a proof of the performance guarantees of local search for general instances (Chapter 2). In Chapter 3 we show how to use separation in order to prove Theorems 1.2 and 1.3. Chapter 4 is dedicated

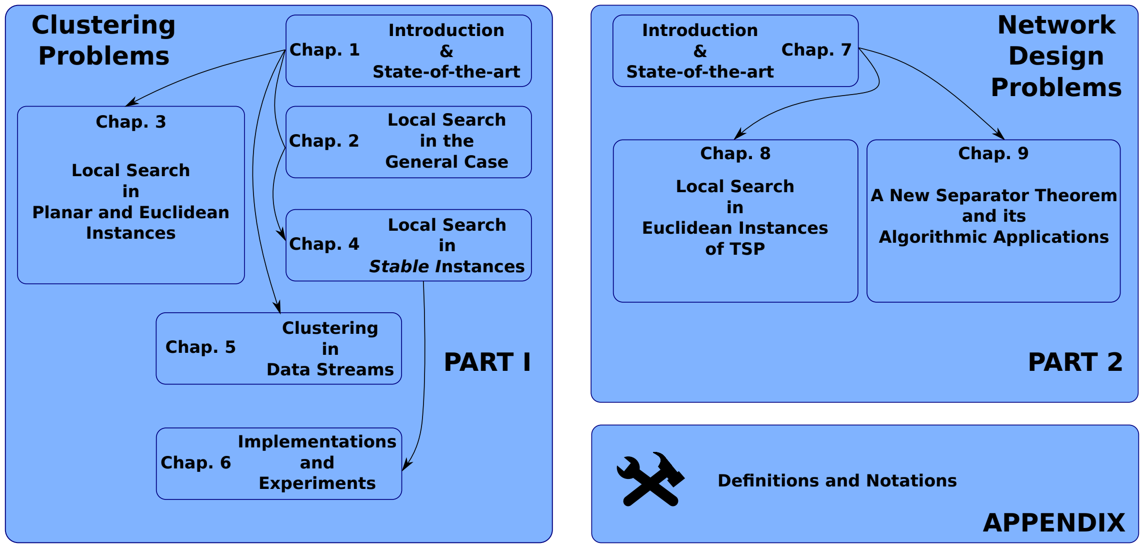

Clustering Problems
Chap. 1 Introduction & State-of-the-art
Chap. 2 Local Search in the General Case
Chap. 3 Local Search in Planar and Euclidean Instances
Chap. 4 Local Search in *Stable* Instances
Chap. 5 Clustering in Data Streams
Chap. 6 Implementations and Experiments
PART I
Network Design Problems
Introduction & State-of-the-art Chap. 7
Chap. 8 Local Search in Euclidean Instances of TSP
Chap. 9 A New Separator Theorem and its Algorithmic Applications
PART 2
Definitions and Notations
APPENDIX

Figure 7: Order of the chapters.

to the various characterizations of real-world instances and to the proof of Theorems 1.8, 1.5 and 1.10. Finally, Chapter 5 introduces an algorithm for the case of massive datasets. In this setting, due to the huge amount of data no separator can be computed and so, we introduce a different greedy approach.

We tackle network design problems in Part II. We prove the efficiency of local search for random d-dimensional instances of TSP (Theorem 7.1) in Chapter 8. We then introduce our new framework for network design problems in Chapter 9.

For a global picture of the organization see Figure 7.

Publications of the Author

[1] V. Cohen-Addad, É. Colin de Verdière, P. N. Klein, C. Mathieu, D. Meierfrankenfeld, *Approximating connectivity domination in weighted bounded-genus graphs*, Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016.

[2] V. Cohen-Addad and A. Eden and M. Feldman and A. Fiat, *The Invisible Hand of Dynamic Market Pricing*, To appear in the Proceedings of the 17th ACM Conference on Electronic Commerce, EC 2016.

[3] V. Cohen-Addad, M. Habib, and F. de Montgolfier, *Algorithmic aspects of switch cographs*, Discrete Applied Mathematics, 2016.

[4] V. Cohen-Addad, M. Hebdige, D. Kral, Z. Li, and E. Salgado, *Steinberg's Conjecture is false*, To appear in Journal of Combinatorial Theory, Series B.

[5] V. Cohen-Addad and V. Kanade, *Online Optimization of Smoothed Piecewise Constant Functions*, Submitted, 2016.

[6] V. Cohen-Addad and P. N. Klein and C. Mathieu, *Local search yields approximation schemes for k-means and k-median in Euclidean and minor-free metrics*, To appear in the Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science FOCS 2016.

[7] V. Cohen-Addad, Z. Li, C. Mathieu, and Ioannis Milis, *Energy-Efficient Algorithms for Non-preemptive Speed-Scaling*, Proceeding of the 12th International Workshop on Approximation and Online Algorithms WAOA 2014, Wrocław, Poland, September 11-12, 2014.

[8] V. Cohen-Addad and C. Mathieu, *Effectiveness of Local Search for Geometric Optimization*, Proceedings of the 31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands, 2015.

[9] V. Cohen-Addad, C. Mathieu, and C. Schwiegelshohn, *One Size Fits All: Effectiveness of Local Search on Structured Data*, Submitted, 2016.

[10] V. Cohen-Addad and A. de Mesmay, *A Fixed Parameter Tractable Approximation Scheme for the Optimal Cut Graph of a Surface*, Proceedings of the 23rd Annual European Symposium ESA 2015, Patras, Greece, September 14-16, 2015.

[11] V. Cohen-Addad and C. Schwiegelshohn and C. Sohler, *Diameter and k-Center Clustering in Sliding Windows*, To appear in the Proceeding of 43rd International Colloquium on Automate, Languages and Programming ICALP 2016.

Part I

Separation and Clustering

CHAPTER 1

Introduction to Clustering

Clustering data according to similarity is ubiquitous in computer and data sciences. Similarity between data is often modeled by a distance function: two data points are close if and only if they are similar. This induces a metric space in which each data point is associated to a point of the space. Thus, a clustering according to similarity is a partition of the points such that the distance between two points in the same part, called *cluster*, is small. However, making this problem more formal is hard. From a practitioner's point of view, the appropriateness of a particular clustering depends on the underlying structure of the data. For instance, if the data is assumed to be generated from a mixture of unit Gaussians, the problem of finding the k mixtures generating the points is often modeled by the k-means problem. Yet, even when the data points lie in $\mathbb{R}^d$, this problem is APX-Hard (assuming k and d are part of the input).

This induces a gap between theory and practice: on the one hand, with an appropriate model a benchmark algorithm often yields a good clustering. On the other hand, many clustering objectives are NP-hard to approximate. Thus, to bridge the gap between theory and practice, prior approaches usually proceed in two steps: (1) restrict attention to either data lying in some specific metric spaces or data satisfying some properties characterizing real-world instances and (2) design an algorithm leveraging the properties of the data.

Clustering in the Classic Memory Model

Well-separated instances. As argued in the introduction, clustering problems come up in a variety of contexts. The problem of locating warehouses induces a clustering problem whose instances are very structured since road or transport networks are planar (or embeddable on a surface of small genus) or can be modeled by embedding the points in $\mathbb{R}^2$ or $\mathbb{R}^3$. Thus, it is fairly natural to restrict attention to inputs consisting of metrics induced by, more generally, minor-free graphs or points lying in Euclidean spaces of small dimensions.

Researchers have been trying to find a polynomial-time approximation scheme for the planar restriction of facility location (see Definition A.2) for many years. An unpublished manuscript by Ageev [6] dating back at least to 2001 addressed the planar case via a straightforward application of Baker's method [24], giving an algorithm whose performance on an instance depends on how much of the cost of the optimal solution is the cost of opening a facility. Despite the title of the manuscript[1], the algorithm is *not* an approximation scheme for instances with arbitrary weights. Since then there have been no results on the problem despite efforts by several researchers in the area (*e.g.,* [77]). We tackle more general instances consisting of graphs excluding a fixed minor or d-dimensional Euclidean space for constant d and a more general problem, the k-clustering problem.

Definition A.1 (k-Clustering)**.** *Given a finite set of* clients $A \subseteq \mathcal{A}$*, a set of* candidate centers $F \subseteq \mathcal{A}$*, two positive integers k and p, the k-*clustering problem *asks for a set of* centers $S \subseteq F$*, of cardinality at most k, that minimizes*

$$\mathit{cost}(S) = \sum_{x \in A} \min_{c \in S} \mathit{dist}(x, c)^p.$$

Instances of the k-clustering problem consisting of graphs excluding a fixed minor or d-dimensional Euclidean space are known to have *small* separators (see Appendix A for a more formal definition). However, previous work failed in applying the most successful algorithmic approaches based on the existence of small separators.

We show that local search (Algorithm 2) can take advantage of those properties, for both graph and Euclidean instances. As far as we know, this is the first algorithm that has strong theoretical guarantees for both settings simultaneously.

Algorithm 2 Local search for finding k clusters.

```
1: Input: A metric space and associated cost function $\text{cost}(\cdot)$, an $n$-element set $C$ of
   points, error parameter $\varepsilon > 0$, number of clusters $k$
2: Parameter: A positive integer parameter $s$ for the neighborhood size
3: $S \leftarrow$ Arbitrary size-$k$ set of points
4: while $\exists\, S'$ s.t. $|S'| \leqslant k$ and $|S \setminus S'| + |S' \setminus S| \leqslant s$ and $\text{cost}(S') \leqslant (1 - \varepsilon/n)\text{cost}(S)$
5:  do
6:      $S \leftarrow S'$
7: Output: $S$
```

The following lemma has been used in several context and shows that Algorithm 2 runs in polynomial time.

Lemma 1.1 ([16, 112])**.** *If the initial solution has cost at most $c^{poly(n)}$OPT, then the running time of Algorithm 2 is at most $n^{O(c+s)}/\varepsilon$.*

[1]"An approximation scheme for the uncapacitated facility location problem on planar graphs"

Beyond well-separated instances: other real-world instances. Planar and Euclidean instances are not the only practical instances. Machine learning and database systems make use of clustering to extract useful information from the data. Thus, bridging the gap between theory and practice for clustering instances stemming from those fields has led to a large body of work during the last decade. Several groups of researchers have come up with properties that aim at characterizing those instances, together with various algorithms making use of the properties induced. Thus, the characterization of real-world instances has resulted in new, ad-hoc algorithms that bypass NP- and APX-hardness results.

From practice to theory. At this point, there is a wide variety of (1) characterizations of well-clusterable instances and (2) algorithms tuned to those instances. In contrast, we proceed in the reverse order: (1) focus on a single, all-purpose, easy-to-implement algorithm that is already widely used in practice: local search (Algorithm 2)[1], and (2) prove that it works well for both well-clusterable and well-separated instances for the k-clustering problem.

Beyond the Classic Memory Model: Clustering in Data Streams

The increasing size of datasets has made clustering in data streams an important problem that has received considerable theoretical and practical attention. For example, clustering data *on the fly*, as new elements are inserted into the database allows to obtain useful information at any time at low memory cost. Hence, in the standard streaming setting, algorithms are constrained to use as little space as possible while computing high-quality solutions. The complexity of clustering is well understood for insertion-only streams where input points arrive one by one. The more general settings, like *dynamic streams* and the *sliding window* model, have also recently received some attention for other clustering objectives. Both generalizations aim to incorporate dynamic behavior; in dynamic streams input points are removed via a dedicated delete operation and in the sliding window model older elements expire as new ones arrive.

We focus on maintaining a k-center clustering in the sliding window model. We consider a greedy approach and prove strong theoretical guarantee, showing that the approach is almost optimal.

Contributions and Techniques

Well-separated instances. We distinguish two kinds of well-separated instances. First, we address instances of the k-clustering problem that consists of an edge-weighted graph G belonging to a fixed nontrivial minor-closed family of graphs[2]. We then consider the

[1] See [112] or Chapter 6 for a experimental study and comparison with other heuristics.

[2] A family is nontrivial if it contains a graph with a least one edge.

metric completion of G, i.e., the metric space whose points are the vertices of G and where the distance between two vertices u and v is the length of the shortest u-to-v path in G with respect to the given edge-weights.

In this setting we show that local search yields a polynomial-time approximation scheme for the k-clustering problem. This is the first PTAS for both k-median and k-means in this setting.

Theorem 1.2 (Graphs – Chapter 3, Section 3.4)**.** *Let $\mathcal{K}$ be a nontrivial minor-closed family of edge-weighted graphs. For any integer $p > 0$, there is a constant c such that the following holds. For any $0 < \varepsilon < 1/2$, for any graph $G = (V, E)$ in $\mathcal{K}$ with clients set $A \subseteq V$ and set of centers locations $F \subseteq V$, Algorithm 2 applied to the metric completion of G with cost function*

$$cost(S) = \sum_{a \in A} (\min_{u \in S} dist(a, u))^p$$

and with $s = 1/\varepsilon^{cp}$ outputs a solution S whose cost is at most $1 + \varepsilon$ times the cost of the optimal solution to the k-clustering problem with parameter p.

Second, we consider the instances of the k-clustering problem where $A \subset \mathbb{R}^d$ and $F \subseteq \mathbb{R}^d$ for some fixed d equipped with Euclidean distance. In this thesis, we are mainly interested in polynomial-time algorithms. In order for local search to run in polynomial-time, the set of candidate centers F needs to be of polynomial size. We define a pair of finite sets $A, F \subseteq \mathbb{R}^d$ to be an *ε-discretization* of a d-dimensional Euclidean space for the k-clustering problem if F is of size polynomial in the size of A and $\text{OPT}(A, F) \leqslant (1 + \varepsilon)\text{OPT}(A, \mathbb{R}^d)$, where $\text{OPT}(S_0, S_1)$ denote the value of the optimal solution for the k-clustering problem on the instance (S_0, S_1) equipped with Euclidean distance. There is an important literature on finding good candidate centers since this is a common preprocessing step for many algorithms (see *e.g.,* [138]).

We also show that local search is a PTAS for the k-clustering problem in this setting. Again, this is the first PTAS for k-means in this setting.

Theorem 1.3 (Euclidean Spaces – Chapter 3, Section 3.5)**.** *For any fixed integers $p, d > 0$, there is a constant c such that the following holds. For any $0 < \varepsilon < 1/2$, client set $A \subset \mathbb{R}^d$, candidate centers $F \subset \mathbb{R}^d$, Algorithm 2 applied to A, F with*

$$cost(S) = \sum_{u \in C} (\min_{f \in S} dist(u, f))^p$$

and $s = 1/\varepsilon^{cpd}$ yields a solution S whose cost is at most $1 + \varepsilon$ times cost of the optimal solution to the k-clustering problem with parameter p.

The proofs of the two theorems are very similar. The first key ingredient in our analysis is the existence of a certain kind of decomposition of the input called *weak r-division*. The concept (in a stronger form) is due to Frederickson [86] in the context of planar graphs. It is straightforward to extend it to any family of graphs with balanced separators of sublinear size. Building upon the work of Bhattiprolu and Har-Peled [34], we also define a weak r-division for points in a Euclidean space, and show that such a decomposition always exists.

The second ingredient is the notion of *isolation*. As we will see in Section 3.4, a simple analysis shows that a local search algorithm that is allowed to pick up to $(1+\varepsilon)k$ centers yields a solution of cost at most $(1+\varepsilon)$ times the optimal solution with k centers. When looking at this analysis, it seems hard to avoid an increase in the number of centers when comparing a locally optimal solution to a globally optimal solution.

The idea then is to compare the output L of the local search algorithm to a globally optimal solution with *fewer* than k centers. However, it is easy to see that, for some instances the cost of such a solution might be significantly larger than the cost of the optimal solution using k centers and so, comparing the cost of L to the cost of such a solution is useless. When this happens though, the intuition is that the clusters are *obvious* or rather very far from one another and so, local search can find them efficiently.

Our analysis leverages these observations, proving that local search is a $(1+\varepsilon)$ approximate solution. It relies on finding pairs of centers of L and OPT that serve roughly the same set of clients in the two solutions. This yields a structure theorem that could have applications beyond the analysis of local search.

Well-clusterable instances. We then consider instances stemming from data analysis and machine learning. Those instances are often call well-clusterable because they are assumed to possess a "natural" clustering that needs to be identified. Various characterizations of well-clusterable instances, called *stability conditions*, have been proposed (see Figure 1.1). The three incomparable notions that have received the most attention are *distribution stability*, *perturbation resilience*, and *spectral separability*.

In one of the earliest attempts to formalize the notion of a real-world instance, Ostrovsky et al. [151] assumed that the cost of an optimal k-clustering of the instance is smaller than an ε-fraction of the cost of an optimal clustering with $k-1$ centers. It is motivated by the commonly used "elbow method" of determining the correct value of k: run an algorithm for an incrementally increasing number of clusters until the cost drops significantly. The condition of Ostrovsky et al. was later generalized to the distribution stability by Awashti et al. [17]. We state the latter condition.

Definition 1.4 (Distribution Stability [17]). *Let $(A, F, cost, k)$ be an input for the k-clustering problem and let $\{C_1^*, \dots, C_k^*\}$ denote the optimal k-clustering of A with centers $S = \{c_1^*, \dots c_k^*\}$. Given $\beta > 0$, the instance is β-*distribution stable *if, for any i, for any $x \notin C_i^*$,*

$$cost(x, c_i^*) \geqslant \beta \frac{OPT}{|C_i^*|}.$$

A PTAS for β-distribution stable instances was previously given by Awasthi et al. [17]. Guided by our work on isolation, we show that local search is already a PTAS: no ad-hoc

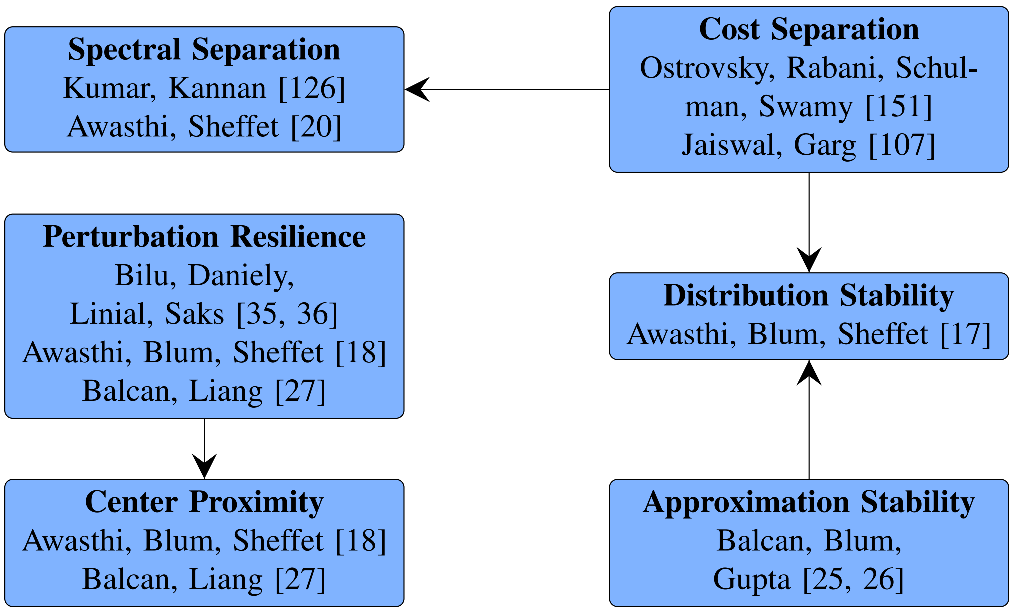

Spectral Separation
Kumar, Kannan [126]
Awasthi, Sheffet [20]
Cost Separation
Ostrovsky, Rabani, Schulman, Swamy [151]
Jaiswal, Garg [107]
Perturbation Resilience
Bilu, Daniely, Linial, Saks [35, 36]
Awasthi, Blum, Sheffet [18]
Balcan, Liang [27]
Distribution Stability
Awasthi, Blum, Sheffet [17]
Center Proximity
Awasthi, Blum, Sheffet [18]
Balcan, Liang [27]
Approximation Stability
Balcan, Blum, Gupta [25, 26]

Figure 1.1: An overview over all definitions of well-clusterability. Arrows correspond to implication. For example, if an instance is cost-separated then it is distribution-stable; therefore the algorithm by Awasthi at al. also works for cost-separated instances.

algorithm is needed. Moreover, β-distribution stability is also implied by "cost separation" as defined by Ostrovsky et al. [151], so local search is also a PTAS in that setting.

Theorem 1.5 (β-Distribution Stability – Chapter 4, Section 4.2)**.** *Let $\beta, p > 0$. There exists a constant c such that the following holds. For any $0 < \varepsilon < 1/2$, for any β-stable instance, the solution output by local search with parameter $s = c^p\varepsilon^{-3}\beta^{-1}$ (Algorithm 2) has cost at most $(1+\varepsilon)$ time the cost of the optimal solution to k-clustering problem with parameter p.*

Furthermore, we are able to show that local search is efficient for a slightly more general definition of β-distribution stability (see Chapter 4).

We now turn to the second notion of stability. This definition, due to Awashti et al. [18], adapts the definition of stability used by Bilu and Linial [36] for the max-cut problem to the k-clustering problem. It allows to characterize instances for which a small perturbation of the data (due, for example, to measurement errors) does not change the natural clustering of the data.

Definition 1.6 (Perturbation Resilience [18])**.** *Let $(A, F, cost, k)$ be an input for the k-clustering problem and let $\{C_1^*, \dots, C_k^*\}$ denote the optimal k-clustering of A with centers $S = \{c_1^*, \dots c_k^*\}$. Given $\alpha \geqslant 1$, the instance is* α-perturbation-resilient *if for any cost* *function cost' on A with*

$$\forall\ (p,q) \in A \times F,\ cost(p,q) \leqslant cost'(p,q) \leqslant \alpha cost(p,q),$$

$\{C_1^*,\dots,C_k^*\}$ is the unique optimal clustering of the instance $(A,F,cost',k)$.

To make this more formal we introduce the notion of locally-optimal solution. Given a solution S_0 to the k-clustering problem, we say that S_0 is $1/\varepsilon$-*locally optimal* if any solution S_1 such that $|S_0 \backslash S_1| + |S_1 \backslash S_0| \leqslant 2/\varepsilon$ has cost at least $\text{cost}(S_0)$. We derive the following theorem.

Theorem 1.7 (α-Perturbation Resilience – Chapter 4, Section 4.1)**.** *For any $p > 0$, there exists $\alpha_p > 0$ such that, for any instance of the k-clustering problem with parameter p if the instance is α_p-perturbation-resilient then any $f(\alpha_p)$-locally optimal solution is the optimal clustering $\{C_1^*,\dots,C_k^*\}$, for some function f.*

This notion of stability was mainly defined for cost $=$ dist. In this case, we show that for any instance that is α-perturbation-resilient for $\alpha > 3$, for any large enough neighborhood size, a local optimum must be a global optimum as well.

Theorem 1.8. *Let $\alpha > 3$. For any instance of the k-median problem that is α-perturbation-resilient, any $2(\alpha-3)^{-1}$-locally optimal solution is the optimal clustering $\{C_1^*,\dots,C_k^*\}$.*

An optimal algorithm was already known for $1+\sqrt{2}$-perturbation resilient k-median instances [27]. Our contribution is to show that local search is already optimal for $3+\varepsilon$-perturbation resilient instances; no ad-hoc algorithm is needed.[1]

We now turn to the third stability condition. In the Euclidean setting, Kumar and Kannan [126] introduced a condition under which a target clustering may be found. There have been several results on clustering data points generated from a mixture of k probability distributions, under the assumption that the means of those distributions are far enough from one another. Kumar and Kannan [126] showed that assuming such a *proximity condition*, one can retrieve the underlying clustering of the data. The spectral separation defined below generalizes the proximity condition of Kumar and Kannan.

Definition 1.9 (Spectral Separation [126])**.** *Let $(A,\mathbb{R}^d,||\cdot||^2,k)$ be an input for k-means clustering in Euclidean space and let $\{C_1^*,\dots C_k^*\}$ denote an optimal clustering of A with centers $S = \{c_1^*,\dots c_k^*\}$. Denote by C an $n \times d$ matrix such that the row $C_i = \underset{c_j^* \in S}{argmin}\ ||A_i - c_j^*||^2$. Denote by $||\cdot||_2$ the spectral norm of a matrix. Then $\{C_1^*,\dots C_k^*\}$ is γ-*spectrally separated*, if for any pair (i,j) the following condition holds:*

$$||c_i^* - c_j^*|| \geqslant \gamma \cdot \left(\frac{1}{\sqrt{|C_i^*|}} + \frac{1}{\sqrt{|C_j^*|}}\right)||A-C||_2.$$

[1] We do not quite match [27]: One limitation is that local search is not necessarily optimal for $(1+\sqrt{2})$-perturbation resilient instances, see Proposition 4.1

Since this stability is defined over non-finite metric spaces, we again require standard preprocessing steps in order to reduce the number of dimensions and discretize the space in order to bound the number of candidate centers. We obtain the following theorem.

Theorem 1.10 (δ-Spectral Separation – Chapter 4, Section 4.4)**.** *Let $(A, \mathbb{R}^d, ||\cdot||^2, k)$ be an instance of Euclidean k-means clustering with optimal clustering $C = \{C_1^*, \dots C_k^*\}$ and centers $S = \{c_1^*, \dots c_k^*\}$. If C is more than $3\sqrt{k}$-spectrally separated, then Algorithm 4 (which consists in projecting the points and before applying local search (Algorithm 2)) is a polynomial time approximation scheme.*

In previous work by Kumar and Kannan [126], an algorithm is given with approximation ratio $1 + O(\text{OPT}_k/\text{OPT}_{k-1})$, where OPT_i denotes the value of the optimal clustering with i clusters. Assuming that $\text{OPT}_k/\text{OPT}_{k-1} \leqslant \varepsilon$ implies that the optimal k-clustering C is $\Omega(\sqrt{k/\varepsilon})$-spectrally separated [126]. Thus our assumption in Theorem 1.10, that C is $\sqrt{k}$-spectrally separated is weaker (it does not depend on ε) and therefore our result is stronger since the approximation guarantee does not depend on the properties of the instance.

Clustering in data streams In the context of massive datasets, different algorithms are needed. Indeed, it is not possible to store all the input points in memory while performing computations. Therefore, we are interested in algorithms that only stores a very small number of elements at the expense of a slightly worse approximation ratio. In the sliding window model (see Chapter 5 for a formal definition), we are given an integer W which represents the *window size* and a stream of data (input points in a metric space in our case). The goal is to maintain over time a solution to a problem where the input consists of the last W elements of the stream (the element that are in the "window"). Additionally, the algorithm is required to use a memory of size sublinear[1] in W.

Since the main applications of clustering in data streams lie in machine learning and databases, we aim at avoiding any dependency on the number of dimensions. We will assume an *oracle* distance function: given two points of the metric space, the oracle can provide at any time of the execution the distance between those two points. Then, the goal our the algorithm is to store the most relevant points for our problem.

We focus on the problem of computing the diameter of a set of points (the maximum distance between two points) and the *k-center* problem where we aim to find k points such that the maximum distance over all points to their closest center is minimized. For Euclidean space this is equivalent to finding k spheres of minimum radius containing the entire point set. Throughout this section, we define the aspect ratio of a set of points $\mathcal{A}$ in a metric space to be $\max_{a,b\in\mathcal{A}} \text{dist}(a,b)/\min_{a\neq b\in\mathcal{A}} \text{dist}(a,b)$.

For the diameter problem, we obtain the following theorem.

[1]Note that for some problems an exact solution can be computed trivially using memory of size $O(W)$.

Theorem 1.11 (Diameter in Sliding Window – Chapter 5, Section 5.1)**.** *Given a set of points A with aspect ratio α, there exists an algorithm computing a $(3+\varepsilon)$-approximate solution for the metric diameter problem storing at most $O(\varepsilon^{-1}\log\alpha)$ points in memory. The update time is $O(\varepsilon^{-1}\log\alpha)$.*

This is a substantial improvement over the best sliding window algorithm for diameter in general metric spaces by Chan and Sadjad [50] which computes a $(2^{m+2}-2+\varepsilon)$-approximation using $O(W^{1/(m+1)}\log\alpha)$ memory. Observe that the memory dependency in $\log\alpha$ is inherent to the model: Feigenbaum et al. [82] showed that even in the case of 1-dimensional Euclidean space, for any algorithm using less than $\Omega(\log W\log\alpha)$ memory there exists an instance in which the approximation ratio of the algorithm is arbitrarily bad. Moreover, we obtain a lower bound of $\Omega(\sqrt[2]{W})$ for any deterministic algorithm achieving a $(3-\varepsilon)$-approximation to the diameter problem for any $\varepsilon>0$. Together with the result of Feigenbaum et al. [82], this implies that our algorithm is tight up to a $O(1/\varepsilon)$ factor.

To our knowledge there exists no previous work on k-center in the sliding window model. For the 2-center problem our diameter algorithm yields a $(4+\varepsilon)$-approximate clustering. We are also able to obtain a matching lower bound. For arbitrary values of k, we obtain the following theorem.

Theorem 1.12 (k-center in Sliding Window – Chapter 5, Section 5.2)**.** *Given a set of points A with aspect ratio α, there exists a $(6+\varepsilon)$-approximation algorithm for the metric k-center problem storing at most $O((k+1)\varepsilon^{-1}\log(\alpha))$ points in memory. The update time is $O(k^3\varepsilon^{-1}\log\alpha)$.*

We obtain a lower bound of 6 for the k-center problem, with $k>2$.

The techniques are very basic and different from the approaches used in the other chapters. Given an *estimate* of the value of the optimal solution, the algorithm either finds a solution of this value or provide a *certificate* that our estimate is incorrect. For example, in the case of the diameter, given an estimate γ of the optimal value of the instance, the goal of an ideal algorithm would be to either provide two points at distance at least γ or provide a point that is at distance less than $\gamma/2$ to every point of the instance. Then, the triangular inequality ensures that no two points are at distance at least γ from each other.

We will see that such an ideal algorithm is required to store at least $\Omega(\sqrt{n})$ points. Thus, we relax the condition on the certificate and show how to derive an algorithm satisfying the conditions of Theorem 1.11.

State-of-the-Art

Clustering problems have been extensively studied from an algorithmic perspective since, at least, the 60s. The heterogeneity of the contexts in which clustering problems arise has led various communities to develop ad-hoc algorithms. In this section, we review the general results that are particularly relevant for our work.

In the Classic Memory Model

Clustering problems are NP-hard: k-median and k-means are already NP-hard in the Euclidean plane (see Meggido and Supowit [140], Mahajan et al. [137], and Dasgupta and Freud [65]). In terms of hardness of approximation, both problems are APX-hard, even in the Euclidean setting (when both k and d are part of the input) (see Gua and Khuller [95], Jain et al. [105], Guruswami et al. [100] and very recently Awasthi et al. [19] for the first APX-hardness proof for Euclidean k-means when both k and d are part of the input).

On the positive side, constant factor approximations are known in metric space for both k-median and k-means (see for example [129, 106, 142]). The current best approximation factor for metric k-median is $2.675 + \varepsilon$ (see Byrka et al. [46] improving upon Li and Svensson [129].)

Several algorithms that bypass those hardness results for restricted inputs have been developed. For example PTASs have been proved, for Euclidean k-means and fixed k (see [83, 127] for example), or Euclidean space and fixed d [12, 123, 101, 102], or some stability assumption on the input structure such as cost separation, approximation stability, perturbation resilience, or spectral separability. However, as far as we know, there is no algorithm achieving better bounds than the general ones for well-separated instances yet.

Cost Separation The condition by Ostrovsky et al. [151] represent one of the earliest attempt to formally define real-world instances, see also Schulman [169] for an earlier condition for two clusters and the irreducibility condition by Kumar et al. [127]. An instance I satisfies the ε-"ORSS" condition if the cost of a solution to the instance using k centers is smaller than ε times the cost of the optimal solution to the instance using $k-1$ centers. This condition has several appealing properties. It is robust towards small perturbations of the data set and it implies that two low-cost clusterings agree on a large fraction of points. Moreover, the popular D^2 sampling technique (also known as k-means++) has an improved performance for cost separated instances compared to the worst-case $O(\log k)$-approximation ratio [14, 45, 107, 151]. A related, slightly weaker condition called weak-deletion stability was introduced by Awasthi et al. [17] where the cost of assigning all the points from one cluster in the optimal k-clustering to another center increases the objective by some factor $(1 + \alpha)$.

Perturbation Resilience The motivation behind this stability condition is that bounded modifications to the input should not affect the optimum solution. Bilu et al. [36, 35] formalized this as allowing edge weights in a graph to be modified by a factor of at most γ without changing the structure of the max-cut. Perturbation resilience has some similarity to smoothed analysis (see Arthur et al. [13, 15] for work on k-means). The main difference is that smoothed analysis takes a worst case instance and applies a random perturbation, while perturbation resilience takes a well-behaved instance and applies an adversarial perturbation. For results using perturbation resilience, see [18, 27, 32].

Spectral Separation Kumar and Kannan [126] introduced a deterministic condition under which a target clustering may be found in Euclidean spaces as opposed to the probabilistic assumptions used by previous work. For other deterministic conditions we refer to [3, 60]. Viewing each data point as a row of a matrix A and the rows of the center matrix K containing the centroid of the respective target cluster of A, they impose a proximity condition on each point A_i when projected onto the line connecting its target centroid c_j and some other centroid c_ℓ. The condition requires that the projection of A_i is closer to c_j than to any c_ℓ by a factor of $\Omega\left(k \cdot \left(\frac{1}{\sqrt{|C_j|}} + \frac{1}{\sqrt{|C_\ell|}}\right) \cdot ||A - K||_2\right)$, where $||A - K||_2$ is the spectral norm and C_j and C_ℓ are the target clusters. This condition implies spectral separability with $\gamma \in \Omega(k)$. See also [20].

Thus, the various definitions are attempts to model real-world inputs by adding stability assumptions; in this paper, we analyze a real-world algorithm, local search (see [2]). There is much prior research on analyzing local search without stability assumptions.

Local search There exist a large body of bicriteria approximations for k-median and k-means [125, 53, 28]. Arya et al. [16] (see also [99]) gave the first analysis showing that local search with a neighborhood parameter $s = 1/\varepsilon$ gives a $(3 + 2\varepsilon)$-approximation to k-median and showed that this bound is tight. Kanungo et al. [112] proved an approximation ratio of $9 + \varepsilon$ for k-means clustering by local search, currently the best known algorithm with polynomial running time in metric and Euclidean spaces[1]. For further very recent results on local search, we refer to [89]. Due to its simplicity, local search is also a popular subroutine for clustering tasks in various computational models [38, 97, 30].

In Data Streams

The diameter problem. In the streaming metric distance oracle model, there exists a naive 2-approximation algorithm for the diameter problem which consists in maintaining the first point p of the stream and the point with maximum distance from p inserted after p. Guha [93] showed that this algorithm is essentially optimal: no algorithm storing fewer than $\Omega(n)$ points can achieve a ratio better than $2 - \varepsilon$ for any $\varepsilon > 0$. For Euclidean spaces, the best streaming algorithm with a polynomial dependency on d is due to Agarwal and Sharathkumar [5] with an almost tight approximation ratio of $\sqrt{2} + \varepsilon$ in $O(d\varepsilon^{-3} \log(1/\varepsilon))$ space. Agarwal et al. [4] proposed a $(1 + \varepsilon)$-approximation using $O(\varepsilon^{-(d-1)/2})$ points.

Feigenbaum et al. [82] were the first to consider the diameter in the sliding window model. For d-dimensional Euclidean space, their algorithm uses $O(\varepsilon^{-(d+1)/2} \log^3 W(\log \alpha + \log\log W + \varepsilon^{-1}))$ bits of space. They also give a lower bound of $\Omega(\varepsilon^{-1} \log W \log \alpha)$ for

[1]They use an algorithm from Matousek [138] that "discretizes" $\mathbb{R}^d$, *i.e.,* finds a good set of candidate centers. However, as stated, the running time of their algorithm has an exponential dependency in d. As discussed earlier, this is an important drawback for several applications, we show in Chapter 4 how to circumvent this issue by using the celebrated Johnson-Lindenstrauss lemma.

a $(1+\varepsilon)$-approximation factor in one dimension and, implicitly, a $\Omega(\log W \log \alpha)$ space bound for any multiplicative approximation factor. This lower bound was later matched by Chan and Sadjad [50], who also gave an improved space bound of $O(\varepsilon^{-(d+1)/2}\log(\alpha/\varepsilon))$ points for higher dimensions. For more general metric spaces, they obtain a $(2^{m+2}-2+\varepsilon)$-approximation with $O(W^{1/(m+1)})$ points.

The k-Center problem. In one of the earliest works on clustering in streams, Charikar et al. [52] gave a number of incremental clustering algorithms for metric k-center, among other results. While storing no more than $k+1$ points at any given time, they were able to derive a deterministic 8-approximation and a randomized $2e \approx 5.437$-approximation. They also show that no incremental algorithm can be better than 3. McCutchen and Khuller [139] and Guha [93] independently derived a $(2+\varepsilon)$-approximate algorithm using $O(k/\varepsilon \log 1/\varepsilon)$ space, with Guha giving an almost tight lower bound of $\Omega(n)$ space for any algorithm achieving a better approximation ratio than 2. In their paper, McCutchen and Khuller [139] also studied the problem with z outliers, giving a $(4+\varepsilon)$-approximate algorithm that stores $O(\varepsilon^{-1}kz)$ points, see also Charikar et al. [54] for an earlier treatment of the problem. Further improvements are possible in Euclidean spaces see Zarrabi-Zadeh [177] or Kim and Ahn [116].

This is the first attempt to address the k-center problem in the sliding window model. For k-median and k-means, we remark that Babcock et al. [21] and more recently Braverman et al. [43, 44] gave an $O(1)$-approximation for the metric case and a $(1+\varepsilon)$-approximation for the Euclidean case.

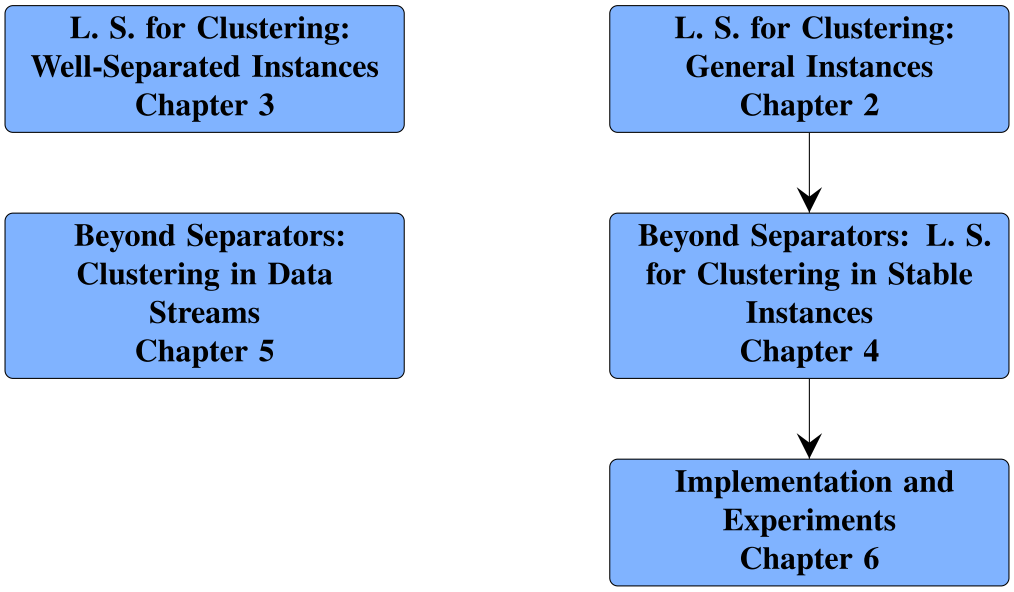

L. S. for Clustering: Well-Separated Instances Chapter 3
L. S. for Clustering: General Instances Chapter 2
Beyond Separators: Clustering in Data Streams Chapter 5
Beyond Separators: L. S. for Clustering in Stable Instances Chapter 4
Implementation and Experiments Chapter 6

Figure 1.2: Organization of this part.

Corresponding Publications

[1] V. Cohen-Addad and P. N. Klein and C. Mathieu, *Local search yields approximation schemes for k-means and k-median in Euclidean and minor-free metrics*, To appear in the Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science FOCS 2016.

[2] V. Cohen-Addad and C. Mathieu, *Effectiveness of Local Search for Geometric Optimization*, Proceedings of the 31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands, 2015.

[3] V. Cohen-Addad, C. Mathieu, and C. Schwiegelshohn, *One Size Fits All: Effectiveness of Local Search on Structured Data*, Submitted, 2016.

[4] V. Cohen-Addad and C. Schwiegelshohn and C. Sohler, *Diameter and k-Center Clustering in Sliding Windows*, To appear in the Proceeding of 43rd International Colloquium on Automate, Languages and Programming ICALP 2016.

CHAPTER 2

Local Search for Clustering: General Instances

In this chapter, we consider the following general k-clustering problem.

Definition A.1 (k-Clustering)**.** *Given a finite set of* clients $A \subseteq \mathcal{A}$*, a set of* candidate centers $F \subseteq \mathcal{A}$*, two positive integers k and p, the* k-clustering problem *asks for a set of* centers $S \subseteq F$*, of cardinality at most k, that minimizes*

$$\textit{cost}(S) = \sum_{x \in A} \min_{c \in S} \textit{dist}(x, c)^p.$$

We prove the following structural theorem (Theorem 2.1) about local optima. It can be seen as a strengthening of Theorem 2.2 (see below) that states that local search is a constant factor approximation for the k-clustering problem and which was already proven in [99] who showed that local search achieves an $O(p)$-approximation. We will see in Chapter 4, how Theorem 2.1 implies any local optimum is actually a global optimum for perturbation resilient instances.

Consider a solution S_0 to the k-clustering problem with parameter p. We say that S_0 is $1/\varepsilon$-*locally optimal* if any solution S_1 such that $|S_0 \backslash S_1| + |S_1 \backslash S_0| \leqslant 2/\varepsilon$ has cost at least $\text{cost}(S_0)$. Furthermore, for each client a, we denote by g_a and ℓ_a the costs induced by client a in solution $\mathcal{C}$ and $\mathcal{L}$ respectively. Finally, for any $S' \subseteq S_0$, denote by the $N_{S_0}(S')$ the set of clients that are closer to a center of S' than to any other center in S_0. We refer to those clients as the clients *served* by S' in solution S_0.

Theorem 2.1 (Local-Approximation Theorem.)**.** *Let $\mathcal{L}$ be a $1/\varepsilon$-locally optimal solution and $\mathcal{C}$ be any solution to the k-clustering problem with parameter p. Let $S = \mathcal{L} \cap \mathcal{C}$ and $\tilde{\mathcal{L}} = \mathcal{L} \backslash S$ and $\tilde{\mathcal{C}} = \mathcal{C} \backslash S$. Then, there exists $\gamma_{\varepsilon,p}$ depending on ε and p such that*

$$\sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}})} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}})} g_a + \gamma_{\varepsilon,p} \sum_{a \in N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_a.$$

In other words, given a locally optimal solution $\mathcal{L}$ and a solution $\mathcal{C}$, the theorem states that the cost of the clients served by a center of the solution $\mathcal{C}$ that is "not found" by the algorithm, *i.e.,* not in solution $\mathcal{L}$, is at most its cost in $\mathcal{C}$ plus $\gamma_{\varepsilon,p}$ times the costs in $\mathcal{C}$ of the clients served by the "incorrect" centers in $\mathcal{L}$. Note that this is not necessarily the case for any $\gamma_{\varepsilon,p}$-approximate solution.

Obviously, for the clients served by the "correctly" guessed centers, their total cost is at most their cost in OPT. Thus, this implies that local search is an $O(\gamma_{\varepsilon,p})$-approximation for the k-clustering problem. This is the purpose of the following theorem. Loosely speaking, Theorem 2.2 states that any $\Theta(\gamma_{\varepsilon,p})$-locally optimal solution is an $O(\gamma_{\varepsilon,p})$-approximation for the k-clustering problem.

Theorem 2.2 (General form of [99])**.** *Let $\mathcal{L}$ be a $1/\varepsilon$-locally optimal solution and $\mathcal{C}$ be any solution to the k-clustering problem with parameter p. There exists a $\gamma_{\varepsilon,p}$ such that cost($\mathcal{L}$) $\leqslant (\gamma_{\varepsilon,p}+1)$cost($\mathcal{C}$).*

Proof of Theorem 2.2. Consider a client $a \notin N_{\mathcal{C}}(\tilde{\mathcal{C}})$, it is served in solution $\tilde{\mathcal{C}}$ by a center in S. Since $S \subseteq \mathcal{L}$, its cost in solution $\mathcal{L}$ is at most g_a. Thus,

$$\sum_{a \notin N_{\mathcal{C}}(\tilde{\mathcal{C}})} \ell_a + \sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}})} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}})} g_a + \gamma_{\varepsilon,p} \sum_{a \in N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_a + \sum_{a \notin N_{\mathcal{C}}(S)} g_a.$$

Therefore,

$$\sum_{a \in A} \ell_a \leqslant (\gamma_{\varepsilon,p}+1) \sum_{a \in A} g_a.$$

□

For ease of exposition we proceed to the proof for the k-median problem ($p = 1$). Lemmas 2.8, 2.9 at the end of the chapter imply the results for general values of p. We prove the following theorem.

Theorem 2.3 (Local-Approximation Theorem.)**.** *Let $\mathcal{L}$ be a $1/\varepsilon$-locally optimal solution and $\mathcal{C}$ be any solution to the k-median problem. Define $S = \mathcal{L} \cap \mathcal{C}$, $\tilde{\mathcal{L}} = \mathcal{L} \backslash S$, and $\tilde{\mathcal{C}} = \mathcal{C} \backslash S$. Then,*

$$\sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}})} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}})} g_a + (2+2\varepsilon) \sum_{a \in N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_a.$$

Additionally, we mention that Arya et al. [16] showed that this is tight (see also Figure 3.1).

We now provide some useful lemmas for the proof of Theorem 2.3. We first introduce some definitions, following the terminology of [16, 99]. Consider the following bipartite graph $\Gamma = (\tilde{\mathcal{L}} \cup \tilde{\mathcal{C}}, \mathcal{E})$ where $\mathcal{E}$ is defined as follows. For any center $f \in \tilde{\mathcal{C}}$, we have $(f, \ell) \in \mathcal{E}$ where ℓ is the center of $\tilde{\mathcal{L}}$ that is the closest to f. Denote $N_\Gamma(\ell)$ the neighbors of the point corresponding to center ℓ in Γ.

For each edge $(f, \ell) \in \mathcal{E}$, for any client $a \in N_{\mathcal{C}}(f) \backslash N_{\mathcal{L}}(\ell)$, we define Reassign_a as the cost of a client a after reassignment to ℓ. We derive the following lemma.

Lemma 2.4. *For any client a, Reassign$_a \leqslant \ell_a + 2g_a$.*

Proof. By definition we have Reassign$_a = \text{dist}(a,\ell)$. By the triangular inequality $\text{dist}(a,\ell)$ is at most $\text{dist}(a,f) + \text{dist}(f,\ell)$. Since f serves a in $\mathcal{C}$ we have $\text{dist}(a,f) = g_a$, hence $\text{dist}(a,\ell) \leqslant g_a + \text{dist}(f,\ell)$. We now bound $\text{dist}(f,\ell)$. Consider the center ℓ' that serves a in solution $\mathcal{L}$. By the triangular inequality we have $\text{dist}(f,\ell') \leqslant \text{dist}(f,a) + \text{dist}(a,\ell') = g_a + \ell_a$. Finally, since ℓ is the closest center of f in $\mathcal{L}$, we have $\text{dist}(f,\ell) \leqslant \text{dist}(f,\ell') \leqslant g_a + \ell_a$ and the lemma follows. □

We partition the centers of $\tilde{\mathcal{L}}$ as follows.

- Let $\tilde{\mathcal{L}}_0$ be the set of centers of $\tilde{\mathcal{L}}$ that have degree 0 in Γ.
- Let $\tilde{\mathcal{L}}_{\leqslant \varepsilon^{-1}}$ be the set of centers of $\tilde{\mathcal{L}}$ that have degree at least one and at most $1/\varepsilon$ in Γ.
- Let $\tilde{\mathcal{L}}_{>\varepsilon^{-1}}$ be the set of centers of $\tilde{\mathcal{L}}$ that have degree greater than $1/\varepsilon$ in Γ.

We now partition the centers of $\tilde{\mathcal{L}}$ and $\tilde{\mathcal{C}}$ using the neighborhoods of the vertices of $\tilde{\mathcal{L}}$ in Γ. We start by iteratively constructing two set of pairs $S_{\leqslant \varepsilon^{-1}}$ and $S_{>\varepsilon^{-1}}$. For each center $\ell \in \tilde{\mathcal{L}}_{\leqslant \varepsilon^{-1}} \cup \tilde{\mathcal{L}}_{>\varepsilon^{-1}}$, we pick a set A_ℓ of $|N_\Gamma(\ell)| - 1$ centers of $\tilde{\mathcal{L}}_0$ and define a pair $(\{\ell\} \cup A_\ell, N_\Gamma(\ell))$. We then remove A_ℓ from $\tilde{\mathcal{L}}_0$ and repeat. Let $S_{\leqslant \varepsilon^{-1}}$ be the pairs that contain a center of $\tilde{\mathcal{L}}_{\leqslant \varepsilon^{-1}}$ and let $S_{>\varepsilon^{-1}}$ be the remaining pairs.

The following fact follows from the definition of the pairs.

Fact 2.5. *Let $(R^{\tilde{\mathcal{L}}}, R^{\tilde{\mathcal{C}}})$ be a pair in $S_{\leqslant p} \cup S_{>p}$. If $\ell \in R^{\tilde{\mathcal{L}}}$, then for any f such that $(f,\ell) \in \mathcal{E}$, $f \in R^{\tilde{\mathcal{C}}}$.*

Lemma 2.6. *For any pair $(R^{\tilde{\mathcal{L}}}, R^{\tilde{\mathcal{C}}}) \in S_{\leqslant p}$ we have that*

$$\sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} g_a + 2 \sum_{a \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) - N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} g_a.$$

Proof. Consider the mixed solution $M = \mathcal{L} \backslash R^{\tilde{\mathcal{L}}} \cup R^{\tilde{\mathcal{C}}}$. For each point a, let m_a denote the cost of a in solution M. We have

$$m_a \leqslant \begin{cases} g_a & \text{if } a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}}). \\ \text{Reassign}_a & \text{if } a \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) - N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}} \cup S) \text{ and by Fact 2.5.} \\ \ell_a & \text{Otherwise.} \end{cases}$$

Now, observe that the solution M differs from $\mathcal{L}$ by at most $2/\varepsilon$ centers. Thus, by $1/\varepsilon$-local optimality we have $\text{cost}(\mathcal{L}) \leqslant \text{cost}(\mathcal{C})$. Summing over all clients and simplifying, we obtain

$$\sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}}) \cup N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}})} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} g_a + \sum_{a \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) - N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} \text{Reassign}_a.$$

The lemma follows by combining with Lemma 2.4. □

We now analyze the cost of the clients served by a center of $\mathcal{L}$ that has degree greater than ε^{-1} in Γ.

Lemma 2.7. *For any pair* $(R^{\tilde{\mathcal{L}}}, R^{\tilde{\mathcal{C}}}) \in S_{>\varepsilon^{-1}}$ *we have that*

$$\sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} g_a + 2(1+\varepsilon) \sum_{a \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) - N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} g_a.$$

Proof. Consider the center $\hat{\ell} \in R^{\tilde{\mathcal{L}}}$ that has in-degree greater than ε^{-1}. Let $\hat{L} = R^{\tilde{\mathcal{L}}} \backslash \{\hat{\ell}\}$. For each $\ell \in \hat{L}$, we associate a center $f(\ell)$ in $R^{\tilde{\mathcal{C}}}$ in such a way that each $f(\ell) \neq f(\ell')$, for $\ell \neq \ell'$. Note that this is possible since $|\hat{L}| = |R^{\tilde{\mathcal{C}}}| - 1$. Let $\tilde{f}$ be the center of $R^{\tilde{\mathcal{C}}}$ that is not associated with any center of $\hat{L}$.

Now, for each center ℓ of $\hat{L}$ we consider the mixed solution $M^\ell = \mathcal{L} - \{\ell\} \cup \{f(\ell)\}$. For each client a, we bound its cost m_a^ℓ in solution M^ℓ. We have

$$m_a^\ell \leqslant \begin{cases} g_a & \text{if } a \in N_{\mathcal{C}}(f(\ell)). \\ \text{Reassign}_a & \text{if } a \in N_{\mathcal{L}}(\ell) - N_{\mathcal{C}}(f(\ell)) \text{ and by Fact 2.5.} \\ \ell_a & \text{Otherwise.} \end{cases}$$

Summing over all center $\ell \in \hat{L}$ and all the clients in $N_{\mathcal{C}}(f(\ell)) \cup N_{\mathcal{L}}(\ell)$, we have by ε^{-1}-local optimality

$$\sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}} - \tilde{f}) \cup N_{\mathcal{L}}(\hat{L})} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}} - \tilde{f})} g_a + \sum_{a \in N_{\mathcal{L}}(\hat{L}) - N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}} - \tilde{f})} \text{Reassign}_a. \tag{2.1}$$

We now complete the proof of the lemma by analyzing the cost of the clients in $N_{\mathcal{C}}(\tilde{f})$. We consider the center $\ell^* \in \hat{L}$ that minimizes the reassignment cost of its clients. Namely, the center ℓ^* such that $\sum_{a \in N_{\mathcal{L}}(\ell^*)} \text{Reassign}_a$ is minimized. We then consider the solution $M^{(\ell^*, \tilde{f})} = \mathcal{L} - \{\ell^*\} \cup \{\tilde{f}\}$. For each client a, we bound its cost $m_a^{(\ell^*, \tilde{f})}$ in solution $M^{(\ell^*, \tilde{f})}$. We have

$$m_a^{(\ell^*, \tilde{f})} \leqslant \begin{cases} g_a & \text{if } a \in N_{\mathcal{C}}(\tilde{f}). \\ \text{Reassign}_a & \text{if } a \in N_{\mathcal{L}}(\ell^*) - N_{\mathcal{C}}(\tilde{f}) \text{ and by Fact 2.5.} \\ \ell_a & \text{Otherwise.} \end{cases}$$

Thus, summing over all clients a, we have by local optimality

$$\sum_{a \in N_{\mathcal{C}}(\tilde{f}) \cup N_{\mathcal{L}}(\ell^*)} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(\tilde{f})} g_a + \sum_{a \in N_{\mathcal{L}}(\ell^*) - N_{\mathcal{C}}(\tilde{f})} \text{Reassign}_a. \tag{2.2}$$

By Lemma 2.4, combining Equations 2.1 and 2.2 and averaging over all centers of $\hat{L}$ we have

$$\sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} g_a + 2(1+\varepsilon) \sum_{a \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) - N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} g_a.$$

□

We now turn to the proof of Theorem 2.3.

Proof of Theorem 2.3. We sum the equations of Lemmas 2.6 and 2.7 over all pairs and obtain

$$\sum_{(R^{\tilde{\mathcal{L}}},R^{\tilde{\mathcal{C}}})} \sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} \ell_a \leqslant \sum_{(R^{\tilde{\mathcal{L}}},R^{\tilde{\mathcal{C}}})} \left(\sum_{a \in N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} g_a + (2+2\varepsilon) \sum_{a \in N_{\mathcal{L}}(R^{\tilde{\mathcal{L}}}) - N_{\mathcal{C}}(R^{\tilde{\mathcal{C}}})} g_a \right)$$
$$\sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}})} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}})} g_a + (2+2\varepsilon) \sum_{a \in N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_a.$$

□

The two following lemmas show how to handle a relaxed version of the triangular inequality (when $p > 1$).

Lemma 2.8. *Let $p \geqslant 0$ and $1/2 > \varepsilon > 0$. For any $a, b, c \in A \cup F$, we have cost$(a,b) \leqslant (1+\varepsilon)^p($cost$(a,c) +$ cost$(c,b)/\varepsilon^p)$.*

Proof. Suppose first that dist$(c,b) < \varepsilon$dist(a,c), then by the triangular inequality cost$(a,b) =$ dist$(a,b)^p \leqslant ($dist$(a,c) +$ dist$(c,b))^p$. Hence, cost$(a,b) \leqslant (1+\varepsilon)^p$dist$(a,c)^p = (1+\varepsilon)^p$cost$(a,c)$. Now suppose that dist$(c,b) \geqslant \varepsilon$dist$(a,c)$. Thus, by the triangular inequality cost$(a,b) \leqslant ((1+\varepsilon)$dist$(c,b)^p/\varepsilon) = (1+\varepsilon)^p$cost$(c,b)/\varepsilon^p$. □

The following lemma follows from the binomial theorem.

Lemma 2.9. *Let $p \geqslant 0$. For any $a, b, c \in A \cup F$, we have cost$(a,b) \leqslant 2^{p-1}($cost$(a,c) +$ cost$(c,b))$.*

Proof. By the triangular inequality cost$(a,b) \leqslant ($dist$(a,c) +$ dist$(c,b))^p$. By the binomial theorem we have that $(\text{dist}(a,c) + \text{dist}(c,b))^p = \sum_{i=0}^{p} \binom{p}{i} \text{dist}(a,c)^i \text{dist}(c,b)^{p-i}$. Thus, $(\text{dist}(a,c) + \text{dist}(c,b))^p = \text{cost}(a,c) + \text{cost}(c,b) + \sum_{i=1}^{p-1} \binom{p}{i} \text{dist}(a,c)^i \text{dist}(c,b)^{p-i}$. Observe now that dist$(a,c)^i$dist$(c,b)^{p-i} \leqslant$ dist$(a,c)^p +$ dist$(c,b)^p$. Hence, $(\text{dist}(a,c) + \text{dist}(c,b))^p \leqslant \text{cost}(a,c) + \text{cost}(c,b) + (\text{cost}(a,c) + \text{cost}(c,b)) \sum_{i=1}^{p-1} \binom{p}{i}$ and so $(\text{dist}(a,c) + \text{dist}(c,b))^p \leqslant \text{cost}(a,c) + \text{cost}(c,b) + (\text{cost}(a,c) + \text{cost}(c,b))(2^{p-1} - 1)$ and the lemma follows. □

Rescaling ε as a function of p yields the proof of Theorem 2.1.

CHAPTER 3

Local Search for Clustering: Well-Separated Instances

This chapter is dedicated to the proof of Theorems 1.2 and 1.3, restated below.

We consider the k-clustering problem and recall its definition for completeness.

Definition A.1 (k-Clustering). *Given a finite set of* clients $A \subseteq \mathcal{A}$*, a set of* candidate centers $F \subseteq \mathcal{A}$*, two positive integers k and p, the k-*clustering problem *asks for a set of* centers $S \subseteq F$*, of cardinality at most k, that minimizes*

$$\textit{cost}(S) = \sum_{x \in A} \min_{c \in S} \textit{dist}(x, c)^p.$$

Theorem 1.2 (Graphs – Section 3.4). *Let $\mathcal{K}$ be a nontrivial minor-closed family of edge-weighted graphs. For any integer $p > 0$, there is a constant c such that the following holds. For any $0 < \varepsilon < 1/2$, for any graph $G = (V, E)$ in $\mathcal{K}$ with clients set $A \subseteq V$ and set of centers locations $F \subseteq V$, Algorithm 2 applied to the metric completion of G with cost function*

$$\textit{cost}(S) = \sum_{a \in A} (\min_{u \in S} \textit{dist}(a, u))^p$$

and with $s = 1/\varepsilon^{cp}$ outputs a solution S whose cost is at most $1 + \varepsilon$ times the cost of the optimal solution to the k-clustering problem with parameter p.

Theorem 1.3 (Euclidean Spaces – Section 3.5). *For any fixed integers $p, d > 0$, there is a constant c such that the following holds. For any $0 < \varepsilon < 1/2$, client set $A \subset \mathbb{R}^d$, candidate centers $F \subset \mathbb{R}^d$, Algorithm 2 applied to A, F with*

$$\textit{cost}(S) = \sum_{u \in C} (\min_{f \in S} \textit{dist}(u, f))^p$$

and $s = 1/\varepsilon^{cpd}$ yields a solution S whose cost is at most $1 + \varepsilon$ times cost of the optimal solution to the k-clustering problem with parameter p.

Section 3.1 highlights the properties of the well-separated instances and provides the key ingredients to our analysis: r-division and isolation. Section 3.3 introduces our structure theorem for the analysis of Algorithm 2. The analysis of Algorithm 2 for instances consisting of graphs with small separators is done in Section 3.4. In Section 3.5 we describe how a similar analysis holds for instances consisting of points lying in $\mathbb{R}^d$.

3.1 Properties of Well-Separated Instances

As mentioned in the introduction, the analysis of the output of local search on well-separated instances relies on a decomposition of the input called a *weak r-division*.

Chan and Har-Peled [49] showed that local search can be used to obtain a PTAS for (unweighted) maximum independent pseudo-disks in the plane, which implies the analogous result for planar graphs. More generally, Har-Peled and Quanrud [103] show that local search can be used to obtain PTASs for several problems including *independent set*, *set cover*, and *dominating set*, in graphs with polynomial expansion. These graphs have small separators and therefore r-divisions. However, our analysis of local search for clustering requires not only that the *input graph* have an r-division but that a *minor* of the input graph have an r-division. This is not true of graphs of polynomial expansion. Indeed, we show in Section 3.1.4 that there are low-density graphs in low-dimensional space (which are therefore polynomial-expansion graphs) for which local-search produces a solution that is worse than the optimum by at least a constant factor.

Thus one of our technical contributions is showing how to take advantage of a property possessed by nontrivial minor-closed graph families that is not possessed by polynomial-expansion graph families.

3.1.1 r-Division in Graph with Small Separators

For a graph G, we use $V(G)$ and $E(G)$ to denote the set of vertices of G and the set of edges of G, respectively. For a subgraph H of G, the *vertex boundary* of H in G, denoted $\partial_G(H)$, is the set of vertices v such that v is in H but has an incident edge that is not in H. (We might write $\partial(H)$ if G is unambiguous.) A vertex in the vertex boundary of H is called a *boundary vertex* of H. A vertex of H that is not a boundary vertex of H is called an *internal vertex*. We denote the set of internal vertices of H as $\mathcal{I}(H)$.

Definition 3.1. *Let c_1 and c_2 be constants. For a number r, a* weak r-division *of a graph G (with respect to c_1, c_2) is a collection $\mathcal{R}$ of subgraphs of G, called* regions, *with the following properties.*

1. *Each edge of G is in exactly one region.*

2. *The number of regions is at most $c_1|V(G)|/r$.*

3. *Each region contains at most r vertices.*

4. *The number of boundary vertices, summed over all regions, is at most $c_2|V(G)|/r^{1/2}$.*

A family of graphs F is said to be closed under taking minor (*minor-closed*) if for any graph $G \in F$, for any minor H of G, we have $H \in F$.

Theorem 3.2 (Frederickson [86] combined with Alon, Seymour, and Thomas [7])**.** *Let $\mathcal{K}$ be a nontrivial minor-closed family of graphs. There exist c_1, c_2 such that every graph in $\mathcal{K}$ has a weak r-division with respect to c_1, c_2.*

Proof. Alon, Seymour, and Thomas [7] proved a separator theorem for the family of graphs excluding a fixed graph as a minor. Any nontrivial minor-closed family excludes some graph as a minor (else it is trivial). Frederickson [86] gave a construction for a stronger kind of r-division of a planar graph. The construction uses nothing of planar graphs except that they have such separators. □

Let G be an undirected graph with edge-lengths. Fix an arbitrary priority ordering of the vertex set $V(G)$. For every subset S of $V(G)$, we define the *Voronoi partition with respect to S*. For each vertex $v \in S$, the *Voronoi cell with center v*, denoted $V_S(v)$, is the set of vertices that are closer to v than to any other vertex in S, breaking ties in favor of the highest-priority vertex of S.

Fact 3.3. *For any S, for any vertex $v \in S$, the induced subgraph $G[V_S(v)]$ is a connected subgraph of G.*

Proof. Let $u \in V_S(v)$, and let p denote a v-to-u shortest path. Let w be a vertex on P. Assume for a contradiction that, for some vertex $v' \in S$, either the v'-to-w shortest path p' is shorter than the shortest v-to-w path, or it is no longer and v' has higher priority than v. Replacing the v-to-w subpath of p with p' yields a v'-to-u path that either is shorter than p or is no longer than p and originates at a higher-priority vertex than v. □

It follows that, for any vertex v of G, contracting the edges of the subgraph $G[V_S(v)]$ yields a single vertex.

Definition 3.4. *We define $G_{Vor(S)}$ as the graph obtained from G by contracting every edge of $G[V_S(v)]$ for every vertex $v \in S$. For each vertex $v \in S$, we denote by $\hat{v}$ the vertex of $G_{Vor(S)}$ resulting from contracting every edge of $G[V_S(v)]$.*

For any set graph $G = (V, E)$ and $S \subseteq E$, if G belongs to a minor-closed family $\mathcal{K}$ then so does $G_{\text{Vor}(S)}$.

3.1.2 r-Division in Euclidean Space

We define analogous notions for the case of Euclidean spaces of fixed dimension d. Consider a set of points C in $\mathbb{R}^d$. For a set Z of points in $\mathbb{R}^d$ and a bipartition $C_1 \cup C_2$ of C, we say that Z *separates* C_1 and C_2 if, in the Voronoi diagram of $C \cup Z$, the boundaries of cells of points in C_1 are disjoint from the boundaries of cells of points in C_2.

Definition 3.5. *Let c_1 and c_2 be constants. Let C be a set of points in $\mathbb{R}^d$. For an integer $r > 1$, a* weak r-division *of C (with respect to c_1, c_2) is a set of* boundary points $Z \subset \mathbb{R}^d$ *together with a collection of subsets $\mathcal{R}$ of $C \cup Z$ called* regions, *with the following properties.*

1. *Each element of C appears in exactly one subset of $\mathcal{R}$.*
2. *The number of regions is at most $c_1|C|/r$.*
3. *Each region contains at most r points of $C \cup Z$.*
4. $\sum_{R \in \mathcal{R}} |R \cap Z| \leqslant c_2|C|/r^{1/d}$.

Moreover, for any region R_i, $R_i \cap Z$ is a Voronoi separator for $R_i - Z$ and $(C \cup Z) - R_i$.

The following theorem is from [34, Theorem 3.7].

Theorem 3.6. *[34, Theorem 3.7] Let P be a set of n points in $\mathbb{R}^d$. One can compute, in expected linear time, a sphere S, and a set $Z \subseteq S$, such that*

- $|Z| \leqslant cn^{1-1/d}$,
- *There are most σn points of P in the sphere S and at most σn points of P not in S, and*
- *Z is a Voronoi separator of the points of P inside S from the points of P outside S.*

Here c and $\sigma < 1$ are constants that depends only on the dimension d.

From that theorem we can easily derive the following.

Theorem 3.7. *Let r be a positive integer and d be fixed. Then there exist c_1, c_2 such that every set of points $C \subset \mathbb{R}^d$ has a weak r-division with respect to c_1, c_2.*

Proof. We describe a recursive procedure to construct the set Z in the definition of weak r-division of C. Assuming that $|C| > r$, find a sphere S and a set Z_0 satisfying Theorem 3.6. Let Z_1 be the result of applying the procedure to the union of Z_0 with the set of points inside C, and similarly obtain Z_2 from the set of points outside C. Return $Z_0 \cup Z_1 \cup Z_2$.

It is clear that the set Z together with its induced partition $\mathcal{R}$ of C returned by the procedure satisfies all the properties of a weak r-division except for Property 4, which requires some calculation. Let $b(n) = \sum_{R \in \mathcal{R}} |R \cap Z|$ when the procedure is applied to a set C of size at most n, where $n > (1-\sigma)r$. If $n \leqslant r$ then $b(n) = 0$, and if $n > r$ then

$$b(n) \leqslant cn^{1-1/d} + \max_{\alpha \in [1-\sigma,\sigma]} b(\alpha n + cn^{1-1/d}) + b((1-\alpha)n + cn^{1-1/d}).$$

We show by induction that $b(n) \leqslant \beta \frac{n}{r^{1/d}} - \gamma n^{1-1/d}$ for suitable constants $\beta, \gamma > 0$ to be determined. We postpone the basis of the induction until β, γ are selected.

By the inductive hypothesis,

$$\begin{aligned} b(\alpha n + cn^{1-1/d}) &\leqslant \beta \frac{\alpha n}{r^{1/d}} + \beta \frac{cn^{1-1/d}}{r^{1/d}} - \gamma \alpha^{1-1/d} n^{1-1/d} \\ b((1-\alpha)n + cn^{1-1/d}) &\leqslant \beta \frac{(1-\alpha)n}{r^{1/d}} + \beta \frac{cn^{1-1/d}}{r^{1/d}} - \gamma (1-\alpha)^{1-1/d} n^{1-1/d} \end{aligned}$$

so

$$b(n) \leqslant \left(c + \frac{2c}{r^{1/d}}\right) n^{1-1/d} + \beta \frac{n}{r^{1/d}} - \gamma \left[\alpha^{1-1/d} + (1-\alpha)^{1-1/d}\right] n^{1-1/d} \tag{3.1}$$

The function $f(x) = x^{1-1/d} + (1-x)^{1-1/d}$ is strictly concave for $x \in [0,1]$, as can be seen by taking its second derivative. For any $\alpha \in [1-\sigma, \sigma]$, there exists a number $0 < \mu < 1$ such that $\alpha = (1-\mu)(1-\sigma) + \mu\sigma$. By concavity, therefore, $f(\alpha) \geqslant (1-\mu)f(1-\sigma) + \mu f(\sigma)$. Since a weighted average is at least the minimum, $(1-\mu)f(1-\sigma) + \mu f(\sigma) \geqslant \min\{f(1-\sigma), f(\sigma)\}$. Write $f(1-\sigma) = f(\sigma) = 1 + \delta$. Since f is strictly concave, $\delta > 0$. We choose $\gamma = (c + 2c/r^{1/d})/\delta$, for then the first term in Inequality 3.1 is bounded by $\gamma \delta n^{1-1/d}$, and we obtain $b(n) \leqslant \beta \frac{n}{r^{1/d}} - \gamma n^{1-1/d}$.

For the basis of the induction, suppose $n > (1-\sigma)r$. Then

$$\beta \frac{n}{r^{1/d}} - \gamma n^{1-1/d} = \left(\beta \frac{n^{1/d}}{r^{1/d}} - \gamma\right) n^{1-1/d} \geqslant \left(\beta \frac{(1-\sigma)^{1/d} r^{1/d}}{r^{1/d}} - \gamma\right) = \left(\beta(1-\sigma)^{1/d} - \gamma\right)$$

which is nonnegative for an appropriate choice of β depending on σ and γ. □

3.1.3 Contraction of Voronoi cells and Properties of the r-Divisions

We present the properties of the r-divisions that we will be using for the analysis of the solution output by the local search algorithm.

Lemma 3.8. *Let $G = (V, E)$ be a graph excluding a fixed minor H and let $\mathcal{F} \subseteq V$. Let H_i be a region of the r-division of $G_{Vor(\mathcal{F})}$. Suppose c and v are vertices of G such that one of the vertices in $\{\hat{c}, \hat{v}\}$ is a vertex of H_i and the other is* not *an internal vertex of H_i. Then there exists a vertex $x \in \mathcal{F}$ such that $\hat{x}$ is a boundary vertex of the region H_i and dist$(c, x) \leqslant$ dist(c, v).*

Proof. Let p be a shortest c-to-v path in G. By the conditions on $\hat{c}$ and $\hat{v}$, there is some vertex w of p such that $\hat{w}$ is a boundary vertex of H_i. Let x be the center of the Voronoi cell whose contraction yields $\hat{w}$. By definition of Voronoi cell, $\text{dist}(w, x) \leqslant \text{dist}(w, v)$. Therefore replacing the w-to-v subpath of p with the shortest w-to-x path yields a path no longer than p. □

We obtain the analogous lemma for the Euclidean case, whose proof follows directly from the definition of r-division (i.e.: the fact that Z is a Voronoi separator).

Lemma 3.9. *Let C be a set of points in $\mathbb{R}^d$ and Z be an r-division of C. For any two different regions R_1, R_2, for any points $c \in R_1, v \in R_2$ there exists a boundary vertex $x \in Z \cap R_1$ such that dist$(c, x) \leqslant$ dist(c, v).*

3.1.4 Tightness of the Analysis

In this section, we show that our analysis of the performance of local search for the k-clustering problem cannot be extended to other classes of graphs such as t-shallow-minor free. Loosely speaking, there exist t-shallow-minor free graphs for which local search might return a solution of cost at least 3 times the optimal.

A graph G has a graph H as a *t-shallow minor* if each subgraph of G that is contracted to a single vertex of H has small diameter.

This shows that having small separators is not the only ingredient needed for local search to work and the fact that the class of planar graphs is closed under edge contraction is crucial.

Proposition 3.10. *For any w, t, there exists an infinite family of graphs excluding K_w as a t-shallow minor such that for any constant ε, there exists a $1/\varepsilon$-locally optimal solution of cost at least 3OPT.*

See Figure 3.1 and [16] for a complete proof that local search performs badly on the instance depicted in the figure.

We additionally remark that Awasthi et al. [19] show that the k-means problem is APX-Hard for inputs in d-dimensional Euclidean space and $d = \Omega(\log n)$. Moreover, Kanungo et al. [112] give an example where local search returns a solution of cost at least 9OPT.

3.2 Isolation

In order to obtain our approximation schemes for k-means and k-median clustering, we need another technique. As mentioned earlier, a bicriteria approximation scheme for k-means was already known; the solution it returns has more than k centers. It seems hard to avoid an increase in the number of centers in comparing a locally optimal solution to a

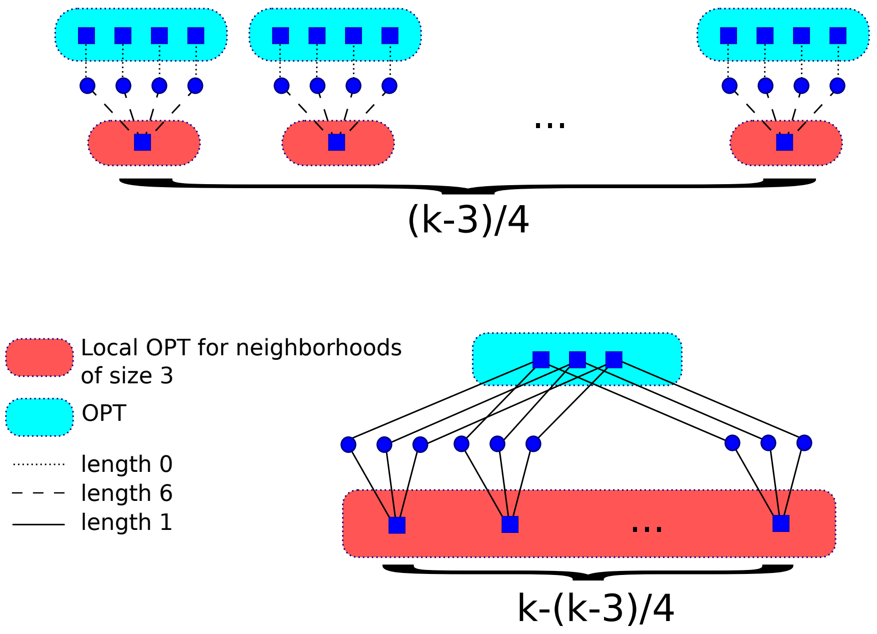


Figure 3.1: This instance contains the complete bipartite graph $K_{3,3}$ as a 1-shallow minor (and so is not planar) but no non-trivial 0-shallow minor. Clients are denoted by circle and candidate centers by squares. There exists a local (for neighborhoods of size 3) optimum whose cost is at least 3OPT. This example can be generalized to handle locality for neighborhoods of size $1/\varepsilon$ for any constant $\varepsilon > 0$. This is based on an example of [16] and can be extended to form a i-shallow minor graph for any $i = o(n)$.

globally optimal solution. It would help if we could show that the globally optimal solution could be modified so as to *reduce* the number of centers below k while only slightly increasing the cost; we could then compare the local solution to this modified global solution, and the increase in the number of centers would leave the number no more than k.

Unfortunately, we cannot unconditionally reduce the number of centers. However, consider a globally optimal solution $\mathcal{C}$ and a locally optimal solution $\mathcal{L}$. A center f in $\mathcal{C}$ might correspond to a center ℓ in $\mathcal{L}$ in the sense that they serve almost exactly the same set of clients. In this case, we say the pair (f, ℓ) is *1-1 isolated* (the formal definition is below). Such centers do not contribute much to the increase in cost in going from global solution to local solution, so let's ignore them. Among the remaining centers of $\mathcal{C}$, there are a substantial number that can be removed without the cost increasing much. The analysis of the local solution then proceeds as discussed above.

We now give the formal definition of 1-1 isolated.

Definition 3.11. *Let $\varepsilon < 1/2$ be a positive number and let $\mathcal{L}$ and $\mathcal{C}$ be two solutions for the*

k-clustering problem with parameter p. Given a facility $f_0 \in \mathcal{C}$ and a facility $\ell \in \mathcal{L}$, we say that the pair (f_0, ℓ) is 1-1 ε-isolated *if most of the clients served by ℓ in $\mathcal{L}$ are served by f_0 in $\mathcal{C}$, and most of the clients served by f_0 in $\mathcal{C}$ are served by ℓ in $\mathcal{L}$; formally, if*

$$|V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)| \geqslant \max \left\{ \begin{array}{l} (1-\varepsilon)|V_{\mathcal{L}}(\ell)|, \\ (1-\varepsilon)|V_{\mathcal{C}}(f_0)| \end{array} \right\}$$

We will sometimes simply say the pair is 1-1 isolated if the value of ε is clear.

We define the more general concept of *isolated* regions; 1-1-isolated regions correspond to the special case of isolated regions when $\mathcal{L}_0$ consists of a single center.

Definition 3.12 (Isolated Region)**.** *Given a facility $f_0 \in \mathcal{C}$ and a set of facilities $\mathcal{L}_0 \subseteq \mathcal{L}$, we say that the pair $(f_0, \mathcal{L}_0)$ is an* ε-isolated region *if the following hold:*

- *For each facility $f \in \mathcal{L}_0$, most of the clients served by f in $\mathcal{L}$ are served by f_0 in $\mathcal{C}$: formally, $|V_{\mathcal{L}}(f) \cap V_{\mathcal{C}}(f_0)| \geqslant (1-\varepsilon)|V_{\mathcal{L}}(f)|$, and*

- *most of the clients served by f_0 in $\mathcal{C}$ are served by facilities of $\mathcal{L}_0$ in $\mathcal{L}$: formally, $|V_{\mathcal{L}}(\mathcal{L}_0) \cap V_{\mathcal{C}}(f_0)| \geqslant (1-\varepsilon)|V_{\mathcal{C}}(f_0)|$.*

If $(f_0, \mathcal{L}_0)$ is ε-isolated, we say that f_0 and the the elements of $\mathcal{L}_0$ are *ε-isolated.* If ε is clear, we might simply say *isolated.*

Finally, if $(f_0, \mathcal{L}_0)$ is an isolated region, we say that f_0 and the elements of $\mathcal{L}_0$ are *isolated.*

We can now state the structural theorem arising from the notion of isolation. In simple words, it claims that it is possible to delete a small fraction of the *non-isolated* optimal centers (*i.e.,* the centers that are not part of any isolated region) while preserving roughly the same cost (*i.e.,* increasing it by an ε fraction of the cost of $\mathcal{L}$ and the cost of $\mathcal{C}$).

Theorem 3.13. *Let $\varepsilon < 1/2$ be a positive number and let $\mathcal{L}$ and $\mathcal{C}$ be two solutions for the k-clustering problem with exponent p. Let $\bar{k}$ denote the number of facilities f of $\mathcal{C}$ that are not in a 1-1 ε-isolated region. There exists a set S_0 of facilities of $\mathcal{C}$ of size at least $\varepsilon^3\bar{k}/6$ that can be removed from $\mathcal{C}$ at low cost: cost$(\mathcal{C} \backslash S_0) \leqslant (1+2^{3p+1}\varepsilon)$cost$(\mathcal{C})+2^{3p+1}\varepsilon$ cost$(\mathcal{L})$.*

Note that the preceding theorem does not assume that $\mathcal{L}$ is a local optimum and $\mathcal{C}$ is an optimal solution. Thus we believe that this theorem can be of broader interest.

3.3 A Structure Theorem

This section is dedicated to the proof of a theorem that provides a useful handle to compare two solutions of the k-clustering problem. It is the second ingredient of our analysis of local search on well-separated instances. We recall the theorem for completeness.

Theorem 3.13. *Let $\varepsilon < 1/2$ be a positive number and let $\mathcal{L}$ and $\mathcal{C}$ be two solutions for the k-clustering problem with exponent p. Let $\bar{k}$ denote the number of facilities f of $\mathcal{C}$ that are not in a 1-1 ε-isolated region. There exists a set S_0 of facilities of $\mathcal{C}$ of size at least $\varepsilon^3\bar{k}/6$ that can be removed from $\mathcal{C}$ at low cost: cost$(\mathcal{C}\backslash S_0) \leqslant (1+2^{3p+1}\varepsilon)$cost$(\mathcal{C})+2^{3p+1}\varepsilon$ cost$(\mathcal{L})$.*

Observe that this theorem has a broader scope than well-separated instance since $\mathcal{L}$ and $\mathcal{C}$ can be solutions to a general instance.

Theorem 3.13 relies on the following lemma, whose proof we momentarily defer.

Lemma 3.14. *There exists a function $\phi : \tilde{\mathcal{C}} \mapsto \mathcal{C}$ such that reassigning all the clients of $V_{\mathcal{C}}(f)$ to $\phi(f)$ for every center $f \in \tilde{\mathcal{C}}$ increases the cost of $\mathcal{C}$ by at most $2^{3p+1}\varepsilon^{-2}($cost$(\mathcal{L})$ + cost$(\mathcal{C}))$.*

Proof of Theorem 3.13. Consider the abstract graph H where the nodes are the elements of $\mathcal{C}$ and there is a directed arc from f to $\phi(f)$. More formally, $H = (\mathcal{C}, \{\langle f, \phi(f)\rangle \mid f \in \tilde{\mathcal{C}}\})$. Notice that every node of H has outdegree at most 1. Thus, there exists a coloring of the nodes of H with three colors, such that all arcs are bichromatic. Let S denote the color set with the largest number of nodes of $\tilde{\mathcal{C}}$. We have that S contains at least $|\tilde{\mathcal{C}}|/3$ nodes of $\tilde{\mathcal{C}}$.

Arbitrarily partition S into $1/\varepsilon^3$ parts, each of cardinality at least $\varepsilon^3|\tilde{\mathcal{C}}|/3$. By Lemma 3.14 and an averaging argument, there exists a part S_0 such that reassigning each center $f \in S_0$ to $\phi(f)$ increases the cost by at most

$$\frac{2^{3p+1}\varepsilon^{-2}}{\varepsilon^{-3}}(\text{cost}(\mathcal{L}) + \text{cost}(\mathcal{C})) = 2^{3p+1}\varepsilon(\text{cost}(\mathcal{L}) + \text{cost}(\mathcal{C})).$$

Since the arcs of H are bichromatic, if $f \in S_0$ then $\phi(f) \notin S_0$. Consider the solution $\mathcal{C}\backslash S_0$. Client that belong to $V_{\mathcal{C}}(f)$ for some $f \in S_0$ can be assigned in $\mathcal{C}\backslash S_0$ to a center that is no farther than $\phi(f)$. Therefore, the cost of the solution $\mathcal{C}\backslash S_0$ is at most cost$(\mathcal{C}) + 2^{3p+1}\varepsilon \cdot$ (cost$(\mathcal{L})$ + cost$(\mathcal{C})$).

We now relate $|\tilde{\mathcal{C}}|$ to $\bar{k}$. Let $k(\mathcal{L})_1$ be the number of centers of $\mathcal{L}$ that belong to an isolated region that is not 1-1 isolated. Let $k(\mathcal{C})_1$ be the number of centers of $\mathcal{C}$ that belong to an isolated region that is not 1-1 isolated. Finally, let $k(\mathcal{C})_2 = |\tilde{\mathcal{C}}|$. By definition, we have $k(\mathcal{C})_1 + k(\mathcal{C})_2 = \bar{k} \geqslant k(\mathcal{L})_1$.

Now, observe that there are at least two centers of $\mathcal{L}$ per isolated region that is not 1-1 isolated. Thus, $2k(\mathcal{C})_1 \leqslant k(\mathcal{L})_1$. Hence, $\bar{k} = k(\mathcal{C})_1 + k(\mathcal{C})_2 \leqslant k(\mathcal{L})_1/2 + k(\mathcal{C})_2$. But for any $k(\mathcal{C})_2 < k(\mathcal{C})_1$, $k(\mathcal{L})_1/2 + k(\mathcal{C})_2 < k(\mathcal{L})_1 \leqslant \bar{k}$. Therefore, we must have $k(\mathcal{C})_2 \geqslant k(\mathcal{C})_1$, and so $k(\mathcal{C})_2 \geqslant \bar{k}/2$. Thence $\varepsilon|\tilde{\mathcal{C}}|/3 \geqslant \varepsilon\bar{k}/6$ and the theorem follows. □

We now define g_c to be the cost of client c in solution $\mathcal{C}^*$ and ℓ_c to be the cost of client c in solution $\mathcal{L}$.

Proof of Lemma 3.14. For each center $f \in \tilde{\mathcal{C}}$, we define $\phi(f) = \text{argmin}\{\text{dist}(f, f') \mid f' \in \mathcal{C}\backslash\{f\}\}$. Instead of analyzing the cost increase when reassigning clients of $V_{\mathcal{C}}(f)$ to $\phi(f)$

we will analyze the cost increase of the following fractional assignment. First for a center $f \in \mathcal{C}$, we denote by $\hat{\mathcal{L}}(f)$ the set

$$\hat{\mathcal{L}}(f) = \{\ell \in \mathcal{L} \mid 1 \leqslant |V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)| < (1-\varepsilon)|V_{\mathcal{L}}(\ell)|\}. \tag{3.2}$$

By definition of isolated regions (Definition 3.12), for any $f \in \tilde{\mathcal{C}}$ we have

$$\sum_{\ell \in \hat{\mathcal{L}}(f)} |V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f)| > \varepsilon |V_{\mathcal{C}}(f)|. \tag{3.3}$$

Thus, we partition the clients in $V_{\mathcal{C}}(f)$ into parts indexed by $\ell \in \hat{\mathcal{L}}(f)$, in a such a way that the part associated to ℓ has size at most $\varepsilon^{-1}|V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)|$. For any $\ell \in \hat{\mathcal{L}}(f)$, the clients in the associated part are reassigned to the center $\psi(\ell, f) \in \mathcal{C}\backslash\{f\}$ that is the closest to ℓ.

We now bound the cost increase Δ induced by the reassignment. For each client $c \in V_{\mathcal{C}}(f)$ assigned to a part associated to a center ℓ, the new cost for c is $\text{cost}'_c = \text{dist}(c, \psi(\ell, f))^p$. By the triangular inequality and Lemma 2.9, $\text{cost}'_c \leqslant 2^p(\text{dist}(c, f)^p + \text{dist}(f, \psi(\ell, f))^p) = 2^p(g_c + \text{dist}(f, \psi(\ell, f))^p)$. Summing over all clients, we have that the new cost is at most

$$\sum_c 2^p g_c + \sum_{f \in \tilde{\mathcal{C}}} \sum_{\ell \in \hat{\mathcal{L}}(f)} \varepsilon^{-1}|V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)| 2^p \text{dist}(f, \psi(\ell, f))^p.$$

Let $\Delta = \sum_{f \in \tilde{\mathcal{C}}} \sum_{\ell \in \hat{\mathcal{L}}(f)} \varepsilon^{-1}|V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)| 2^p \text{dist}(f, \psi(\ell, f))^p$. By Lemma 2.9, we have

$$\Delta \leqslant \sum_{f \in \tilde{\mathcal{C}}} \sum_{\ell \in \hat{\mathcal{L}}(f)} \varepsilon^{-1}|V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)| 4^p (\text{dist}(f, \ell)^p + \text{dist}(\ell, \psi(\ell, f))^p).$$

Inverting summations, we have that Δ is at most

$$4^p \varepsilon^{-1} \left(\sum_{\ell \in \mathcal{L}} \sum_{\substack{f \in \tilde{\mathcal{C}}: \\ \ell \in \hat{\mathcal{L}}(f)}} |V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)| \text{dist}(f, \ell)^p + \sum_{\ell \in \mathcal{L}} \sum_{\substack{f \in \tilde{\mathcal{C}}: \\ \ell \in \hat{\mathcal{L}}(f)}} |V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)| \text{dist}(\ell, \psi(\ell, f))^p \right).$$

Define

$$\Delta_1 = \sum_{\ell \in \mathcal{L}} \sum_{f \in \tilde{\mathcal{C}} : \ell \in \hat{\mathcal{L}}(f)} |V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)| \text{dist}(f, \ell)^p$$

and

$$\Delta_2 = \sum_{\ell \in \mathcal{L}} \sum_{f \in \tilde{\mathcal{C}} : \ell \in \hat{\mathcal{L}}(f)} |V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)| \text{dist}(\ell, \psi(\ell, f))^p.$$

We first bound Δ_1. By Lemma 2.9, for any client $c \in V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)$, we have $\text{dist}(f, \ell)^p \leqslant 2^p(\text{dist}(f, c)^p + \text{dist}(\ell, c)^p) = 2^p(g_c + \ell_c)$. Therefore,

$$\begin{aligned}\Delta_1 &\leqslant \varepsilon^{-1} \sum_{\ell \in \mathcal{L}} \sum_{f \in \tilde{\mathcal{C}} \,:\, \ell \in \hat{\mathcal{L}}(f)} |V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)| \frac{2^p}{|V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)|} \sum_{c \in V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)} (g_c + \ell_c) \\ &\leqslant 2^p \varepsilon^{-1} \sum_{\ell \in \mathcal{L}} \sum_{f \in \tilde{\mathcal{C}} \,:\, \ell \in \hat{\mathcal{L}}(f)} \sum_{c \in V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)} (g_c + \ell_c) \leqslant 2^p \varepsilon^{-1}(\text{cost}(\mathcal{C}) + \text{cost}(\mathcal{L})).\end{aligned}$$

We now turn to bound the cost of Δ_2. Let $f^{\ell}_{\min}$ be the center of $\mathcal{C}$ that is the closest to ℓ. Let

$$\begin{aligned}\Delta_3 &= \varepsilon^{-1} \sum_{\ell \in \mathcal{L}} \sum_{f \neq f^{\ell}_{\min} : \ell \in \hat{\mathcal{L}}(f)} |V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)| \text{dist}(\ell, \psi(\ell, f))^p \\ \Delta_4 &= \varepsilon^{-1} \sum_{\ell \in \mathcal{L}} |V_{\mathcal{C}}(f^{\ell}_{\min}) \cap V_{\mathcal{L}}(\ell)| \text{dist}(\ell, \psi(\ell, f^{\ell}_{\min}))^p.\end{aligned}$$

For any client $c \in V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)$, by Lemma 2.9, $\text{dist}(\ell, \psi(\ell, f))^p$, for $f \neq f^{\ell}_{\min}$ yields $\text{dist}(\ell, \psi(\ell, f))^p \leqslant 2^p(\text{dist}(\ell, c)^p + \text{dist}(c, \psi(\ell, f))^p) \leqslant 2^p(\text{dist}(\ell, c)^p + \text{dist}(c, f)^p) = 2^p(\ell_c + g_c)$. Thus, Δ_3 is at most

$$\varepsilon^{-1} \sum_{\ell \in \mathcal{L}} \sum_{f \neq f^{\ell}_{\min} : \ell \in \hat{\mathcal{L}}(f)} \frac{2^p |V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)|}{|V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)|} \sum_{c \in V_{\mathcal{C}}(f) \cap V_{\mathcal{L}}(\ell)} (\ell_c + g_c) \leqslant 2^p \varepsilon^{-1}(\text{cost}(\mathcal{C}) + \text{cost}(\mathcal{L})).$$

We conclude by analyzing Δ_4. Observe that if $\ell \notin \hat{\mathcal{L}}(f^{\ell}_{\min})$ then we are done: the clients in $V_{\mathcal{C}}(f^{\ell}_{\min})$ are not reassigned through ℓ. Thus we assume $\ell \in \hat{\mathcal{L}}(f^{\ell}_{\min})$. We now apply Lemma 2.9 to $\text{dist}(\ell, \psi(\ell, f^{\ell}_{\min}))^p$, for any client $c \in V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f^{\ell}_{\min})$ we have $\text{dist}(\ell, \psi(\ell, f^{\ell}_{\min}))^p \leqslant 2^p(\text{dist}(\ell, c)^p + \text{dist}(c, \psi(\ell, f^{\ell}_{\min}))^p) \leqslant 2^p(\ell_c + g_c)$, since $\psi(\ell, f^{\ell}_{\min})$ is the center of $\mathcal{C}$ that is the second closest to ℓ. Replacing we have,

$$\Delta_4 \leqslant \varepsilon^{-1} \sum_{\ell \in \mathcal{L}} \frac{2^p |V_{\mathcal{C}}(f^{\ell}_{\min}) \cap V_{\mathcal{L}}(\ell)|}{|V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f^{\ell}_{\min})|} \sum_{c \in V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f^{\ell}_{\min})} (\ell_c + g_c)$$

Now, since $\ell \in \hat{\mathcal{L}}(f^{\ell}_{\min})$, we have that $|V_{\mathcal{C}}(f^{\ell}_{\min}) \cap V_{\mathcal{L}}(\ell)| / |V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f^{\ell}_{\min})| \leqslant (1 - \varepsilon)/\varepsilon$. Therefore,

$$\Delta_4 \leqslant 2^p (1 - \varepsilon) \varepsilon^{-2} \sum_{c \in V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f^{\ell}_{\min})} (\ell_c + g_c).$$

Putting $\Delta_1, \Delta_2, \Delta_3, \Delta_4$ together we obtain that the total cost increase induced by the reassignment is at most $2^{3p+1}(\text{cost}(\mathcal{C}) + \text{cost}(\mathcal{L}))/\varepsilon^2$. $\square$

3.4 Clustering in Graphs with Small Separators

Before proving Theorem 1.3, we give a much simpler proof that Local Search is a PTAS for the uniform facility location problem (Definition A.2). The goal is to provide the reader with more intuition on our analysis using r-division. In Section 3.4.2, we add second ingredient, isolation, in order to prove Theorem 1.2. In this section we will use the terms facility and centers interchangeably to denotes the elements of a solution set for both the facility location and k-clustering problems.

3.4.1 Local Search for Facility Location

We consider the following adaption of Algorithm 2 to the facility location problem.

Algorithm 3 Local Search for Uniform Facility Location

```
1: Input: A metric space and associated cost function cost($\cdot$), an $n$-element set $A$ of
   points, error parameter $\varepsilon > 0$, facility opening cost $f > 0$, positive integer parameter $s$
2: $S \leftarrow$ Arbitrary subset of $\mathcal{F}$.
3: while $\exists\, S'$ s.t. $|S \backslash S'| + |S' \backslash S| \leqslant s$ and cost($S'$) $\leqslant (1 - \varepsilon/n)$ cost($S$)
4: do
5:     $S \leftarrow S'$
6: Output: $S$
```

We show that Algorithm 3 is a PTAS for uniform facility location in well-separated instances.

Theorem 3.15. *Fix a nontrivial minor-closed family $\mathcal{K}$ of graphs. There is a constant c such that, when Algorithm 3 is applied to the metric completion of any graph $G = (A \cup F, E)$ in $\mathcal{K}$ with*

$$\mathit{cost}(S) = |S|f + \sum_{a \in A} (\min_{u \in S} \mathit{dist}(a, u))^p$$

and $s = 1/\varepsilon^c$, the output has cost at most $1 + \varepsilon$ times the minimum.

Throughout this section on facility location, define $\mathcal{L}$ to be a solution output by Algorithm 3 (the "local" solution) and a globally optimal solution $\mathcal{C}$ of value OPT. Let $\mathcal{F} = \mathcal{L} \cup \mathcal{C}$. Let $r = 1/\varepsilon^2$. Consider the graph $G_{\text{Vor}(\mathcal{F})}$ defined in Definition 3.4, and recall that each vertex of G maps to a vertex $\hat{v}$ in the contracted graph $G_{\text{Vor}(\mathcal{F})}$.

Since G belongs to $\mathcal{K}$ and $G_{\text{Vor}(\mathcal{F})}$ is obtained from G by contraction, it belongs to $\mathcal{K}$ too and hence it has an r-division. Let $H_1, \ldots H_\kappa$ be the regions of this r-division. For $i = 1, \ldots, \kappa$, define V_i and B_i as follows:

$$\begin{aligned} V_i &= \{v \in \mathcal{F} \;:\; \hat{v} \text{ is a vertex of } H_i\} \\ B_i &= \{v \in \mathcal{F} \;:\; \hat{v} \text{ is a boundary vertex of } H_i\} \end{aligned}$$

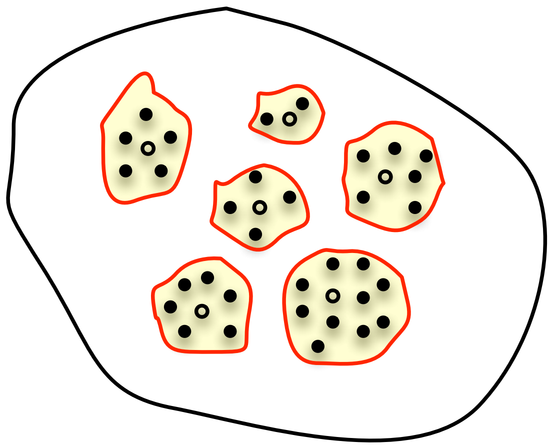


Figure 3.2: The diagram shows a region of the weak r-division. The blobs represent vertices of the region. Each blob is obtained by coalescing a set of vertices of the input graph. These vertices are indicated by circles. The unfilled circles represent the centers of the Voronoi cells.

That is, V_i is the set of vertices in the union of the local solution and the global solution that map via contraction to vertices of the region H_i, and B_i is the set of vertices in the union that map to *boundary* vertices of H_i.

Let $\mathcal{C}' = \mathcal{C} \cup \bigcup_{i=1}^{\kappa} B_i$ and fix a region H_i of the r-division of $G_{\text{Vor}(\mathcal{F})}$. We define $\mathcal{L}_i = \mathcal{L} \cap V_i$ and $\mathcal{C}'_i = \mathcal{C}' \cap V_i$. We consider the mixed solution $\mathcal{M}^i$ defined as follows:

$$\mathcal{M}^i = (\mathcal{L} \backslash \mathcal{L}_i) \cup \mathcal{C}'_i.$$

Lemma 3.16. *We have,* $|\mathcal{M}^i \backslash \mathcal{L}| + |\mathcal{L} \backslash \mathcal{M}^i| \leqslant 1/\varepsilon^2$.

Proof. To obtain $\mathcal{M}^i$ from $\mathcal{L}$, one can remove the vertices in $\mathcal{L} \cap V_i$ that are not in $\mathcal{G}'$, and add the vertices in $\mathcal{G}' \cap V_i$ that are not in $\mathcal{L}$. Thus the size of the symmetric difference is at most $|(\mathcal{L} \cup \mathcal{C}') \cap V_i|$. Since the vertices of $\mathcal{L} \cup \mathcal{C}'$ are centers of Voronoi cells, these vertices all map to different vertices in the contracted graph $G_{\text{Vor}(\mathcal{F})}$. Therefore $|(\mathcal{L} \cup \mathcal{C}') \cap V_i|$ is at most the number of vertices in region H_i, which is at most $r = 1/\varepsilon^2$. □

Denote by m_a^i the cost induced by client a in solution $\mathcal{M}^i$

Lemma 3.17. *Let a be a vertex of G and H_i a region. Then:*

$$m_a^i - \ell_a \leqslant \begin{cases} g_a - \ell_a & \text{if } \hat{a} \text{ is an internal vertex of } H_i \\ 0 & \text{otherwise.} \end{cases}$$

Proof. First suppose $\hat{a}$ is an internal vertex of H_i, and let v be the facility in $\mathcal{M}^i$ closest to a. If v is in V_i then it is in $\mathcal{C}_i$, so $m_a^i = g'_a$. Suppose v is not in V_i. Then by Lemma 3.8

there is a vertex $x \in \mathcal{F}$ such that $\hat{x}$ is a boundary vertex of H_i and $\text{dist}(a, x) \leqslant \text{dist}(a, v)$. As before, x is in $\mathcal{C}'_i$ so $m^i_a \leqslant g'_a$. Since $g'_a \leqslant g_a$, this proves the claimed upper bound.

Now, suppose $\hat{a}$ is not an internal vertex of H_i and let v be the facility in $\mathcal{L}$ closest to a. If v is not in V_i then it is in the mixed solution $\mathcal{M}^i$, so $m^i_a = \ell_a$. Suppose v is in V_i. Then by Lemma 3.8 there is a vertex $x \in \mathcal{F}$ such that $\hat{x}$ is a boundary vertex of H_i and $\text{dist}(a, x) \leqslant \text{dist}(a, v)$. Since x is in $\mathcal{F}$ and $\hat{x}$ is a boundary vertex of H_i, we know x is in $\mathcal{C}' \cap V_i$, which is $\mathcal{C}'_i$. Therefore x is in $\mathcal{M}^i$. Since $\text{dist}(a, x) \leqslant \text{dist}(a, v)$, we obtain $m^i_a \leqslant \ell_a$, which proves the claimed upper bound. □

Lemma 3.18. *We have,*

$$\sum_{i=1}^{\kappa} |\mathcal{C}'_i| \leqslant |\mathcal{C}| + c_2\varepsilon(|\mathcal{C}| + |\mathcal{L}|).$$

Proof. Let v be a vertex of $\mathcal{C}'$. For $i = 1, \ldots, \kappa$, if $\hat{v}$ is an internal vertex of the region H_i then v contributes only one towards the left-hand side. If $\hat{v}$ is a boundary vertex of H_i then $v \in B_i$. Therefore

$$\sum_{i=1}^{\kappa} |\mathcal{C}' \cap V_i| \leqslant |\mathcal{C}| + \sum_{i=1}^{\kappa} |B_i|.$$

To finish the proof, we bound the sum in the right-hand side. Each vertex in $\mathcal{F}$ is the center of one Voronoi cell, so $G_{\text{Vor}(\mathcal{F})}$ has $|\mathcal{F}|$ vertices. For each region H_i, there is one vertex in B_i that corresponds to each boundary vertex of H_i, so $\sum_{i=1}^{\kappa} |B_i|$ is the sum over all regions of the number of boundary vertices of that region, which, by Property 4 of r-divisions, is at most $c_2|\mathcal{F}|/r^{1/2}$, which, by choice of r, is at most $c_2\varepsilon|\mathcal{F}|$, which in turn is at most $c_2\varepsilon(|\mathcal{C}| + |\mathcal{L}|)$. □

Proof. (Proof of Theorem 3.15) Lemma 3.16 and the stopping condition of Algorithm 3 imply the following:

$$-\frac{1}{n}\text{cost}(\mathcal{L}) \leqslant \text{cost}(\mathcal{M}^i) - \text{cost}(\mathcal{L}). \tag{3.4}$$

We now decompose the right-hand side. For a client a, we denote by ℓ_a, g'_a and m^i_a the distance from the client a to the closest facilities in $\mathcal{L}$, $\mathcal{C}'$ and $\mathcal{M}^i$ respectively. This gives

$$\text{cost}(\mathcal{M}^i) - \text{cost}(\mathcal{L}) = (|\mathcal{C}'_i| - |\mathcal{L}_i|) \cdot f + \sum_a (m^i_a - \ell_a). \tag{3.5}$$

Using Lemma 3.17 and summing over a shows that

$$\sum_a (m^i_a - \ell_a) \leqslant \sum_{a:\hat{a} \in \mathcal{I}(H_i)} (g_a - \ell_a). \tag{3.6}$$

Combining Inequalities (3.4), (3.5) and (3.6), we obtain

$$-\frac{1}{n}\text{cost}(\mathcal{L}) \leqslant (|\mathcal{C}'_i| - |\mathcal{L}_i|) \cdot f + \sum_{a:\hat{a} \in \mathcal{I}(H_i)} (g'_a - \ell_a) \tag{3.7}$$

We next sum this inequality over all κ regions of the weak r-division and use Lemma 3.18.

$$\begin{aligned}
-\frac{\kappa}{n}\text{cost}(\mathcal{L}) &\leqslant \left(\sum_{i=1}^{\kappa}|\mathcal{C}'_i| - \sum_{i=1}^{\kappa}|\mathcal{L}_i|\right)\cdot f + \sum_{i=1}^{\kappa}\sum_{a:\hat{a}\in\mathcal{I}(H_i)}(g_a - \ell_a)\\
&\leqslant (|\mathcal{C}| + (c_2\varepsilon(|\mathcal{C}| + |\mathcal{L}|) - |\mathcal{L}|)\cdot f + \sum_a (g_a - \ell_a)\\
&= ((1+c_2\varepsilon)|\mathcal{C}| - (1-c_2\varepsilon)|\mathcal{L}|)\cdot f + \sum_a (g_a - \ell_a)\\
&\leqslant (1+c_2\varepsilon)\text{cost}(\mathcal{C}) - (1-c_2\varepsilon)\text{cost}(\mathcal{L})
\end{aligned}$$

Since $\kappa \leqslant c_1|\mathcal{F}|/r \leqslant c_1\varepsilon^2 n$, we obtain

$$-c_1\varepsilon^2\text{cost}(\mathcal{L}) \leqslant (1+c_2\varepsilon)\text{cost}(\mathcal{C}) - (1-c_2\varepsilon)\text{cost}(\mathcal{L})$$

so

$$\text{cost}(\mathcal{L}) \leqslant (1 - c_2\varepsilon - c_1\varepsilon^2)^{-1}(1+c_2\varepsilon)\text{cost}(\mathcal{C})$$

This completes the proof of Theorem 3.15. □

3.4.2 Local Search for the k-Clustering Problem

We now prove Theorem 1.2. The proof is similar for graphs and for points lying in R^d. It builds on the notions of isolation and 1-1 isolation introduced in Section 3.2. As in the previous section, $\mathcal{L}$ denotes the output of Algorithm 2 and $\mathcal{C}$ denotes an optimal solution. We use a parameter $0 < \varepsilon < 1/2$ whose value depends on δ and is chosen later. Let $\bar{\mathcal{F}}$ be the set of facilities of $\mathcal{L}$ and $\mathcal{C}$ that are not in 1-1 ε-isolated regions and let $\bar{k} = |\bar{\mathcal{F}}|$.

Theorem 1.2. *Let $\mathcal{K}$ be a nontrivial minor-closed family of edge-weighted graphs. For any integer $p > 0$, there is a constant c with the following property. For any $0 < \varepsilon < 1/2$, for any graph $G = (A \cup F, E)$ in $\mathcal{K}$, Algorithm 2 applied to the metric completion of G with cost function*

$$\textit{cost}(S) = \sum_{a\in A}(\min_{u\in S}\textit{dist}(a,u))^p$$

and with $s = 1/\varepsilon^c$ yields a solution S whose cost is at most $1+\varepsilon$ times the minimum.

Recall Algorithm 2.

Applying Theorem 3.13 to $\mathcal{C}$ and $\mathcal{L}$ yields a set $S_0 \subset \mathcal{C}$ such that $\text{cost}(\mathcal{C} - S_0) \leqslant (1 + 2^{3p+1}\varepsilon)\text{cost}(\mathcal{C}) + 2^{3p+1}\varepsilon\,\text{cost}(\mathcal{L})$ and $|S_0| \geqslant \varepsilon^3\bar{k}/6$. Let $\mathcal{C}_1 = \mathcal{C}\backslash S_0$.

For a client c, $\mathcal{L}(c)$ and $\mathcal{C}(c)$ denote, respectively, the facility of $\mathcal{L}$ serving c and the facility of $\mathcal{C}$ serving c. We say c is *bad* if $\mathcal{L}(c)$ and $\mathcal{C}(c)$ do not belong to the same ε-isolated region but at least one is isolated; otherwise c is *good*. We define a subset E' of

Algorithm 2 Local search for finding k clusters.

```
1: Input: A metric space and associated cost function $\text{cost}(\cdot)$, an $n$-element set $C$ of points, error parameter $\varepsilon > 0$, number of clusters $k$
2: Parameter: A positive integer parameter $s$ for the neighborhood size
3: $S \leftarrow$ Arbitrary size-$k$ set of points
4: while $\exists\, S'$ s.t. $|S'| \leqslant k$ and $|S \setminus S'| + |S' \setminus S| \leqslant s$ and $\text{cost}(S') \leqslant (1 - \varepsilon/n)\text{cost}(S)$
5:  do
6:      $S \leftarrow S'$
7: Output: $S$
```

edges of G as follows: for each good client c, include in E' the shortest c-to-$\mathcal{L}(c)$ and c-to-$\mathcal{C}(c)$ paths. Let $G' = (V, E')$ be the corresponding subgraph of G.

Let $\mathcal{F} = \mathcal{C}_1 \cup \mathcal{L}$. Let $R_1, R_2, \ldots$ be an r-division of $G'_{\text{Vor}(\mathcal{F})}$ where $r = 1/\varepsilon^7$. Define $\mathcal{C}^* = \mathcal{C}_1 \cup \{\text{boundary vertices of the } r\text{-division}\}$.

The vertex sets of regions are of course not disjoint—a boundary vertex is in multiple regions—but it is convenient to represent them by disjoint sets. We therefore define a ground set $\Omega = \{(v, R) : v \text{ a vertex of } G'_{\text{Vor}(\mathcal{F})}, R \text{ a region containing } v\}$, and, for each region R, we define $\widehat{R} = \{(v, R) : v \text{ a vertex of } R\}$. Now $\widehat{R_1}, \widehat{R_2}, \ldots$ form a partition of Ω. To allow us to go from an element of Ω back to a vertex, if $x = (v, R)$ we define $\breve{x} = v$. Finally, define $\widehat{\mathcal{G}} = \{(v, R) \in \Omega : v \in \mathcal{C}^*\}$.

Let $\bar{\mathcal{F}}$ be the set of facilities of $local$ and $\mathcal{C}$ that are not in 1-1 isolated regions.

Lemma 3.19. *We have, $|\widehat{\mathcal{G}}| \leqslant |\mathcal{C}_1| + c_2\varepsilon^{3.5}|\bar{\mathcal{F}}|$, where c_2 is the constant in the definition of r-division.*

Proof. Consider the r-division. Each 1-1 isolated region results in a connected component of size 2 in $G'_{\text{Vor}(\mathcal{F})}$ and so no boundary vertices arise from such connected components. By the definition of r-division, the sum over regions of boundary vertices is at most $c_2 \cdot |\bar{n}_0|/r^{1/2}$, where $\bar{n}_0$ is the total number of elements of $\mathcal{C}_1$ and $\mathcal{L}$ that are not in 1-1 isolated regions. Since $r = 1/\varepsilon^7$, we have that $|\widehat{\mathcal{G}}| \leqslant |\mathcal{C}_1| + c_2 \cdot \varepsilon^4|\bar{\mathcal{F}}|$. □

Lemma 3.20. *We have, $|\widehat{\mathcal{G}}| \leqslant k$.*

Proof. By Theorem 3.13, we have that $|\mathcal{C}_1| \leqslant k - \varepsilon^3\bar{k}/12$. By Lemma 3.19 we thus have

$$|\widehat{\mathcal{G}}| \leqslant |\mathcal{C}_1| + c_2\varepsilon^{3.5}\bar{k} \leqslant k - \varepsilon^3\bar{k}/12 + c_2\varepsilon^{3.5}\bar{k} \leqslant k,$$

for ε small enough. □

Throughout the rest of the proof, we will bound the cost of $\mathcal{L}$ by the cost of $\mathcal{C}^*$. We now slightly abuse notations in the following way: each facility ℓ of $\mathcal{L}$ that belongs to an isolated region and that is a boundary vertex is now in $\mathcal{C}^*$. We say that this facility is isolated.

Lemma 3.21 (Balanced Partitioning)**.** *Let $\mathcal{S} = \{S_1, ..., S_p\}$ and $\{A, B\}$ be partitions of some ground set. Suppose $|A| \geqslant |B|$ and, for $= 1, \dots, p$, $1/(2\varepsilon^2) \leqslant |S_i| \leqslant 1/\varepsilon^2$. There exists a partition that is a coarsening of $\mathcal{S}$ such that every part C of the coarser partition satisfies the following:*

- ***Small Cardinality**: C is the union of $\mathcal{O}(1/\varepsilon^5)$ parts of $\mathcal{S}$.*
- ***Balanced:** $|C \cap A| \geqslant |C \cap B|$.*

Proof. We first define for each set S_i, $v(S_i) := |A \cap S_i| - |B \cap S_i|$.

The sizes of the S_i imply that $v(S_i)$ is an integer in the range $[-1/\varepsilon^2, 1/\varepsilon^2]$. We iteratively construct a coarsening of $\mathcal{S}$ such that for each part P of the coarsening, P contains $O(1/\varepsilon^5)$ sets of $\mathcal{S}$ and $\sum_{S_i \in P} v(S_i) \geqslant 0$. It is easy to see that this coarsening satisfies the condition of the lemma.

For any set S_i such that $v(S_i) = 0$, we create a new part that contains only this set. This part trivially satisfies the above property.

We now consider the remaining sets. For i in $[-1/\varepsilon^2, 1/\varepsilon^2]$, define $\mathcal{V}_i$ to be the collection of sets $S_\ell \in \mathcal{S}$ such that $v(S_\ell) = i$.

While there exists $1 \leqslant i, j$, such that $1/\varepsilon^2 < |\mathcal{V}_i|, |\mathcal{V}_{-j}|$, We take i sets from $\mathcal{V}_{-j}$ and j sets from $\mathcal{V}_i$ and create a new part containing all of them. This part satisfies the property of the Lemma and contains at most $2/\varepsilon^2$ sets of $\mathcal{S}$.

Let $\Delta = |A| - |B|$. By a direct induction, at each step before the final step we have that the sum of $\mathcal{V}(S_i)$ for the remaining set S_i is equal to Δ. We now consider the final step. We have $\forall j \geqslant 0$, $|\mathcal{V}_j| \leqslant 1/\varepsilon^2$ or $\forall j \leqslant 0$ $|\mathcal{V}_j| \leqslant 1/\varepsilon^2$. Since $\Delta > 0$, it must be the case that $\forall j \leqslant 0$ $|\mathcal{V}_j| \leqslant 1/\varepsilon^4$. We now construct a part containing all the sets in the $\mathcal{V}_j$, for $j < 0$. Additionally, we add the smallest number of sets of $\mathcal{V}_j$, for $j > 0$, such that the balanced property is satisfied. Not that since $\Delta \geqslant 0$, this is possible. Furthermore, this part has size at most $O(1/\varepsilon^4)$. Finally, we create a new part for each remaining set of the $\mathcal{V}_j$, for $j > 0$. Each such part satisfies the conditions of the lemma. □

We now apply Lemma 3.21 to the partition $\widehat{R_1}, \widehat{R_2}, \dots$ with $A = \{(v, R) \in \Omega \ : \ v \in \mathcal{L}\}$ and $B = \widehat{\mathcal{G}}$. We refer to the parts of the resulting coarse partition as *super-regions*. Each super-region $\mathcal{R}$ naturally corresponds to a subgraph of $G'_{\text{Vor}(\mathcal{F})}$, the subgraph induced by $\{v \ : \ (v, R) \in \mathcal{R}\}$, and we sometimes use $\mathcal{R}$ to refer to this subgraph.

For a super-region $\mathcal{R}$, let $\mathcal{L}(\mathcal{R})$ (resp. $\mathcal{C}^*(\mathcal{R})$) be the set of facilities of $\mathcal{L}$ (resp. $\mathcal{C}^*$) in the super-region $\mathcal{R}$, i.e.: the set $\{\ell \mid \ell \in \mathcal{L}$ and $(\ell, R) \in \Omega\}$ (resp. $\{f \mid f \in \mathcal{C}^*$ and $(f, R) \in \Omega\}$). We consider the mixed solution

$$\mathcal{M}_{\mathcal{R}} = (\mathcal{L} \backslash \mathcal{L}(\mathcal{R})) \cup \mathcal{C}^*(\mathcal{R}).$$

Lemma 3.22. *$|\mathcal{M}_{\mathcal{R}} \backslash \mathcal{L}| + |\mathcal{L} \backslash \mathcal{M}_{\mathcal{R}}| = O(1/\varepsilon^{12})$ and $|\mathcal{M}_{\mathcal{R}}| \leqslant k$.*

Proof. Each region of the r-division contains at most $1/\varepsilon^7$ facilities. By Lemma 3.21, each super-region is the union of $\mathcal{O}(1/\varepsilon^5)$ regions □

For each client c, we define $g_c = \text{dist}(c, \mathcal{C}(f))^p$ and $\ell_c = \text{dist}(c, \mathcal{L}(c))^p$. For any client $c \in V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)$ for some isolated region $(f_0, \mathcal{L}_0)$, define $\text{Reassign}_{\mathcal{C}* \mapsto \mathcal{L}}(c)$ to be the cost of assigning c to the facility of $\mathcal{L}_0$ that is the closest to f_0. We let ε_1 denote a positive constant to be chosen later.

Lemma 3.23. *Consider an isolated region* $(f_0, \mathcal{L}_0)$.

$$\sum_{\substack{c \in \\ V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)}} \mathit{Reassign}_{\mathcal{C}* \mapsto \mathcal{L}}(c) \leqslant (1+\varepsilon_1)^p \sum_{\substack{c \in \\ V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)}} g_c + \frac{2^p (1+\varepsilon_1)^p \varepsilon_1^{-p} \varepsilon}{1-\varepsilon} \sum_{c \in V_{\mathcal{C}}(f_0)} (g_c + \ell_c),$$

Proof. Consider a client $c \in V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)$, and let ℓ denote the facility of $\mathcal{L}$ that is the closest to f_0. By Lemma 2.9, $\text{dist}(c, \ell)^p \leqslant (1+\varepsilon_1)^p (\text{dist}(c, f_0)^p + \varepsilon_1^{-p} \text{dist}(\ell, f_0)^p) = (1+\varepsilon_1)^p (g_c + \varepsilon_1^{-p} \text{dist}(\ell, f_0)^p)$. Summing over $c \in V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)$,

$$\sum_{c \in V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)} \text{Reassign}_{\mathcal{C}* \mapsto \mathcal{L}}(c) \leqslant (1+\varepsilon_1)^p \Big(\sum_{c \in V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)} g_c + \frac{|V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)|}{\varepsilon_1^p} \text{dist}(\ell, f_0)^p \Big).$$

To upper bound $\text{dist}(\ell, f_0)^p$, we use an averaging argument. Since ℓ is the facility in $\mathcal{L}$ that is closest to f_0, $\text{dist}(\ell, f_0)^p \leqslant \text{dist}(\mathcal{L}(c), f_0)^p$. By Lemma 2.9, $\text{dist}(\mathcal{L}(c), f_0)^p \leqslant 2^p(\text{dist}(\mathcal{L}(c), c')^p + \text{dist}(c, f_0)^p) = 2^p(\ell_c + g_c)$, thus

$$\text{dist}(\ell, f_0)^p \leqslant \frac{2^p}{|V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)|} \sum_{c \in V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)} (\ell_c + g_c).$$

Substituting, we infer that

$$\sum_{c \in V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)} \text{Reassign}_{\mathcal{L} \mapsto \mathcal{C}*}(c)$$

is at most the sum of

$$(1+\varepsilon_1)^p \sum_{c \in V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)} g_c$$

and

$$2^p (1+\varepsilon_1)^p \varepsilon_1^{-p} \frac{|V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)|}{|V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)|} \sum_{c \in V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)} (\ell_c + g_c).$$

By definition of isolated regions, $V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0) \leqslant \varepsilon |V_{\mathcal{C}}(f_0)|$ and $|V_{\mathcal{C}}(f_0) \backslash V_{\mathcal{L}}(\mathcal{L}_0)| \geqslant (1-\varepsilon)|V_{\mathcal{C}}(f_0)|$, so the ratio is at most $\varepsilon/(1-\varepsilon)$. Summing over $\ell \in \mathcal{L}_0$ proves the lemma. □

Similarly, for any client $c \in V_{\mathcal{L}}(\mathcal{L}_0) \backslash V_{\mathcal{C}}(f_0)$ for some isolated region $(f_0, \mathcal{L}_0)$, define $\text{Reassign}_{\mathcal{L} \mapsto \mathcal{C}*}$ as the cost of assigning c to f_0.

Lemma 3.24. *For an isolated region $(f_0, \mathcal{L}_0)$, the sum*

$$\sum_{c \in V_{\mathcal{L}}(\mathcal{L}_0) \backslash V_{\mathcal{C}}(f_0)} \mathit{Reassign}_{\mathcal{L} \mapsto \mathcal{C}^*}(c)$$

is at most

$$(1+\varepsilon_1)^p \sum_{c \in V_{\mathcal{L}}(\mathcal{L}_0) \backslash V_{\mathcal{C}}(f_0)} \ell_c$$

plus

$$\frac{2^p(1+\varepsilon_1)^p \varepsilon_1^{-p} \varepsilon}{1-\varepsilon} \sum_{c \in V_{\mathcal{C}}(f_0)} (g_c + \ell_c).$$

Proof. Let ℓ be a facility in $\mathcal{L}_0$. For each client $c \in V_{\mathcal{L}}(\mathcal{L}_0) \backslash V_{\mathcal{C}}(f_0)$, by Lemma 2.8, $\text{dist}(c, f_0)^p \leqslant (1+\varepsilon_1)^p(\text{dist}(c,\ell)^p + \varepsilon_1^{-p}\text{dist}(\ell, f_0)^p) = (1+\varepsilon_1)^p(\ell_c + \varepsilon_1^{-p}\text{dist}(\ell, f_0)^p)$. Therefore

$$\sum_{c \in V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f_0)} \text{Reassign}_{\mathcal{L} \mapsto \mathcal{C}^*}(c)$$

is at most $(1+\varepsilon_1)^p$ times

$$\sum_{c \in V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f_0)} \ell_c + \varepsilon_1^{-p} |V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f_0)| \, \text{dist}(\ell, f_0)^p.$$

To upper-bound $\text{dist}(\ell, f_0)$, we use an averaging argument. For each client $c \in V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)$, by Lemma 2.9 we have $\text{dist}(\ell, f_0)^p \leqslant 2^p(\text{dist}(\ell, c)^p + \text{dist}(c, f_0)^p) = 2^p(\ell_c + g_c)$, thus

$$\text{dist}(\ell, f_0)^p \leqslant \frac{2^p}{|V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)|} \sum_{c \in V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)} (\ell_c + g_c).$$

Substituting, we infer that

$$\sum_{c \in V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f_0)} \text{Reassign}_{\mathcal{L} \mapsto \mathcal{C}^*}(c)$$

is at most $(1+\varepsilon_1)^p$ times

$$\sum_{c \in V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f_0)} \ell_c \qquad +$$

$$\frac{2^p \varepsilon_1^{-p} |V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f_0)|}{|V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)|} \sum_{c \in V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)} (\ell_c + g_c).$$

By definition of isolated regions, $|V_{\mathcal{L}}(\ell) \backslash V_{\mathcal{C}}(f_0)| \leqslant \varepsilon |V_{\mathcal{L}}(\ell)|$ and $|V_{\mathcal{L}}(\ell) \cap V_{\mathcal{C}}(f_0)| \geqslant (1-\varepsilon)|V_{\mathcal{L}}(\ell)|$, so the ratio is at most $\varepsilon/(1-\varepsilon)$. Summing over $\ell \in \mathcal{L}_0$ proves the lemma. □

Lemma 3.25. *Consider an isolated region $(f, \mathcal{L}_0)$. Let ℓ be a facility of $\mathcal{L}_0$. For any super-region $\mathcal{R}$, $\mathcal{M}_\mathcal{R}$ contains f or a facility that is at distance at most dist(ℓ, f) from f.*

Proof. Since ℓ and f belong to the same isolated region $(f, \mathcal{L}_0)$ and $\ell \in \mathcal{L}_0$, they belong to the same connected component of $G'_{\text{Vor}}(\mathcal{F})$. Now consider a super-region $\mathcal{R}$ such that $\mathcal{M}_\mathcal{R}$ does not contain ℓ. Then $\ell \in \mathcal{L}(\mathcal{R})$. Thus, either $f \in \mathcal{R}$ or by Lemma 3.8, a boundary element $\ell' \in \mathcal{R}$ of the r-division is on the path from ℓ to f and dist$(\ell', f) \leqslant$ dist(ℓ, f). Thus, $\ell' \in \mathcal{M}_\mathcal{R}$, proving the lemma. □

For a client c and a super-region $\mathcal{R}$, we define $m_\mathcal{R}(c)$ to be the cost of c in the mixed solution $\mathcal{M}_\mathcal{R}$. Moreover, for each client c, we consider the facilities $\mathcal{L}(c)$ and $\mathcal{C}^*(c)$ that serve this client in solution $\mathcal{L}$ and $\mathcal{C}^*$ respectively. We define $\ell(c)$ to be an arbitrary pair $(\mathcal{L}(c), R) \in \Omega$ and $g^*(c)$ to be an arbitrary pair $(\mathcal{C}^*(c), R) \in \Omega$. We say that (v, R) is *isolated* if v belongs to one of the isolated regions.

Lemma 3.26. *Let c be a good client and $\mathcal{R}$ a super-region. The value of $m_\mathcal{R}(c) - \ell_c$ is less than or equal to:*

$$\begin{cases} g_c - \ell_c & \textit{if } g^*(c) \in \mathcal{R} \\ 0 & \textit{otherwise} \end{cases}$$

Proof. Observe that if $g^*(c) \in \mathcal{R}$, then $m_\mathcal{R}(c) \leqslant g_c$ and the first case holds. Now, for any super-region $\mathcal{R} \not\ni l(c), g^*(c)$, $\mathcal{M}_\mathcal{R}$ contains the facility serving client c in local. Thus its cost is at most ℓ_c and the second case holds. Finally, assume that $\mathcal{R}$ contains $l(c)$ and does not contain $g^*(c)$. If $\hat{c}$ belongs to $\mathcal{R}$, then by the separation property of the r-division (see Lemmas 3.8, 3.9), $g^*(c) \in \mathcal{R}$ and $m_\mathcal{R}(c) \leqslant g_c$. Otherwise, $\hat{c} \notin \mathcal{R}$, and so, by the separation property there must be a boundary vertex of $\mathcal{R}$ that is closer to c than the facility that serves it in $\mathcal{L}$. Therefore, we have $m_\mathcal{R}(c) \leqslant \ell_c$ and the second case holds. □

We now turn to the bad clients.

Lemma 3.27. *Let c be a bad client and $\mathcal{R}$ a super-region. If $\ell(c) \in \mathcal{R}$ then $m_\mathcal{R}(c) - \ell_c$ is at most*

$$\begin{cases} g_c - \ell_c & \textit{if } g^*(c) \in \mathcal{R} \\ \textit{Reassign}_{\mathcal{C}^* \mapsto \mathcal{L}}(c) - \ell_c & \textit{if } g^*(c) \notin \mathcal{R} \textit{ and } g^*(c) \textit{ is isolated} \\ \textit{Reassign}_{\mathcal{L} \mapsto \mathcal{C}^*}(c) - \ell_c & \textit{if } g^*(c) \notin \mathcal{R} \textit{ and } g^*(c) \textit{ is not isolated} \\ 0 & \textit{otherwise.} \end{cases}$$

if $\ell(c) \notin \mathcal{R}$ then $m_\mathcal{R}(c) - \ell_c$ is at most

$$\begin{cases} g_c - \ell_c & \textit{if } g^*(c) \in \mathcal{R} \textit{ and } g^*(c) \textit{ is not isolated} \\ 0 & \textit{otherwise.} \end{cases}$$

Proof. Observe that the super-regions form a partition of the $l(c)$ and $g^*(c)$. Let $\mathcal{R}(\ell(c))$ be the region that contains $\ell(c)$ and $\mathcal{R}(g^*(c))$ be the region that contains $g^*(c)$. If $\mathcal{R}(\ell(c)) = \mathcal{R}(g^*(c))$ then, the facility serving c in $\mathcal{C}^*$ is in $\mathcal{M}_{\mathcal{R}(\ell(c))}$, hence $m_{\mathcal{R}(\ell(c))}(c) \leqslant g_c$. Moreover for any other region $\mathcal{R}' \neq \mathcal{R}(\ell(c))$, we have $\ell(c) \notin \mathcal{R}'$ and so the facility serving c in $\mathcal{L}$ is in $\mathcal{M}_{\mathcal{R}'}$. Therefore $m_{\mathcal{R}'}(c) \leqslant \ell_c$.

Thus, we consider c such that $\mathcal{R}(\ell(c)) \neq \mathcal{R}(g^*(c))$. Since c is bad, we have that $\ell(c)$ or $g^*(c)$ is isolated. Consider the case where $g^*(c)$ is isolated. The cost of c in solution $\mathcal{M}_{\mathcal{R}(\ell(c))}$ is, by Lemma 3.25, at most $\text{Reassign}_{\mathcal{C}^* \mapsto \mathcal{L}}(c)$ satisfying the lemma. Now, for any other region $\mathcal{R}' \neq \mathcal{R}(g^*(c))$, again we have $\ell(c) \notin \mathcal{R}'$ and so the facility serving c in $\mathcal{L}$ is in $\mathcal{M}_{\mathcal{R}'}$. Therefore, $m_{\mathcal{R}'}(c) \leqslant \ell_c$.

Therefore, we consider the case where c is such that $\mathcal{R}(\ell(c)) \neq \mathcal{R}(g^*(c))$ and such that $g^*(c)$ is not isolated. Since c is bad, $\ell(c)$ is isolated. Thence, by Lemma 3.25, the cost in solution $\mathcal{M}_{\mathcal{R}(\ell(c))}$ is at most $\text{Reassign}_{\mathcal{L} \mapsto \mathcal{C}^*}(c)$, satisfying the Lemma. Moreover, in solution $\mathcal{R}(g^*(c))$, the cost is at most g_c. Finally, for any other region $\mathcal{R}' \neq \mathcal{R}(\ell(c)), \mathcal{R}(g^*(c))$, $\ell(c) \notin \mathcal{R}'$ and so the facility serving c in $\mathcal{L}$ is in $\mathcal{M}_{\mathcal{R}'}$. Therefore, $m_{\mathcal{R}'}(c) \leqslant \ell_c$, concluding the proof of the lemma. □

We now partition the clients into three sets, $\Lambda_1, \Lambda_2, \Lambda_3$. Let Λ_1 be the set of bad clients such that there exists a super-region $\mathcal{R}$ such that $\ell(c) \in \mathcal{R}$ and $g^*(c) \notin \mathcal{R}$ and $g^*(c)$ is not isolated. Let Λ_2 be the set of bad clients such that there exists a super-region $\mathcal{R}$ such that $\ell(c) \in \mathcal{R}$ and $g^*(c) \notin \mathcal{R}$ and $g^*(c)$ is isolated. Finally let Λ_3 be the remaining clients : $\Lambda_3 = C \backslash \Lambda_1 \backslash \Lambda_2$. The next corollary follows from combining Lemmas 3.26 and 3.27 and by observing that the super-regions form a partition of the $\ell(c)$ and $g^*(c)$.

Corollary 3.28. *For any client c, we have that*

$$\sum_{\mathcal{R}} (m_{\mathcal{R}}(c) - \ell_c) \leqslant \begin{cases} \textit{Reassign}_{\mathcal{L} \mapsto \mathcal{C}^*} + g_c - 2\ell_c & \textit{if } c \in \Lambda_1 \\ \textit{Reassign}_{\mathcal{C}^* \mapsto \mathcal{L}} - \ell_c & \textit{if } c \in \Lambda_2 \\ g_c - \ell_c & \textit{if } c \in \Lambda_3 \end{cases}$$

We now turn to the proof of Theorem 1.2.

Let $\alpha > 1$ be a constant to be chosen later. Now we choose ε_1 and ε. We maximize ε_1 subject to $(1 + \varepsilon_1)^p \leqslant 1 + \delta/\alpha$. Note that $\varepsilon_1 = \Theta(\delta/p)$. We maximize ε subject to $(2(1 + \varepsilon_1)/\varepsilon_1)^p \varepsilon/(1 - \varepsilon) \leqslant \delta/\alpha$ and $2^{3p+1}\varepsilon \leqslant \delta/\alpha$. Note that $\varepsilon = \Theta(\delta^{p+1})$. By Lemma 3.22, therefore, there is a constant c such that using neighborhood parameter $s = 1/\delta^c$ in Algorithm 1 ensures that, for any super-region $\mathcal{R}$ the solution $\mathcal{M}_{\mathcal{R}}$ is in the local neighborhood of $\mathcal{L}$. By local optimality, we have

$$(1 - \delta/2n) \sum_c \ell_c \leqslant \sum_c m_{\mathcal{R}}(c).$$

Hence,

$$-\frac{\delta}{2n}\text{cost}(\mathcal{L}) \leqslant \sum_c (m_{\mathcal{R}}(c) - \ell_c).$$

Observe that the number of regions is at most $k \leqslant n$. Thus, summing over all regions, we have

$$-(\delta/2)\text{cost}(\mathcal{L}) \leqslant \sum_{\mathcal{R}} \sum_{c} (m_{\mathcal{R}}(c) - \ell_c).$$

Inverting summations and applying Corollary 3.28 shows that $-(\delta/2)\,\text{cost}(\mathcal{L})$ is at most

$$\sum_{c \in \Lambda_1} (\text{Reassign}_{\mathcal{L} \mapsto \mathcal{C}*}(c) + g_c - 2\ell_c) + \sum_{c \in \Lambda_2} (\text{Reassign}_{\mathcal{C}* \mapsto \mathcal{L}}(c) - \ell_c) + \sum_{c \in \Lambda_3} (g_c - \ell_c). \quad (3.8)$$

Since each client in Λ_1 is bad, applying Lemma 3.24 shows $\sum_{c \in \Lambda_1} (\text{Reassign}_{\mathcal{L} \mapsto \mathcal{C}*}(c) + g_c - 2\ell_c)$ is at most

$$\sum_{c \in \Lambda_1} (g_c - (1 - \delta/\alpha)\ell_c) + (\delta/\alpha)\,(\text{cost}(\mathcal{L}) + \text{cost}(\mathcal{C}^*))$$

Since each client in Λ_2 is bad, Lemma 3.23 shows $\sum_{c \in \Lambda_2} (\text{Reassign}_{\mathcal{C}* \mapsto \mathcal{L}}(c) - \ell_c)$ is at most

$$\sum_{c \in \Lambda_2} ((1 + \delta/\alpha)g_c - \ell_c) + (\delta/\alpha)\,(\text{cost}(\mathcal{L}) + \text{cost}(\mathcal{C}^*))$$

Since $\Lambda_1, \Lambda_2, \Lambda_3$ is a partition of the clients, therefore, the sum (3.8) is bounded by

$$\sum_{c} ((1 + \delta/\alpha)g_c - (1 - \delta/\alpha)\ell_c) + (2\delta/\alpha)(\text{cost}(\mathcal{L}) + \text{cost}(\mathcal{C}^*))$$

which is $(1 + 3\delta/\alpha)\text{cost}(\mathcal{C}^*) - (1 - 3\delta/\alpha)\text{cost}(\mathcal{L})$ Since $-(\delta/2)\,\text{cost}(\mathcal{L})$ is at most the sum (3.8),

$$(1 - c_1\frac{2\varepsilon}{1-\varepsilon} - \varepsilon)\text{cost}(\mathcal{L}) \leqslant (1 + c_1\frac{2\varepsilon}{1-\varepsilon})\text{cost}(\mathcal{C}^*)$$
$$(1 - c_1\frac{2\varepsilon}{1-\varepsilon} - \varepsilon)\text{cost}(\mathcal{L}) \leqslant (1 + c_1\frac{2\varepsilon}{1-\varepsilon})(1+\varepsilon)\text{cost}(\mathcal{C}) + c_1\varepsilon\text{cost}(\mathcal{L})$$

because $\mathcal{C}_1 \subseteq \mathcal{C}^*$ implies $\text{cost}(\mathcal{C}^*) \leqslant \text{cost}(\mathcal{C}_1)$ and Theorem 3.13 implies $\text{cost}(\mathcal{C}_1) \leqslant (1 + \delta/\alpha)\text{cost}(\mathcal{C}) + (\delta/\alpha)\text{cost}(\mathcal{L})$. Thus there is a choice of the constant α for which $\text{cost}(\mathcal{L}) \leqslant (1 + \delta)\text{cost}(\mathcal{C})$.

3.5 Clustering in the Euclidean Setting

The proof is similar for $\mathbb{R}^d$. We explain how to modify the beginning of the proof of the graph case, the rest of the proof applies directly.

As before, we define a client c as *bad* if $\mathcal{L}(c)$ and $\mathcal{C}(c)$ do not belong to the same ε-isolated region but at least one is isolated; otherwise c is *good*.

Again we obtain a solution $\mathcal{C}_1$ from $\mathcal{C}$ by applying Theorem 3.13 to find a set S_0 of facilities to remove. Let $\mathcal{F} = \mathcal{L} \cup \mathcal{C}_1$. We now consider each isolated region $(\mathcal{L}_0, f_0)$, with $|\mathcal{L}_0| > 1/\varepsilon^{7d} - 1$, and proceed to an r-division of $\mathcal{L}_0 \cup \{f_0\}$ with $r = 1/\varepsilon^{7d}$. Moreover, for the remaining facilities $\mathcal{F}$ of that are not in any isolated region, we proceed to an r-division of those points with $r = 1/\varepsilon^{7d}$. We denote by $R_1, R_2 \dots$ the subset of all the regions defined by the above r-divisions. Let Z denote the set of boundary elements of all the r-divisions. Define $\mathcal{C}^* = \mathcal{C}_1 \cup Z$.

The point sets of regions are not disjoint since points of Z appear in various regions. Thus, we again define a ground set $\Omega = \{(v, R) \ : v \text{ a point of } \mathcal{F},\ R \text{ a region containing } v\}$, and, for each region R, we define $\widehat{R} = \{(v, R) \ : \ v \text{ a point of } R\}$. Now $\widehat{R_1}, \widehat{R_2}, \dots$ form a partition of Ω. To allow us to go from an element of Ω back to a point, if $x = (v, R)$ we define $\breve{x} = v$. Finally, define $\widehat{\mathcal{G}} = \{(v, R) \in \Omega : \ v \in \mathcal{C}^*\}$.

We now follow the rest of the proof of Theorem 1.2, starting from Lemma 3.19.

CHAPTER 4

Beyond Separators: Local Search for Clustering in Stable Instances

In this section we prove the following statements:

- If an instance is α-perturbation-resilient, for $\alpha > 3$, then Algorithm 2 computes the optimal solution (Theorem 1.8, Section 4.1).
- If an instance is β-distribution stable then Algorithm 2 is a PTAS (Theorem 1.5, Section 4.2).
- If an instance is δ-spectrally separated for $\delta > 3\sqrt{k}$, then Algorithm 4 is a PTAS (Theorem 1.10, Section 4.4).

Recall Algorithms 2 and 4.

Algorithm 2 Local search for finding k clusters.

```
1: Input: A metric space and associated cost function $\text{cost}(\cdot)$, an $n$-element set $C$ of
   points, error parameter $\varepsilon > 0$, number of clusters $k$
2: Parameter: A positive integer parameter $s$ for the neighborhood size
3: $S \leftarrow$ Arbitrary size-$k$ set of points
4: while $\exists$ $S'$ s.t. $|S'| \leqslant k$ and $|S \backslash S'| + |S' \backslash S| \leqslant s$ and $\text{cost}(S') \leqslant (1 - \varepsilon/n)\text{cost}(S)$
5:  do
6:      $S \leftarrow S'$
7: Output: $S$
```

Algorithm 4 Project and local search

```
1: Project points $A$ onto the best rank $k/\varepsilon$ subspace
2: Embed points into a random subspace of dimension $O(\varepsilon^{-2}\log n)$
3: Compute candidate centers (Corollary 4.17)
4: local search($\Theta(\varepsilon^{-4})$)
5: Output clustering
```

4.1 α-Perturbation-Resilient Instances

We recall the definition of α-perturbation-resilient instances as it appears in the introduction.

Definition 1.6. *Let $I = (A, F, cost, k)$ be an instance for the k-clustering problem. For $\alpha \geqslant 1$, I is* α-perturbation-resilient *if there exists a unique optimal clustering $\{C_1, \dots, C_k\}$ and for any instance $I' = (A, F, cost', k)$, such that*

$$\forall\, p, q \in A \cup F,\ cost(p, q) \leqslant cost'(p, q) \leqslant \alpha dist(p, q),$$

the unique optimal clustering is $\{C_1, \dots, C_k\}$.

Observe that $(A \cup F, \text{cost}')$ in Definition 1.6 needs not be a metric, even if $(A \cup F, \text{cost})$ was originally one. In the following, we assume that $\text{cost}(a, b) = \text{dist}(a, b)^p$ for some fixed p and some distance function dist defined over $A \cup F$. Consider a solution S_0 to the k-clustering problem with parameter p.

For ease of exposition, we give the proof of Theorem 1.8 for the k-median problem, when $p = 1$. Applying Lemmas 2.8 and 2.9 in the proof of Lemma 2.4 yields the result for general p with α growing exponentially with p (Theorem 1.7). The proof of Theorem 1.8 relies on Theorem 2.3.

Theorem 1.8. *Let $\alpha > 3$. For any instance of the k-median problem that is α-perturbation-resilient, any $2(\alpha - 3)^{-1}$-locally optimal solution is the optimal clustering $\{C_1^*, \dots, C_k^*\}$.*

Proof. Given an instance (A, F, cost, k), we define an instance $I' = (A, F, \text{cost}', k)$ as follows. For each client $a \in N_{\mathcal{C}}(\tilde{\mathcal{C}}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})$, let ℓ_i be the center of $\mathcal{L}$ that serves it in $\mathcal{L}$, for any point $p \neq \ell_i$, we define $\text{cost}'(a, p) = \alpha\text{cost}(a, p)$ and $\text{cost}'(a, \ell_i) = \text{cost}(a, \ell_i)$. For the other clients we set $\text{cost}' = \text{cost}$. Observe that by local optimality, the clustering induced by $\mathcal{L}$ is $\{C_1, \dots, C_k\}$ if and only if $\mathcal{L} = \mathcal{C}$. Therefore, the cost of $\mathcal{C}$ in instance I' is equal to

$$\alpha \sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_a + \sum_{a \notin N_{\mathcal{C}}(\tilde{\mathcal{C}}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_a.$$

On the other hand, the cost of $\mathcal{L}$ in I' is the same than in I, by Theorem 2.3

$$\sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} \ell_a \leqslant \sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} \ell_a \leqslant (3 + \frac{2(\alpha - 3)}{2}) \sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_a$$

and by definition

$$\sum_{a \notin N_{\mathcal{C}}(\tilde{\mathcal{C}}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} \ell_a \leqslant \sum_{a \notin N_{\mathcal{C}}(\tilde{\mathcal{C}}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_a.$$

Hence the cost of $\mathcal{L}$ in I' is at most

$$\alpha \sum_{a \in N_{\mathcal{C}}(\tilde{\mathcal{C}}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_a + \sum_{a \notin N_{\mathcal{C}}(\tilde{\mathcal{C}}) \cup N_{\mathcal{L}}(\tilde{\mathcal{L}})} g_a.$$

By definition of α-perturbation-resilience, we have that the clustering $\{C_1, \ldots, C_k\}$ is the unique optimal solution in I'. Therefore $\mathcal{L} = \mathcal{C}$ and the Theorem follows. □

We show that the analysis is tight:

Proposition 4.1. *There exists an infinite family of* 3*-perturbation-resilient instances such that for any constant* $\varepsilon > 0$*, there exists a* ε^{-1}*-locally optimal solution that has cost at least* 3*OPT.*

This relies on the example from [16] (see also Figure 3.1). It is straightforward to see that the instance they provide is 3-perturbation-resilient.

4.2 Distribution Stability

We work with the notion (β, δ)-distribution stability which generalizes β-distribution stability.

Definition 4.2 ((β, δ)-Distribution Stability)**.** *Let* $(A, F, cost, k)$ *be an instance of the* k*--clustering problem where* $A \cup F$ *are embedded into some metric space and let* $C = \{C_1, \ldots, C_k\}$ *denote a partition of* A *and* $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\} \subseteq F$ *be a set of centers. Further, let* $\beta > 0$ *and* $1/2 > \delta$*. Then* $(C, \mathcal{C})$ *is* (β, δ)-stable *if, for any* i*, there exists a set* $\Delta_i \subseteq C_i$ *such that* $|\Delta_i| \geqslant (1 - \delta)|C_i|$ *and for any* $a \in \Delta_i$*, for any* $j \neq i$*,*

$$cost(a, \mathcal{C}_j) \geqslant \beta \frac{OPT}{|C_j|},$$

where $cost(a, \mathcal{C}_j)$ *is the cost of assigning* a *to* $\mathcal{C}_j$*.*

We show the following theorem.

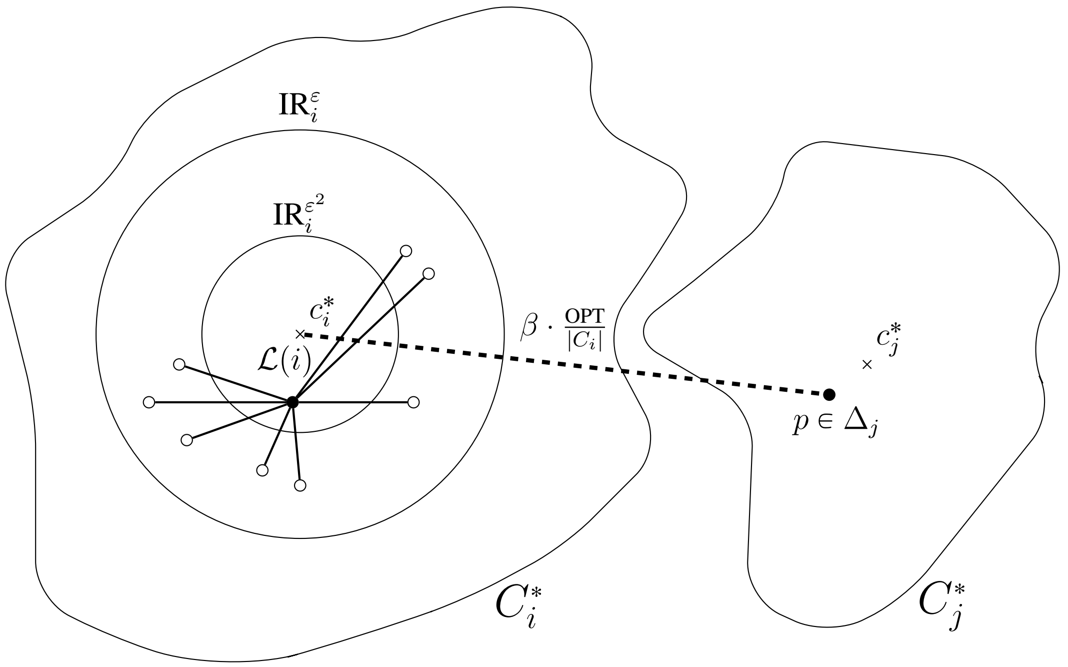

$\mathrm{IR}_i^{\varepsilon}$
$\mathrm{IR}_i^{\varepsilon^2}$
c_i^*
$\mathcal{L}(i)$
$\beta \cdot \frac{\mathrm{OPT}}{|C_i|}$
c_j^*
$p \in \Delta_j$
C_i^*
C_j^*

Figure 4.1: Example of a cluster $C_i^* \notin Z^*$. An important fraction of the points in $\mathrm{IR}_i^{\varepsilon^2}$ are served by $\mathcal{L}(i)$ and few points in $\bigcup_{j \neq i} \Delta_j$ are served by $\mathcal{L}(i)$.

Theorem 4.3. *There exists a constant* c *such that the following holds. Let* $\beta > 0, \delta < 1/2.$ *and* $\varepsilon < 1/2$*. For any* (β, δ)*-stable instance with respect to a clustering* C *and* $\varepsilon < 1/2$*, the cost of the solution output by local search (Algorithm 2) with parameter* $s = 2\varepsilon^{-3}\beta^{-1}$ *is at most* $(1 + c_1(\varepsilon + \delta))$*cost*$(C)$*.*

Note that taking $\delta = 0$, Theorem 1.5 is an immediate corollary of Theorem 4.3.

For ease of exposition, we give the proof of Theorem 4.3 for the case of k-median, namely when $p = 1$. Applying Lemmas 2.8 and 2.9 through the proof yields the result for general values of p.

Throughout this section we consider a set of centers $\mathcal{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_k\}$ whose induced clustering is $C = \{C_1, \dots, C_k\}$ and such that the instance is (β, δ)-stable with respect to this clustering. We denote by *clusters* the parts of a partition $C = \{C_1, \dots, C_k\}$. Let $\mathrm{cost}(C) = \sum_{i=1}^{k} \sum_{a \in C_i} \mathrm{cost}(a, \mathcal{C}_i)$. Moreover, for any cluster C_i, for any client $a \in C_i$, denote by g_a the cost of client a in solution C: $g_a = \mathrm{cost}(a, \mathcal{C}_i) = \mathrm{dist}(a, \mathcal{C}_i)$ since $p = 1$. Let $\mathcal{L}$ denote the output of the LocalSearch($\beta^{-1}\varepsilon^{-3}$) and ℓ_a the cost induced by client a in solution $\mathcal{L}$, namely $\ell_a = \min_{\ell \in \mathcal{L}} \mathrm{cost}(a, \ell)$. The following definition is adapted from [17].

Definition 4.4. *For any* ε_0*, we define the* inner ring *of cluster* i*,* $IR_i^{\varepsilon_0}$*, as the set of clients* a *such that cost*$(a, \mathcal{C}_i) \leqslant \varepsilon_0 \beta$*OPT*$/|C_i|$*.*

We say that cluster i is *cheap* if $\sum_{a \in C_i} g_a \leqslant \varepsilon^3 \beta \text{OPT}$, and *expensive* otherwise. We aim at proving the following structural lemma.

Lemma 4.5. *There exists a set of clusters $Z^* \subseteq C$ of size at most $(\varepsilon^{-3} + 160\varepsilon^{-1})\beta^{-1}$ such that for any cluster $C_i \in C \backslash Z^*$, we have the following properties*

1. *C_i is cheap.*
2. *At least a $(1-\varepsilon)$ fraction of IR$_i^{\varepsilon^2}$ is served by a unique center $\mathcal{L}(i)$ in solution $\mathcal{L}$.*
3. *The total number of clients $p \in \bigcup_{j \neq i} \Delta_j$, that are served by $\mathcal{L}(i)$ in $\mathcal{L}$ is at most $\varepsilon |\text{IR}_i^{\varepsilon^2}|$.*

See Fig 4.1 for a typical cluster of $C \backslash Z^*$. We start with the following lemma which generalizes Fact 4.1 in [17].

Lemma 4.6. *Let C_i be a cheap cluster. For any ε_0, we have $|\text{IR}_i^{\varepsilon_0}| > (1 - \varepsilon^3/\varepsilon_0)|C_i|$.*

Proof. Observe that each client that is not in $IR_i^{\varepsilon_0}$ is at distance at least $\varepsilon_0 \beta \text{OPT}/|C_i^*|$ from $\mathcal{C}_i$. Since i is cheap, the total cost of the clients in C_i^* is at most $\varepsilon^3 \beta \text{OPT}$ and in particular, the total cost of the clients that are not in $IR_i^{\varepsilon_0}$ does not exceed $\varepsilon^3 \beta \text{OPT}$. Therefore, the total number of such clients is at most $\varepsilon^3 |C_i^*|/\varepsilon_0$. □

We then prove that the inner rings of cheap clusters are disjoint.

Lemma 4.7. *Let $\varepsilon_0 < 1/3$. If $C_i \neq C_j$ are cheap clusters, then IR$_i^{\varepsilon_0} \cap$ IR$_j^{\varepsilon_0} = \varnothing$.*

Proof. Assume towards contradiction that this is not true. Consider a client $x \in \text{IR}_i^{\varepsilon_0} \cap \text{IR}_j^{\varepsilon_0}$. Without loss of generality assume $|C_i| \geqslant |C_j|$. By the triangular inequality we have $\text{cost}(\mathcal{C}_j, \mathcal{C}_i) \leqslant \text{cost}(\mathcal{C}_j, x) + \text{cost}(x, \mathcal{C}_i) \leqslant 2\varepsilon_0 \beta \text{OPT}/|C_j|$. Since $\delta < 1/2$, there exists a client $a \in \text{IR}_i^{\varepsilon_0} \cap \Delta_i$. Thus, we have $\text{cost}(a, \mathcal{C}_j) \leqslant 3\varepsilon_0 \beta \text{OPT}/|C_j|$. This is a contradiction to the assumption that the instance is (β, δ)-distribution stable with respect to $(C, \mathcal{C})$ and a being in Δ_i for any $\varepsilon_0 < 1/3$. □

The following observation follows directly from the definition of cheap clusters.

Lemma 4.8. *Let $Z_1 \subseteq C$ be the set of clusters of C that are not cheap. Then $|Z_1| \leqslant \varepsilon^{-3}\beta^{-1}$.*

For each cheap cluster C_i, let $\mathcal{L}(i)$ denote a center of $\mathcal{L}$ that belongs to $\text{IR}_i^{\varepsilon}$ if there is one (and remain undefined otherwise). By Lemma 4.7 those centers are all different.

Lemma 4.9. *Let $C \backslash Z_2$ denote the set of clusters C_i that are cheap, such that $\mathcal{L}(i)$ is defined, and such that at least $(1-\varepsilon)|\text{IR}_i^{\varepsilon^2}|$ clients of IR$_i^{\varepsilon^2}$ are served in $\mathcal{L}$ by $\mathcal{L}(i)$. Then $|Z_2| \leqslant (\varepsilon^{-3} + 120\varepsilon^{-1})\beta^{-1}$.*

Proof. We distinguish five types of clusters in C: expensive clusters (k_1), cheap clusters with $\mathcal{L}(i)$ undefined (k_2), cheap clusters with exactly one center of $\mathcal{L}$ in $\text{IR}_i^{\varepsilon} \backslash \text{IR}_i^{(1-\varepsilon)\varepsilon}$ (k_3), cheap clusters with exactly one center of $\mathcal{L}$ in $\text{IR}_i^{(1-\varepsilon)}$ (k_4), and cheap clusters with at least two centers of $\mathcal{L}$ in $\text{IR}_i^{\varepsilon}$ (k_5). Since $\mathcal{L}$ and C both have k clusters and the inner rings of cheap clusters are disjoint (Lemma 4.7), we have $k_5 \leqslant k_1 + k_2$. By Lemma 4.8, we have $k_1 \leqslant \varepsilon^{-3}\beta^{-1}$.

We now bound the number k_2 of cheap clusters such that $\mathcal{L}(i)$ is undefined. Consider a cheap cluster $C_i \subseteq C \backslash Z_1$ such that at least a $(1-\varepsilon)$ fraction of the clients of $\text{IR}_i^{\varepsilon^2}$ are served in $\mathcal{L}$ by some centers that are either in $\text{IR}_i^{\varepsilon} \backslash \text{IR}_i^{(1-\varepsilon)\varepsilon}$ or not in $\text{IR}_i^{\varepsilon}$. By the triangular inequality, the cost for any client c in $\text{IR}_i^{\varepsilon^2}$ is at least $((1-\varepsilon)\varepsilon - \varepsilon^2)\beta\text{OPT}/|C_i|$. Since $\varepsilon \leqslant 1/2$, it is at least $\varepsilon\beta\text{OPT}/(4|C_i|)$. Since at least $(1-\varepsilon)|\text{IR}_i^{\varepsilon^2}|$ clients of $\text{IR}_i^{\varepsilon^2}$ are served by centers that are not in $\text{IR}_i^{\varepsilon}$, the total cost in $\mathcal{L}$ induced by those clients is at least $(1-\varepsilon)|\text{IR}_i^{\varepsilon^2}|\varepsilon\beta\text{OPT}/(2|C_i|)$. By Lemma 4.6, substituting $|\text{IR}_i^{\varepsilon^2}|$ yields,

$$(1-\varepsilon)|\text{IR}_i^{\varepsilon^2}|\varepsilon\beta\frac{\text{OPT}}{2|C_i|} \geqslant (1-\varepsilon)(1-\varepsilon)|C_i|\varepsilon\beta\frac{\text{OPT}}{2|C_i|} \geqslant \varepsilon\beta\frac{\text{OPT}}{8}$$

since $\varepsilon \leqslant 1/2$. Now, observe that by Theorem 2.2, the cost of $\mathcal{L}$ is at most a 5 approximation to the cost of OPT in the worst case. Thus, $k_2 \leqslant 40(\varepsilon^{-1}\beta^{-1})$.

By a similar argument, we can bound the number of clusters k_4 such that a $(1-\varepsilon)$ fraction the clients in $\text{IR}_i^{\varepsilon^2}$ are served by a center not in $\text{IR}_i^{(1-\varepsilon)\varepsilon}$. We have $k_4 \leqslant 80(\varepsilon^{-1}\beta^{-1})$ since $\varepsilon < 1/2$. For the remaining clusters, we have that there is a unique center located in $\text{IR}_i^{(1-\varepsilon)\varepsilon}$ and that $\text{IR}_i^{\varepsilon} \backslash \text{IR}_i^{(1-\varepsilon)\varepsilon}$ does not contain any center. Additionally, at least $(1-\varepsilon)|\text{IR}_i^{\varepsilon^2}|$ clients in $\text{IR}_i^{\varepsilon^2}$ are served by a center in $\text{IR}_i^{\varepsilon}$ and so in $\text{IR}_i^{(1-\varepsilon)\varepsilon}$. Thus, by the triangular inequality we have that at least $(1-\varepsilon)|\text{IR}_i^{\varepsilon^2}|$ clients in $\text{IR}_i^{\varepsilon^2}$ are served by the unique center in $\text{IR}_i^{\varepsilon}$, namely $\mathcal{L}(i)$. It follows that $|Z_2| \leqslant (\varepsilon^{-3} + 120\varepsilon^{-1})\beta^{-1}$. □

We continue with the following lemma, whose proof relies on similar arguments.

Lemma 4.10. *There exists a set $Z_3 \subseteq C \backslash Z_2$ of size at most $(\varepsilon^{-3} + 40\varepsilon^{-1})\beta^{-1}$ such that for any cluster $C_j \in C \backslash Z_3$, the total number of clients $a \in \bigcup_{i \neq j} \Delta_i$, that are served by $\mathcal{L}(j)$ in $\mathcal{L}$ is at most $\varepsilon|\text{IR}_i^{\varepsilon^2}|$.*

Proof. Consider a cheap cluster $C_j \in C \backslash Z_2$ such that the total number of clients $a \in \Delta_i$, for $j \neq i$, that are served by $\mathcal{L}(j)$ in $\mathcal{L}$ is greater than $\varepsilon|\text{IR}_j^{\varepsilon^2}|$. By the triangular inequality and the definition of (β, δ)-stability, the total cost for each $a \in \Delta_i$, $j \neq i$ served by $\mathcal{L}(j)$ is at least $(1-\varepsilon)\beta\text{OPT}/|C_j|$. Since there are at least $\varepsilon|\text{IR}_j^{\varepsilon^2}|$ such clients, their total cost is at least $\varepsilon|\text{IR}_j^{\varepsilon^2}|(1-\varepsilon)\beta\text{OPT}/|C_j|$. By Lemma 4.6, this is at least

$$\varepsilon(1-\varepsilon)^2|C_j|(1-\varepsilon)\beta\frac{\text{OPT}}{|C_j|} \geqslant \varepsilon\beta\frac{\text{OPT}}{8},$$

since $\varepsilon \leqslant 1/2$. Recall that by Theorem 2.2, $\mathcal{L}$ is a 5-approximation and so there exist at most $40\varepsilon^{-1}\beta^{-1}$ such clusters. By Lemma 4.8, the total number of not cheap clusters is at most $\varepsilon^{-3}\beta^{-1}$ and so, there exists a set Z_3 of size at most $(\varepsilon^{-3} + 40\varepsilon^{-1})\beta^{-1}$ satisfying the lemma. □

Therefore, the proof of Lemma 4.5 follows from combining Lemmas 4.8,4.9 and 4.10.

We now turn to the cost analysis of the $\mathcal{L}$. Let $C(Z^*) = \bigcup_{C_i \in Z^*} C_i$. For any cluster $C_i \in C \backslash Z^*$, let $\mathcal{L}(i)$ be the unique center of $\mathcal{L}$ that serves a set of clients $A_i \subseteq \text{IR}_i^{\varepsilon^2}$ such that $|A_i| \geqslant (1-\varepsilon)|\text{IR}_i^{\varepsilon^2}|$. Let $\hat{\mathcal{L}} = \bigcup_{C_i \in C \backslash Z^*} \mathcal{L}(i)$ and $\bar{\mathcal{L}} = \mathcal{L} \backslash \hat{\mathcal{L}}$. Define $\bar{A}$ and $\hat{A}$ to be the set of clients that are served in solution $\mathcal{L}$ by centers of $\bar{\mathcal{L}}$ and $\hat{\mathcal{L}}$ respectively. Finally, let $A(\mathcal{L}(i))$ be the set of clients that are served by $\mathcal{L}(i)$ in solution $\mathcal{L}$. Observe that the $A(\mathcal{L}(i))$ partition $\hat{A}$.

Lemma 4.11. *We have*

$$-\varepsilon \textit{cost}(\mathcal{L})/n + \sum_{a \in \bar{A} \cup C(Z^*)} \ell_a \leqslant \sum_{a \in \bar{A} \cup C(Z^*)} g_a + \varepsilon(\textit{cost}(\mathcal{L}) + \textit{cost}(C))/(1-\varepsilon)^2.$$

Proof. Consider the following mixed solution $\mathcal{M} = \hat{\mathcal{L}} \cup \{c_i^* \in Z^* \mid C_i^* \in Z^*\}$. We start by bounding the cost of $\mathcal{M}$. For any client $a \notin \bar{A} \cup C(Z^*)$, the center that serves it in $\mathcal{L}$ belongs to $\mathcal{M}$. Thus its cost is at most ℓ_a. Now, for any client $a \in \bar{C}(Z^*)$, the center that serves it in Z^* is in $\mathcal{M}$, so its cost is at most g_a.

Finally, we evaluate the cost of the clients in $\bar{A} \backslash C(Z^*)$. Consider such a client a and let C_i be the cluster it belongs to in solution C. By definition of $\bar{A}$ we have that $C_i \notin Z^*$. Therefore, $\mathcal{L}(i)$ is defined and so we have $\mathcal{L}(i) \in \hat{\mathcal{L}} \subseteq \mathcal{M}$. Hence, the cost of a in $\mathcal{M}$ is at most $\text{cost}(a, \mathcal{L}(i))$. Observe that by the triangular inequality $\text{dist}(a, \mathcal{L}(i)) \leqslant \text{dist}(a, \mathcal{C}_i) + \text{dist}(\mathcal{C}_i, \mathcal{L}(i)) = g_a + \text{dist}(\mathcal{C}_i, \mathcal{L}(i))$.

Now consider a client $a' \in C_i \cap A_i$. By the triangular inequality, we have that

$$\text{cost}(\mathcal{C}_i, \mathcal{L}(i)) \leqslant \text{cost}(\mathcal{C}_i, a') + \text{cost}(a', \mathcal{L}(i)) = g_{a'} + \ell_{a'}.$$

Hence,

$$\text{cost}(\mathcal{C}_i, \mathcal{L}(i)) \leqslant \frac{1}{|C_i \cap A_i|} \sum_{a' \in C_i \cap A_i} (g_{a'} + \ell_{a'}).$$

It follows that assigning the clients of $\bar{A} \cap C_i$ to $\mathcal{L}(i)$ induces a cost of at most

$$\sum_{a \in \bar{A} \cap C_i} g_a + \frac{|C_i \cap \bar{A}|}{|C_i \cap A_i|} \sum_{a' \in C_i \cap A_i} (g_{a'} + \ell_{a'}).$$

By Lemma 4.9 and the definition of Z^*, we have that $|C_i \cap \bar{A}|/|C_i \cap A_i| \leqslant \varepsilon/(1-\varepsilon)^2$. Summing over all clusters $C_i \notin Z^*$, we obtain that the cost in $\mathcal{M}$ for the clients in $\bar{A} \cap C_i$ is at most

$$\sum_{c \in \bar{A} \backslash C(Z^*)} g_a + \frac{\varepsilon}{(1-\varepsilon)^2}(\text{cost}(C) + \text{cost}(\mathcal{L})).$$

By Lemmas 4.9,4.10, we have that $|\mathcal{M}\backslash\mathcal{L}|+|\mathcal{L}\backslash\mathcal{M}| \leqslant 3(\varepsilon^{-3}+40\varepsilon^{-1})\beta^{-1}$. Thus, by local optimality $(1-\varepsilon/n)\text{cost}(\mathcal{L}) \leqslant \text{cost}(\mathcal{M})$. Therefore, combining the above observations, we have

$$\begin{aligned}
(1-\frac{\varepsilon}{n})\text{cost}(\mathcal{L}) &\leqslant \sum_{a\notin\bar{A}\cup C(Z^*)} \ell_a + \sum_{a\in\bar{A}\cup C(Z^*)} g_a + \frac{\varepsilon}{(1-\varepsilon)^2}(\text{cost}(C)+\text{cost}(\mathcal{L}))\\
-\frac{\varepsilon}{n}\text{cost}(\mathcal{L}) + \sum_{a\notin\bar{A}\cup C(Z^*)} \ell_a + \sum_{a\in\bar{A}\cup C(Z^*)} \ell_a &\leqslant \sum_{a\notin\bar{A}\cup C(Z^*)} \ell_a\\
&\quad + \sum_{a\in\bar{A}\cup C(Z^*)} g_a + \frac{\varepsilon}{(1-\varepsilon)^2}(\text{cost}(C)+\text{cost}(\mathcal{L})).\\
-\frac{\varepsilon}{n}\text{cost}(\mathcal{L}) + \sum_{a\in\bar{A}\cup C(Z^*)} \ell_a &\leqslant \sum_{a\in\bar{A}\cup C(Z^*)} g_a + \frac{\varepsilon}{(1-\varepsilon)^2}(\text{cost}(\mathcal{L})+\text{cost}(C)).
\end{aligned}$$

□

We now turn to evaluate the cost for the clients that are not in $\bar{A} \cup C(Z^*)$, namely the clients in $\hat{A}\backslash C(Z^*)$. For any cluster C_i, for any $a \in C_i\backslash\Delta_i$ define Reassign(a) to be the distance from a to the center in $\mathcal{L}$ that is the closest to $\mathcal{C}_i$. Before going deeper in the analysis, we need the following lemma.

Lemma 4.12. *For any $\delta < 1/2$, we have for any C_i,*

$$\sum_{a\in C_i\backslash\Delta_i} \textit{Reassign}(a) \leqslant \sum_{a\in C_i\backslash\Delta_i} g_a + \frac{\delta}{(1-\delta)}\sum_{a\in C_i}(\ell_a+g_a).$$

Proof. Consider a client $a \in C_i\backslash\Delta_i$. Let ℓ' be the center that serves at least one client of Δ_i that is the closest to $\mathcal{C}_i$. Since $\delta < 1$, ℓ' is well defined. By the triangular inequality we have that Reassign$(a) \leqslant \text{cost}(a,\ell') \leqslant \text{cost}(a,\mathcal{C}_i) + \text{cost}(\mathcal{C}_i,\ell') = g_a + \text{cost}(\mathcal{C}_i,\ell')$. Then,

$$\sum_{a\in C_i\backslash\Delta_i} \text{Reassign}(a) \leqslant \sum_{a\in C_i\backslash\Delta_i} g_a + |C_i\backslash\Delta_i| \cdot \text{cost}(\mathcal{C}_i,\ell').$$

Now, since ℓ' is the center that serves at least one client of Δ_i that is the closest to $\mathcal{C}_i$ we have that for any $a \in \Delta_i$, by the triangular inequality $\text{cost}(\mathcal{C}_i,\ell') \leqslant \text{cost}(\mathcal{C}_i,a) + \text{cost}(a,\ell') \leqslant g_a + \ell_a$. Therefore,

$$\text{cost}(\mathcal{C}_i,\ell') \leqslant \frac{1}{|\Delta_i|}\sum_{a\in\Delta_i}(g_a+\ell_a).$$

Combining, we obtain

$$\begin{aligned}
\sum_{a\in C_i\backslash\Delta_i} \text{Reassign}(a) &\leqslant \sum_{a\in C_i\backslash\Delta_i} g_a + \frac{|C_i\backslash\Delta_i|}{|\Delta_i|}\sum_{a\in\Delta_i}(g_a+\ell_a)\\
&= \sum_{a\in C_i\backslash\Delta_i} g_a + \frac{\delta}{(1-\delta)}\sum_{a\in\Delta_i}(g_a+\ell_a),
\end{aligned}$$

by definition of (β, δ)-stability. □

We now partition the clients of cluster $C_i \in C\backslash Z^*$. For any i, let $\bar{\Delta}_i$ be the set of clients of C_i that are served in solution $\mathcal{L}$ by a center $\mathcal{L}(j)$ for some $j \neq i$ and $C_j \in C\backslash Z^*$. Moreover, let $\tilde{\Delta}_i = (A(\mathcal{L}(i)) \cap (\bigcup_{j \neq i} \bar{\Delta}_j)) \backslash C(Z^*)$. Finally, define $\tilde{C}_i = C_i \backslash (\bar{A} \cup \bigcup_{j \neq i} \tilde{\Delta}_j)$.

Lemma 4.13. *Let C_i be a cluster in $C\backslash Z^*$. Define the solution $\mathcal{M}^i = \mathcal{L}\backslash\{\mathcal{L}(i)\} \cup \{\mathcal{C}_i\}$ and denote by m_c^i the cost of client c in solution $\mathcal{M}^i$. Then*

$$\sum_{a \in A} m_a^i \leqslant \sum_{a \notin A(\mathcal{L}(i)) \cup \tilde{C}_i} \ell_a + \sum_{a \in \tilde{C}_i} g_a + \sum_{a \in \tilde{\Delta}_i} \textit{Reassign}(a) + \sum_{\substack{a \in A(\mathcal{L}(i)) \backslash \\ (\tilde{C}_i \cup \tilde{\Delta}_i)}} \ell_a + \frac{\varepsilon}{(1-\varepsilon)} (\sum_{a \in \tilde{C}_i} g_a + \ell_a).$$

Proof. For any client $a \notin A(\mathcal{L}(i)) \cup \tilde{C}_i$, the center that serves it in $\mathcal{L}$ belongs to $\mathcal{M}^i$. Thus its cost is at most ℓ_a. Moreover, observe that any client $a \in \tilde{C}_i$ can now be served by $\mathcal{C}_i$, and so its cost is at most g_a. For each client $a \in \tilde{\Delta}_i$, since all the centers of $\mathcal{L}$ except for $\mathcal{L}(i)$ are in $\mathcal{M}^i$, we bound its cost by Reassign(a).

Now, we bound the cost of a client $a \in A(\mathcal{L}(i)) \backslash (\tilde{C}_i \cup \tilde{\Delta}_i)$. The closest center in $\mathcal{M}^i$ for a client $a \in A(\mathcal{L}(i)) \backslash (\tilde{C}_i \cup \tilde{\Delta}_i)$ is not farther than $\mathcal{C}_i$. By the triangular inequality we have that the cost of such a client a is at most cost$(a, \mathcal{C}_i) \leqslant$ cost$(a, \mathcal{L}(i))$ + cost$(\mathcal{L}(i), \mathcal{C}_i) = \ell_a$ + cost$(\mathcal{L}(i), \mathcal{C}_i)$, and so

$$\sum_{\substack{a \in A(\mathcal{L}(i)) \backslash \\ (\tilde{C}_i \cup \tilde{\Delta}_i)}} m_a^i \leqslant |A(\mathcal{L}(i)) \backslash (\tilde{C}_i \cup \tilde{\Delta}_i)| \text{cost}(\mathcal{L}(i), \mathcal{C}_i) + \sum_{\substack{a \in A(\mathcal{L}(i)) \backslash \\ (\tilde{C}_i \cup \tilde{\Delta}_i)}} \ell_a. \tag{4.1}$$

Now, observe that, for any client $a \in |A(\mathcal{L}(i)) \cap \tilde{C}_i|$, by the triangular inequality we have that cost$(\mathcal{L}(i), \mathcal{C}_i) \leqslant$ cost$(a, \mathcal{L}(i))$ + cost$(a, \mathcal{C}_i) = \ell_a + g_a$. Therefore,

$$\text{cost}(\mathcal{L}(i), \mathcal{C}_i) \leqslant \frac{1}{|A(\mathcal{L}(i)) \cap \tilde{C}_i|} \sum_{a \in A(\mathcal{L}(i)) \cap \tilde{C}_i} (\ell_a + g_a). \tag{4.2}$$

Combining Equations 4.1 and 4.2, we have that

$$\sum_{\substack{a \in A(\mathcal{L}(i)) \backslash \\ (\tilde{C}_i \cup \tilde{\Delta}_i)}} m_c^i \leqslant \sum_{\substack{a \in A(\mathcal{L}(i)) \backslash \\ (\tilde{C}_i \cup \tilde{\Delta}_i)}} \ell_a + \frac{|A(\mathcal{L}(i)) \backslash \tilde{C}_i|}{|A(\mathcal{L}(i)) \cap \tilde{C}_i|} \sum_{a \in A(\mathcal{L}(i)) \cap \tilde{C}_i} (\ell_a + g_a). \tag{4.3}$$

We now remark that since $\tilde{C}_i$ is not in Z^*, we have by Lemmas 4.9 and 4.10, $|A(\mathcal{L}(i)) \backslash \tilde{C}_i| \leqslant \varepsilon |IR_i^{\varepsilon^2}|$ and $(1-\varepsilon)|IR_i^{\varepsilon^2}| \leqslant |A(\mathcal{L}(i)) \cap \tilde{C}_i|$. Thus, combining with Equation 4.3 yields the lemma. □

We can thus prove the following lemma, which concludes the proof.

Lemma 4.14. *There exists a constant η such that*

$$-\varepsilon \cdot \mathit{cost}(\mathcal{L}) + \sum_{a \in \hat{A} \backslash C(Z^*)} \ell_a \leqslant \sum_{a \in \hat{A} \backslash C(Z^*)} g_a + (\eta(\varepsilon + \frac{\delta}{1-\delta}))(\mathit{cost}(\mathcal{L}) + \mathit{cost}(C)).$$

Proof. We consider a cluster C_i in $C \backslash Z^*$. and the solution $\mathcal{M}^i = \mathcal{L} \backslash \{\mathcal{L}(i)\} \cup \{\mathcal{C}_i\}$. Observe that $\mathcal{M}^i$ and $\mathcal{L}$ only differs by $\mathcal{L}(i)$ and $\mathcal{C}_i$. Therefore, by local optimality and Lemma 4.13, we have that $(1 - \frac{\varepsilon}{n})\text{cost}(\mathcal{L}_i) \leqslant \text{cost}(\mathcal{M}^i)$. Then,

$$(1 - \frac{\varepsilon}{n})\text{cost}(\mathcal{L}_i) \leqslant \sum_{a \notin A(\mathcal{L}(i)) \cup \tilde{C}_i} \ell_a + \sum_{a \in \tilde{C}_i} g_a + \sum_{\substack{a \in A(\mathcal{L}(i)) \backslash \\ (\tilde{C}_i \cup \tilde{\Delta}_i)}} \ell_a + \sum_{a \in \tilde{\Delta}_i} \text{Reassign}(a) + \frac{\varepsilon}{(1-\varepsilon)} \sum_{a \in C_i} (g_a + \ell_a)$$

and so, simplifying

$$\frac{\varepsilon}{n}\text{cost}(\mathcal{L}_i) + \sum_{a \in \tilde{C}_i} \ell_a + \sum_{a \in \tilde{\Delta}_i} \ell_a \leqslant \sum_{a \in \tilde{C}_i} g_a + \sum_{a \in \tilde{\Delta}_i} \text{Reassign}(a) + \frac{\varepsilon}{(1-\varepsilon)} \sum_{a \in C_i} (g_a + \ell_a)$$

We now apply this analysis to each cluster $\tilde{C}_i \in C \backslash Z^*$. Summing over all clusters $\tilde{C}_i \in C \backslash Z^*$, we obtain,

$$-k\frac{\varepsilon}{n}\text{cost}(\mathcal{L}) + \sum_{i=1}^{|C \backslash Z^*|} \left(\sum_{a \in \tilde{C}_i} \ell_a + \sum_{a \in \tilde{\Delta}_i} \ell_a \right) \leqslant \sum_{i=1}^{|C \backslash Z^*|} \left(\sum_{a \in \tilde{C}_i} g_a + \sum_{a \in \tilde{\Delta}_i} \text{Reassign}(c) \right) + \frac{\varepsilon}{(1-\varepsilon)}(\text{cost}(\mathcal{L}) + \text{cost}(C))$$

By Lemma 4.12 and the definition of $\tilde{C}_i$,

$$-k\frac{\varepsilon}{n}\text{cost}(\mathcal{L}) + \sum_{i=1}^{|C \backslash Z^*|} \sum_{a \in C_i \cap \hat{A}} \ell_a \leqslant \sum_{i=1}^{|C \backslash Z^*|} \sum_{a \in C_i \cap \hat{A}} g_a + (\frac{\varepsilon}{(1-\varepsilon)} + \frac{\delta}{(1-\delta)})(\text{cost}(\mathcal{L}) + \text{cost}(C)).$$

Therefore, $-\varepsilon\text{cost}(\mathcal{L}) + \sum_{c \in \hat{A} \backslash C(Z^*)} \ell_a \leqslant \sum_{a \in \hat{A} \backslash C(Z^*)} g_a + (\eta(\varepsilon + \frac{\delta}{1-\delta}))(\text{cost}(\mathcal{L}) + \text{cost}(C))$, for some constant η and any $\delta < 1/2$. □

The proof of Theorem 4.3 follows from observing that $\bar{A} \cup \hat{A} = A$ and $\bar{A} \cap \hat{A} = \varnothing$ and summing the equations from Lemmas 4.11 and 4.14.

4.3 Euclidean Distribution Stability

In this section we show how to reduce the Euclidean problem to the discrete metric one. Our analysis is focused on the k-means problem, however we note that the discretization works for all values of cost $= \text{dist}^p$, where the dependency on p grows exponentially. For constant p, we obtain polynomial sized candidate solution sets in polynomial time. For k-means itself, we could alternatively combine Matousek's approximate centroid set [138] with the Johnson Lindenstrauss lemma and avoid the following construction; however this would only work for optimal distribution stable clusterings and the proof Theorem 1.10 requires it to hold for non-optimal clusterings as well.

First, we describe a discretization procedure. It will be important to us that the candidate solution preserves (1) the cost of any given set of centers and (2) distribution stability.

For a set of points P, a set of points $\mathcal{N}_\varepsilon$ is an *ε-net* of P if for every point $x \in P$ there exists some point $y \in \mathcal{N}_\varepsilon$ with $||x-y|| \leqslant \varepsilon$. It is well known that for unit Euclidean ball of dimension d, there exists an ε-net of cardinality $(1+2/\varepsilon)^d$, see for instance Pisier [154]. We will use such ε-nets in our discretization.

Lemma 4.15. *Let A be a set of n points in d-dimensional Euclidean space and let $\beta, \varepsilon > 0$ with $\min(\beta, \varepsilon, \sqrt{7-4\sqrt{2}}-1) > 2\eta > 0$ be constants. Suppose there exists a clustering $C = \{C_1, \ldots, C_k\}$ with centers $S = \{c_1, \ldots c_k\}$ such that*

1. *cost$(C,S) = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - c_i||^2$ is a constant approximation to the optimum clustering and*
2. *C is β-distribution stable.*

Then there exists a discretization D of the solution space such that there exists a subset $S' = \{c'_1, \ldots c'_k\} \subset D$ of size k with

1. *$\sum_{i=1}^{k} \sum_{x \in C_i} ||x - c'_i||^2 \leqslant (1+\varepsilon) \cdot$ cost(C,S) and*
2. *C with centers S' is $\beta/2$-distribution stable.*

The discretization consists of $O(n \cdot \log n \cdot \eta^{d+2})$ many points.

Proof. Let OPT being the cost of an optimal k-means clustering. Define an exponential sequence to the base of $(1+\eta)$ starting at $(\eta \cdot \frac{\text{OPT}}{n})$ and ending at $(n \cdot \text{OPT})$. It is easy to see that the sequence contains $t = \log_{1+\eta}(n^2/\eta) \in O(\eta^{-1}(\log n + \log(1/\eta)))$ many elements. For each point $p \in A$, define $B(p, \ell_i)$ as the d-dimensional ball centered at p with radius $\sqrt{(1+\eta)^i \cdot \eta \cdot \frac{\text{OPT}}{n}}$. We cover the ball $B(p, \ell_i)$ with an $\eta/8 \cdot \ell_i$ net $\mathcal{N}_{\varepsilon/8}(p, \ell_i)$. As the set of candidate centers, we let $D = \cup_{p \in A} \cup_{i=0}^{t} \mathcal{N}_{\eta/8}(p, \ell_i)$. Clearly, $|D| \in O(n \cdot \log n \cdot (1 + 16/\eta)^{d+2})$.

Now for each $c_i \in S$, set $c_i' = \underset{q \in D}{\text{argmin}}\, ||q - c_i||$. We will show that $S' = \{c_1', \dots c_k'\}$ satisfies the two conditions of the lemma.

For (1), we first consider the points p with $||p - c_i|| \leqslant \sqrt{\varepsilon \cdot \frac{\text{OPT}}{n}}$. Then there exists a c_i' such that $||p - c_i'||^2 \leqslant \eta^2/64 \cdot \varepsilon\frac{\text{OPT}}{n}$ and summing up over all such points, we have a total contribution to the objective value of at most $\eta^2 \cdot \varepsilon/64 \cdot \text{OPT} \leqslant \eta^3/64 \cdot \text{OPT}$.

Now consider the remaining points. Since the cost(C, S) is a constant approximation, the center c_i of each point p satisfies $\sqrt{(1+\eta)^i \cdot \eta \cdot \frac{\text{OPT}}{n}} \leqslant ||c_i - p|| \leqslant \sqrt{(1+\eta)^{i+1} \cdot \eta \cdot \frac{\text{OPT}}{n}}$ for some $i \in \{0, \dots t\}$. Then there exists some point $q \in N_{\eta/8}(p, \ell_{i+1})$ with $||q - c_i|| \leqslant \eta/\cdot\sqrt{(1+\eta)^{i+1} \cdot \eta \cdot \frac{\text{OPT}}{n}} \leqslant \eta/8 \cdot \sqrt{1+\eta}||p - c_i|| \leqslant \eta/4||p - c_i||$. We then have $||p - c_i'||^2 \leqslant (1 + \eta/4)^2||p - c_i||^2$. Summing up over both cases, we have a total cost of at most $\eta^3/64 \cdot \text{OPT} + (1+\eta/4)^2 \cdot \text{cost}(C, S') \leqslant (1+\eta) \cdot \text{cost}(C, S') \leqslant (1+\varepsilon) \cdot \text{cost}(C, S')$.

To show (2), let us consider some point $p \notin C_j$ with $||p - c_j||^2 > \beta \cdot \frac{\text{OPT}}{|C_j|}$. Since $\beta \cdot \frac{\text{OPT}}{|C_j|} \geqslant 2\eta \cdot \frac{\text{OPT}}{n}$, there exists a point q and an $i \in \{0, \dots t\}$ such that $\sqrt{\frac{\beta}{1+\eta} \cdot \frac{\text{OPT}}{n}} \leqslant ||c_i - q|| \leqslant \sqrt{\beta \cdot (1+\eta) \cdot \varepsilon \cdot \frac{\text{OPT}}{n}}$. Then $||c_j' - c_j|| \leqslant \eta \cdot (1+\eta)\sqrt{\beta \cdot (1+\eta) \cdot \frac{\text{OPT}}{n}}$. Similarly to above, the point c_j' satisfies $||p - c_j'||^2 \geqslant (||p - c_j|| - ||c_j - c_j'||)^2 \geqslant (\sqrt{\beta \cdot \frac{\text{OPT}}{|C_j|}} - \sqrt{\beta \cdot \eta(1+\eta) \cdot \frac{\text{OPT}}{n}})^2 \geqslant (1 - \eta(1+\eta))\beta \cdot \frac{\text{OPT}}{|C_j|} > \beta \cdot \frac{\text{OPT}}{|C_j|}$ where the last inequality holds for any $\eta < \frac{1}{2} \cdot (\sqrt{7 - 4\sqrt{2}} - 1)$. □

To reduce the dependency on the dimension, we combine this statement with the seminal theorem originally due to Johnson and Lindenstrauss [110].

Lemma 4.16 (Johnson-Lindenstrauss lemma)**.** *For any set of n points N in d-dimensional Euclidean space and any $0 < \varepsilon < 1/2$, there exists a distribution $\mathcal{F}$ over linear maps $f : \ell_2^d \to \ell_2^m$ with $m \in O(\varepsilon^{-2}\log n)$ such that*

$$\mathbb{P}_{f \sim \mathcal{F}}[\forall x, y \in N,\ (1-\varepsilon)||x - y|| \leqslant ||f(x) - f(y)|| \leqslant (1+\varepsilon)||x - y||] \geqslant \frac{2}{3}.$$

It is easy to see that Johnson-Lindenstrauss type embeddings preserve the Euclidean k-means cost of any clustering, as the cost of any clustering can be written in terms of pairwise distances (see also Fact 4.18 in Section 4.4). Since the distribution over linear maps $\mathcal{F}$ can be chosen obliviously with respect to the points, this extends to distribution stability of a set of k candidate centers as well.

Combining Lemmas 4.16 and 4.15 gives us the following corollary.

Corollary 4.17. *Let A be a set of points in d-dimensional Euclidean space with a clustering $C = \{C_1, \dots C_k\}$ and centers $S = \{c_1, \dots c_k\}$ such that C is β-perturbation stable. Then there exists a $(A, F, ||\cdot||^2, k)$-clustering instance with clients A, $n^{poly(\varepsilon^{-1})}$ centers F and a* *subset $S' \subset F \cup A$ of k centers such that C and S' is $O(\beta)$ stable and the cost of clustering A with S' is at most $(1+\varepsilon)$ times the cost of clustering A with S.*

Remark. This procedure can be adapted to work for general powers of cost functions. For Lemma 4.15, we simply rescale η. The Johnson-Lindenstrauss lemma can also be applied in these settings, at a slightly worse target dimension of $O((p+1)^2 \log((p+1)/\varepsilon)\varepsilon^{-3} \log n)$, see Kerber and Raghvendra [115].

4.4 Spectral Separability

In this Section we will study the spectral separability condition for the Euclidean k-means problem. Our main result will be a proof of Theorem 1.10. The algorithm we consider is as follows (Algorithm 4).

Algorithm 4 Project and local search

```
1: Project points $A$ onto the best rank $k/\varepsilon$ subspace
2: Embed points into a random subspace of dimension $O(\varepsilon^{-2} \log n)$
3: Compute candidate centers (Corollary 4.17)
4: local search($\Theta(\varepsilon^{-4})$)
5: Output clustering
```

We first recall the basic notions and definitions for Euclidean k-means. In this section, let $A \in \mathbb{R}^{n\times d}$ be a matrix representing set of points in d-dimensional Euclidean space, where the row A_i contains the coordinates of the ith point. The singular value decomposition is defined as $A = U\Sigma V^T$, where $U \in \mathbb{R}^{n\times d}$ and $V \in \mathbb{R}^{d\times d}$ are orthogonal and $\Sigma \in \mathbb{R}^{d\times d}$ is a diagonal matrix containing the singular values where per convention the singular values are given in descending order, i.e. $\Sigma_{1,1} = \sigma_1 \geqslant \Sigma_{2,2} = \sigma_2 \geqslant \ldots \Sigma_{d,d} = \sigma_d$. Denote the Euclidean norm of a d-dimensional vector x by $||x|| = \sqrt{\sum_{i=1}^d x_i^2}$. The spectral norm and Frobenius norm are defined as $||A||_2 = \sigma_1$ and $||A||_F = \sqrt{\sum_{i=1}^d \sigma_i^2}$, respectively.

The best rank k approximation $\min\limits_{\text{rank}(X)=k} ||A - X||_F$ is given via $A_k = U_k\Sigma V^T = U\Sigma_k V^T = U\Sigma V_k^T$, where U_k, Σ_k and V_k^T consist of the first k columns of U, Σ and V^T, respectively, and are zero otherwise. The best rank k approximation also minimizes the spectral norm, that is $||A - A_k||_2 = \sigma_{k+1}$ is minimal among all matrices of rank k. The following fact is well known throughout k-means literature and will be used frequently throughout this section.

Fact 4.18. *Let A be a set of points in Euclidean space and denote by $c(A) = \frac{1}{|A|}\sum_{x\in A} x$ the centroid of A. Then the 1-means cost of any candidate center c can be decomposed via*

$$\sum_{x\in A} ||x - c||^2 = \sum_{x\in A} ||x - c(A)||^2 + |A| \cdot ||c(A) - c||^2$$

and

$$\sum_{x\in A} ||x - c(A)||^2 = \frac{1}{2\cdot |A|}\sum_{x\in A}\sum_{y\in A} ||x-y||^2.$$

Note that the centroid is the optimal 1-means center of A. For a clustering $C = \{C_1, \dots C_k\}$ of A with centers $S = \{c_1, \dots c_k\}$, the cost is then $\sum_{i=1}^{k}\sum_{p\in C_i} ||p - c_i||^2$. Further, if $c_i = \frac{1}{|C_i|}\sum_{p\in C_i} p$, we can rewrite the objective function in matrix form by associating the ith point with the ith row of some matrix A and using the cluster matrix $X \in \mathbb{R}^{n\times k}$ with $X_{i,j} = \begin{cases}\frac{1}{\sqrt{|C_j^*|}} & \text{if } A_i \in C_j^* \\ 0 & \text{else}\end{cases}$ to denote membership. Note that $X^TX = I$, i.e. X is an orthogonal projection and that $||A - XX^TA||_F^2$ is the cost of the optimal k-means clustering. k-means is therefore a constrained rank k-approximation problem.

We first restate the separation condition.

Definition 4.19 (Spectral Separation)**.** *Let A be a set of points and let $\{C_1, \dots C_k\}$ be a clustering of A with centers $\{c_1, \dots c_k\}$. Denote by C an $n \times d$ matrix such that $C_i = \underset{j\in\{1,\dots,k\}}{\text{argmin}}\, ||A_i - c_j||^2$. Then $\{C_1, \dots C_k\}$ is γ* spectrally separated, *if for any pair of centers c_i and c_j the following condition holds:*

$$||c_i - c_j|| \geqslant \gamma \cdot \left(\frac{1}{\sqrt{|C_i|}} + \frac{1}{\sqrt{|C_j|}}\right) ||A - C||_2.$$

The following crucial lemma relates spectral separation and distribution stability.

Lemma 4.20. *For a point set A, let $C = \{C_1, \dots, C_k\}$ be an optimal clustering with centers $S = \{c_1, \dots, c_k\}$ associated clustering matrix X that is at least $\gamma \cdot \sqrt{k}$ spectrally separated, where $\gamma > 3$. For $\varepsilon > 0$, let A_m be the best rank $m = k/\varepsilon$ approximation of A. Then there exists a clustering $K = \{C'_1, \dots C'_2\}$ and a set of centers S_k, such that*

1. *the cost of clustering A_m with centers S_k via the assignment of K is less than $||A_m - XX^TA_m||_F^2$ and*
2. *(K, S_k) is $\Omega((\gamma-3)^2 \cdot \varepsilon)$-distribution stable.*

We note that this lemma would also allow us to use the PTAS of Awasthi et al. [17]. Before giving the proof, we outline how Lemma 4.20 helps us prove Theorem 1.10. We first notice that if the rank of A is of order k, then elementary bounds on matrix norm show that spectral separability implies distribution stability. We aim to combine this observation with the following theorem due to Cohen et al. [59]. Informally, it states that for every rank k approximation, (an in particular for every constrained rank k approximation such as k-means clustering), projecting to the best rank k/ε subspace is cost-preserving.

Theorem 4.21 (Theorem 7 of [59])**.** *For any $A \in \mathbb{R}^{n\times d}$, let A' be the rank $\lceil k/\varepsilon \rceil$-approximation of A. Then there exists some constant c such that for any rank k orthogonal projection P,*

$$||A - PA||_F^2 \leqslant ||A' - PA'||_F^2 + \sum_{i=\lceil \frac{k}{\varepsilon}\rceil+1}^{\min(n,d)} \sigma_i^2 \leqslant (1+\varepsilon)||A - PA||_F^2.$$

The combination of the low rank case and this theorem is not trivial as points may be closer to a wrong center after projecting, see also Figure 4.2. Lemma 4.20 determines the existence of a clustering whose cost for the projected points A_m is at most the cost of C^*. Moreover, this clustering has constant distribution stability as well which, combined with the results from Section 4.3, allows us to use Local Search. Given that we can find a clustering with cost at most $(1+\varepsilon)\cdot ||A_m - XX^TA_m||_F^2$, Theorem 4.21 implies that we will have a $(1+\varepsilon)^2$-approximation overall.

To prove the lemma, we will require the following steps:

- A lower bound on the distance of the projected centers $||c_iV_mV_m^T - c_jV_mV_m^T|| \approx ||c_i - c_j||$.
- Find a clustering K with centers $S_m^* = \{c_1V_mV_m^T, \ldots, c_k^*V_mV_m^T\}$ of A_m with cost less than $||A_m - XX^TA_m||_F^2$.
- Show that in a well-defined sense, K and C^* agree on a large fraction of points.
- For any point $x \in K_i$, show that the distance of x to any center not associated with K_i is large.

We first require a technical statement.

Lemma 4.22. *For a point set A, let $C = \{C_1, \ldots C_k\}$ be a clustering with associated clustering matrix X and let A' and A'' be optimal low rank approximations where without loss of generality $k \leqslant rank(A') < rank(A'')$. Then for each cluster C_i*

$$\left|\left|\frac{1}{|C_i|}\sum_{j\in C_i}\left(A_j'' - A_j'\right)\right|\right|_2 \leqslant \sqrt{\frac{k}{|C_i|}}\cdot ||A - XX^TA||_2.$$

Proof. By Fact 4.18 $|C_i|\cdot||\frac{1}{|C_i|}\sum_{j\in C_i}(A_i'' - A_i')||_2^2$ is, for a set of point indexes C_i, the cost of moving the centroid of the cluster computed on A'' to the centroid of the cluster computed on A'. For a clustering matrix X, $||XX^TA' - XX^TA'||_F^2$ is the sum of squared distances of moving the centroids computed on the point set A'' to the centroids computed on A'. We then have

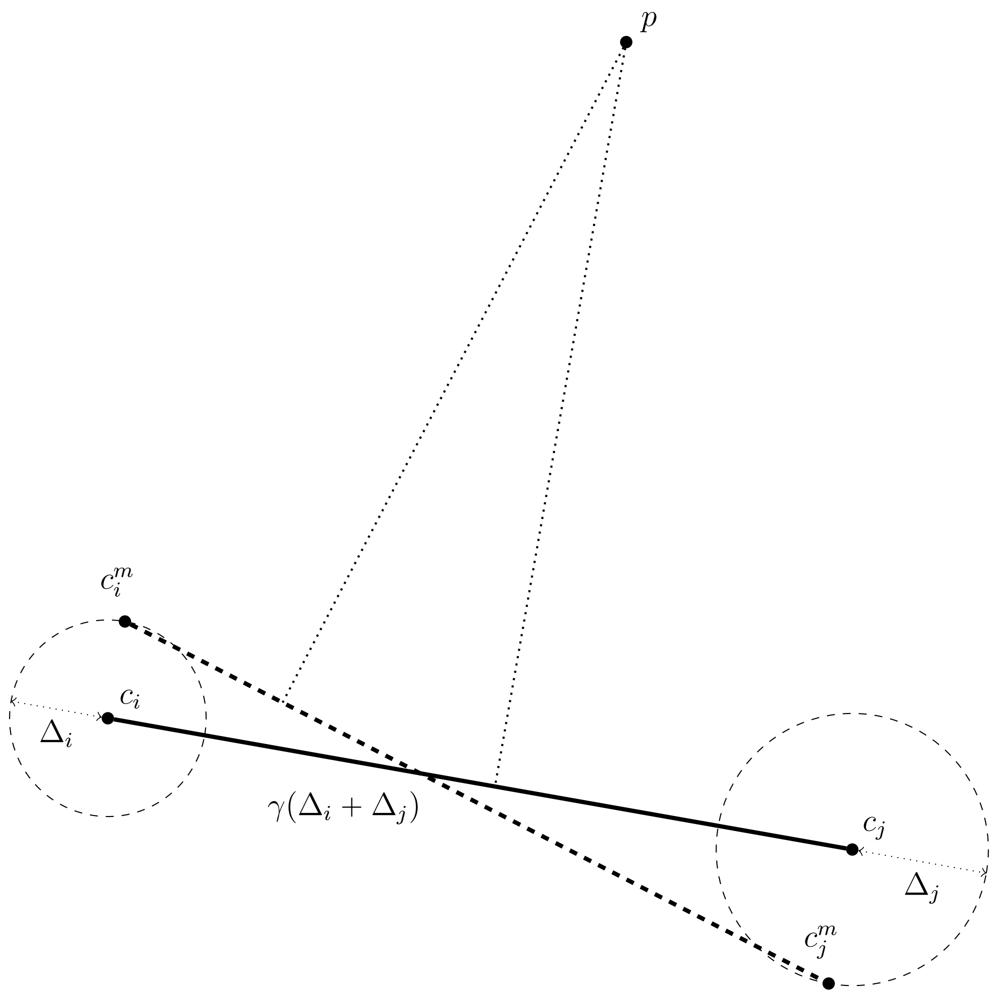


Figure 4.2: Despite the centroids of each cluster being close after computing the best rank m approximation, the projection of a point p to the line connecting the centroid of cluster C_i and C_j can change after computing the best rank m approximation. In this case $||p - c_j|| < ||p - c_i||$ and $||p - c_i^m|| < ||p - c_j^m||$. (Here $\Delta_i = \sqrt{\frac{k}{|C_i|}}||A - XX^T A||_2$.)

$$|C_i| \cdot \left|\left| \frac{1}{|C_i|} \sum_{j \in C_i} (A''_j - A'_j) \right|\right|_2^2 \leqslant ||XX^T A'' - XX^T A'||_F^2 \leqslant ||X||_F^2 \cdot ||A'' - A'||_2^2$$
$$\leqslant k \cdot \sigma_{k+1}^2 \leqslant k \cdot ||A - XX^T A||_2^2.$$

□

Proof of Lemma 4.20. For any point p associated with some row of A, let $p^m = pV_m V_m^T$ be the corresponding row in A_m. Similarly, for some cluster C_i, denote the center in A by c_i and the center in A_m by c_i^m. Extend these notion analogously for projections p^k and c_i^k to the span of the best rank k approximation A_k.

We have for any $m \geqslant k$ $i \neq j$

$$\begin{aligned} ||c_i^m - c_j^m|| &\geqslant ||c_i - c_j|| - ||c_i - c_i^m|| - ||c_j - c_j^m|| \\ &\geqslant \gamma \cdot \left(\frac{1}{\sqrt{C_i}} + \frac{1}{\sqrt{|C_j|}} \right) \sqrt{k} ||A - XX^T A||_2 \\ &\quad - \frac{1}{\sqrt{C_i}} \sqrt{k} ||A - XX^T A||_2 - \frac{1}{\sqrt{|C_j|}} \sqrt{k} ||A - XX^T A||_2 \\ &= (\gamma - 1) \cdot \left(\frac{1}{\sqrt{C_i}} + \frac{1}{\sqrt{|C_j|}} \right) \sqrt{k} ||A - XX^T A||_2, \end{aligned} \tag{4.4}$$

where the second inequality follows from Lemma 4.22.

In the following, let $\Delta_i = \frac{\sqrt{k}}{\sqrt{|C_i|}} ||A - XX^T A||_2$. We will now construct our target clustering K. Note that we require this clustering (and its properties) only for the analysis. We distinguish between the following three cases.

Case 1: $p \in C_i$ **and** $c_i^m = \underset{j \in \{1,\ldots,k\}}{\textbf{argmin}} ||p^m - c_j||$**:**

These points remain assigned to c_i^m. The distance between p_m and a different center c_j^m is at least $\frac{1}{2}||c_i^m - c_j^m|| \geqslant \frac{\gamma-1}{2}\varepsilon(\Delta_i + \Delta_j)$ due to Equation 4.4.

Case 2: $p \in C_i$**,** $c_i^m \neq \underset{j \in \{1,\ldots,k\}}{\textbf{argmin}} ||p^m - c_j||$**, and** $c_i^k \neq \underset{j \in \{1,\ldots,k\}}{\textbf{argmin}} ||p^k - c_j^k||$**:**

These points will get reassigned to their closest center.

The distance between p_m and a different center c_j^m is at least $\frac{1}{2}||c_i^m - c_j^m|| \geqslant \frac{\gamma-1}{2}\varepsilon(\Delta_i + \Delta_j)$ due to Equation 4.4.

Case 3: $p \in C_i$**,** $c_i^m \neq \operatorname*{\mathbf{argmin}}_{j\in\{1,\ldots,k\}} ||p^m - c_j^m||$**, and** $c_i^k = \operatorname*{\mathbf{argmin}}_{j\in\{1,\ldots,k\}} ||p^k - c_j^k||$**:**

We assign p^m to c_i^m at the cost of a slightly weaker movement bound on the distance between p^m and c_j^m. Due to orthogonality of V, we have for $m > k$, $(V_m - V_k)^T V_k = V_k^T(V_m - V_k) = 0$. Hence $V_m V_m^T V_k = V_m V_k^T V_k + V_m(V_m - V_k)^T V_k = V_k V_k^T V_k + (V_m - V_k)V_k^T V_k = V_k V_k^T V_k = V_k$. Then $p^k = pV_kV_k^T = pV_mV_m^TV_kV_k^T = p_mV_kV_k^T$.

Further, $||p^k - c_j^k|| \geqslant \frac{1}{2}||c_j^k - c_i^k|| \geqslant \frac{\gamma-1}{2}(\Delta_i + \Delta_j)$ due to Equation 4.4. Then the distance between p_m and a different center c_j^m is at least

$$\begin{aligned} ||p^m - c_j^m|| &\geqslant ||p^m - c_j^k|| - ||c_j^m - c_j^k|| \\ &= \sqrt{||p^m - p^k||^2 + ||p^k - c_j^k||^2} - ||c_j^m - c_j^k|| \\ &\geqslant ||p^k - c_j^k|| - \Delta_j \geqslant \frac{\gamma - 3}{2}(\Delta_i + \Delta_j), \end{aligned}$$

where the equality follows from orthogonality and the second to last inequality follows from Lemma 4.22.

Now, given the centers $\{c_1^m, \ldots c_k^m\}$, we obtain a center matrix M_K where the ith row of M_K is the center according to the assignment of above. Since both clusterings use the same centers but K improves locally on the assignments, we have $||A_m - M_K||_F^2 \leqslant ||A_m - XX^TA_m||_F^2$, which proves the first statement of the lemma. Additionally, due to the fact that $A_m - XX^TA_m$ has rank $m = k/\varepsilon$, we have

$$||A_m - M_K||_F^2 \leqslant ||A_m - XX^TA_m||_F^2 \leqslant m \cdot ||A_m - XX^TA_m||_2^2 \leqslant k/\varepsilon \cdot ||A - XX^TA||_F^2 \tag{4.5}$$

To ensure stability, we will show that for each element of K there exists an element of C, such that both clusters agree on a large fraction of points. This can be proven by using techniques from Awasthi and Sheffet [20] (Theorem 3.1) and Kumar and Kannan [126] (Theorem 5.4), which we repeat for completeness.

Lemma 4.23. *Let $K = \{C_1', \ldots C_k'\}$ and $C = \{C_1, \ldots C_k\}$ be defined as above. Then there exists a bijection $b : C \to K$ such that for any $i \in \{i, \ldots, k\}$*

$$\left(1 - \frac{32}{(\gamma-1)^2}\right)|C_i| \leqslant b(|C_i|) \leqslant \left(1 + \frac{32}{(\gamma-1)^2}\right)|C_i|.$$

Proof. Denote by $T_{i\to j}$ the set of points from C_i such that $||c_i^k - p^k|| > ||c_j^k - p^k||$. We first note that $||A_k - XX^TA||_F^2 \leqslant 2k \cdot ||A_k - XX^TA||_2^2$ which is at most

$$2k \cdot \left(||A - A_k||_2 + ||A - XX^TA||_2\right)^2 \leqslant 8k \cdot ||A - XX^TA||_2^2 \leqslant 8 \cdot |C_i| \cdot \Delta_i^2$$

for any $i \in \{1, \ldots, k\}$. The distance

$$||p^k - c_i^k|| \geqslant \frac{1}{2}||c_i^k - c_j^k|| \geqslant \frac{\gamma - 1}{2} \cdot \left(\frac{1}{\sqrt{C_i}} + \frac{1}{\sqrt{|C_j|}}\right)\sqrt{k}||A - XX^TA||_2^2.$$

Assigning these points to c_i^k, we can bound the total number of points added to and subtracted from cluster C_j by observing

$$\Delta_j^2 \sum_{i \neq j} |T_{i \to j}| \leqslant \sum_{i \neq j} |T_{i \to j}| \cdot \left(\frac{\gamma - 1}{2}\right)^2 \cdot (\Delta_i + \Delta_j)^2 \leqslant ||A_k - XX^TA||_F^2 \leqslant 8 \cdot |C_j| \cdot \Delta_j^2$$

$$\Delta_j^2 \sum_{i \neq j} |T_{j \to i}| \leqslant \sum_{j \neq i} |T_{j \to i}| \cdot \left(\frac{\gamma - 1}{2}\right)^2 \cdot (\Delta_i + \Delta_j)^2 \leqslant ||A_k - XX^TA||_F^2 \leqslant 8 \cdot |C_j| \cdot \Delta_j^2.$$

Therefore, the cluster sizes are up to some multiplicative factor of $\left(1 \pm \frac{32}{(\gamma-1)^2}\right)$ identical. □

We now have for each point $p^m \in C_i'$ a minimum cost of which is at least

$$||p^m - c_j^m||^2 \geqslant \left(\frac{\gamma - 3}{2}\left(\frac{1}{\sqrt{|C_i|}} + \frac{1}{\sqrt{|C_j|}}\right)\sqrt{k}||A - XX^TA||_2\right)^2$$

$$\geqslant \left(\frac{\gamma - 3}{2}\left(\sqrt{\frac{1}{\left(1 + \frac{32}{(\gamma-1)^2}\right)|C_i'|}} + \sqrt{\frac{1}{\left(1 + \frac{32}{(\gamma-1)^2}\right)|C_j'|}}\right)\sqrt{k}||A - XX^TA||_2\right)^2$$

$$\geqslant \frac{4(\gamma - 3)^2}{81}\varepsilon\frac{||A_m - M_K||_F^2}{|C_j'|}$$

where the first inequality holds due to Case 3, the second inequality follows from Lemma 4.23 and the last inequality follows from $\gamma > 3$ and Equation 4.5. This ensures that the distribution stability condition is satisfied. □

Proof of Theorem 1.10

Proof. Given the optimal clustering C^* of A with clustering matrix X, Lemma 4.20 guarantees the existence of a clustering K with center matrix M_K such that $||A_m - M_K||_F^2 \leqslant ||A_m - XX^TA_m||$ and that C has constant distribution stability. If $||A_m - M_K||_F^2$ is not a constant factor approximation, we are already done, as Local Search is guaranteed to find a constant factor approximation. Otherwise due to Corollary 4.17 (Section 4.3, there exists a discretization $(A_m, F, ||\cdot||^2, k)$ of $(A_m, \mathbb{R}^d, ||\cdot||^2, k)$ such that the clustering C of the first instance has at most $(1 + \varepsilon)$ times the cost of C in the second instance and such that C has constant distribution stability. By Theorem 1.5, Local Search with appropriate (but constant) neighborhood size will find a clustering C' with cost at most $(1+\varepsilon)$ times the cost of K in $(A_m, F, ||\cdot||^2, k)$. Let Y be the clustering matrix of C'. We then have $||A_m - YY^T A_m||_F^2 + ||A - A_m||_F^2 \leqslant (1+\varepsilon)^2 ||A_m - M_K||_F^2 + ||A - A_m||_F^2 \leqslant (1+\varepsilon)^2 ||A_m - XX^T A_m||_F^2 + ||A - A_m||_F^2 \leqslant (1+\varepsilon)^3 ||A - XX^T A||_F^2$ due to Theorem 4.21. Rescaling ε completes the proof. $\square$

CHAPTER 5

Beyond Separators: Clustering in Data Streams

In this chapter, we focus on the sliding window model and describe two algorithms, for the diameter and k-center problems.

In Section 5.1 we prove the following theorem.

Theorem 1.11. *Given a set of points A with aspect ratio α, there exists an algorithm computing a $(3 + \varepsilon)$-approximate solution for the metric diameter problem storing at most at most $O(\varepsilon^{-1} \cdot \log \alpha)$ points in memory. The update time is $O(\varepsilon^{-1} \log \alpha)$.*

Section 5.2 is dedicated to the proof of the following theorem.

Theorem 1.12. *Given a set of points A with aspect ratio α, there exists a $(6+\varepsilon)$-approximation algorithm for the metric k-center problem storing at most $O((k + 1)\varepsilon^{-1} \log(\alpha))$ points in memory. The update time is $O(k^3\varepsilon^{-1} \log \alpha)$.*

We remark that $\log \alpha$ is a very natural dependency: the distance between two points has to be encoded using at least $\log \alpha$ bits and so, any algorithm has to store this amount of bits in its temporary memory. This was made more formal by Feigenbaum et al. [82] who gave a space lower bound of $\Omega(\log \alpha)$ for any multiplicative approximation factor. We provide the following better lower bounds for the problem in Section 5.3.

Theorem 5.1. *For windows of size W, any deterministic sliding window algorithm outputting a solution of cost greater than or equal to $\frac{1}{3}$OPT for the distance oracle metric diameter problem with constant aspect ratio requires $\Omega(\sqrt{W})$ memory.*

Theorem 5.2. *Any randomized sliding window algorithm achieving an approximation factor less than 4 with probability bounded away from $\frac{1}{2}$ for the Window Metric Distance Oracle 2-Center problem with constant aspect ratio requires $\Omega(\sqrt[3]{W})$ memory.*

Preliminaries on data streams.

We first provide some intuition about the model. Let (A, dist) be a metric space where A is a set of points and $\text{dist} : A \times A \mapsto R_+$ is a distance function. A stream is a (potentially infinite) sequence of points from the metric space A (note that a point can appear multiple times in the stream). The sliding window of size N contains the most recent N elements of the stream.

We define the aspect ratio $\alpha = \frac{\max_{p,q \in A} \text{dist}(p,q)}{\min_{p \neq q \in A} \text{dist}(p,q)}$. To query the distance between two points p and q, we invoke a distance oracle $\text{dist}(p, q)$. We assume that the oracle can accessed only for those points we currently keep in memory and that the oracle itself requires no additional space.

We now turn to definitions of the two problems studied in this chapter. We adapt the definition of k-center problem given in Chapter A to the sliding window context.

Definition 5.3 (Diameter in the sliding window model)**.** *Given a stream of a metric space* $\sigma_1, \sigma_2, \ldots$*,* $\langle dist_t \rangle_{t>0}$*, the* sliding window diameter problem *asks at any time t, for* $s_1, s_2 \in \mathcal{W}_t$ *that maximize* $dist_t(s_1, s_2)$*.*

Definition 5.4 (k-center in the sliding window model)**.** *Given a stream of a metric space* $\sigma_1, \sigma_2, \ldots$*,* $\langle dist_t \rangle_{t>0}$*, and a non-negative integer k, the* sliding window k-center problem *asks at any time t, for a set of* centers $S \subseteq \mathcal{W}_t$*, of cardinality at most k, that minimizes*

$$cost(S) = \max_{x \in \mathcal{W}_t} \min_{c \in S} dist_t(x, c).$$

5.1 The Diameter Problem

For a given estimate γ of the diameter, our algorithm for the metric diameter problem maintains two witness points x, y (x appearing before y in the stream) at distance greater than γ together with a point q that has a certain degree of centrality among the points that are more recent that x. More formally, all points inserted after x up to the insertion time of q (including q) will be proven to have distance at most 2γ from one another and points inserted after q will have distance at most γ from q. Thus, the diameter is at most 3γ.

Specifically, Algorithm 5 aims at maintaining a certificate for the diameter consisting of two points x and y such that $\text{dist}(x, y) > \gamma$ and x inserted before y. In addition, we also store the point q submitted immediately prior to y and the most recent point. When a new point arrives, we test whether, based on the points we currently keep in memory, we can produce two points that were both inserted after x with distance more than γ. If we find such a pair, we update the points accordingly.

Our algorithm is independent of the size of the window. For any request asking what is the diameter of the window of size W, it either returns two points that are still in the

Algorithm 5 Sliding Window Algorithm for $(\gamma, 3\gamma)$-gap Diameter

```
1:  **Initialization:**
2:  $x \leftarrow$ first point of the stream
3:  $y \leftarrow$ first point $\sigma_{t_0}$ of the stream s.t dist$(\sigma_{t_0}, x) > \gamma$
4:  **if** dist$(\sigma_{t_0-1}, y) > \gamma$ **then**
5:      $x \leftarrow \sigma_{t_0-1}$
6:
7:  *Notation: $q$ last point inserted before $y$*
8:  **Main Algorithm:**
9:  Upon reception of a new point $\sigma_t$ $(t > t_0)$:
10: **if** dist$(\sigma_{t-1}, \sigma_t) > \gamma$ **then**
11:     $x \leftarrow \sigma_{t-1}$
12:     $y \leftarrow \sigma_t$
13: **else if** dist$(y, \sigma_t) > \gamma$ **then**
14:     $x \leftarrow y$
15:     $y \leftarrow \sigma_t$
16: **else if** dist$(q, \sigma_t) > \gamma$ and $q \neq x$ **then**
17:     $x \leftarrow q$
18:     $y \leftarrow \sigma_t$
```

window and at distance greater than γ or, if the oldest of the two has already expired, a point with a certain degree of centrality.

The properties of our algorithm are summarized by the following invariants.

Lemma 5.5. *The following invariants hold at the reception of a new point:*

1. *dist$(x, y) > \gamma$.*
2. *Any point inserted after x and before q (including q) is at distance at most γ from x.*
3. *Any point inserted after y is at distance at most γ from y.*
4. *If $q \neq x$, any point inserted after q is at distance at most γ from q.*

Proof. Invariant 1 follows directly from the specification of the algorithm. We thus focus on the proof of Invariants 2, 3, and 4. We proceed by induction on the time t. It is easy to check that the invariants hold at the end of the initialization step. Assuming all invariants holds for points x, y, q at time $t-1$ for $t > 1$, we will show they hold for x', y', q' at time t.

Suppose first that x did not change. Thus, q did not change. Therefore Invariant 2 holds by the inductive hypothesis. Since q did not change, we have that dist$(q, \sigma_t) \leqslant \gamma$. Thus, combining this observation with the inductive hypothesis (on Invariant 4) implies that Invariant 4 holds. Finally, since x did not change, y did not change either and so, we have dist$(\sigma_t, y) \leqslant \gamma$. Again, combining this observation with the inductive hypothesis (on Invariant 3) implies that Invariant 3 holds.

We now assume that x did change upon insertion of σ_t. Observe that $y' = \sigma_t$. Thus, Invariant 3 is vacuously true. Moreover, if $x' = \sigma_{t-1}$, then Algorithm 5 assigns σ_{t-1} to q. Thus, Invariants 2 and 4 hold by observing that $q = x$.

Otherwise, $q' = \sigma_{t-1}$ and so, we have either $x' = y$ or $x' = q$. Assume first that we have $x' = y$. Thus, by the inductive hypothesis on Invariant 3, we have that for each point σ_i inserted after y and strictly before σ_t, dist$(y, \sigma_i) \leqslant \gamma$. Thus, after assigning $x' = y$ and $q' = \sigma_{t-1}$, Invariant 2 holds. Moreover, in this case we have dist$(\sigma_{t-1}, \sigma_t) \leqslant \gamma$, and so Invariant 4 holds.

Assume now that we have $x' = q$. This implies that we had $q \neq x$ (before insertion of σ_t). Thus, by the inductive hypothesis on Invariant 4, we have that for each point σ_i inserted after q and strictly before σ_t, dist$(q, \sigma_i) \leqslant \gamma$. Thus, after assigning $x' = q$ and $q' = \sigma_{t-1}$, Invariant 2 holds. In this case again we have dist$(\sigma_{t-1}, \sigma_t) \leqslant \gamma$, and so Invariant 4 holds.

This completes the proof that all the invariants are satisfied at time t, and the lemma follows. □

We can thus turn to the proof of the approximation guarantee of the main theorem.

Proof of Theorem 1.11. Our algorithm for the diameter problem is Algorithm 6. It mainly consists in running several instances of Algorithm 5 for different values of γ.

Algorithm 6 Sliding Window Approximation Algorithm for the Diameter. At each time $t > 1$, returns a pair of points whose distance is within a factor of $1/3 - \varepsilon$ of the diameter of the window.

```
1: Let $\gamma_i = (1+\varepsilon)^i \min_{p \neq q \in A} \text{dist}(p, q)$
2: In parallel, run Algorithm 5($\gamma_i$), for $i = 0, \ldots, \log_{1+\varepsilon} \alpha$, and
3: Let $x_i, y_i$, be the variables corresponding to variables $x, y$ of Algorithm 5($\gamma_i$), and
4: In addititon, at each time $t$, find $i^* = \max\{i \mid x_i, y_i \in \mathcal{W}_t\}$
5: Return $x_{i^*}, y_{i^*}$
```

We first argue about the approximation guarantee of Algorithm 6. Observe that by Lemma 5.5, Invariant 1 holds and so, dist$(x_i^*, y_i^*) > \gamma$. We now claim that there are no two points at distance at least $3(1+\varepsilon)^{i^*+1} \min_{p \neq q \in A}$ dist(p, q) in the current window.

Consider Algorithm 5(γ_{i^*+1}). Let x, y, q denote x^{i^*+1}, y^{i^*+1} and q^{i^*+1} respectively. By Lemma 5.5, Invariants 2, 3, and 4 hold. Since i^* is maximum, we have that $x \notin \mathcal{W}_t$.

First, if $q \in \mathcal{W}_t$, we have $q \neq x$ and by Invariant 4, any point inserted after q is at distance at most γ from q and so, by the triangular inequality at most 2γ from each other. Now, since x has expired, the only points in the window that were inserted before q were inserted after x. Thus, by Invariant 2 and the triangular inequality, we have that all those points are at distance at most 2γ from one another and at most 2γ from q. Thus there are no two points at distance greater than 3γ in the window.

Second, if $q \notin \mathcal{W}_t$, then recall that q is the predecessor of y in the stream and thus Invariant 3 implies that all the points inserted strictly after q and before σ_t are at distance at most γ from y. Thus, there are at distance at most 2γ from each other. Again, we conclude that there are no two points at distance greater than 3γ in the window.

This concludes the proof of the approximation guarantee of our Algorithm.

The memory usage of the algorithm consists of 4 points per instance of Algorithm 5 and $\log_{1+\varepsilon} \alpha = \frac{\ln \alpha}{\ln(1+\varepsilon)} \leqslant \frac{2}{\varepsilon} \ln \alpha$ instances. □

5.2 The k-Center Problem

This section is dedicated to the proof of the following theorem.

Theorem 1.12. *Given a set of points A with aspect ratio α, there exists a $(6{+}\varepsilon)$-approximation algorithm for the metric k-center problem storing at most $O((k+1)\varepsilon^{-1}\log(\alpha))$ points in memory. The update time is $O(k^3\varepsilon^{-1}\log\alpha)$.*

The algorithm is similar to the algorithm we proposed in the previous section; It can be seen as a generalization of the diameter algorithm. The idea is again to "estimate" the value of the optimal solution. For a given estimate γ of the k-center solution, our algorithm for the metric diameter problem either produces $k+1$ witness points at distance greater than 2γ – this ensures that there is no solution with k centers of cost γ – or a set of points that have a certain degree of centrality among the points in the current window. However, the algorithm is a bit more technical than the one for the diameter.

We introduce the *Time To Live* value of a point p: Upon insertion $TTL(p)$ is set to the window size N and with each subsequently inserted point it is decremented. We say that p *expires* if $TTL(p) = 0$. We extend the common use of TTL to negative numbers to indicate the number of points submitted after expiration, *i.e.,* $TTL(p) = -10$ means that 10 points were submitted after the expiration of p.

A high level description of our algorithm is as follows, see Algorithm 7 for the pseudocode.

We maintain a set A of at most $k+1$ *attraction points*. For each attraction point a, we maintain the newest point $R(a)$ within radius 2γ as a *representative*, i.e. $R(a) = \underset{p:\ \mathrm{dist}(p,a)\leqslant 2\gamma}{\mathrm{argmax}}\ TTL(R(a))$. When an attraction point expires, the representative point remains in memory. Call the set of representative points whose attraction points expired, the *orphaned representatives* O, and the set of representative points whose attraction points are still in the current window *active representatives* R. A new point p may become an attraction point if its distance is greater than 2γ to any point in A upon insertion. If the cardinality of A is greater than k, we retain the newest $k+1$ attraction points of A and all points with a greater TTL than the minimum TTL of A.

When asked to provide a clustering, we iterate through all estimates and either provide a counter example, or find a clustering which is then guaranteed to be a $6(1+\varepsilon)$-approximation. Our set of centers C first consists of an arbitrarily chosen point $p \in A \cup R \cup O$. Thereafter we greedily add any point $q \in A \cup R \cup O$ with distance dist$(q, C) > 2\gamma$. If upon termination $|C| > k$, we have a certificate for OPT $> \gamma$ and move to the next higher estimate. The smallest estimate with $|C| \leqslant k$ is then guaranteed to be a 6 approximation.

We start by giving the space bound.

Algorithm 7 Sliding Window Algorithm for $(\gamma, 6 \cdot \gamma)$-gap k-Center

```
1:  $A, R, O \leftarrow \emptyset$;
2:  for all element $p$ of the stream do
3:      if $q \in O$ expires then
4:          $O \leftarrow O \setminus \{q\}$;
5:      if $a \in A$ expires then
6:          DeleteAttraction($a$);
7:      Insert($p$);
8:  procedure DeleteAttraction($a$)
9:      $O \leftarrow O \cup \{R(a)\}$;
10:     $R \leftarrow R \setminus \{R(a)\}$;
11:     $A \leftarrow A \setminus \{a\}$;
12: procedure Insert($p$)
13:     $D \leftarrow \{a \in A \mid \text{dist}(p, a) \leqslant 2 \cdot \gamma\}$;
14:     if $D = \emptyset$ then
15:         $A \leftarrow A \cup \{p\}$
16:         $R(p) \leftarrow p$
17:         $R \leftarrow R \cup \{R(p)\}$
18:         if $|A| > k + 1$ then
19:             $a_{old} \leftarrow \operatorname{argmin}_{a \in A} TTL(a)$;
20:             DeleteAttraction($a_{old}$);
21:         if $|A| > k$ then
22:             $t \leftarrow \min_{a \in A} TTL(a)$;
23:             for all $q \in O$ do
24:                 if $TTL(q) < t$ then
25:                     $O \leftarrow O \setminus \{q\}$;
26:     else
27:         for all $a \in D$ do
28:             Exchange $R(a)$ with $p$ in $R$;
```

Lemma 5.6. *At any given time, the number of points kept in memory is bounded by at most* $3(k+1)$.

Proof. We number all attraction points we keep in memory via the sequence in which they arrived, i.e. a_1 is the first attraction point, a_2 the second, etc. Call this sequence S. Note that in this sequence a_1 also expires before a_2.

At any given time, we maintain at most $k+1$ attraction points A and $k+1$ active representative points R due to lines 18-20 and the subroutine DeleteAttraction (lines 8-11). What remains to be shown is that the number of orphaned representative points O also never exceeds $k+1$.

First, we show that $TTL(a_{i+k+1}) > TTL(R(a_i)) \geqslant TTL(a_i)$. We distinguish between two cases. If a_i expires, then a_{i+k+1} gets inserted after a_i exits the window, hence $TTL(a_{i+k+1}) > N + 1 + TTL(a_i)$ and $TTL(R(a_i)) + N \leqslant TTL(a_i)$. Otherwise, a_i gets deleted via lines 18-20 in the exact same time step in which a_{i+k+1} got inserted, in which case the claim also holds.

Now consider any point of time and let j be the maximum index of any attraction point in S that has expired. By the above reasoning, any representative spawned by $a_{j-(k+1)}$ is no longer in memory, and the space bounds holds. □

Lemma 5.7. *Let P be a set of points in a given window, $\gamma > 0$ an estimate of the clustering cost, $A \cup R \cup O$ the set of points we currently keep in memory with $|A| \leqslant k$. Then $\max_{q \in P} \operatorname{dist}(p, R \cup O) \leqslant 4\gamma$.*

Proof. We note that for any attraction point a, the representative $R(a)$ has maximum TTL among all points with distance at most 2γ. When a point p arrives, it has distance at most 2γ to some attraction point (which may be identical to p if we create a new one). Hence, if $R(a)$ is still in memory, the claim holds for p.

We now argue that by executing lines 18-25, all points p with $\operatorname{dist}(p, R \cup O) > 4\gamma$ have $TTL(p) < \min_{a \in A} TTL(a)$. If $TTL(p) > \min_{a \in A} TTL(a)$, then there exists an attraction point a' such that $\operatorname{dist}(p, a') \leqslant 2\gamma$. Then we have $TTL(R(a')) \geqslant TTL(p) > \min_{a \in A} TTL(a)$ and $\operatorname{dist}(p, R(a')) \leqslant 4\gamma$. Due to lines 24-25, $R(a')$ is never deleted until it expires. □

Combining these lemmas and using arguments analogous to those of the proof of Theorem 1.11, we have:

Theorem 1.12. *Given a set of points P with aspect ratio α, there exists an algorithm computing a $6(1+\varepsilon)$-approximate solution for the metric k-center problem storing $6(k+1)\ln(\alpha)/\varepsilon)$ points. The update time per point is $O(k^2\varepsilon^{-1}\log\alpha)$.*

Proof. Again define an exponential sequence to the base $(1+\varepsilon)$ and run Algorithm 7 in parallel for all powers of $(1+\varepsilon)$ as objective value estimates. The space bound then follows from Lemma 5.6.

For each estimate γ, we greedily compute a clustering of $A \cup R \cup O$ where the pairwise distance between centers is greater than 2γ. Now consider the smallest estimate γ' for which the greedy clustering requires at most k centers C.

We have $\max_{p \in A \cup R \cup O} \operatorname{dist}(p, C) \leqslant 2\gamma'$. We further have for any point q in the current window $\max_{q \in P} \operatorname{dist}(q, C) \leqslant \max_{q \in P} \operatorname{dist}(q, R \cup O) + \max_{p \in A \cup R \cup O} \operatorname{dist}(p, C) \leqslant 4\gamma' + 2\gamma' \leqslant 6\gamma'$ due to Lemma 5.7. Since we have OPT $> \frac{\gamma'}{1+\varepsilon}$, C is a $6(1+\varepsilon)$ approximation. □

5.3 Lower bounds

Our lower bounds for the studied problems hold for the metric oracle distance model. Whenever we wish to know the distance between two points p, q, we have to store the points in their entirety in order to invoke the oracle. The fundamental assumption used in the proofs of this section is that the algorithm cannot create new points, unlike, for instance, in Euclidean spaces, where we can store projections, means and similar points. In particular, this implies that once a point is discarded by the algorithm, it cannot be recalled by any means at a later date. Without any assumptions as to how the points are encoded, we measure the space complexity of an algorithm via the number of stored points, rather than the number of bits. In this model, the space required to store the distance oracle, the TTL of each point or any other information we might wish to store are not considered. A similar model can be also found in the paper by Guha [93], where the author was able to derive a lower bound of $\Omega(k^2)$ points for any deterministic single-pass streaming algorithm approximating the cost of the optimal k-center clustering up to a factor $2 + 1/k$.

We first describe an adversarial input sequence for deterministic algorithms for the diameter problem and give a proof for deterministic algorithms.

With some modification, these ideas can be extended to the k-center problem and to randomized algorithm as well. We aim for a lower bound of $\Omega(\sqrt{W})$ points for deterministic algorithms. We divide the input into $\sqrt{W}$ buckets containing $\sqrt{W}$ points each. Unless the distances between two points are further specified, we will set these distances to 1. Denote the ith bucket by B_i and the jth point of bucket B_i by $p_{i,j}$ with $i, j \in \{0, \ldots, \sqrt{W} - 1\}$. The points appear bucket by bucket, that is, $p_{i,j}$ is the $(i \cdot \sqrt{W} + j + 1)$th input point.

We assume that an algorithm always stores less than $\sqrt{W}$ points. Therefore, the algorithm must discard at least one point of bucket B_i before reading the first point of bucket B_{i+1}. Let f_i be such a discarded point in B_i. To any point from some bucket B_j, $j > i$, we then set the distance to f_i to be 2. For the same reason, there is at least one bucket without any stored points when the $(W + 1)$st input point is read. Let B_t be this bucket. We now introduce the $(W + 1)$st input point p that satisfies the distances dist$(p, p_{i,j}) = 1$ if $i > t$, dist$(p, p_{i,j}) = 3$ if $p_{i,j} = f_t$, and dist$(p, p_{i,j}) = 2$ otherwise.

We proceed to insert copies of p until all points in buckets B_i with $i < t$ are expired. Therefore, there is no pair of points in memory with distance larger than 1. The algorithm can only output two points at distance 1 whereas the true diameter is 3.

Theorem 5.1. *For windows of size W, any deterministic sliding window algorithm outputting a solution of cost greater than or equal to $\frac{1}{3}$OPT for the distance oracle metric diameter problem with constant aspect ratio requires $\Omega(\sqrt{W})$ memory.*

Proof. For the sake of contradiction, we assume that there exists an algorithm ALG that returns a solution whose cost is a factor 3 from the optimal solution while the algorithm stores less than $\sqrt{W}$ points. We start with the adversarial input sequence by submitting

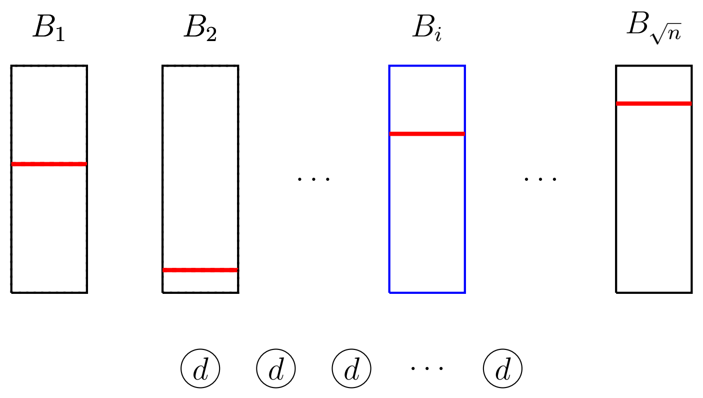


Figure 5.1: The lower bound instance. Points of each bucket are inserted one after the other from bucket B_1 to bucket $B_{\sqrt{W}}$. One of the point f_i, in red, of each bucket B_i has to be "forgotten" by the algorithm upon insertion of the first point of the following bucket. Each of those "forgotten" point has distance 1 to its predecessor but distance 2 to all the points of the subsequent buckets. The crux of the argument is that the algorithm cannot know this since the point is forgotten before the next bucket arrives. Thus this forgotten point looks similar to the others. Since there are $\sqrt{W}$ buckets, all the points of one bucket B_t, in blue, have to be "forgotten" upon insertion of the first "d" point. Thus, the distance of a "d" point can be set to 2 to the point of buckets $B_1, \dots, B_t$ and to 1 to the other bucket. Additionally the distance from the "d" points to f_t can be set to 3. However, after all the point inserted before B_t have expired, the algorithm only has points at distance 1 from each other. Thus, it cannot determine whether the diameter of this instance is 1 or 3.

$\sqrt{W}$ *buckets* each containing $\sqrt{W}$ points. We denote the ith bucket by B_i and the jth point of bucket B_i by $a_{i,j}$ with $i, j \in \{0, \dots, \sqrt{W} - 1\}$. Any point of bucket B_i is only read after all $\sqrt{W}$ points of bucket B_{i-1} are received. The points within each bucket are at distance 1 from one another.

Since the algorithm stores less than $\sqrt{W}$ points, there is at least one point in a bucket that must be discarded before an point of the next bucket is read. Let f_i be such a point for bucket B_i. The points of bucket B_i have the following distance to all future points:

$$\text{dist}(a_{i',j}, a_{i,l}) = \begin{cases} 2 & \text{for all } a_{i',j} = f_{i'} \\ 1 & \text{otherwise} \end{cases}$$

for all $i' < i$.

It is easy to see that the distances satisfy the triangle inequality. Since there are $\sqrt{W}$ buckets and the algorithm stores less than $\sqrt{W}$ points, there is at least one bucket for which all the points are missing. Let B_t be this bucket.

We now introduce the $(W+1)$st input point a with distances

$$\mathrm{dist}(a, a_{i,j}) = \begin{cases} 1 & \text{if } i > t \\ 3 & \text{if } a_{i,j} = f_t \\ 2 & \text{otherwise} \end{cases}.$$

To show that the distances still satisfy the triangle inequality, we first observe that only $\mathrm{dist}(a, f_t)$ is neither 1 or 2 and thus requires special consideration. Here, we have $3 = \mathrm{dist}(a, f_t) \leqslant \mathrm{dist}(a, a_{i,j}) + \mathrm{dist}(a_{i,j}, f_t) = 1 + 2$ for $i > t$, and $3 = \mathrm{dist}(a, f_t) \leqslant \mathrm{dist}(a, a_{i,j}) + \mathrm{dist}(a_{i,j}, f_t) \leqslant 2 + 1$ for $i \leqslant t$.

First we note that f_t is at distance at least 1 from all the buckets B_i, $i \leqslant t$ so $\mathrm{dist}(a, f_t) = 3$ satisfies the triangle inequality. Since all the points of buckets $B_{i'}$, $i' > t$ are at distance 1 from the points of buckets B_i, $i \leqslant t$ and distance 2 from f_t, $\mathrm{dist}(a, a_{i',j'}) = 1$ satisfies the triangle inequality as well.

Now, we keep inserting copies of a at distance 1 from one another until all the points of buckets $t' < t$ have expired. Since all the remaining points in memory are a subset of $\bigcup_{t'>t}(B_{t'} \backslash f_{t'})$ plus copies of p, the points in memory are all at distance 1 from one another. It follows that the algorithm can only output a solution of value 1 whereas the pair (a, f_t) induces a solution of value 3. □

We now turn to the k-center lower bound.

Theorem 5.2. *Any randomized sliding window algorithm achieving an approximation factor less than* 4 *with probability bounded away from* $\frac{1}{2}$ *for the Window Metric Distance Oracle* 2*-Center problem with constant aspect ratio requires* $\Omega(\sqrt[3]{W})$ *memory.*

Proof. The proof uses arguments similar to those given in the proof of Theorem 5.1. Again, for ease of exposition, we will use a window of size $\Theta(W)$, the theorem then holds by rescaling W. We first pick a constant $\ell = 32$. Now, define $8W^{1/3}$ buckets consisting of $128W^{2/3} + \ell$ points each. The points of the ith bucket are at distance 2 from each other. We iteratively replace one randomly chosen point of the last $128W^{2/3}$ points from bucket B_i with f_i, where $\mathrm{dist}(p_{i,j}, p_{i',*}) = \begin{cases} 4 & \text{for all } p_{i,j} = f_i \\ 2 & \text{otherwise} \end{cases}$ for all points $i' > i$.

We now define ℓ points $p_0, \dots, p_{\ell-1}$ whose distances are specified below but we do not insert them yet. Finally, we randomly choose a bucket t and a point $p* \in \{p_0, \dots, p_{\ell-1}\}$ such that for all $p_{i'} \in \{p_0, \dots, p_{\ell-1}\} \backslash \{p*\}$ and point p, we have

$$\mathrm{dist}(p_{i'}, p) = \begin{cases} 4 & \text{if } p = f_t \\ 1 & p \in \{p_0, \dots, p_{\ell-1}\} \backslash \{p_{i'}\} \\ 2 & p \in \left(\bigcup_{t' \geqslant t} B_{t'}\right) \backslash \{f_t\} \end{cases} \text{ and } \mathrm{dist}(p*, p) = \begin{cases} 3 & \text{if } p = f_t \\ 1 & p \in \left(\bigcup_{t' \geqslant t} B_{t'}\right) \backslash \{f_t\} \end{cases}.$$

Let T_t denote the time at which the first point of B_t is inserted. Note that we do not specify the distance from the points of $\{p_0, \ldots, p_{\ell-1}\}$ to the buckets $B_{t''}$ for $t'' < t$, since we insert copies of the last point of the last bucket in order to make all the points of buckets $0, \ldots, t-1$ expire. Then we insert the points $p_0, \ldots, p_{\ell-1}$.

Several things should be noted about the input. First, all distances obey the triangle inequality. Second, f_t expires after time $T_t + \ell$. And lastly, at any given time from $T_t + \ell$ until f_t expires, there exists a solution of cost 1 which consists of $p*$ and f_t.

By Yao's minimax principle, it is sufficient to bound the number of points used by any deterministic algorithm against the above input distribution.

We first bound the probability that the algorithm stores some point f_i. Call this event A. Assuming that the algorithm never stored a point $f_{i'}$, the points stored by any future bucket B_i with $i > i'$ are fixed. The probability that f_i is one of these points is bounded by the hypergeometric distribution with population $128W^{2/3}$, $W^{1/3}$ samples and 1 success in both population and sample: $\frac{\binom{128W^{2/3}-1}{W^{1/3}-1}\cdot\binom{1}{1}}{\binom{128W^{2/3}}{W^{1/3}}}$. Then the probability that no f_i is stored for any of the $8W^{1/3}$ buckets can be lower bounded as follows.

$$\begin{aligned} 1-\mathbb{P}[A] &\geqslant \left(1-\frac{\binom{128W^{2/3}-1}{W^{1/3}-1}\cdot\binom{1}{1}}{\binom{128W^{2/3}}{W^{1/3}}}\right)^{8W^{1/3}} = \left(1-\frac{W^{1/3}}{128W^{2/3}}\right)^{8W^{1/3}} \geqslant 1-\frac{1}{16}=\frac{15}{16} \\ \Leftrightarrow \mathbb{P}[A] &\leqslant \frac{1}{16}. \end{aligned}$$

Now we bound the probability that the algorithm retains any point from bucket t upon submission, which we call event B. Again, conditioned on the fact that event A does not hold ($\overline{A}$), the buckets from which the algorithm stores at least one point are fixed. The probability that B_t is among the stored buckets again follows a hypergeometric distribution with population $8W^{1/3}$, $W^{1/3}$ samples and 1 success in both population and sample. Therefore $\mathbb{P}[B|\overline{A}] = \frac{\binom{8W^{1/3}-1}{W^{1/3}-1}\cdot\binom{1}{1}}{\binom{8W^{1/3}}{W^{1/3}}} = \frac{1}{8}$.

Finally, we consider the event that the algorithm does pick $p*$ as a center. Let C denote this event. In order for C to happen, the algorithm has to be able to distinguish between points of $p_0, \ldots, p_{\ell-1}$ or it has to pick $p*$ randomly from $p_0, \ldots, p_{\ell-1}$. The first case is identical to event B, the second case happens with probability at most $2/\ell$, since the algorithm can open 2 centers. It follows that $\mathbb{P}[C] \leqslant \mathbb{P}[B] + 2/\ell$.

If none of the events A, B, or C hold, the algorithm will not output two centers with approximation factor less than 4, The probability that an algorithm storing $W^{1/3}$ points achieves this is at most

$$\begin{aligned}
\mathbb{P}[A \cup B \cup C] &\leqslant \mathbb{P}[A] + \mathbb{P}[B] + \mathbb{P}[C] \leqslant \mathbb{P}[A] + 2\mathbb{P}[B] + 2/\ell \\
&= \mathbb{P}[A] + 2\left(\mathbb{P}[B|A] \cdot \mathbb{P}[A] + \mathbb{P}[B|\overline{A}] \cdot \mathbb{P}[\overline{A}]\right) + 2/\ell \\
&\leqslant 3 \cdot \mathbb{P}[A] + 2\mathbb{P}[B|\overline{A}] + 2/\ell \leqslant \frac{3}{16} + \frac{2}{8} + \frac{2}{32} = \frac{1}{2}.
\end{aligned}$$

□

CHAPTER 6

Implementations and Experiments

In this chapter, we provide evidence that local search is competitive in practice. Our theoretical trade-off between running time and quality of approximation seems quite prohibitive; for a 3/2-approximation, the running time is at least n^8 for 2-dimensional instances. However, Johnson et al. [108] showed that local search for TSP is a competitive algorithm for both real-world and random instances. Thus, we ask the following natural question.

How does local search for clustering compare to practical algorithms on real-world and random instances?

6.1 The Questions

The goal of clustering is to partition data according to similarity. Throughout this thesis, we model this goal by the k-clustering objective (Definition A.1). Thus, we address the natural question of the appropriateness of the k-clustering objective (and in particular the k-means objective) for finding "good" clusters. More precisely, we ask: when is an optimal solution with respect to the k-means objective a partitioning of the data according to similarity?

To answer this question we consider instances that possess a "natural" partitioning of the data and compare this clustering to an optimal clustering with respect to the k-means. We consider the following instances.

- Real instances stemming from machine learning for which the goal is to classify the input elements into k classes.
- Random Euclidean instances generated from a mixture of k Gaussians for which the goal is to find the sets of points generated by each of the k Gaussians[2].

[2]Note that this has also been studied as a "learning" problem, see [146] for example.

We say that a clustering is *optimal* if its k-means cost is minimum. We define the *ground truth* clustering of an instance to be the clustering that we would like to compute. More precisely, the ground truth clustering of a real instance is the partition of the elements into *the correct classes* (see below for a more formal definition) and the ground truth clustering of a random instance is the family consisting of the k sets of elements generated by the k Gaussians.

In the following, when we discuss the value of a clustering we always refer to the value of the clustering with respect to the k-means objective. We thus aim at answering the following question:

When do the ground truth and optimal clusterings agree?

More precisely, when is the value the ground truth clustering at most 1.05 times the value of the optimal clustering?

In Chapter 4, we have analyzed three characterizations of the real-world instances. We have shown that local search is a good heuristic for those instances (see Theorems 1.5, 1.8 and 1.10). In this chapter, we focus on the β-distribution-stability. Theorem 1.5 states that local search with neighborhood of size $n^{\Omega(\varepsilon^{-3}\beta^{-1})}$ returns a solution of cost at most $(1+\varepsilon)$OPT. Thus, we ask the following question.

For which values of β are the random and real instances β-distribution-stable?

Finally, we compare local search to Lloyd's algorithm, a widely-used algorithm in practice (see Algorithm 8). We focus on the following question.

Is local search a competitive heuristic for random and real-world instances?

Algorithm 8 Lloyd's algorithm.

```
1: Input: An instance of $k$-means
2: $S' \leftarrow$ greedy solution to the problem, $S \leftarrow \varnothing$
3: while $S \neq S'$ do
4:     $S \leftarrow S'$
5:     Partition $\mathcal{F} \leftarrow \left\langle \{a \mid \text{dist}(c,a) = \min_{c' \in S} \text{dist}(c',a)\} \right\rangle_{c \in S}$ (breaking ties arbitrarily)
6:     $S' \leftarrow \bigcup_{F \in \mathcal{F}}$ (optimal center for the elements in $F$)
7: Output: $S$
```

Related Work

The most prominent experimental work on the analysis of Lloyd algorithm for the k-means objective is due to Kanungo et al. [111, 112] in 2004. They compare Lloyd algorithm to a slightly weaker version of local search and different combinations of those algorithms. The local search algorithm they study is the following. Instead of enumerating all the possible swaps they only sample a few swaps. They show that this local search algorithm (with $s \in \{1, 2\}$) converges much slowly than Lloyd algorithm for real-world datasets stemming from image compression and consisting of low-dimensional Euclidean instances and for mixtures of Gaussians. More precisely, they show that after 500 *steps* (a step is a re-computation of the clusters in Lloyd's algorithm and a sampling of a few possible swaps in local search), local search is worse than Lloyd's algorithm.

However, the emergence of multicore processors calls for a new study. Indeed, it is fairly easy to boost the performances of local search using multi-threading. Whereas, because of its sequential nature, it seems hard to take advantage of multicore processors to improve the running time of Lloyd algorithm. Note that we only consider multicore processors; Researchers have recently come up with variants of Lloyd algorithm for GPUs and MapReduce environments, see [23] for example.

Technical Aspects

We implemented the Algorithms in C++ and Python. The C++ compiler is `g++ 4.6.3`. At each step, the neighborhood of the current solution was explored in parallel: 8 threads were created by a Python script and each of them correspond to a C++ subprocess that explores a 1/8 fraction of the space of the neighboring solutions. The best neighboring solution found by the 8 threads was taken for the next step. For Lloyd's algorithm we use the C++ implementation by Kanungo et al. [112] available online [113].

The LP for the linear program was generated via a Python script and solved using the solver `CPLEX` [1].

The synthetic data was generated via a Python script using the `numpy` library [176].

The machines used for the experiments have a processor `Intel(R) Core(TM) i73770 CPU, 3.40GHz` with four cores and a total virtual memory of 8GB. The system is Ubuntu 12.04.5 LTS.

6.2 Experimental Results

We focus on the k-means objective and we consider real-world and random instances with ground truth clustering and study under which conditions the value of the solution induced by the ground truth clustering is close to the value of the optimal clustering with respect to the k-means objective.

We approximate the optimal value of the k-means objective using the fractional solution of a standard LP relaxation for the problem (see Algorithm 9). It thus provides a lower bound on the value of the optimal solution. We also obtain an upper bound on the optimal value by running our local search algorithm (Algorithm 2) with $s = 1$ and until it reaches a local optimum. The average ratio between our upper and lower bounds is 1.15. Therefore, we can estimate the average optimal value within a 1.15 factor. we make use of this estimate in the following sections to compute the β-distribution-stability of the instance. Note that the variance for the value of the optimal fractional solution we observe is less than 0.5% of the value of the optimal solution. Therefore, our estimate of β is quite accurate.

Algorithm 9 Linear relaxation for the k-means problem.

Input: A set of clients A, a set of candidates centers F, a number of centers k, a distance function dist.

$$\min \sum_{a\in A}\sum_{b\in F} x_{a,b}\cdot \text{dist}(a,b)^2$$

subject to,

$$\begin{array}{lrcl}
 & \sum_{b\in F} y_b & \leqslant & k \\
\forall a\in A, & \sum_{b\in F} x_{a,b} & = & 1 \\
\forall a\in A,\ \forall b\in F, & y_b & \geqslant & x_{a,b} \\
\forall a\in A,\ \forall b\in F, & x_{a,b} & \geqslant & 0
\end{array}$$

6.2.1 Real Data

In this section, we focus on four classic real-world datasets with ground truth clustering: `abalone`, `digits`, `iris`, and `movement_libras`. `abalone`, `iris`, and `movement_libras` have been used in various works (see [68, 76, 84, 85, 172] for example) and are available online at the UCI Machine learning repository [130].

The `abalone` dataset consists of 8 physical characteristics of all the individuals of a population of abalones. Each abalone corresponds to a point in a 8-dimensional Euclidean space. The ground truth clustering consists in partitioning the points according to the age of the abalones.

The `digits` dataset consists of 8px-by-8px images of handwritten digits from the standard machine learning library scikit-learn [153]. Each image is associated to a point in a 64-dimensional Euclidean space where each pixel corresponds to a coordinate. The ground truth clustering consists in partitioning the points according to the number depicted in their corresponding images.

The `iris` dataset consists of the sepal and petal lengths and widths of all the individuals of a population of iris plant containing 3 different types of iris plant. Each plant is associated to a point in 4-dimensional Euclidean space. The ground truth clustering consists in partitioning the points according to the type of iris plant of the corresponding individual.

The `Movement_libras` dataset consists of a set of instances of 15 hand movements in LIBRAS[1]. Each instance is a curve that is mapped in a representation with 90 numeric values representing the coordinates of the movements. The ground truth clustering consists in partitioning the points according to the type of the movement they correspond to.

Properties	`Abalone`	`Digits`	`Iris`	`Movement_libras`
Number of points	636	1000	150	360
Number of clusters	28	10	3	15
Value of ground truth clustering	169.19	938817.0	96.1	780.96
Value of fractional relaxation	4.47	855567.0	83.96	366.34
Value of Algorithm 2 ($s = 1$)	4.53	855567.0	83.96	369.65
$\%$ of pts correct. class. by Alg. 2 ($s = 1$)	17	76.2	90	39
β-stability	1.27e-06	0.0676	0.2185	0.0065

Table 6.1: Properties of the real-world instances with ground truth clustering.

Table 6.1 shows the properties of the four instances.

For the `Abalone` and `Movement_libras` instances, the values of an optimal solution is much smaller than the value of the ground truth clustering. Thus, since local search optimizes with respect to the k-means objective, the clustering output by local search is far from the ground truth clustering for those instances: the percentage of points correctly classified by Algorithm 2 is at most 17% for the `Abalone` instance and at most 39% for the `Movement_libras` instance. For the `Digits` and `Iris` instances the value of the ground truth clustering is at most 1.15 times the optimal value. In those cases, the number of points correctly classified is much higher: 90% for the `Iris` instance and 76.2% for the `Digits` instance.

The experiments also show that the β-distribution-stability condition is satisfied for $\beta > 0.06$ for the `Digits`, `Iris` and `Movement_libras` instances. This shows that the β-distribution-stability condition captures the structure of some famous real-world instances for which the k-means objective is meaningful for finding the optimal clusters. We thus make the following observations.

Observation 6.1. *If the value of the ground truth clustering is at most 1.15 times the value of the optimal solution, then the instance is β-distribution-stable for $\beta > 0.06$.*

The experiments show that Algorithm 2 with neighborhood size 1 ($s = 1$) is very efficient for all those instances since it returns a solution whose value is within 2% of the

[1]LIBRAS is the official Brazilian sign language

optimal solution for the `Abalone` instance and a within 0.002% for the other instances. Note that the running time of Algorithm 2 with $s = 1$ is $\tilde{O}(k \cdot n/\varepsilon)$ (using a set of $O(n)$ candidate centers) and less than 15 minutes for all the instances. We make the following observation.

Observation 6.2. *If the value of the ground truth clustering is at most 1.15 times the value of the optimal solution, then local search with neighborhood of size 1 computes a clustering that agrees with the ground truth clustering on more than 75% of the points.*

Finally, observe that for those instances the value of an optimal solution to the fractional relaxation of the linear program is very close to the optimal value of an optimal integral solution (since the cost of the integral solution is smaller than the cost returned by Algorithm 2). This suggests that the fractional relaxation (Algorithm 9) might have a small integrality gap for real-world instances.

6.2.2 Data generated from a mixture of k Gaussians

We generate d-dimensional instances consisting of 1000 points generated from a mixture of k Gaussians with the same variance σ, for $d \in \{5, 10, 50\}$ and $k \in \{5, 50, 100\}$. We generate 100 instances for each possible choice of the parameters. The means of the k Gaussians are chosen uniformly and independently at random in $\mathbb{Q}^d$. The ground truth clustering is the family of sets of points generated by the same Gaussian. We compare the value of the ground truth clustering to the optimal value clustering.

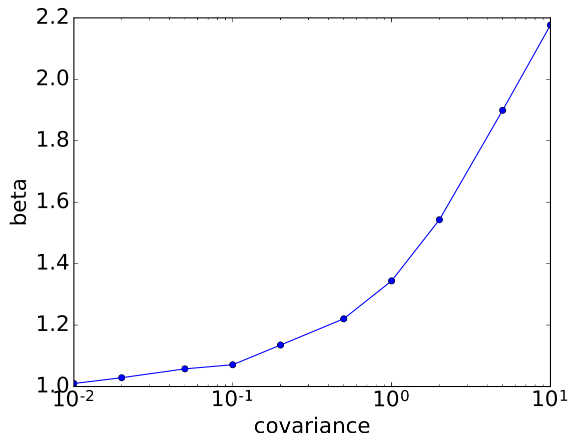
The results are presented in Figures 6.1 and 6.2. We observe that when the variance σ is large, the ratio between the average value of the ground truth clustering and the average value of the optimal clustering becomes more important. Indeed, the ground truth clusters start to overlap, allowing to improve the objective value by defining slightly different clusters.Therefore, the use of the k-means problem as a model is not suitable anymore for recovering the ground truth. In those cases, since local search optimizes the solution with respect to the current cost, the clustering output by local search is very different from the ground truth clustering.

We thus identify instances for which the k-means objective is meaningful and so, local search is a relevant heuristic.

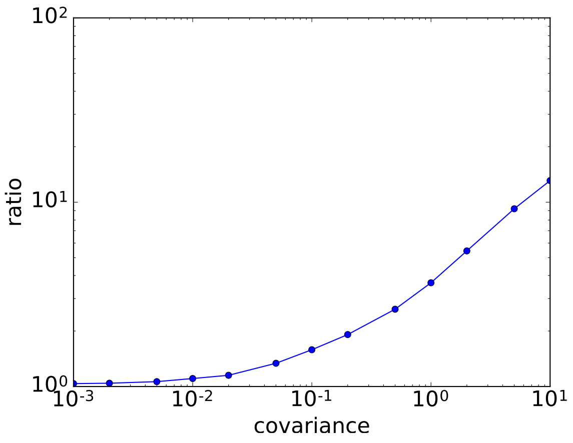
Definition 6.1. *We say that a variance $\hat{\sigma}$ is* relevant *if, for the k-means instances generated with variance $\hat{\sigma}$ the ratio between the average value of the ground truth clustering and the optimal clustering is less than 1.05.*

We summarize in Table 6.2 the relevant variances observed.

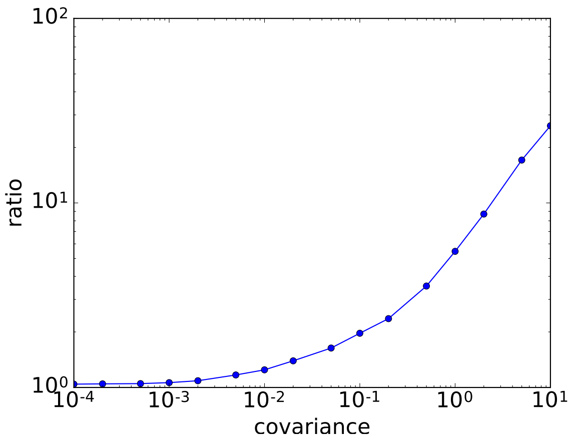
Local search is a suitable approach for the instances generated by a relevant variance only. In Chapter 4, we showed that local search has strong performance guarantees for β-distribution-stable instances (*i.e.,* returns a $(1 + \varepsilon)$ approximation in time $n^{O(\varepsilon^{-3}\beta^{-1})}$). We



(a) $k = 5$, $d = 2$.



(b) $k = 50$, $d = 2$.



(c) $k = 100$, $d = 2$.

Figure 6.1: The ratio of the average k-means cost induced by the means over the average optimal cost vs the variance for 2-dimensional instances generated from a mixture of k Gaussians ($k \in \{5, 50, 100\}$). We observe that the k-means objective becomes "relevant" (*i.e.,* is less than 1.05 times the optimal value) for finding the clustering induced by Gaussians when the variance is less than 0.1 for $k = 5$, less than 0.02 when $k = 50$, and less than 0.0005 when $k = 100$.

thus turn to the β-distribution-stability condition and ask whether the instances generated from a relevant variance satisfy this condition for constant values of β. We remark that β can take arbitrarily small values.

We thus identify *relevant* variances (see Table 6.2) for each pair k, d, such that optimizing the k-means objective in a d-dimensional instances generated from a relevant variance corresponds to finding the underlying clusters.

Values of k / Number of dimensions	5	50	100
2	< 0.05	< 0.002	< 0.0005
10	< 15	< 1	< 0.5
50	< 1000000.0	< 100	< 7

Table 6.2: Relevant variances for $k \in \{5, 50, 100\}$ and $d \in \{2, 10, 50\}$.

On stability conditions. We now study the β-distribution-stability condition for random instances generated from a mixture of k Gaussians. The results are depicted in Figures 6.4 and 6.3.

We observe that for random instances that are not generated from a relevant variance, the instances are β-distribution-stable for very small values of β (*e.g.,* $\beta < 1e - 07$). We also make the following observation.

Observation 6.3. *Instances generated using relevant variances satisfy the β-distribution-stability condition for $\beta > 0.001$.*

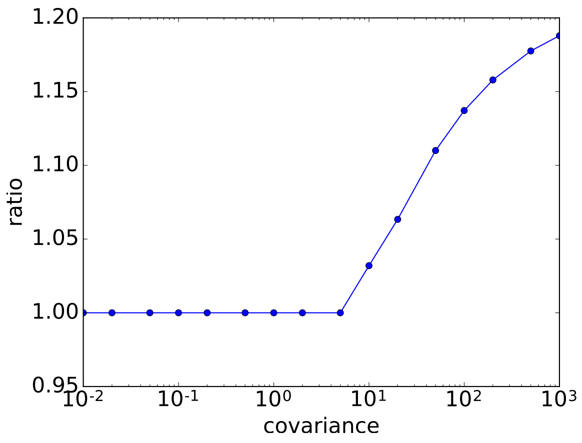
6.2.3 Experimental Analysis of Local Search on Mixtures of Gaussians

In this section we present the results of our experiments on local search for inputs consisting of 1000 points drawn from a mixture of k Gaussians ($1000/k$ points drawn from each Gaussian) in d-dimensional Euclidean space, for $k \in \{5, 50, 100\}$ and $d \in \{2, 10, 50\}$. We focus on instances generated from relevant variances. 100 instances were created for each choice of the parameters.

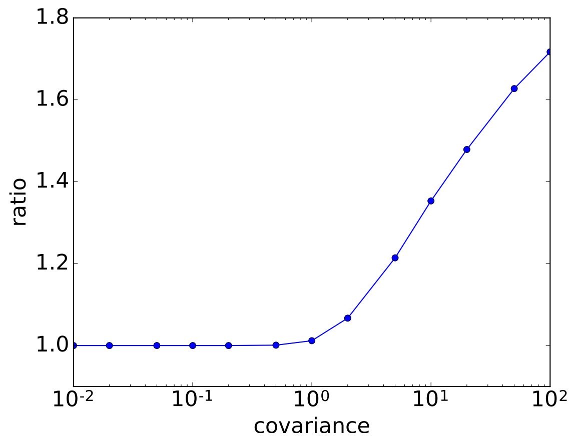
The study of Kanungo et al [112] showed that the drawback of local search is its rate of convergence to the local optimum. We follow the study of [112]: we compare the value of the solution output by Algorithm 2 to the solution output by Lloyd's algorithm. We perform 2000000 iterations of the main loop of Lloyd's algorithm for $d = 2$, 200000 iterations for $d = 10$ and 20000 iterations for $d = 50$. The computation time induced by this number of iterations is depicted in Figure 6.3. We allow the same amount of computation time to Algorithm 2. The corresponding number of iterations of the main loop of Algorithm 2 are presented in Table 6.3. We observe that for $k = 5$, it is enough time for Algorithm 2 to reach a local optimum. Otherwise, none of the two algorithms reaches a local optimum in the amount of computation time we allowed.

The results are presented in Figures 6.6, and 6.5. We obtain similar results for $d = 50$. We make the following observations.

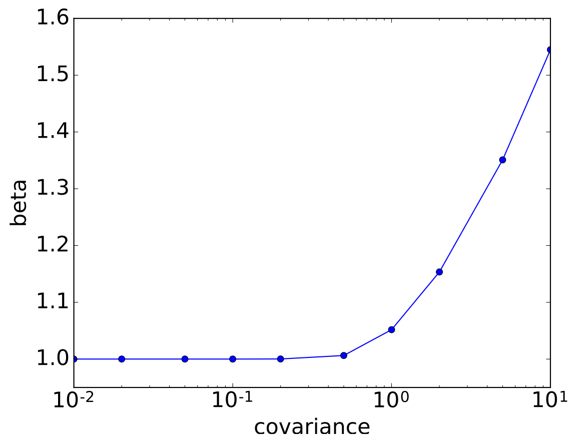
Observation 6.4. *The approximation ratio of local search is a monotonic increasing function of the variance used to generate the instance.*



(a) $k = 5$, $d = 10$.



(b) $k = 50$, $d = 10$.

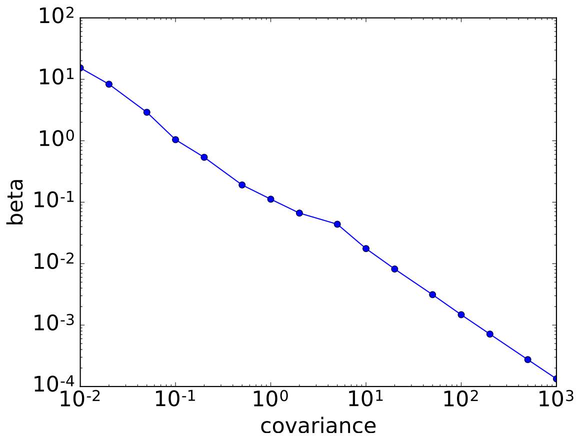


(c) $k = 100$, $d = 10$.

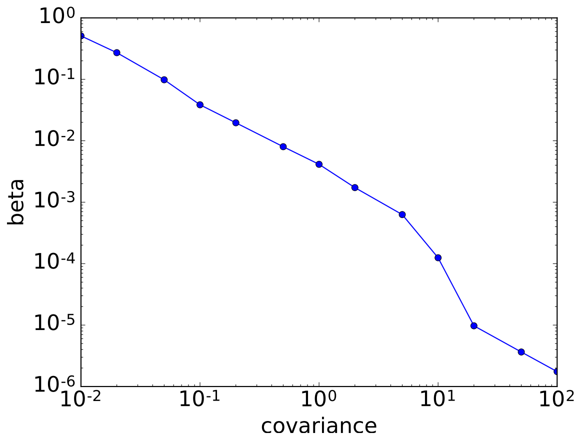
Figure 6.2: The ratio of the average k-means cost induced by the means over the average optimal cost vs the variance for 10-dimensional instances generated from a mixture of k Gaussians ($k \in \{5, 50, 100\}$). We observe that the k-means objective becomes "relevant" (*i.e.,* is less than 1.05 times the optimal value) for finding the clustering induced by Gaussians when the variance is less than 0.1 for $k = 5$, less than 0.02 when $k = 50$, and less than 0.0005 when $k = 100$.

In other words, we observe that the smaller the variance, the better the approximation of local search is.

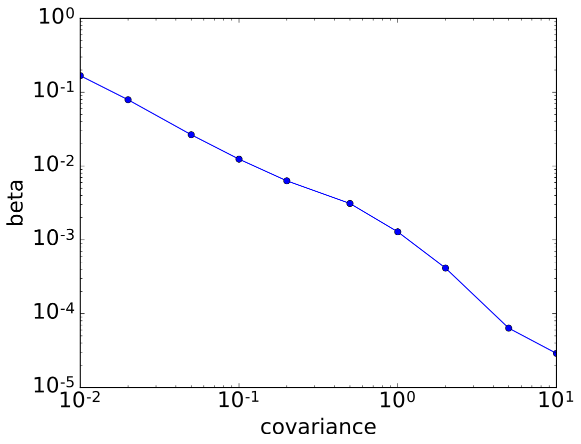
More formally, define $\text{cost}_{\text{Lloyd}}$ to be the average cost of a solution returned by Lloyd's algorithm and cost_{LS} the cost of a solution returned by Algorithm 2. On average, $\text{cost}_{\text{Lloyd}}$'s algorithm returns solutions of cost less than than Algorithm 2 for high values of σ but often end up in a bad local optimum for small values of σ and high values of k. On the



(a) $k = 5$, $d = 10$.



(b) $k = 50$, $d = 10$.



(c) $k = 100$, $d = 10$.

Figure 6.3: The average minimum value of β for which the instance is β-distribution-stable vs the variance for 10-dimensional instances generated from a mixture of k Gaussians ($k \in \{5, 50, 100\}$). We observe that for relevant variances, the value of β is greater than 0.001.

comparison with Lloyd's algorithm, we make the following observations.

Observation 6.5. *On average, for $k = \{50, 100\}$, and $d = 2$,*

- *If $\sigma_{k,d} > \sigma \geqslant 0.001 * \sigma_{k,d}$*
$$cost_{Lloyd} \leqslant cost_{LS} \leqslant 2.5 cost_{Lloyd}.$$
- *If $\sigma < 0.001 * \sigma_{k,d}$*
$$2cost_{LS} \leqslant cost_{Lloyd} \leqslant 20 cost_{LS}.$$

k	Average Comput. time (in sec.)
$k = 5$	2.82
$k = 50$	17.43
$k = 100$	35.80

k	Average Number of Iterations
$k = 5$	9.59
$k = 50$	12.45
$k = 100$	9.18

Table 6.3: On the left, the computation time corresponding to the 2000000 iterations of the main loop of Lloyd's algorithm for 2-dimensional inputs. On the right, the number of iterations of the main loop of Algorithm 2 for the same computation time than Lloyd's algorithm (for 2-dimensional input).

For $d = 10$, we observe the following.

Observation 6.6. *On average, for $k = \{50, 100\}$, and $d = 10$,*

- *If $\sigma_{k,d} > \sigma \geqslant 0.00005\sigma_{k,d}$*

$$\mathit{cost}_{\mathit{Lloyd}} \leqslant \mathit{cost}_{\mathit{LS}} \leqslant 2.5\mathit{cost}_{\mathit{Lloyd}}.$$

- *If $\sigma < 0.00005 * \sigma_{k,d}$*

$$5\mathit{cost}_{\mathit{LS}} \leqslant \mathit{cost}_{\mathit{Lloyd}} \leqslant 10000\mathit{cost}_{\mathit{LS}}.$$

We also observe a similar phenomenon, with slightly different bounds, for $d = 10$ and $d = 50$.

6.3 Interpretation

We now provide possible interpretations of our observations. Combining Observations 6.1, 6.2, and 6.3 we make the following conjecture.

Conjecture 6.1. *For real-world and random instances, if any nearly-optimal solution to the k-means objective induces a good classification of the data (i.e., more than 75% of the points are correctly classified), then the instance satisfies the β-distribution-stability for $\beta > 0.05$.*

Assuming this conjecture and combining it with the following theorem proven in Chapter 4 we have that local search computes the ground truth clustering in those cases.

Theorem 1.5. *There exists a constant c such that the following holds. Let $\beta > 0$ and $\varepsilon < 1/2$. For any β-stable instance, the solution output by local search with parameter $s = c\varepsilon^{-3}\beta^{-1}$ (Algorithm 2) has cost at most $(1 + \varepsilon)OPT$.*

The global message of this chapter is the following.

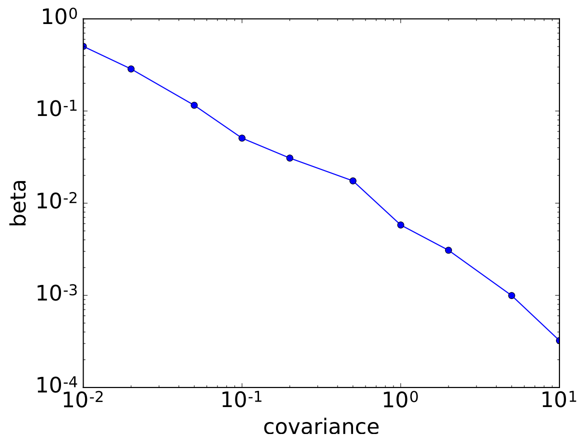
If the use of the k-means objective is appropriate for finding the underlying clustering, then local search is a good heuristic to use.

We remark that this Conjecture 6.1 is formally provable or disprovable for the random input cases. More concretely, we ask whether the conjecture holds for inputs drawn from a mixture of Gaussians (with variance σ) and β being some function of σ, k, and n.

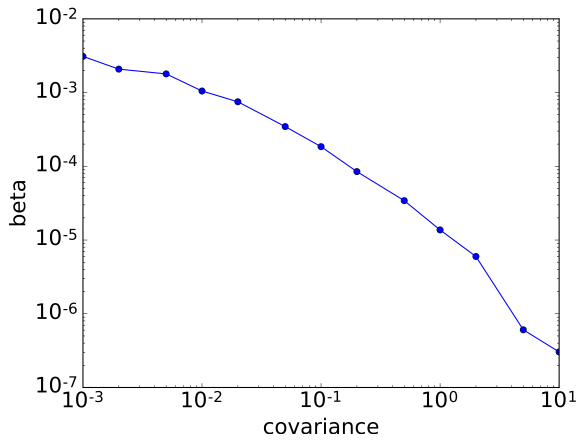
On Lloyd's algorithm. Finally, Observation 6.5 states that Lloyd's algorithm is slightly better than Algorithm 2 for high values of σ but can be very bad for small values of σ and high values of k. We propose the the following explanation to this surprising behavior. When the variance is small, the cost of not opening a center close to a mean is very expensive. Thus, local search quickly locates a center near each mean since this significantly improves the cost of the solution. However, if k is large, Lloyd's algorithm might still merge two clusters into a single cluster. Since there is no objective function to *guide* the algorithm, Lloyd's algorithm might not end up assigning a center close to each mean. This leads Lloyd's algorithm to find poor solutions in those cases.

Practical vs theoretical bounds. Observation 6.4 states that, for instances generated from a mixture of Gaussians, the approximation ratio of local search with $s = 1$ is a monotonic increasing function of the variance. The approximation ratio we observe after a few iterations of the main loop is less than 4. From Observation 6.3, we have that the instances considered are β-distribution-stable for $0.001 < \beta < 1$. Thus, the approximation ratio is less than the theoretical bounds we proved in Chapter 4 $(1 + \varepsilon)$-approximation for $s = \Theta(\varepsilon^{-3}\beta^{-1})$). We thus make the following conjecture.

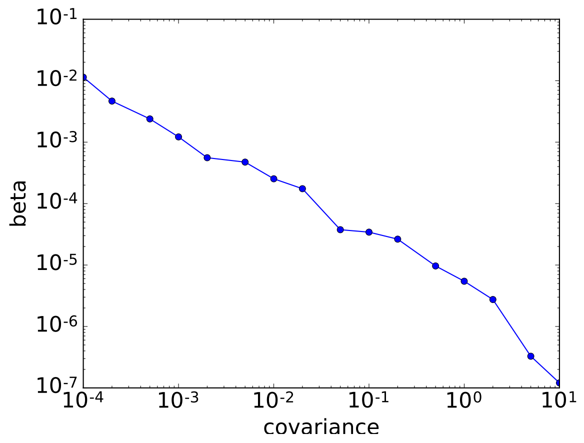
Conjecture 6.2. *The analysis of local search for β-distribution-stable or random instances can be tightened.*



(a) The average minimum value of β for which the instances is β-distribution-stable vs the variance for 2-dimensional instances generated from a mixture of 5 Gaussians.

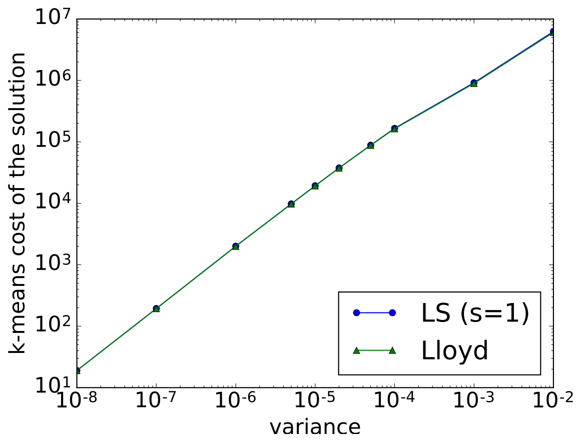


(b) The average of the minimum value of β for which the instances is β-distribution-stable vs the variance for 2-dimensional instances generated from a mixture of 50 Gaussians.

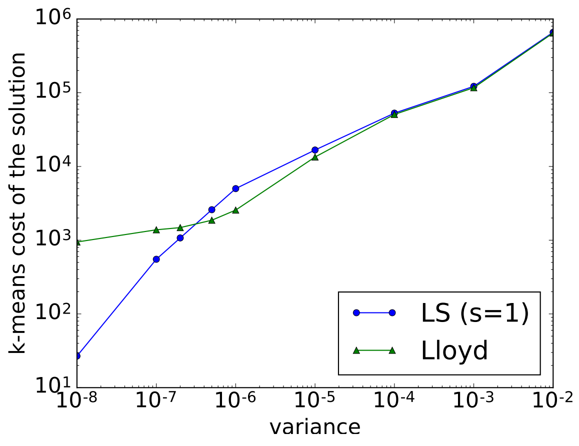


(c) The average minimum value of β for which the instance is β-distribution-stable vs the variance for 2-dimensional instances generated from a mixture of 100 Gaussians.

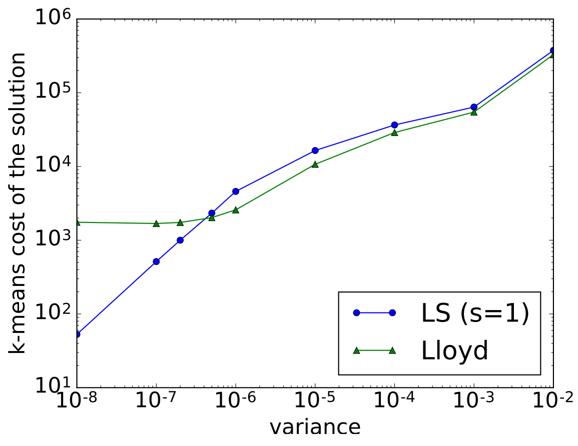
Figure 6.4: The average minimum value of β for which the instance is β-distribution-stable vs the variance for 2-dimensional instances generated from a mixture of k Gaussians ($k \in \{5, 50, 100\}$). We observe that for relevant variances, the value of β is greater than 0.001.



(a) $k = 5$, $d = 2$.

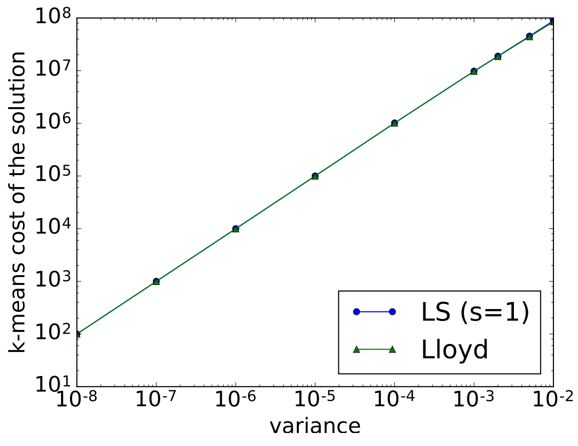


(b) $k = 50$, $d = 2$.

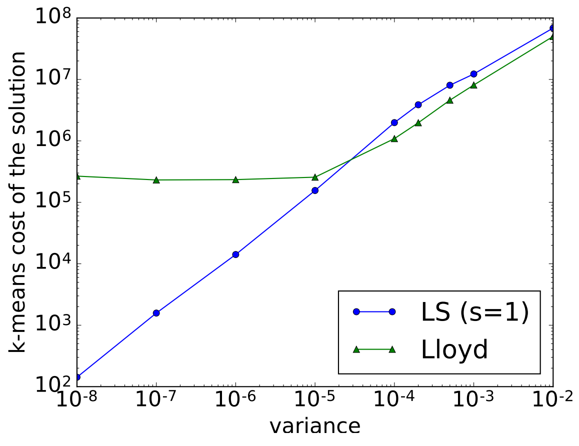


(c) $k = 100$, $d = 2$.

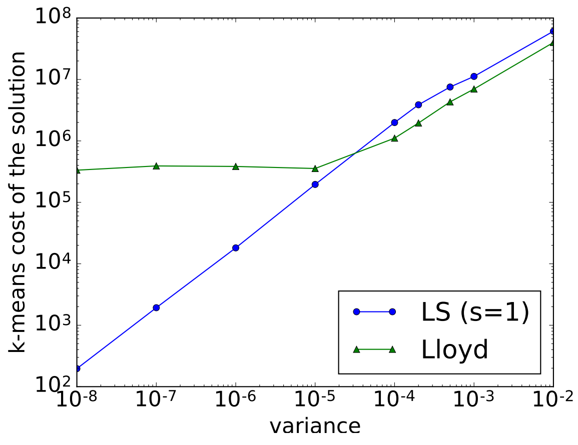
Figure 6.5: The average approximation ratio for Algorithm 2 (denoted LS) with $s = 1$ and Lloyd's algorithm for 2-dimensional instances generated from a mixture of 5, 10, and 100 Gaussians.



(a) $k = 5$, $d = 10$.



(b) $k = 50$, $d = 10$.



(c) $k = 100$, $d = 10$.

Figure 6.6: The average approximation ratio for Algorithm 2 (denoted LS) with $s = 1$ Lloyd's algorithm for 10-dimensional instances generated from a mixture of 5, 10, and 100 Gaussians.

Part II

Separation and Network Design

CHAPTER 7

Introduction to Network Design

Network design problems arise in a broad range of contexts, from connecting people at the surface of the earth to connecting electronic components on a printed circuit board. They often entail hard and specific constraints that ensure some level of quality-of-service for the application. In this thesis, we see a network design problem as the problem of building a graph satisfying some specified constraints.

Among the numerous network design problems, some of them are of more general interest because of their tremendous number of applications or because their study has led to the development of new algorithmic techniques over the years. This is for example the case for the traveling salesman problem which received a considerable amount of attention since at least the 1940s and the work of Tutte [174] and Dantzig et al. [64] and until very recently (see *e.g.,* [170]). This problem has led to (1) new approaches to design approximation algorithms (see *e.g.,* the results of Arora [9] and Mitchell [144] in the Euclidean setting) and (2) the design of benchmarks of real-world and random instances to compare the different heuristics used in practice (see *e.g.,* [109]).

From a theoretical perspective, network design problems are often hard to approximate when dealing with general instances. Thus, people have looked at more restricted instances. For example, metrics induced by a graph embeddable on a surface of small genus or points lying in a Euclidean space of small dimension represent a significant fraction of the instances addressed in practice[2].

In this part, we study network connectivity type problems on d-dimensional Euclidean space and graphs embedded on a surface of small genus. We consider the traveling salesman problem (TSP) and the Steiner tree problem (STP) in the Euclidean setting and the connected dominating set problem together with several related problems in the context of planar graphs and more generally graphs embedded on a surface of small genus.

[2]There is also a variety of practical instances that cannot be modeled by a graph embeddable on a surface of small genus, *e.g.,* instances stemming from computer networks.

Random TSP and Steiner tree. For the traveling salesman problem, the gap between theory and practice is significant. Indeed, the performances of several heuristics used in practice on real-world and random instances have been thoroughly analyzed during the TSP challenge organized by DIMACS [70] 15 years ago. However, quite surprisingly, the PTAS designed by Arora et al. [12] and Mitchell [144] is absent from the challenge[1]. Quoting Johnson and McGeoch [109]:

> "These heuristics, despite their impressive theoretical guarantees, have significant drawbacks compared to the competition we shall be describing. Because of the perturbation of the instances that they initially perform, the versions of the heuristics guaranteeing $1 + \varepsilon$ worst-case ratios are likely to be off by a significant fraction of e even in the average case. Thus, to be competitive with heuristics that typically get within 1 or 2 percent of optimum in practice, one probably must choose $\varepsilon < 0.05$. This is likely to make the running times prohibitive, given the large constant factor overheads involved and the fact that the running times are exponential in $1/\varepsilon$. It would be interesting to verify that this is indeed the case, but as of this date we know of no attempt at a serious implementation of any of the schemes."

The very strong performances achieved by the heuristics, like 2-OPT, 3-OPT, Lin-Kernighan or nearest neighbors, suggest a theoretical analysis of local search approaches.

Connectivity and Domination Problem in Planar Graphs. There is a long history of research on approximation algorithms for combinatorial optimization problems in graphs embedded on a surface of small genus. Three approaches yield most polynomial-time approximation schemes known for planar graphs: Baker's method [24], approximation via bidimensionality (Demaine and Hajiaghayi) [66], and a framework of Klein [117, 118], sometimes called *brick decomposition*. However, those approaches seem hard to extend to some important network design problems.

For example, in the case of the weighted connected dominating set problem, the non-local structure of the problem together with the domination requirement and the vertex weights make it hard to handle with the standard techniques mentioned above (this is further discussed in Chapter 9). We will derive a new separator whose properties allow us to develop a new algorithmic framework for various weighted network design problems in planar graphs.

[1]Despite the fact that a substantial parts of the instances consisted of points lying in a 2-dimensional Euclidean space.

Contributions and Techniques

Tight Result for Random TSP in Euclidean Space. We consider the following standard local search algorithm for TSP and the Steiner tree problem, Algorithm 10. This is one of the local search algorithms that were studied during the DIMACS challenge.

Algorithm 10 Local search for TSP or the Steiner tree problem

```
1: Input: A set $\mathcal{C}$ of points in the a $\mathbb{R}^d$.
2: $S \leftarrow$ Arbitrary feasible solution[1] (set of edges for TSP, set of Steiner points for the
   Steiner tree problem).
3: while $\exists\ S'$ s.t. $S'$ is feasible and $|S - S'| + |S' - S| \leqslant s$ and cost($S'$) $\leqslant (1 - 1/n)$
   cost($S$)
4: do
5:     $S \leftarrow S'$
6: Output: $S$
```

For TSP, we start with the worst-case scenario. We give a family of examples where for any constant value of s, there exists an example such that the solution output by Algorithm 10 has cost more than $(2 - \varepsilon)OPT$. This bounds is very far from what is observed in practice. Thus, we try to explain the success of the local search approach by analyzing its performance on random inputs. We assume that the input points are random uniform in $[0, 1]^2$. We show the following theorem.

Theorem 7.1 (Random Euclidean Instances for TSP and Steiner tree – Chapter 8)**.** *Consider a set of points chosen independently and uniformly in $[0, 1]^d$. There exists a constant c such that Algorithm 10 with $s = c/\varepsilon^{d/(d-1)}$ produces:*

- *In the case of the Traveling Salesman problem, a tour of length at most $(1 + \varepsilon)$OPT;*
- *In the case of the Steiner Tree problem, a tree whose length is at most $(1 + \varepsilon)$OPT;*

with probability 1.

The techniques are very intuitive. We rely on the existence of a recursive dissection of the space, based on cheap separators by Karp [114]. This leads to an upper bound on the length of the tour output by the algorithm of OPT + $\varepsilon n^{(d-1)/d}$ (Theorem 8.1). Classical results on random TSP and the Steiner tree problem show that in the random scenario OPT$/n^{(d-1)/d} \rightarrow \beta$ as n tends to infinity for some absolute constant β with probability 1. Theorem 7.1 follows directly by combining those observations.

A New Framework for Graphs of Bounded Genus: Ubiquity. We present a new approach that yields approximation schemes for some weighted, non-local problems for which no PTAS was previously known in graphs of bounded genus: connected dominating set, connected vertex cover, feedback vertex set, tree cover, tour cover, spanning tree maximizing the weight of leaves, and other problems.

A solution to any of those problems consists either of a subgraph or of a set of edges and/or vertices. In the latter case, the solution can be equivalently expressed by the subgraph induced by that set. We proceed at a more abstract level and define a family of problems that can be solved by our technique.

Definition 7.2. *Let t be an integer. We say a graph problem is* t-ubiquitous *(or simply* ubiquitous*) if, for every input graph G and every feasible solution S, S is connected and G/S has treewidth at most t.*

Let g be a positive integer, considered a constant for the purpose of stating running times in our main theorem, which is as follows:

Theorem 7.3 (Ubiquity Framework – Chapter 9, Section 9.1)**.** *Let P be a minimization problem on edge- and vertex-weighted graphs with genus at most g, such that contracting an edge of an input graph can only reduce the optimal value, and there exist*

1. An $O(1)$-approximation: *For some constant t, P is t-ubiquitous, and there is a polynomial-time algorithm that, given an input G for P, outputs an α-approximation*[1] *for P, for some constant α.*

2. A Dynamic program: *There is a $2^{O(b)}$poly(n) algorithm to find an optimal or $1+\varepsilon$-approximate solution to instances of P with branchwidth at most b.*

3. A Lifting: *There is a constant β and a polynomial-time algorithm that, given a graph G and a subgraph K, and given a solution S to problem P for input G/K, outputs a solution for G of weight at most $w(S)+\beta\cdot w(K)$.*

Then there is a polynomial-time approximation scheme for P.

This theorem is a consequence of a slightly more general version in which Condition 1 is replaced with:
Condition $1'$**.** There are constants $\alpha \geqslant 1$ and t and a polynomial-time algorithm that, given an input G for P, outputs a connected subgraph B such that B has weight at most α times the optimal value for G, and G/B has treewidth at most t. A bound on the length of B with respect to OPT is needed, not B being a solution to the problem.

The key ingredient to prove Theorem 7.3 is the following "well-structured" separator theorem for graphs of bounded genus. Here the *mass* ensures that the separator has some balance property.

[1]This is a variation to what Demaine and Hajiaghayi call a "backbone" in the bidimensional approach [66].

Theorem 7.4 (Separator Theorem – Chapter 8, Section 9.1)**.** *Let b and k be integers. Let $G = (V, E)$ be a planar graph, and H denote a connected subgraph of G such that G/H has branchwidth at most $b-1$. Let w_V and w_E be functions assigning nonnegative weights to, respectively, the vertices and the edges of G.*

Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be the union of G and its face-vertex incidence graph. Suppose the vertices and faces of $\tilde{G}$ have been assigned nonnegative masses, summing to M, with no face or vertex having mass more than $M/2$.

Then $\tilde{G}$ has a cycle C, which may repeat vertices and edges but does not cross itself and has no spurs, such that:

- *C is a balanced separator: the mass of the vertices of G strictly inside (resp., strictly outside) C is at most $3M/4$;*
- *C is mostly a light piece of H: there exists a set V' of $O(bk)$ vertices of $\tilde{G}$, and a set E' of $O(bk)$ edges of $\tilde{G}$, such that $C \backslash (E' \cup V')$ is a subgraph of H and $w_V(V(C)\backslash V') + w_E(E(C)\backslash E') \leqslant W/k$, where W is the total weight of the vertices and edges in H.*

Moreover, C, V', and E' can be computed in time $O(k^2 n)$.

The proof of Theorem 7.3 in a nutshell is as follows. Applying this theorem recursively on the graph yields a subgraph S of small edge- and vertex-weight. Then, we show that contracting this subgraph results in a new graph G' of small treewidth in which dynamic programming can be used to find a solution. Finally, since S is of small weight, extending the solution obtained for G' by adding S yields a solution of small cost for G. Finally, we show how to handle the case of graphs of bounded genus by finding a *planarizing* subgraph to finally apply Theorem 7.4.

More concretely, the results obtain by thanks to our new framework are displayed in Table 7.1.

State-of-the-Art

TSP and Steiner Tree. The TSP problem has a long history. A folklore 3/2-approximation algorithm has been known from a long time (formally stated by Christofides [58] in 1976 as a step of his algorithm).

In this thesis, we focus on Euclidean metrics. TSP in Euclidean plane has also been extensively studied, including work on local search [62, 131, 132]. In the 90s, Arora [9] and Mitchell [144], independently gave the first approximation schemes for TSP and the Steiner tree problem in Euclidean space of fixed dimension. Their algorithms rely on a hierarchical dissection technique and so, on the existence of well-structured separators. This work was subsequently improved [10, 156]. However, the idea of using separators

Table 7.1: Summary of our results. All the problems are APX-hard in general graphs and the approximation ratios of the Weighted Dominating Set, the Vertex-Weighted Connected Vertex Cover and the Vertex-Weighted Connected Dominating Set problems are $\Omega(\log(n))$ for general graphs assuming P $\neq$ NP. All the problems are NP-hard in planar graphs. Previous to our work, polynomial-time approximation schemes were known [66] for the unweighted versions of these problems in planar graphs, and "almost-PTASs" were known for bounded-genus graphs. For each of the weighted versions, the best approximation known before our work was the approximation for general graphs. We obtain PTASs for bounded-genus weighted graphs, except for feedback vertex set, where the algorithm is restricted to weighted planar graphs.

Problem	General Weights	
	Prev. (for general graphs)	**New**
(Edge-weights) Tree Cover	2 [149]	$1+\varepsilon$
(Edge-weights) Tour Cover	3 [124]	$1+\varepsilon$
(Vertex-weights) Connected Dominating Set	$O(\log(n))$ [96]	$1+\varepsilon$
(Vertex-weights) Maximum Leaf Spanning Tree	$O(\log(n))$ [96]	$1+\varepsilon$
(Vertex-weights) Connected Vertex Cover	$O(\log(n))$ [90]	$1+\varepsilon$
(Vertex-weights) Feedback Vertex Set	2 [22]	$1+\varepsilon$

to apply dynamic programming date back to Karp [114] in 1977. Indeed, Karp gave a simple algorithm that computes a near-optimal tour when points are drawn from a random distribution.

The relationship between separator and local search was already around in the 90s. Arora noted the relation between hierarchical dissections and local search, observing:

> *Local-exchange algorithms for the TSP work by identifying possible edge exchanges in the current tour that lower the cost [...].Our dynamic programming algorithm can be restated as a slightly more inefficient backtracking [...]. Thus it resembles k-OPT for $k = O(c)$, except that cost-increasing exchanges have to be allowed in order to undo bad guesses. Maybe it is closer in spirit to more ad-hoc heuristics such as genetic algorithms, which do allow cost-increasing exchanges.*

In fact, even with neighborhoods of size $f(\varepsilon)$, even in the Euclidean plane, local search for TSP can get stuck in a local optimum whose value is far from the global optimum (See Figure 8.2). However, in the case of random inputs the intuition is correct. Local search algorithms have been widely studied for TSP, but mostly for either a local neighborhood limited to size of 2 or 3 (the so called 2-OPT or 3-OPT algorithms), or for the general metric case. Those studies lead to proofs of constant factor approximations, see [51, 108, 132, 141, 166]. In particular, in [51], it is proved (by example) that for Euclidean TSP

2-OPT cannot be a constant-factor approximation in the worst case. For the Steiner Tree problem, the best approximation algorithm up to 2010 was a constant factor approximation due to Robins and Zelikovsky and was by local search [164].

Network Design in Graphs Embedded of Bounded Genus. As mentioned earlier, we distinguish three main approaches to obtain approximation schemes on planar graphs: Baker's method [24], approximation via bidimensionality (Demaine and Hajiaghayi) [66], and a framework of Klein [117, 118]. Note that the two former methods have been generalized to graphs excluding a fixed minor. Some recent results show that the later can be extended to graphs of bounded genus [42].

Each of these methods has its limitations. Baker's method only addresses problems that are local in character, e.g., min-weight vertex cover and dominating set. Bidimensionality is only defined for problems without weights, and this approach only yields approximation schemes for such problems, though it does address very non-local problems such as feedback vertex set and connected dominating set. The framework of [117, 118] has been used for a variety of weighted, non-local problems; it has yielded, for example, a linear-time approximation scheme for traveling salesman, near-linear-time approximation schemes for Steiner tree and generalizations, and polynomial-time approximation schemes for cut problems such as multiway cut and graph bisection. However, for each problem addressed, it requires a kind of sparsification that approximately preserves optimality; for some problems, and in particular the connected dominating set problem, obtaining such a sparsification seems difficult.

Corresponding Publications

[1] V. Cohen-Addad, É. Colin de Verdière, P. N. Klein, C. Mathieu, D. Meierfrankenfeld, *Approximating connectivity domination in weighted bounded-genus graphs*, Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016.

[2] V. Cohen-Addad and C. Mathieu, *Effectiveness of Local Search for Geometric Optimization*, Proceedings of the 31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands, 2015.

CHAPTER 8

Local Search for Random Traveling Salesman and Steiner Tree

This chapter is dedicated to the proof of Theorem 7.1.

Theorem 7.1. *Consider a set of points chosen independently and uniformly in* $[0,1]^d$. *There exists a constant* c *such that Algorithm 10 with* $s = c/\varepsilon^{d/(d-1)}$ *produces:*

- *In the case of the Traveling Salesman problem, a tour of length at most* $(1+\varepsilon)$*OPT.*
- *In the case of the Steiner Tree problem, a tree whose length is at most* $(1+\varepsilon)$*OPT.*

We recall the local search algorithm we are considering in this chapter.

Algorithm 10 Local search for TSP or the Steiner tree problem

```
1: Input: A set $\mathcal{C}$ of points in the a $\mathbb{R}^d$.
2: $S \leftarrow$ Arbitrary feasible solution[2] (set of edges for TSP, set of Steiner points for the
   Steiner tree problem).
3: while $\exists$ $S'$ s.t. $S'$ is feasible and $|S - S'| + |S' - S| \leqslant s$ and cost($S'$) $\leqslant (1 - 1/n)$
   cost($S$)
4: do
5:     $S \leftarrow S'$
6: Output: $S$
```

As mentioned in the introduction, the proof of Theorem 7.1 relies on a worst-case analysis of the output of Algorithm 10 that is summarized by the following theorem.

Theorem 8.1. *Consider an arbitrary set of points in* $[0,1]^d$. *There exists a constant* c *such that Algorithm 10 with* $s = c/\varepsilon^d$ *produces:*

- *In the case of the Traveling Salesman problem, a tour whose length is at most* $(1+\varepsilon)T_{OPT} + \varepsilon n^{(d-1)/d}$*, where* T_{OPT} *is the length of the optimal solution.*
- *In the case of the Steiner Tree problem, a tree whose length is at most* $(1+\varepsilon)T_{OPT} + \varepsilon n^{(d-1)/d}$*, where* T_{OPT} *is the length of the optimal solution.*

We model a random distribution of points in a region $\mathcal{P}$ of the d-dimensional Euclidean space by a d-dimensional Poisson distribution $\Pi_n(\mathcal{P})$. This is a standard model that has been extensively studied until recently (see *e.g.,* [88]) since Beardwood et al. [31]. The distribution $\Pi_n(\mathcal{P})$ is determined by the following assumptions:

1. the numbers of points occurring in two or more disjoint sub-regions are distributed independently of each other;
2. the expected number of points in a region A is $nv(A)$ where $v(A)$ is the area of A; and
3. as $v(A)$ tends to zero, the probability of more than one point occurring in A tends to zero faster than $v(A)$.

From these assumptions it follows that Pr[A contains exactly m points] $= e^{-\lambda}\lambda^m/m!$, where $\lambda = nv(A)$.

The following result is known.

Theorem 8.2. *[31] Let* $\mathcal{P}$ *be a set of* n *points distributed according to a* d*-dimensional Poisson distribution* $\Pi_n(\mathcal{P})$ *in* $[0,1]^d$ *and let TSP*$_n(\mathcal{P})$ *and ST*$_n(\mathcal{P})$ *be the random variables that denote the value of the optimal solution to TSP and the value of the optimal solution to STP where the inputs are the points in* $\mathcal{P}$ *respectively. There exists positive constants* β_{TSP}, β_{ST} *(independent of* $\mathcal{P}$*) such that TSP*$_n(\mathcal{P})/n^{(d-1)/d} \to \beta_{TSP}$ *and ST*$_n(\mathcal{P})/n^{(d-1)/d} \to \beta_{ST}$ *with probability 1.*

Assuming Theorem 8.1 Theorem 7.1 follows easily.

Proof of Theorem 7.1. Consider the case of TSP. Let L be the tour produced by Algorithm 10 and T_{OPT} be the optimal tour. Combining, Theorems 8.2 and 8.1 implies that there exists a constant β such that

$$\text{cost}(L) \leqslant \text{cost}(T_{\text{OPT}}) + \varepsilon \cdot n^{(d-1)/d} \leqslant (1+\beta' \cdot \varepsilon)\text{cost}(T_{\text{OPT}}),$$

with probability 1. The exact same reasoning applies to prove the Steiner Tree case. Theorem 7.1 follows by rescaling ε. □

The rest of the chapter is dedicated to the proof of Theorem 8.1. For ease of exposition, we give the proof for the case of the plane and show how to handle d-dimensional Euclidean space for fixed d. To this aim, we define a recursive dissection of the unit square according

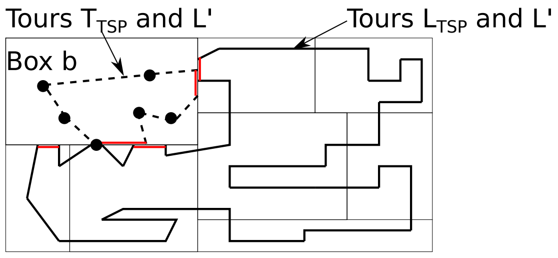


Figure 8.1: The solid black segments form the tour L_{TSP} outside b. The dotted line segments are the tour T_{TSP} inside b. The red segments are the one needed to connect the two tours.

to a set of points $\mathcal{P}$. At each step we cut the longer side of each rectangle produced by the previous step in such a way that each of the two parts contains half the points of $\mathcal{P}$ that lie in the rectangle. The process stops when each rectangle contains $\Theta(1/\varepsilon^2)$ points of $\mathcal{P}$. We now consider the final rectangles and we refer to them as *boxes*. Let $\mathcal{B}$ be the set of boxes.

Lemma 8.3 ([114]). $\sum\limits_{b\in\mathcal{B}} |\partial b| = O(\varepsilon\sqrt{n})$, *where* $|\partial b|$ *is the perimeter of box* b.

For any set of segments S and box b and for each segment s, let s_b be the part of s that lies inside b. We define $\text{in}(S, b) := \{s_b \mid s \in S \text{ and } s \text{ has at least one endpoint in } b\}$ and $\text{Cross}(S, b) := \{s_b \mid s \in S \text{ and } s \text{ has no endpoint in } b\}$. Moreover we define $\text{out}(S, b) := \{s_{b'} \mid s \in S \text{ and } b \neq b'\}$. Additionally, let $S(b) = \sum_{s\in S} \text{length}(s_b)$.

We can now prove the two following structural Lemmas. See Fig. 8.1 for an illustration of the proof.

Lemma 8.4. *Let* L_{ST} *be a locally optimal solution to the Steiner Tree problem and let* T_{ST} *be any Steiner Tree. Let* $\mathcal{B}$ *be a set of boxes produced by a dissection of* $\mathcal{P} \cup L_{ST} \cup T_{ST}$. *Using the same notation for a set of segments and their total length, we then have for any box* $b \in \mathcal{B}$

$$(1 - O(\varepsilon^2))L_{ST}(b) \leqslant \textit{in}(T_{ST}, b) + |\partial b| + L_{ST}/n,$$

where $|\partial b|$ *is the perimeter of* b.

Proof. For each box b, the segments of $\text{Cross}(L_{\text{ST}}, b)$ can be distributed into 6 different classes according to which side of b they intersect.

We divide further. Since the segments of a class are pairwise disjoint, there is a natural ordering of the segments inside each class. For each class that contains more than $1/\varepsilon^2$ segments, we partition them into subsets that contain $\Theta(1/\varepsilon^2)$ consecutive segments (in the natural order of the class). We define a sub-box for each subset of each class as follows. Let s and s' be the two extreme segments of the set in the ordering of the class. The sides of the sub-box associated to this subset consists of s and s' and the two shortest paths p, p' along the sides of b that connects the endpoints of s and s'.

Remark that the sum of the lengths of the sides of all the sub-boxes is at most $|\partial b| + O(\varepsilon^2 L_{\text{ST}}(b))$. For each sub-box b_0, let L' be the set of vertices of L_{ST} that are outside b_0, plus the set of vertices of T_{ST} that are inside b_0, plus the set of the intersection points of the edges of L_{ST} and T_{ST} with the sides of b_0. Thus, $L' \leqslant \text{out}(L_{\text{ST}}, b_0) + \text{in}(T_{\text{ST}}, b_0) + |\partial b_0|$. Moreover, we have $|L_{\text{ST}} \triangle L'| = O(1/\varepsilon^2)$ and the local near-optimality argument applies. Namely, we obtain that $(1 - 1/n)L_{\text{ST}} \leqslant L'$, and so

$$-1/n \cdot L_{\text{ST}} + \text{in}(L_{\text{ST}}, b_0) + \text{Cross}(L_{\text{ST}}, b_0) \leqslant \text{in}(T_{\text{ST}}, b_0) + |\partial b_0|.$$

We now sum over all sub-boxes of box b and we obtain

$$L_{\text{ST}}(b) = \text{in}(L_{\text{ST}}, b_0) + \text{Cross}(L_{\text{ST}}, b_0) \leqslant \text{in}(T_{\text{ST}}, b) + |\partial b| + O(\varepsilon^2 L_{\text{ST}}(b)) + L_{\text{ST}}/n.$$

□

Lemma 8.5. *Let L_{TSP} be a locally optimal solution to the Traveling Salesman problem and let T_{TSP} be any tour. Let $\mathcal{B}$ be a set of boxes produced by a dissection of $\mathcal{P}$. Using the same notation for a set of segments and their total length, we then have for any box $b \in \mathcal{B}$*

$$(1 - O(\varepsilon^2))L_{TSP}(b) \leqslant in(T_{TSP}, b) + 3|\partial b|/2 + L_{TSP}/n,$$

where $|\partial b|$ is the perimeter of b.

Proof. We again further divide the boxes into sub-boxes as we did for Lemma 8.4. For each sub-box b_0, we define a tour L' obtained by a traversal of the following Eulerian graph. The graph vertices are $\mathcal{P}$, plus the corners of ∂b_0, plus all points of intersection of L_{TSP} and T_{TSP} with ∂b_0. The edges are the segments of $\text{out}(L_{\text{TSP}}, b_0)$, plus the segments of $\text{in}(T_{\text{TSP}}, b_0)$, plus ∂b_0 (so that the result is connected), plus a minimum length matching of the odd vertices of ∂b_0 (so that the result is Eulerian). Thus, $L' \leqslant \text{out}(L_{\text{TSP}}, b_0) + \text{in}(T_{\text{TSP}}, b_0) + 3|\partial b_0|/2$.

Since the number of edges of L intersecting b_0 is $O(1/\varepsilon^2)$ and the number of edges in $\text{in}(T_{\text{TSP}}, b_0)$ is $O(1/\varepsilon^2)$, we have $|L_{\text{TSP}} \triangle L'| = O(1/\varepsilon^2)$ and the local near-optimality argument applies. Namely, we obtain $(1 - 1/n)L_{\text{TSP}} \leqslant L'$, and so

$$-1/n \cdot L_{\text{TSP}} + \text{in}(L_{\text{TSP}}, b_0) + \text{Cross}(L_{\text{TSP}}, b_0) \leqslant \text{in}(T_{\text{TSP}}, b_0) + 3|\partial b_0|/2.$$

We now sum over all sub-boxes of box b and we obtain

$$L_{\text{TSP}}(b) = \text{in}(L_{\text{TSP}}, b) + \text{Cross}(L_{\text{TSP}}, b) \leqslant \text{in}(T_{\text{TSP}}, b) + 3|\partial b|/2 + O(\varepsilon^2 L_{\text{TSP}}(b)) + L_{\text{TSP}}/n.$$

□

We can now prove Theorem 8.1.

Proof of Theorem 8.1. We first consider the Traveling Salesman case. Let L_{TSP} be a tour produced by Algorithm 10 and T_{TSP} be any tour. Lemma 8.5 implies that for any box b, we have

$$(1 - O(\varepsilon^2))L_{\text{TSP}}(b) \leqslant \text{in}(T_{\text{TSP}}, b) + 3|\partial b|/2 + L_{\text{TSP}}/n.$$

Since there are $O(\varepsilon^2 n)$ boxes in total, by summing over all boxes, we obtain

$$-O(\varepsilon^2 L_{\text{TSP}}) + \sum_{b\in\mathcal{B}} L_{\text{TSP}}(b) = (1 - O(\varepsilon^2))L_{\text{TSP}} \leqslant \sum_{b\in\mathcal{B}} (\text{in}(T_{\text{TSP}}, b) + 3|\partial b|/2)$$
$$\leqslant T_{\text{TSP}} + \frac{3}{2}\sum_{b\in\mathcal{B}} |\partial b|.$$

By Lemma 8.3, $\sum_{b\in\mathcal{B}} |\partial b| = O(\varepsilon\sqrt{n})$ and so,

$$(1 - O(\varepsilon^2)) \cdot L_{\text{TSP}} \leqslant T_{\text{TSP}} + O(\varepsilon\sqrt{n}).$$

To prove the Steiner Tree case, it is sufficient to notice that the total number of vertices in $\mathcal{P} \cup L_{\text{ST}} \cup T_{\text{ST}}$ is at most $3n$. It follows that the total number of boxes is $O(\varepsilon^2 n)$ and by Lemma 8.3, $\sum_{b\in\mathcal{B}} |\partial b| = O(\varepsilon\sqrt{n})$. We apply a reasoning similar to the one for the TSP case to conclude the proof.

$\square$

We observe that Lemmas 8.4 and 8.5 and Theorem 8.1 hold in the worst-case since we do not assume that the points are randomly distributed in the $[0, 1]^2$.

Higher dimensions Karp remarks in [114] that for a d-dimensional Euclidean space, Lemma 8.3 can be generalized to obtain a bound of $O((\varepsilon \cdot n)^{(d-1)/d})$. The proof of Theorem 8.1 for d-dimensional Euclidean space follows directly by rescaling ε.

Tightness of the Analysis Figure 8.2 shows that there exists a set of points such that there is a local optimum whose length is at least $(2 - o(\varepsilon))\text{cost}(\text{OPT})$.

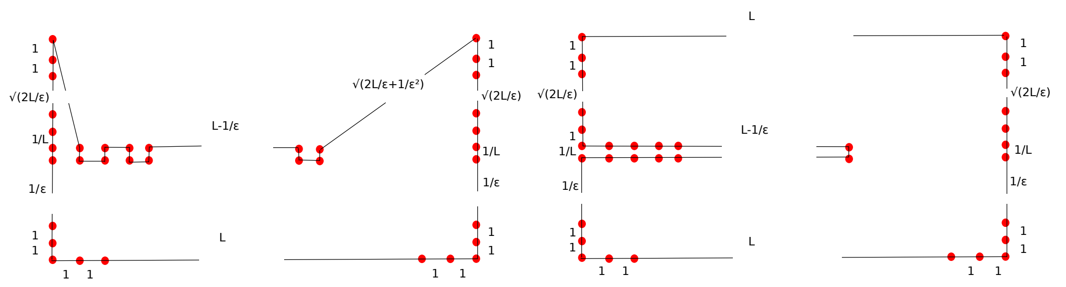


Figure 8.2: The tour on the right is $1/\varepsilon$-locally optimal for any $\varepsilon^{-1} = o(\sqrt{L})$ but is $(2 - O(\varepsilon))$ times longer than the tour on the left.

CHAPTER 9

Computing More Structured Separators

This chapter introduces our new framework: "Ubiquity".

We provide the proof of Theorems 7.3 and 7.4 in Section 9.1. The algorithmic applications are presented in Section 9.2.

Recall that the Ubiquity framework relies on the following definition.

Definition 7.2. *Let t be an integer. We say a graph problem is t-*ubiquitous *(or simply* ubiquitous*) if, for every input graph G and every feasible solution S, S is connected and G/S has treewidth at most t.*

The framework is summarized by the Theorem 7.3 that we recall for completeness.

Theorem 7.3. *Let P be a minimization problem on edge- or vertex-weighted graphs with genus at most g, such that contracting an edge of an input graph can only reduce the optimal value, and there exist*

1. An $O(1)$-approximation: *For some constant t, P is t-ubiquitous, and there is a polynomial-time algorithm that, given an input G for P, outputs an α-approximation*[2] *for P, for some constant α.*

2. A Dynamic program: *There is an $2^{O(b)}$poly(n) algorithm to find an optimal or $1+\varepsilon$-approximate solution to instances of P with branchwidth at most b.*

3. A Lifting: *There is a constant β and a polynomial-time algorithm that, given a graph G and a subgraph K, and given a solution S to problem P for input G/K, outputs a solution for G of weight at most $w(S) + \beta \cdot w(K)$.*

Then there is a polynomial-time approximation scheme for P.

We start by giving a more detailed background together with the main motivation behind this work.

[2]This is a variation to what Demaine and Hajiaghayi call a "backbone" in the bidimensional approach [66].

Background. Several techniques relying on the small-separator property of graphs of bounded genus have been developed through the years.

For the unweighted versions of the problems we consider, the *bidimensionality* framework gives polynomial-time approximation schemes. Bidimensionality applies to a problem only if a solution is necessarily *dense* in the graph; in particular if for grid graphs a solution necessarily uses a constant fraction of the edges. Bidimensionality applies only when every element (vertex/edge) has the same weight. Thus density is measured in terms of the *value* of the optimum.

Similarly, the first step of an algorithm employing the brick decomposition framework of Klein [117, 118] is to thin the graph (using deletions or contractions) so that the total weight of the graph is a small factor times the value of the optimum. Thus again the key is ensuring that the optimum value is a large fraction of the weight of the graph. It remains unknown for many optimization problems whether such a thinning step can be carried out in polynomial time.

We therefore want to identify a property of problems for which no thinning step is needed, like bidimensionality, but we want our property to work for problems with weights, unlike bidimensionality. The key idea is to define the property in terms of graph structure rather than solely in terms of the optimum value. As is often the case in recent research in graph algorithms, "simple structure" is formalized as small treewidth or, equivalently, small branchwidth. Several state-of-the-art algorithms use the fact that many NP-hard graph problems can be solved quickly on graphs of small treewidth since those graphs have very small separators and therefore dynamic programming can be used. However, in our framework it wouldn't help to solve the problem in the subgraph of edges not in a solution. We make a different use of the small treewidth of the graph of edges not in the solution; it enables us to prove the existence of a certain kind of separator structure for the entire graph.

Planar graphs, bounded-genus graphs, and, more generally, members of a minor-closed graph family excluding some apex graph all have the diameter-treewidth property [78]: the treewidth of such a graph is upper-bounded by some function of its unweighted diameter. When referring to unweighted distance in a graph, we use the term *hops* to distinguish this from measuring distance according to edge- or vertex-weights.

Fact 9.1. *Suppose a graph problem restricts the input graphs to have bounded genus. Suppose also that for some integer t, for every input graph G and for every feasible solution S, S is connected and every vertex of G is within t hops of S. Then the graph problem is $O(t)$-ubiquitous.*

By the observation, for a problem on bounded-genus graphs to be considered ubiquitous, it is enough that every solution be "everywhere" in the sense that every vertex is close to the solution in terms of number of hops[1].

[1]In fact, it is sufficient even if we measure number of hops in the *face-vertex incidence graph* (a.k.a. the *radial graph*).

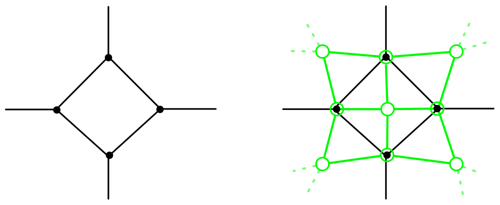


Figure 9.1: On the left is a fragment of an embedded graph G. On the right is the corresponding fragment of $\tilde{G}$, where we have added vertices and edges of the face-vertex incidence graph of G.

Preliminaries and notations.

We first introduce the definitions and notations that are specific to this section. The following definitions are illustrated in Figure 9.1. Given a connected graph $G = (V, E)$ cellularly embedded on a surface, the *radial graph* (a.k.a. the *face-vertex incidence graph*) of G is the embedded graph whose vertex set includes the vertices of G and also a vertex v_f for each face f of G; it contains an edge between v and v_f if v is a vertex of G that is incident to the face f of G. We define $\tilde{G} = (\tilde{V}, \tilde{E})$ to be the union of G and its radial graph. That is, $\tilde{G}$ contains the vertices of the radial graph and the edges of G and of the radial graph.

In a walk in a graph G, a *spur* is the occurence of a single edge used twice consecutively in oppposite directions.

A *branch decomposition* [171] of a graph is a maximal noncrossing collection of subsets of edges of the graph, equivalently a rooted binary tree in which each node corresponds to a subset of edges, and the two children of an internal node correspond to disjoint subsets whose union corresponds to the parent. Each subset of edges in a branch decomposition induces a subgraph of the graph, which we call a *cluster* of the branch decomposition. The *boundary* of a cluster is the set of vertices that are incident both to edges belonging to the cluster and edges not belonging to the cluster. The *width* of a cluster is the number of boundary vertices, and the width of a branch decomposition is the maximum cluster width. The *branchwidth* of a graph is the minimum width of a branch decomposition. The branchwidth w and treewidth t of a graph are related by

$$w - 1 \leqslant t \leqslant \lfloor \frac{3}{2} w \rfloor - 1.$$

For fixed w, there is a linear-time algorithm [41] to determine if a graph has branchwidth at most w and, if so, construct a branch decomposition of width at most w. There is a polynomial-time algorithm [171] to find an optimal branch decomposition of a planar graph.

A *noose* of an embedded graph is a Jordan curve that intersects only vertices of the graph and not edges.

Consider a planar embedded graph G. A *sphere-cut decomposition* [73] of a planar graph G is a branch decomposition in which for each cluster there is a noose that encloses

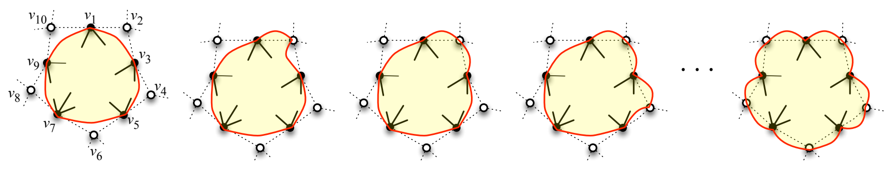


Figure 9.2: This figure illustrates the proof of Lemma 9.2. The first diagram shows a cluster in the original branch decomposition. Original vertices of G are represented by solid circles, and vertices of $\tilde{G}$ that represent faces of G are represented by open circles. The remaining diagrams show some new clusters added to form the branch decomposition of $\tilde{G}$.

exactly the edges in the cluster. The vertices that the noose intersects are exactly the boundary of the cluster. The nooses can be assumed to be mutually noncrossing.

Building on work of Seymour and Thomas [171], Dorn et al. [73] show that every planar embedded graph has a sphere-cut decomposition whose width equals the graphs' branchwidth.

Lemma 9.2. *If G is a planar graph G of branchwidth at most w then $\tilde{G}$, the union of G with the radial graph, has branchwidth at most $2w$.*

Proof. Since G has branchwidth at most w, there exists a sphere-cut decomposition T of width at most w. Consider a cluster C of T and the noose N that encloses the edges of that cluster. The noose can be represented as a cycle in $\tilde{G}$ that uses only edges of the radial graph. The noose passes through at most w vertices, so the cycle passes through at most $2w$ vertices of $\tilde{G}$. Let these vertices be $v_1, v_2, \ldots, v_{2k}$ in the order in which they appear on the cycle, where $v_1, v_3, \ldots, u_{2k-1}$ are original vertices of G and $v_2, v_4, \ldots, v_{2k}$ are the vertices of $\tilde{G}$ representing faces of G. To form the branch decomposition of $\tilde{G}$, we replace the cluster C with $2k+1$ clusters $C_0, C_1, \ldots, C_{2k}$, where C_i is obtained from C by modifying it to include the edges $v_1v_2, v_2v_3, \ldots, v_{i-1}v_i$. The new cluster with the largest boundary is C_{2k}, which has a boundary of size $2k$. In addition, we add the singleton clusters $\{v_1v_2\}, \{v_2v_3\}, \ldots, \{v_{2k-1}v_{2k}\}$. □

9.1 A New Framework

We present the ubiquity framework. As mentioned earlier, the framework relies on a new separator theorem: Theorem 7.4.

Applying this theorem recursively leads to the following theorem.

Theorem 9.3 (Branchwidth Reduction Theorem)**.** *Let $\varepsilon > 0$ and b, g be two integers. There is a polynomial-time algorithm achieving the following:*

Taking as Input: *Graph G_0 of genus at most g with edge weights and/or vertex weights, a* *connected subgraph H_0 of G_0 such that G_0/H_0 has branchwidth at most $b-1$.*

Output: *Subgraph K of H_0 such that the total weight of the edges and vertices of K is at most ε times the total weight of the edges and vertices of H_0, and G_0/K has branchwidth $O(\log n)$, where n is the number of vertices of G_0.*

The branchwidth depends linearly on b and ε^{-1}, and polynomially on g. Assuming Theorem 9.3, the proof of Theorem 7.3 is easy.

Proof of Theorem 7.3 (Ubiquity Framework). Here is the algorithm.

Algorithm 11 Meta-Algorithm for Ubiquitous Problems

```
1: Input: A graph $G = (V, E)$ of genus $g$ with nonnegative vertex and edge weights.
2: $H \leftarrow$ an $\alpha$-approximation for the problem in $G$.
3: $K \leftarrow$ BranchwidthReduction$(G, H)$. By Theorem 9.3, $S_1$ has total weight at
   most $\varepsilon \cdot \alpha \cdot OPT$ and $G/K$ has branchwidth $O(t/\varepsilon) \cdot \log n$.
4: $Y \leftarrow$ an optimal solution for the problem in $G/K$.
5: Output: $S_3 \leftarrow$ a solution for $G$ based on $Y$ and $K$.
```

For the analysis, combining the three assumptions and Theorem 9.3, the running time of the algorithm is polynomial. The solution obtained has cost $w(S_2) + \beta \cdot w(S_1)$, combining the three assumptions and Theorem 9.3, the total cost is $(1 + \alpha \cdot \beta \cdot \varepsilon)$OPT. □

We start with the proof of the Separator theorem (Theorem 7.4) for planar graphs, Section 9.1.1. In Section 9.1.2, we show how to handle graphs of bounded genus. Finally Section 9.1.3 proves Theorem 9.3.

9.1.1 A Separator Theorem

In this section, we prove the following separator theorem for planar graphs. The balance is with respect to a given *mass* function that assigns a nonnegative number to each face, called the mass of that face.

Theorem 7.4. *Let b and k be integers. Let $G = (V, E)$ be a planar graph, and H denote a connected subgraph of G such that G/H has branchwidth at most $b-1$. Let w_V and w_E be functions assigning nonnegative weights to, respectively, the vertices and the edges of G.*

Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be the union of G and its face-vertex incidence graph. Suppose the vertices and faces of $\tilde{G}$ have been assigned nonnegative masses, summing to M, with no face or vertex having mass more than $M/2$.

Then $\tilde{G}$ has a cycle C, which may repeat vertices and edges but does not cross itself and has no spurs, such that:

- *C is a balanced separator: the mass of the vertices of G strictly inside (resp., strictly outside) C is at most $3M/4$;*
- *C is mostly a light piece of H: there exists a set V' of $O(bk)$ vertices of $\tilde{G}$, and a set E' of $O(bk)$ edges of $\tilde{G}$, such that $C \backslash (E' \cup V')$ is a subgraph of H and $w_V(V(C)\backslash V') + w_E(E(C)\backslash E') \leqslant W/k$, where W is the total weight of the vertices and edges in H.*

Moreover, C, V', and E' can be computed in time $O(k^2 n)$.

This theorem is used recursively, with H an $O(1)$-approximation of the ubiquitous problem we consider, to prove Theorem 9.3 (see Section 9.1.3). We note that in the problems described in this paper, b is a small constant.

Algorithm 12 Balanced Separator Algorithm for Planar Graphs

```
1: Input: A planar graph $G = (V, E)$ and a subgraph $H$ such that $G/H$ has branchwidth
   at most $b - 1$. Nonnegative weights on vertices and edges of $H$. Masses on the faces
   of $\tilde{G}$ summing to $M$, each at most $M/2$.
2: $w_1 \leftarrow$ new edge weights on $H$ derived from Lemma 9.4 to represent both edge and
   vertex weights.
3: $w'_E \leftarrow$ edge weights on $\tilde{G}$ defined in Proposition 9.5
4: $\tilde{G}_1 \leftarrow$ ShortFacesSubgraph($\tilde{G}, H, w'_E$) as per Proposition 9.7.
5: if there exists a face $f$ of $\tilde{G}_1$ with mass at least $M/2$ then
6:     return $C \leftarrow$ fundamental cycle separator of $f$ in $\tilde{G}$ as per Proposition 9.8.
7: else
8:     return $C \leftarrow$ cycle separator in $\tilde{G}_1$ as per Proposition 9.9.
9: Output: A cycle separator $C$, fulfilling the requirements of Theorem 7.4.
```

9.1.1.1 Reduction to a simpler problem with no vertex weights

Lemma 9.4. *Without loss of generality we may assume that all vertices have weight zero.*

Proof. Recall that H is connected; let H' be a spanning tree of H. A cycle satisfying the conclusion of the theorem with H' instead of H also satisfies the conclusion of the original theorem. This is because W can only decrease by this operation, and the branchwidth of G/H' equals the branchwidth of G/H (indeed, G/H' can be obtained from G/H by adding loop edges). Henceforth we assume that H is a tree.

The algorithm for Theorem 7.4 proceeds as follows. Select an arbitrary vertex r of H to be the root, direct the edges of H toward r, and for each edge uv of H (directed from u to v), define $\hat{w}_E(uv) = w_E(uv) + w_V(u)$. For every vertex u, define $\hat{w}_V(u) = 0$ for every vertex u. Assume that Theorem 7.4 holds when vertex weights are zero, and apply it with the weight functions $\hat{w}_E$ and $\hat{w}_V$. Let C be the resulting cycle, and let E' be the resulting edge subset, i.e. such that $\hat{w}_E(E(C)\backslash E') \leqslant W/k$ where W is the sum of weights. We prove below that C is also a solution for the weights w_V and w_E, for the same subset E' of edges and for a suitable subset V' of vertices.

Since $|E'| = O(bk)$, we have that $C \cap H$ is made of $O(bk)$ paths of H. Let P be such a path. The path's weight $\hat{w}_E(P)$ includes the weight $w_E(e)$ for every edge e in P. Moreover, for every vertex v of P, since the weight $w_V(v)$ of a vertex v is transferred to the parent edge, $w_V(v)$ is also included in $\hat{w}_E(P)$ except if (i) v is equal to the root r of H, or (ii) v has no outgoing edge in P. Since every vertex of H has at most one outgoing edge, and P has no spur, every vertex of P has at most one outgoing edge. So, when walking along P (oriented arbitrarily), we first encounter an arbitrary nonnegative number of forward edges, and then an arbitrary nonnegative number of backward edges. It follows that at most one vertex of P, let us call it v_P, has no outgoing edge in P. Let V' consist of the root r together with the vertex v_P for each path P comprising $C \cap H$. Then $|V'| = O(bk)$ and the weight of $C\backslash(V' \cup E')$ with respect to w_V and w_E is at most the weight of $C\backslash E'$ with respect to $\hat{w}_E$, which is at most W/k. □

Let c be a a constant $c \geqslant 4$ to be determined.

Proposition 9.5. *Finding a balanced separator satisfying the Separator Theorem (Theorem 7.4) can be reduced to finding a balanced separator cycle in $\tilde{G}$ that has weight at most W/k with respect to the edge-weight assignment*

$$w'_E(e) = \begin{cases} \min\{w_E(e), W/(cbk^2)\} & \text{if } e \in E(H) \\ W/(cbk^2) & \text{otherwise} \end{cases} \tag{9.1}$$

Proof. By Lemma 9.4, we can assume that there are no vertex weights. Assume that we find a cycle C as above. Let E' be the set of edges e used by C such that $w'_E(e) = W/(cbk^2)$. Since C has weight at most W/k, we have $|E'| \leqslant cbk$. For each edge $e \in C\backslash E'$, we have $w'_E(e) = w_E(e)$. Since C has weight at most W/k, we have $w_E(C\backslash E') \leqslant W/k$. □

9.1.1.2 Adding edges to reduce the weight of face boundaries

The algorithm described in Proposition 9.5 first selects edges to add to H so as to ensure that each face of the resulting graph has small weight.

Let $\ell = W/ck^2$. Every edge has weight at most ℓ/b.

Lemma 9.6. *Let $\tilde{H}$ be a subgraph of $\tilde{G}$ containing H. Let f be a face of $\tilde{H}$ with boundary weight at least $(12 + \frac{3}{b})\ell$ with respect to w'_E. Then there are two vertices u and v of $\tilde{G}$ on the boundary of f, and a path p in $\tilde{G}$ lying in f (possibly touching its boundary), such that:*

- *each of the two paths between u and v on the boundary of f has weight at least 3ℓ;*

- *p has at most $2b$ edges.*

Moreover, p can be computed in time linear in the complexity of the subgraph of $\tilde{G}$ inside f.

Proof. Let ∂f be the closed walk that is the boundary of f. We write ∂f as the concatenation of four paths N, W, S, E in this order, such that each of these paths has weight at least 3ℓ. (To prove that this is possible, first take N, W, and S with weight between 3ℓ and $(3+\frac{1}{b})\ell$, which is always possible since each edge has weight at most ℓ/b; then the remaining part E has weight at least 3ℓ, since the boundary of ∂f has weight at least $(12+\frac{3}{b})\ell$.)

Let $\tilde{G}'$ be the part of $\tilde{G}$ inside or on the boundary of f. Similarly, let $G' := G \cap \tilde{G}'$. Since G/H has branchwidth at most $b-1$, $G'/\partial f$ has branchwidth at most $b-1$. By Lemma 9.2, it follows that $\tilde{G}'/\partial f$ has branchwidth at most $2b-2$.

Observe that the distance from any vertex of N to any vertex of S along ∂f is at least 3ℓ. Assume that there is no path p as stated in the lemma. Then every path in $\tilde{G}'$ connecting N to S has at least $2b+2$ vertices. This implies that any vertex cut separating W and E has at least $2b$ vertices. (Indeed, any vertex cut of size j in $\tilde{G}'$ separating W and E corresponds to a closed curve separating W and E in the plane, intersecting j vertices of $\tilde{G}'$; since $\tilde{G}'$ is a triangulation, except for the outer face, the part of that curve inside f can be pushed to $\tilde{G}'$, leading to a path of hop-length j.) Menger's theorem now implies that there are at least $2b$ vertex-disjoint paths between W and E.

Similarly, there are at least $2b$ vertex-disjoint paths between N and S. This implies the existence of a grid minor of size $2b \times 2b$ in $\tilde{G}'$ (similar arguments were used elsewhere [40, p. 88], [55, Proof of Theorem 3.1], and seem to originate from Robertson, Seymour, and Thomas [163]), hence a grid minor of size $2b \times 2b$ in $\tilde{G}'/\partial f$, which contradicts the fact that $\tilde{G}'/\partial f$ has branchwidth at most $2b-2$. So there exists such a path p. Computing such a path takes linear time using two shortest path computations in the planar graph $\tilde{G}'$ [104]. □

Proposition 9.7. *There is an $O(k^2 n)$ algorithm that computes a subgraph $\tilde{H}$ of $\tilde{G}$ containing H such that:*

- *$\tilde{H}$ has weight at most $2W$, and*
- *every face boundary of $\tilde{H}$ has weight at most $(12+\frac{3}{b})W/ck^2$ (w.r.t. w'_E).*

Proof. Initially, let $\tilde{H} := H$. We iteratively apply Lemma 9.6 by adding edges of the path p in $\tilde{H}$, until every face boundary of $\tilde{H}$ has weight less than $(12+\frac{3}{b})\ell$. The path p has weight at most 2ℓ. This operation splits the face f into at least two faces, among which some number m have boundary length at least $(12+\frac{3}{b})\ell$.

We claim that the value of $\varphi := \sum_f (w'_E(\partial f) - 3\ell)$, where the sum is over all faces f of weight at least $(12+\frac{3}{b})\ell$, decreases by at least ℓ when adding p. Indeed, this is clear if $m = 0$; if $m = 1$, the new face f' with length at least $(12+\frac{3}{b})\ell$ satisfies $w'_E(\partial f') \leqslant$

$w'_E(\partial f) + 2\ell - 3\ell = w'_E(\partial f) - \ell$; and if $m \geqslant 2$, the contribution of the m new sub-faces to φ is at most $w'_E(\partial f) + 2 \cdot 2\ell - 3m\ell$.

Thus the total number of iterations is at most $2W/\ell = 2k^2$. At each step, we add at most $2b$ edges, each of weight $W/(cbk^2)$, so the total weight of the added edges is at most $4W/c$. Since $c \geqslant 4$, the total weight of $\tilde{H}$ is at most $2W$. The time complexity follows from the fact that there are $O(k^2)$ iterations. □

The algorithm of Proposition 9.7 produces a subgraph $\tilde{H}$ of weight $\tilde{W} \leqslant 2W$.

9.1.1.3 A balanced cycle separator for weighted planar graphs with light faces

Recall that the faces and vertices of $\tilde{G}$ have been assigned nonnegative masses, that M is the sum of masses, and that no single mass exceeds $M/2$. Our goal is to give a separator algorithm for the subgraph $\tilde{H}$ whose existence is guaranteed by Proposition 9.7. Recall that the total weight $\tilde{W}$ of $\tilde{H}$ is at most $2W$

Note that each face of $\tilde{H}$ is (essentially) the union of a collection of faces and vertices and edges of $\tilde{G}$. We define the mass of a face of $\tilde{H}$ to be the sum of the masses of the corresponding faces and vertices of $\tilde{G}$.

There are two cases: when a face of $\tilde{H}$ has mass greater than $M/2$ and when no such face does. In the first case, we use a simple construction based on sphere-cut decomposition.

Proposition 9.8. *Suppose $\tilde{H}$ has a face f whose mass is greater than $M/2$. Then there is a cycle C, which may repeat vertices and edges but does not cross itself and has no spur, of weight $4\tilde{W}/k^2$, such that the mass of the faces inside (resp., outside) C is at most $3M/4$. Moreover, C can be computed in linear time.*

Proof of Proposition 9.8. Let $\tilde{G}_f$ be the subgraph of $\tilde{G}$ consisting of the interior and boundary of f. Since f has mass greater than $M/2$, every face of $\tilde{G}_f$ that is not part of f has mass less than $M/2$. Let L be the graph obtained from $\tilde{G}_f$ by contracting all but one of the edges of the boundary of f. Since $\tilde{G}/H$ has branchwidth at most $2b-2$, so does L. Since f has mass greater than $M = 2$, the face of L corresponding to the part of $\tilde{G}$ not in f has mass at most $M/2$. Thus each face of L has mass at most $M/2$.

Consider a sphere-cut decomposition of L. It defines a rooted binary tree in which each node corresponds to a noose and a cluster consisting of the edges enclosed by the noose. Define the mass of a node of the binary tree to be the mass of the faces fully enclosed by, or intersecting, the corresponding noose. Let v be a deepest node in the binary tree such that v's mass is greater than $M/2$. Among v's two children let v_1 be the child with the greater weight. The sum of the masses of v's two children is greater than or equal to the mass of v, thus the mass of v_1 is at least $M/4$.

Let C_1 be the Jordan curve corresponding to v_1. The total mass of the faces strictly enclosed by C_1 is at most the mass of v_1, which is at most $M/2$. The total mass of the faces strictly outside C_1 equals M minus the mass of v_1, which is at most $M - M/4 = 3M/4$.

We construct a cycle C_2 in L from C_1 by pushing each part of the curve which passes through a face onto part of the face's boundary. We sequentially choose the direction in which to push faces: each face is added to the currently lighter side. As the mass of each face is at most $M/2$, the new cycle is $3/4$ balanced. As L has maximum face degree 3, the curve C_1 passes through each face at most once, so the resulting cycle C_2 is non-self-crossing. If any spurs are formed in C_2, we (iteratively) remove them. Removing a spur does not affect the balance at all, and can only reduce the weight of the cycle.

Since L has branchwidth at most $2b-2$, the curve corresponding to v passes through at most $2b-2$ vertices, and thus at most $2b-2$ faces. Since each face has degree at most 3, each path through a face is pushed to at most 2 edges. Thus C_2 contains at most $4b$ edges. Since each edge has weight at most $W/(cbk^2)$, C_2 has weight at most $4W/ck^2$.

C_1 can by lifted to a cycle C in $\tilde{G}$ with the same balance by adding to it some (possibly empty) part of the boundary of f. Since the total weight of the boundary of each face in $\tilde{H}$ is at most $(12+\frac{3}{b})\ell$, C has weight at most W/k for an appropriate choice of the constant c. Each step of the algorithm implied in the proof can be implemented in linear time. □

If no face is massive, we use a variation of Miller's simple cycle separator theorem [143]. The main differences are that we do not require 2-connectivity, and that the edges are weighted. The proof is adapted from the simplified proof of Miller's theorem in [119].

Proposition 9.9. *Suppose that no face of $\tilde{H}$ has mass larger than $M/2$. Then there is a cycle C, which may repeat vertices and edges but does not cross itself and has no spur, of weight $O(\tilde{W}/k)$, such that the mass of the faces inside (resp., outside) C is at most $3M/4$. Moreover, C can be computed in linear time.*

Before proving this proposition, we show why it, together with Proposition 9.8, implies Theorem 7.4:

Proof of the Separator Theorem (Theorem 7.4). Let $\tilde{H}$ be the graph obtained after applying Proposition 9.7. By Proposition 9.5, it is sufficient to show that there exists a balanced cycle separator C of total weight W/k in $\tilde{G}$ (with respect to w'_E). If one of the faces of $\tilde{H}$ has mass at greater than $M/2$, applying Proposition 9.8 yields such a cycle. Otherwise, since $\tilde{H}$ is a subgraph of $\tilde{G}$, it suffices to find such a C in $\tilde{H}$. The advantage is that the total weight $\tilde{W}$ of $\tilde{H}$ is linear in W, the weight of H, while each face boundary has weight $O(\hat{W}/k^2)$. So applying Proposition 9.9 gives a cycle of weight W/k, as needed. □

Proof of Proposition 9.9. Let T be a breadth-first search in the face-vertex incidence graph J of $\tilde{H}$, rooted at an arbitrary face f; for clarity of exposition below, we assume that f is the outer face of $\tilde{H}$ in the planar drawing that we consider (thus the notions of "inside" and "outside" are well defined). We define the *level* of f to be zero, and the levels of the other vertices and faces of $\tilde{H}$ by induction: The level of a vertex of $\tilde{H}$ is equal to one plus the level of the parent face in T, and the level of a face of $\tilde{H}$ is equal to the level of the parent vertex in T.

Define a mass function on vertices of J in which vertices corresponding to faces of $\tilde{H}$ have mass equal to the mass of the corresponding faces, and those corresponding to vertices of $\tilde{H}$ have mass zero. There exists a simple cycle $\tilde{C}$ in J consisting of a path in T and an edge e not in T such that the mass on either side of the cycle is at most $2M/3$ [133]. Such a cycle is called a *fundamental cycle*.

Let i be an integer. The set of faces of $\tilde{H}$ of level at least i can be partitioned into regions, by declaring that two such faces are in the same component if they share an edge, and extending this relation by transitivity. A *component of level* i is the topological closure of such a region.

We claim that the boundary ∂K of such a component K is a simple cycle in $\tilde{H}$. Since K is the closure of a union of faces, ∂K is a subgraph of $\tilde{H}$ with each vertex of even degree. If some vertex v has degree at least four in ∂K, then v has level i, and its incident faces all have level i and $i-1$. Because v has degree at least four, there are two faces f' and f'' of $\tilde{H}$ with level i that are separated by faces of level $i-1$ in the cyclic ordering around v. Since f' and f'' are in K, there is a simple topological cycle γ passing through f', v, and f'', in this order, entirely lying in the interior of K (except at v). But then all vertices and faces inside γ must have level at least i, which contradicts the assumption. So ∂K is the disjoint union of cycles. Two such cycles cannot be nested, for a similar reason, and they cannot be separated as well, because their interiors would not be connected to each other. So ∂K is a single simple cycle in $\tilde{H}$. This proves the claim.

Since $\tilde{C}$ is a fundamental cycle in J, the levels of its vertices and faces are increasing and then decreasing when walking along $\tilde{C}$ starting from the common ancestor in T of the endpoints of e. Therefore, $\tilde{C}$ enters the interior of at most one component at a given level i.

Let $i_{min}-1$ and i_{max} be the minimum and maximum levels faces in $\tilde{C}$; for each i, $i_{min} \leqslant i \leqslant i_{max}$, let K_i be the (unique) component at level i penetrated by $\tilde{C}$. Moreover, let $K_{i_{min}-1} = F(\tilde{H})$ and $K_{i_{max}+1} = \varnothing$. The K_i's are nested, and the boundaries ∂K_i of the K_i's, for $i_{min} \leqslant i \leqslant i_{max}$, form disjoint simple cycles. Indeed, by construction, a vertex on ∂K_i is incident with some faces of level $i-1$, some faces of level i, and no face of other levels.

Let i_{med} be such that $i_{min}-1 \leqslant i_{med} \leqslant i_{max}$, and the mass of the faces of $\tilde{H}$ inside $K_{i_{med}+1}$ or outside $K_{i_{med}}$ is at most $3M/4$. (For this purpose, one can let i_{med} be as large as possible such that the mass of the faces of $\tilde{H}$ outside $K_{i_{med}}$ is at most $3M/4$.) Let i_- be the largest level smaller or equal to i_{med} such that ∂K_{i_-} has weight at most $W_1/8k$. Similarly, let i_+ be the smallest level larger or equal to $i_{med}+1$ such that ∂K_{i_+} has weight at most $W_1/8k$. Then the total weight of $\partial K_{i_+} \cup \partial K_{i_-}$ is at most $2W_1/8k$, which is at most $W/2k$.

Since $K_{i_-+1}, K_{i_-+2}, \ldots, K_{i_+-1}$ each have weight larger than $W_1/8k$, there can be at most $8k$ such levels, so we have $i_+ - i_- \leqslant 8k+1$.

We now consider the part of $\tilde{C}$ inside K_{i_-} but outside K_{i_+}, which consists of two paths in J. We push each of these two paths into $\tilde{H}$: Each time such a path traverses a face of $\tilde{H}$, we push the corresponding part onto one of the face boundaries. We sequentially choose the direction in which to push faces: each face is added to the currently lighter side. Since the mass of each face is at most $M/2$, the new cycle is $3/4$ balanced. Further, as $\tilde{C}$ enters each face at most once, the resulting cycle is non-self-crossing. If any spurs are formed, we (iteratively) remove them. Removing a spur does not affect the balance at all, and can only reduce the weight of the paths. Let P_1 and P_2 be the resulting two paths. Each of them has weight at most $(8k+1) \cdot (12 + \frac{3}{b})W/ck^2$ since the corresponding part of $\tilde{C}$ we pushed was traversing at most $8k+1$ faces of $\tilde{H}$, each of boundary weight at most $(12 + \frac{3}{b})W/ck^2$. By choice of c, we can ensure that the weight of these two paths is at most $W/2k$.

Let $S := P_1 \cup P_2 \cup \partial K_{i_+} \cup \partial K_{i_-}$. By construction, S has weight at most W/k. S separates J into four pieces (some of which can be empty or disconnected):

- the part of J strictly outside K_{i_-};
- the part of J strictly inside K_{i_+};
- the part of J strictly inside K_{i_-}, strictly outside K_{i_+}, and strictly inside $\tilde{C}$;
- the part of J strictly inside K_{i_-}, strictly outside K_{i_+}, and strictly outside $\tilde{C}$.

By construction, each such piece encloses faces of mass at most $3M/4$. The three smallest pieces together have face mass at most $3M/4$, and the largest one at most $3M/4$. Thus, we can take for separator the subset of S that bounds the larger of these pieces; this is indeed a balanced separator, and a non-self-crossing cycle without spur in $\tilde{H}$. Each step of the algorithm implied in the proof can be implemented in linear time. □

Algorithm 13 Balanced Separator Algorithm for Planar Graphs

```
1: Input: A planar graph $G = (V, E)$ and a subgraph $H$ such that $G/H$ has branchwidth
   at most $b - 1$. Nonnegative weights on vertices and edges of $H$. Masses on the faces
   of $\tilde{G}$ summing to $M$, each at most $M/2$.
2: $w_1 \leftarrow$ new edge weights on $H$ derived from Lemma 9.4 to represent both edge and
   vertex weights.
3: $w'_E \leftarrow$ edge weights on $\tilde{G}$ defined in Proposition 9.5
4: $\tilde{G}_1 \leftarrow$ ShortFacesSubgraph($\tilde{G}, H, w'_E$) as per Proposition 9.7.
5: if there exists a face $f$ of $\tilde{G}_1$ with mass at least $M/2$ then
6:     return $C \leftarrow$ fundamental cycle separator of $f$ in $\tilde{G}$ as per Proposition 9.8.
7: else
8:     return $C \leftarrow$ cycle separator in $\tilde{G}_1$ as per Proposition 9.9.
9: Output: A cycle separator $C$, fulfilling the requirements of Theorem 7.4.
```

9.1.2 Planarization Theorem

In this section, we show the following theorem.

Theorem 9.10. *Let $b > 1$ be a constant. There exists a polynomial-time algorithm for the following: The algorithm is given a positive integer parameter k, an edge-weighted graph G that is cellularly embedded on a surface of genus g, and a connected subgraph H of G such that G/H has branchwidth at most $b - 1$.*

The algorithm outputs a subgraph S of $\tilde{G}$ such that $G - S$ is planar, S contains at most $O(g^2 + k)$ edges not in H, S has at most $O(g^2 + k)$ connected components, and the total weight of S is at most $g^{O(1)}W/k$ where W is the total weight of H.

Using the argument of Section 9.1.1.1 (which does not use planarity), we can assume that all vertex weights are zero.

As in Proposition 9.5, the algorithm for Theorem 9.10 assigns edge-weights to $\tilde{G}$ according to Equation 9.1. Edges not in H have weight W/cbk^2. The algorithm then finds a subgraph S whose weight is $O(W/k)$ with respect to this edge-weight assignment. As a consequence, the number of edges in S that are not in H is $O(g^2 + k)$.

The next lemma follows from [161, Theorem 4.1].

Lemma 9.11. *Consider a graph G that is cellularly embedded on a surface of genus g and a subgraph H of G such that G/H has treewidth at most t. Let f be a face of H of genus at least one and G_f be the subgraph of G induced by f and its interior. There exists a non-separating cycle in $\tilde{G}_f$ that intersects at most $O(t)$ vertices of G_f.*

Proposition 9.12. *Consider a graph G_0 with r connected components embedded on a surface of genus g. Then the number of faces of G_0 that are not disks is $O(r + g)$. Moreover, the total number of boundary components of all non-disk faces is at most $O(r + g)$.*

Proof. For any graph G, let $\varphi(G)$ denote the sum, over all non-disk faces f of G, of the number of boundary components of f. We will prove that $\varphi(G_0) = O(g + r)$.

First, we define a graph G_1 obtained from G_0 by adding $r - 1$ edges, so that G_1 is connected. Observe that $\varphi(G_0) \leqslant \varphi(G_1) + O(r)$: indeed, consider the addition of an edge e in some face f, during the transformation of G_0 into G_1. Edge e connects two distinct boundary components of f, so it does not separate f. Moreover, the number of boundary components of f decreases by at most one.

Second, we define a graph G_2 obtained from G_1 by contracting the edges of a spanning tree of G_1; the graph G_2 has a single vertex, and we have $\varphi(G_1) = \varphi(G_2)$.

Third, we iteratively apply the following operation to G_2: While there is a disk of G_2 bounded by a single loop, we remove that loop, and similarly while there is a disk of G_2 bounded by exactly two loops, we remove one of the loops. The non-disk faces of this new graph, G_3, have the same topology as those in G_2, so $\varphi(G_2) = \varphi(G_3)$.

Under these conditions, it is known [48, Lemma 2.1] that the number of loops in G_3 is $O(g)$; in particular, $\varphi(G_3) = O(g)$, which by the above equalities implies $\varphi(G_0) = O(r+g)$.

That immediately implies that the number of faces of G_0 that are not disks is $O(r+g)$, hence the proposition holds. □

We can now prove the following lemma.

Lemma 9.13. *Let $\tilde{H}$ be a subgraph of $\tilde{G}$ containing H. Let f be a face of $\tilde{H}$. Assume f has a boundary component f_0 with weight at least $(12+\frac{3}{b})\ell$. Then there exist two vertices u and v of $\tilde{G}$ on the boundary of f, and a path p in $\tilde{G}$ with at most $2b$ edges and lying in f, such that:*

- *if u and v are both in f_0, then each of the two paths between u and v in f_0 has weight at least 3ℓ;*
- *otherwise, u and v belong to different boundary components of f, and the path p intersects the boundary of f only at u and v.*

Moreover, p can be computed in time linear in the complexity of the subgraph of $\tilde{G}$ inside f.

Proof. The proof is similar to the proof of Lemma 9.6. We explain how to adapt its proof. Observe that by Proposition 9.12, the number of boundary components is at most $O(g^2)$. Thus, the graph G_f which consists of the interior of f where each boundary component is contracted to a vertex has branchwidth $O(g^2)+b$. Indeed, the graph corresponding to the interior of f where all the boundary components are contracted into a single vertex has branchwidth at most b. Form a width-b branch decomposition of this graph. When each boundary component is represented by a single vertex, the width increases by at most the number of such vertices.

Now, we apply the argument of Lemma 9.6. If the short path that is found does not intersect any vertex resulting from the contractions, we just return the path and it satisfies the conditions of the lemma. Otherwise, consider a short path from u to v intersecting at least one vertex resulting from the contractions, and return a shortest sub-path connecting a vertex u' in f_0 with a contracted vertex v'. This path connects two different boundary components of f, without intersecting the boundary of f except at its endpoints, thus satisfying the conditions of the lemma. □

We can derive the following proposition whose proof resembles that of Proposition 9.14.

Proposition 9.14. *Let $\tilde{H}$ be a subgraph of $\tilde{G}$ containing H. There exists an algorithm to compute a subgraph $\tilde{H}_1$ of $\tilde{G}$ containing $\tilde{H}$ such that:*

- *$\tilde{H}_1$ has $O(b(k^2+g))$ edges not in $\tilde{H}$;*

- *every boundary component of every face f of $\tilde{G}_1$ has weight at most $(12 + \frac{3}{b})W/ck^2$.*

The running time of the algorithm is $O((k^2 + g)n)$.

Theorem 9.15. *Let k be an integer. Consider a graph G with positive edge weights that is cellularly embedded on a surface S of Euler genus g and such that every face is a disk. Let W denote its total weight, and assume that every face has boundary weight at most W/k^2. There exists a subgraph G' of G, such that cutting S along G' gives a surface with genus zero (possibly with several boundary components), with the following properties: G' has weight $O(\sqrt{g}W/k)$, and has at most g connected components. Furthermore, G' can be computed in linear time.*

G' is called a *planarizing* subgraph of G.

Proof. The proof is a refinement on a result by Eppstein [79]; see also [80]. Let J be the face-vertex incidence graph of G. Let r be an arbitrary vertex of G, and let T be a breadth-first search tree in J rooted at r. We define the *level* $\ell(u)$ of a face or vertex u of G to be the number of edges of the path in T from r to u.

Let E be the set of edges of J. For each edge $uv \in E$, let $L(uv)$ be the loop rooted at r that is the concatenation of the path from r to u in T, edge uv, and the path from v to r in T. Loop $L(uv)$ has $\ell(uv) = \ell(u) + \ell(v) + 1$ edges. Let C be the primal edges of a *maximum* spanning tree of $(E\backslash T)^*$, where the weight of an edge $(uv)^* \in E^*$ equals $\ell(uv)$. Finally, let $X := E\backslash(T \cup C)$.

Euler's formula implies that $|X| = g$. It is known from [81, Section 3.4]) (and not hard to see) that $\bigcup_{uv\in X} L(uv)$ cuts S into a topological disk, and that an alternative greedy way to compute X is to iteratively add to X the edge $u'v'$ with smallest value of $\ell(u'v')$ such that $\bigcup_{uv\in X} L(uv) \cup L(u'v')$ does not disconnect the surface.

Let $M := 2\left\lceil k/(2\sqrt{g})\right\rceil$. Recall that W denotes the total weight. We choose an even $i \in \{0, \dots, M-1\}$ such that the subgraph G_1 of G induced by the vertices of level equal to i modulo M has weight $O(\sqrt{g}W/k)$.

For each edge $uv \in X$, we consider the smallest level $i_{uv} \geqslant \max(\ell(u), \ell(v)) - M$ that is equal to i modulo M. Define $L'(uv)$ to be the part of $L(uv)$ of level at least i_{uv}. Since $L'(uv)$ traverses $O(k/\sqrt{g})$ faces of G, each of boundary weight $O(W/k^2)$, we can "push" $L'(uv)$ to a walk $L'_p(uv)$ of G, of weight $O(W/(k\sqrt{g}))$.

Let G_2 be the union of the subgraph G_1 and of the walks $L'_p(uv)$, for $uv \in X$. By construction, and since $|X| = g$, the weight of G_2 is $O(\sqrt{g}W/k)$. We will now (i) prove that cutting S along G_2 results in a genus zero surface (possibly with several boundary components), and then (ii) extract from G_2 a subgraph still having that property, but having $O(g)$ connected components.

For (i), let i' be equal to i modulo M. It suffices to prove that the part of the surface S that is the closure of the union of the faces of levels between $i'+1$ and $i'+M-1$, minus G_2, has genus zero; or, equivalently, minus G_1 union the $L'(uv)$ for $uv \in X$. Actually, that latter surface is contained in the closure of the faces of G at level at most $i' + M - 1$ minus the union of the loops $L(uv)$ with $i_{uv} \leqslant i' + M$, so it suffices to prove that this latter surface, S' has genus zero. To simplify the discussion, we attach a disk to each boundary of S'. The restriction of T to S' is also breadth-first search tree of the restriction of J to S'. If S' has positive genus, then it has a non-separating loop based at r that has the form $L(u'v')$ for some edge $u'v'$ [47, Lemma 5]; that loop is also non-separating in S minus the loops $L(uv)$ with $i_{uv} \leqslant i' + M$. But this contradicts the greedy algorithm mentioned above (which should have inserted $u'v'$ in X, since $\ell(u'v') \leqslant i' + M$). This contradiction proves (i).

For (ii), we consider an inclusion-wise maximal subgraph G_3 of G_2 such that cutting S along G_3 results in a connected surface (which therefore has genus zero as well); computing G_3 can be done in linear time, by computing a spanning tree of the "dual" graph of G_2 and keeping the primal edges of the complement. Finally, let G' be obtained from G_3 by removing any connected component of G_3 that is a tree. Cutting S along G' still results in a genus zero surface, so G' is planar. Moreover, G' has at most g cycles, because otherwise the complement of G' would be disconnected (by definition of the genus). Since each connected component of G' contains a cycle, G' has at most g connected components.

Finally, G' can be computed in linear time. □

We can now prove the theorem.

Proof of Theorem 9.10. We consider the following algorithm to construct S.

1. $S \leftarrow \varnothing$
2. While there is a face with positive genus: apply Lemma 9.11 to H in order to obtain a graph H' where each face has genus 0.
3. While there exists a face whose boundary has large weight: apply Lemma 9.13. Obtain a subgraph H'' with $O(g)$ connected components, that contains H, and of maximum face weight at most $O(g^2W/k^2)$.
4. For each face f of H'', if f is not a disk, then add the entire boundary of f to S and remove f. Obtain H'''.
5. Apply Theorem 9.15 to each connected component of H''' to obtain a planarizing subgraph S', and add it to S.
6. Return S

We prove that the subgraph S satisfies the conditions of Theorem 9.10.

We first argue that iteratively applying Lemma 9.11 yields a graph H' of total weight at most $O(g^2W/k)$. Since for each face we add a non-separating cycle, the total genus of all the faces decreases by one at each iteration. By Lemma 9.11, the path added is short and so, the total weight of H' is bounded by $O(g^2W/k)$ and each face of H' has genus 0.

By applying Lemma 9.13 we either decrease the number of connected components of the boundary of a face or we reduce the weight of the face. By Proposition 9.12, the total number of connected components of all the faces is at most $O(g^2)$, thus the total weight of H'' is at most $O(g^4 W/k)$.

We now turn to the analysis of the cost incurred by Step 4. By Proposition 9.12, there are at most $O(g)$ such faces. Again since the face weight of H'' is at most $O(W/k^2)$, the total weight added to S at step 4 is at most $O(gW/k^2)$.

Finally, observe that in the remaining graph, by Step 4 each face of H''' is a disk and contains a subgraph of G of genus 0. Moreover by Step 3, the maximum face weight is at most $O(W/k^2)$. It is thus possible to apply Theorem 9.15 in order to obtain a subgraph S' of H''' of total weight at most $O(\sqrt{g}W/k)$ and such that $H''' - S'$ is planar. The total number of connected components of S' is at most $O(k)$.

Since the number of connected components added at Step 4 is $O(g^2)$, the total number of connected components of S is thus $O(g^2 + k)$. □

9.1.3 Branchwidth reduction

In this section we prove Theorem 9.3: we show that, for constants g, b, ε, there is a polynomial-time algorithm that, given a genus-g edge/vertex-weighted graph G_0 and a connected subgraph H_0 such that G_0/H_0 has branchwidth at most $b-1$, outputs a subgraph K of H_0 of weight at most ε times the weight of H_0 such that G_0/K has branchwidth $O(\log n)$, where n is the number of vertices of G_0. We give a procedure that returns a branch decomposition of G_0/K.

First we assume the graph is planar. At the end of this section, we discuss the case of positive genus.

An overview: The algorithm of Theorem 9.3 uses the algorithm of Theorem 7.4 to recursively find separators and uses them to decompose the graph into clusters of a branch decomposition. The boundaries of these clusters are subsets of the vertex sets of the separators. The boundaries might be large but, after contraction of the edges of H in the separator, the size of the boundary becomes small. Because the separators are balanced, the recursion depth is logarithmic (see also Figure 9.3).

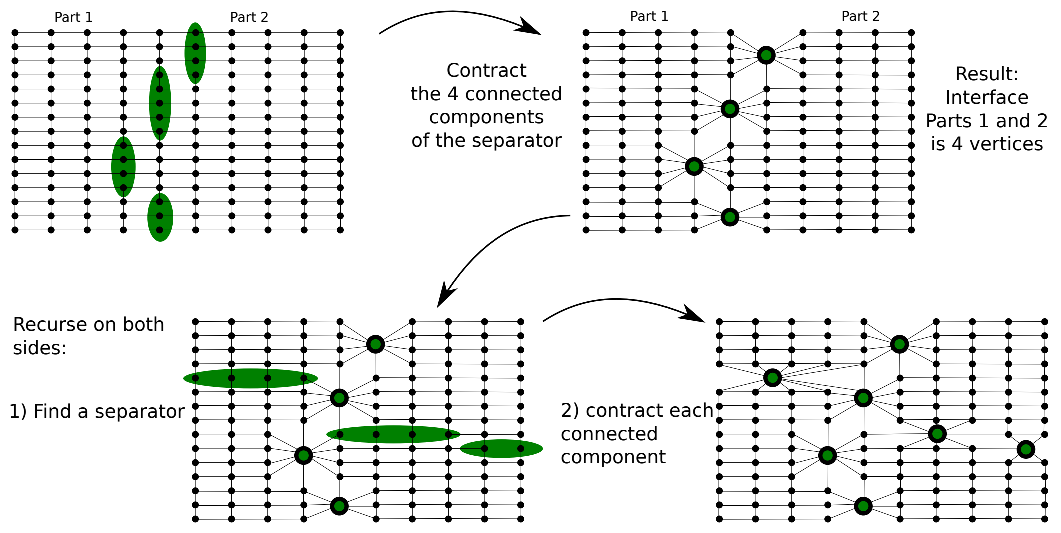

Part 1
Part 2
Contract the 4 connected components of the separator
Part 1
Part 2
Result: Interface Parts 1 and 2 is 4 vertices
Recurse on both sides:
1) Find a separator
2) contract each connected component

Figure 9.3: Two steps of the Branchwidth-Reduction algorithm.

In order to ensure the boundaries in the contracted graph remain small, the algorithm uses a variant of a strategy of [86]. The variant is described, e.g., [119] (and used elsewhere previously); occasionally, instead of ensuring the separator is balanced with respect to the size of the subgraphs, the algorithm ensures that the separator is balanced with respect to the number of vertices appearing on boundaries.

More details: We describe a recursive procedure to select edges to contract and find a branch decomposition for the contracted graph. The procedure is given a planar embedded graph G and a subgraph H such that G/H has branchwidth at most b. The procedure is also given a subset S of vertices of G, which we call *boundary vertices*.

The initial invocation is $\text{BRANCHDECOMP}(G_0, H_0, \emptyset)$ be the initial invocation. In any nonterminal invocation
$\text{BRANCHDECOMP}(G, H, S)$, the two recursive calls in Line 14 operate on disjoint subsets of H. Therefore, for every level of recursion the invocations operate on disjoint subsets of H_0, so the total weight of these subgraphs is at most the weight of H_0. We will see that the recursion depth is $O(\log n)$. Therefore the total weight of all subgraphs H passed to all invocations is $O(\log n)$ times the weight of H_0.

In Line 15, the procedure takes branch decompositions returned by the recursive calls and adds one additional cluster, the cluster consisting of all the edges in the two branch decompositions. Therefore the procedure returns a branch decomposition of the graph induced on all those edges. Thus the initial invocation returns a branch decomposition of the graph obtained from G_0 by contracting all edges ever contracted during recursive invocations of the procedure.

In any nonterminal invocation $\text{BRANCHDECOMP}(G, H, S)$, in Line 8 the procedure finds a cycle separator C using the parameter value $k = c_3\varepsilon^{-1} \log n$. The weight of edges of H in C is at most the weight of H divided by k. It follows that, for an appropriate choice

Algorithm 14 BranchDecomp(G, H, S)

```
1:  Input: a planar graph $G$, a subset $H$ of edges, and a set $S$ of vertices
2:  if $G$ has at most $c_1$ vertices then return a branch decomposition of $G$ of width $\leqslant c_1$
3:  else
4:      if $|S| > c_2\varepsilon^{-1}\log n$ then
5:          assign mass 1 to vertices of $S$ and zero to other vertices
6:      else
7:          assign mass 1 to all vertices of $G$
8:      find a cycle separator $C$ as per Theorem 7.4 using $k = c_3\varepsilon^{-1}\log n$
9:      $G' \leftarrow G/$edges of $C \cap H$
10:     $C' \leftarrow$ noose corresponding to $C$ in $G'$
11:     $S' \leftarrow$ vertices of $G'$ on $C'$
12:     $(E_1, E_2) \leftarrow C'$-induced bipartition of edges of $G'$
13:     $(S_1, S_2) \leftarrow C'$-induced bipartition of vertices of $G'$ not in $C'$
14:     $B_i \leftarrow$BranchDecomp$(G'[E_i], H \cap E_i, S_i \cup S')$ for $i = 1, 2$
15:     return $B_1 \cup B_2 \cup (\bigcup B_1 \cup B_2)$
```

of the constant c_3, the total weight of edges in all cycle separators found is at most ε times the weight of H_0. This bounds the weight of all edges contracted in Line 9 throughout all invocations.

When the edges of $C \cap H$ are contracted, C becomes a noose C' in the contracted graph G'. The noose C' partitions G' into two edge-induced subgraphs, and also partitions the boundary vertices S. In each of the two recursive calls, the vertices on C' are included as boundary vertices. This implies the invariant that, for any invocation BranchDecomp(G, H, S), any vertex of G_0 that is incident to an edge in G and an edge not in G is a member of S.

By Theorem 7.4, there are $O(b\varepsilon^{-1}k)$ vertices on the noose C', which is $O(b\varepsilon^{-1}\log n)$. Because of Line 5, one can choose the constant c_2 in Line 4 so that there is a constant c_4 such that the number of boundary vertices passed to the procedure never exceeds $c_4 b\varepsilon^{-1}\log n$, and that no two consecutive recursive invocations execute Line 5. As a consequence of the first statement, every branch decomposition returned has width at most $c_4 b\varepsilon^{-1}\log n$. As a consequence of the second statement, the recursion depth is $O(\log n)$ as promised.

Finally, consider the case in which the input graph has genus $g > 0$. In this case, the algorithm first applies Theorem 9.10's algorithm to the input graph G and obtain a subgraph S. For each piece L of $G - S$ that is planar, the algorithm recursively applies the planar separator theorem and obtains a set of edges S'_L such that L/S' has bounded branchwidth.

We now argue that the branchwidth of $G/(S \cup \bigcup_L S_L)$ is bounded. For each planar piece L of $G - S$ we take a branch decomposition of L/S_L of small width.

Since by Theorem 9.10, S contains at most $O(g^2 + k)$ connected components, these branch decompositions can be merged to form a branch decomposition of $G/(S \cup \cup_L S_L)$, increasing the width by $O(g^2 + k)$. $\square$

9.2 Algorithmic Implications

A fairly large class of problems to which our framework applies mix structure requirements and domination requirements, and can have several kinds of weights. More specifically, we consider problems that take as input a bounded-genus graph with weights on vertices, edges or faces; a solution must usually be connected, have a connected induced subgraph, or be a tour; and it must dominate all vertices, edges or faces of the graph.

To derive PTASs for those problems, we rely on Algorithm 11 and appeal to Theorem 7.3. To have a subgraph H of total weight $O(\text{OPT})$ such that G/H has bounded tree width, it is sufficient to take an $O(1)$ approximate solution, either available from previous work or obtained by designing $O(1)$-approximation when needed (for example for weighted connected dominating set).

In [167], the authors introduce a new tree-decomposition for graphs embedded on surface, called surface cut decomposition. All the problems listed in Table 7.1 are *packing-encodable* according to the definition in [167]. Even though this theorem is designed for unweighted versions of the problems, it is straightforward to extend it to weighted versions.

Theorem 9.16. *[167, Theorem 3.2] Every connected packing-encodable problem whose input graph G is embedded in a surface of genus g, and has branchwidth at most b, can be solved in time $g^{O(b+g)} b^{O(g)} n^{O(1)}$.*

9.2.0.1 Weighted Connected Dominating Set, Max Weighted Leaf Spanning Tree

Consider the Vertex-Weighted Connected Dominating Set problem defined as follows.

Definition 9.17. ***Weighted Connected Dominating Set**. Given a graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{R}^+$, a* connected dominating set *is a set of vertices S such that $G[S]$ is connected and such that every vertex of V is in S or adjacent to S. The objective is to find a connected dominating set of minimum weight.*

Garey and Johnson's book [91] showed that the problem is NP-Hard, even for bounded degree planar graphs. Guha and Keller [94] obtained a $\log \Delta$ approximation where Δ is the maximum degree of the graph and that no polynomial time algorithm can do better in general graph unless NP $\subseteq$ DTIME[$n^{O(\log \log n)}$]. The vertex-weighted version of the problem received a lot of attention as it has applications in network testing problems (see [152]) and wireless communication problems (see [57]). For the unweighted version of the problem in graphs of bounded genus, a PTAS was obtained through the framework arising from the bidimensionality theory in [66]. A linear kernel was found for planar graphs in [135]. The FPT version of the problem was addressed in [63].

The vertex-weighted version of the problem was also considered by Guha and Keller in a later paper [96] who obtained a $(1.35 + \varepsilon)\log n$ approximation for general graphs and which remained the best approximation ratio until this work. Using Theorem 7.3, we obtain the following result for the connected dominating set problem.

Theorem 9.18. *Let $0 < \varepsilon \leqslant 1/2$ and let g be a fixed integer. There exists an algorithm, based on Algorithm 11, that computes a $1 + \varepsilon$-approximation to the weighted connected dominating set problem in graphs of genus bounded by g. Its running time is $n^{O(f(\varepsilon,g))}$.*

Clearly, any solution in the original graph remains a solution in the contracted graph, and its cost can only be reduced. We show that each of the three conditions of Theorem 7.3 hold, implying Theorem 9.18. **Condition 2.** The second condition is ensured by Theorem 9.16. **Condition 3.** To prove that the last condition holds, we show that given a graph G, a subgraph $G_1 = (V_1, E_1)$ and a solution S for G/G_1, there exists an Algorithm `Lift` which computes a solution for G of total cost at most $w(S) + w(G_1)$: Since each vertex resulting from the contraction of G_1 has to be dominated, at least one of its neighbor belongs to S. Therefore, we can add all the vertices of G_1 to S and the solution remains connected. Furthermore, since each vertex is dominated in G/G_1 by S and since we add the all the vertices of G_1 to S, all the vertices of G are dominated by $S \cup V_1$. The total cost of the new solution $S \cup V_1$ is $w(S) + w(G_1)$. **Condition 1.** To show the first condition, we provide the first known constant factor approximation. Indeed, since for each feasible solution S each vertex of the graph is dominated by $G[S]$, each vertex of $G/G[S]$ is at distance at most 1 from the vertex resulting from the contraction of $G[S]$. Therefore $G/G[S]$ has diameter at most 2. It follows that the branchwidth of $G/G[S]$ is $O(1)$. Therefore, any $O(1)$-approximation for the problem is a connected subgraph H of G such that G/H has branchwidth $O(1)$. We show the following lemma which is immediately subsumed by the Approximation Scheme result (Theorem 9.18).

Lemma 9.19. *There exists an $O(1)$-approximation algorithm for the Vertex-Weighted Connected Dominating Set problem for graphs of genus at most g.*

For any graph G of genus at most g, we first define a *ball* of radius i around a vertex v to be the set of points that are at edge distance at most i from v. We prove the correctness of Algorithm 15.

Lemma 9.20. *Consider the set of vertices V_0 after step 6 of Algorithm 15. There exists a solution S of value at most $2OPT$ such that $V_0 \subseteq S$.*

Proof. Consider an optimal feasible solution S_{OPT} and a ball $b \in B$. Since S_{OPT} is feasible, $\text{argmin}_{v\in b} w(v)$ is in S_{OPT} or at least one of its neighbors is in S_{OPT}. Hence $S = S_{\text{OPT}} \cup \{v \mid \exists b \in B \text{ s.t } v = \text{argmin}_{v\in b} w(v)\}$ is connected. We now argue that the cost of S is at most twice the cost of S_{OPT}. Again, since S_{OPT} is feasible, either the center of b belongs to S_{OPT} or at least one of its neighbors belongs to S_{OPT} It follows that the sum of the weights of the

Algorithm 15 Constant factor approximation algorithm for weighted connected dominating set in bounded genus graphs.

```
1: Input: A graph $G = (V, E)$ of genus at most $g$, a weight function $w : V \to \mathbb{R}_+$.
2: $B \leftarrow$ set of disjoint balls of radius 1 that is maximal under inclusion.
3: $V_0 \leftarrow \varnothing$
4: for all ball $b \in B$ do
5:     $V_0 \leftarrow V_0 \cup \{$ an element of $b$ of minimum weight $\}$
6: $V_1 \leftarrow$ constant-approximation solution to the Vertex-Weighted Steiner Tree problem
   on $G$ with terminals $V_0$.
7: $G_1 \leftarrow G/G[V_1]$, $G_1$ has bounded branchwidth by Lemma 9.22.
8: $V_2 \leftarrow$ an optimal solution to the problem on $G_1$ using Algorithm from Theorem 9.16
   for bounded branchwidth graphs.
9: Output: $V_2 \cup V_1$
```

vertices in $S_{\text{OPT}} \cap b$ is at least $\min_{v \in b} w(v)$ and thus, the sum of the weights of the vertices in $S \cap b$ is at least $2 \min_{v \in b} w(v)$. Therefore, since the balls are disjoint, the total cost of S is at most twice the total cost of S_{OPT}. □

Line 7 of the Algorithm is achieved thanks to the following theorem. See also [33].

Theorem 9.21 ([67, Theorem 1]). *There exists a polynomial-time constant-factor approximation algorithm for the vertex-weighted Steiner tree problem.*

Lemma 9.22. *Consider the set V_1 at step 8 of the algorithm. $G/G[V_1]$ has bounded branchwidth.*

Proof. Note that by the maximality condition of Step 1 of Algorithm 15, we have that each vertex of the graph is at distance at most 1 from some ball b and so, at distance at most 3 from all the vertices of some ball b.

Because $G[V_1]$ is connected, the contraction of $G[V_1]$ in $G/G[V_1]$ results in a single vertex which is at distance at most 3 from all the other vertices of $G/G[V_1]$. Hence, the diameter of $G/G[V_1]$ is constant and so, the branchwidth of $G/G[V_1]$ is constant. □

Proof of Lemma 9.19. Lemma 9.20 and Theorem 9.21 ensure that $\text{cost}(S_1) \leqslant 12 \cdot \text{OPT}$. Moreover, Lemma 9.22 and Theorem 9.16 ensure that $\text{cost}(S_2) \leqslant \text{OPT}$. The running time of the algorithm follows directly from Theorems 9.21 and 9.16. □

The weighted connected dominating set problem is also related to the maximum weighted leaf spanning tree defined as follows.

Definition 9.23. ***Maximum Weighted Leaf Spanning Tree***. *Given a graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{R}^+$, a* weighted leaf spanning tree *is a spanning tree of G*

whose cost is defined as the sum of the weights of its leaves. The objective is to find a leaf spanning tree of maximum weight.

The unweighted version of the problem has been studied in a series of results (see for example [39, 136]). The FPT version of the problem has also been extensively studied, for example in [75, 37].

Using an observation from [74] for the unweighted case, we derive the analogous observation for the weighted case, Lemma 9.25. Using Lemma 9.25, it is easy to derive Theorem 9.24 from the proof of Theorem 9.18.

Theorem 9.24. *Let $0 < \varepsilon \leqslant 1/2$ and g be a fixed integer. There exists an algorithm, based on Algorithm 11, that computes a $1 + \varepsilon$-approximation to the maximum weighted leaf spanning tree problem in graphs of genus bounded by g. Its running time is $n^{O(f(\varepsilon,g))}$.*

This lemma is standard and was proven in previous results on maximum weight leaf spanning tree.

Lemma 9.25. *Let $G = (V, E)$ be a graph with vertex weights $w : V \to \mathbb{R}^+$. Let W denote the sum of the weights of the vertices of G. The sum of the value of the optimal maximum weighted leaf spanning tree and the value of the optimal connected dominating set is equal to W.*

9.2.0.2 Tour Cover and Tree Cover

The tour and tree cover problems are defined as follows.

Definition 9.26. ***Tree cover****. Given a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}^+$, a* tree cover *is a set of edges S such that $G[S]$ is connected and such that for each $(u, v) \in E$, $\exists e \in S$ such that e shares an endpoint with (u, v). The tree cover problem asks for a tree cover of minimum weight.*

In the *tour cover* problem, the solution is required to form a tour instead of a tree. The Tour and Tree cover were introduced by Arkin et al. in [8] who obtained the first constant factor approximation and a proof of MAX-SNP hardness in general graphs. An approximation ratio of 3 was later obtained in [124] for both problems and to 2 for tree cover in [149]. The parameterized version of the problem was addressed in [98, 147].

Theorem 9.27. *Let $0 < \varepsilon \leqslant 1/2$ and let g be a fixed non-negative integer. There exist algorithms, based on Algorithm 11, that compute a $1 + \varepsilon$-approximation to the tour cover problem and to the tree cover problem in graphs of genus bounded by g. Their running times are $n^{O(f(\varepsilon,g))}$.*

Proof of Theorem 9.27. We show that the three conditions of Theorem 7.3 are met.

Condition 1 This condition is fulfilled by an $O(1)$ approximation algorithm, see for example [149]. Consider a graph G and any feasible solution S. Since S is connected, any vertex of G/S is at distance at most 1 of the vertex resulting from the contraction of S and so, G/S has constant diameter and therefore constant branchwidth.

Condition 2 This condition is obtained by Theorem 9.16.

Condition 3 We show how to derive a Lift procedure. Note that each vertex resulting from the contraction of an edge (or more generally a path) has a loop in the contracted graph. Since the solution for the contracted graph has to cover all the edges of the graph, this vertex has to belong to the optimal solution. Therefore it is possible to add the contracted edges to the solution while preserving connectivity. For Tree Cover, the solution in G is simply $S \cup K$, while for tour cover, some edges must be taken twice to form a tour. In either case, the weight of the solution is bounded by $w(S) + 2w(K)$. □

9.2.0.3 Weighted Connected Vertex Cover

We now consider the weighted connected vertex cover.

Definition 9.28. ***Weighted Connected Vertex Cover****. Given a graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{R}^+$, a* connected vertex cover *is a set of vertices S such that $G[S]$ is connected and such that for each $(u, v) \in E$, $u \in S$ or $v \in S$. The weighted connected vertex cover problem asks for a connected vertex cover of minimum weight.*

Savage [168] gave a 2 approximation algorithm which remains the best approximation algorithm for general graphs.

There are PTASs for the unweighted case in restricted classes of graphs (see [178, 66]).

The weighted connected vertex cover problem is very related to the tree cover problem. The difference is that the weights are on the vertices and not the edges. Fujito shows in [90] that, whereas the tree cover problem can be approximated within a constant factor in general graphs, the weighted vertex cover problem cannot be approximated within a factor better than $\log n$ in general graphs unless NP $\subseteq$ DTIME[$n^{O(\log \log n)}$] and provides an $O(\log n)$ approximation algorithm for the problem. See [98, 147] for results in the parameterized case.

Theorem 9.29. *Let $0 < \varepsilon \leqslant 1/2$ and let g be a fixed non-negative integer. There exists an algorithm, based on Algorithm 11, that computes a $1 + \varepsilon$-approximation to the vertex-weighted connected vertex cover problem in graphs of genus bounded by g. Its running time is $n^{O(f(\varepsilon,g))}$.*

Proof of Theorem 9.29. We show that the conditions of Theorem 7.3 hold.

Condition 1'. Here we use the more general version of the first condition, where the backbone is not required to be a solution: Observe that the value of any optimal solution to the weighted connected dominating set problem is a lower bound on the value of the optimal solution for the weighted connected vertex cover problem. Therefore, it is possible to compute a subgraph H of the input graph G of total weight at most $O(\text{OPT})$ such that G/H has treewidth at most $O(1)$ using Lemma 9.19.

Condition 2 is ensured by Theorem 9.16.

Condition 3 is attained by letting the solution on G be $S \cup K$. Since every contracted vertex is either in S or adjacent to S, $S \cup K$ is connected. As S is a vertex cover in G/H, every edge in G has an endpoint in either S or K. □

9.2.0.4 Weighted Feedback Vertex Set

There has been much research on *feedback vertex set*. Here we only mention a few representative results. The first constant-factor approximation algorithm for the *unweighted* case was achieved in [29]. It was later improved to a factor 2 for both weighted and unweighted in [22]. Primal-dual approximation algorithms for these and more general problems were given in [92] The parameterized problem was addressed in [56] and [155]. An approximation scheme for the unweighted version was given in [66].

Theorem 9.30. *There is a PTAS for* weighted feedback vertex set *in undirected planar graphs.*

We provide a reduction from weighted feedback vertex set to vertex-weighted connected dominating set.

Given a planar graph $G = (V, E)$ and a vertex-weight function $w(\cdot)$, we construct an instance for connected dominating set (CDS): the graph $\tilde{G} = (\tilde{V}, \tilde{E})$; weights of vertices of G in $\tilde{G}$ are preserved; others receive a weight of 0. It suffices to show that every FVS in G corresponds to a CDS in $\tilde{G}$ of the same weight, and vice versa.

Let S be a FVS in G. Let $V_f := \tilde{V} \backslash V$ be the vertices in $\tilde{G}$ inside faces of G, and let $\tilde{S} := S \cup V_f$. Thus $w(S) = w(\tilde{S})$. As $\tilde{S}$ contains every vertex in V_f, $\tilde{S}$ is a FVS of $\tilde{G}$. Note that $\tilde{G}$ is triangulated, and in a triangulated planar graph, every minimal vertex cut is a simple cycle. Therefore $\tilde{S}$ hits every vertex cut in $\tilde{G}$, i.e., induces a connected graph. $\tilde{S}$ contains V_f, which dominates $\tilde{G}$. Therefore $\tilde{S}$ is a connected dominating set in $\tilde{G}$.

Now let $\tilde{S}$ be a CDS in $\tilde{G}$. Then $\tilde{S}' := S' \cup V_f$ is a CDS in $\tilde{G}$ with the same weight. Thus $\tilde{S}'$ hits every cycle in $\tilde{G}$ that strictly separates any two vertices in $\tilde{S}'$. Every cycle in G separates some two faces of G, and therefore the corresponding vertices in V_f. Thus $\tilde{S}'$ hits every cycle in G. Thus $S := \tilde{S}' \cap V = \tilde{S} \cap V$ hits every cycle in G, i.e., is a feedback vertex set, and $w(S) = w(\tilde{S})$.

Concluding Remarks and Open Questions

This thesis is a step toward the understanding of both the algorithms used in practice for clustering and network design problems and the complexity of those problems. We have shown that the popular local search heuristic achieves nearly optimal performance for a variety of characterizations of practical instances. Furthermore, local search is, so far, the only algorithm that achieves the best possible theoretical bounds (*i.e.,* PTAS[1]) for clustering instances consisting of graphs excluding a fixed minor or points in a low-dimensional Euclidean space. We also proved theoretical bounds on the approximation guarantee of local search for TSP that match the performances experienced during the DIMACS TSP Challenge. Thus:

The main message is that local search occupies a sweet spot between practical performances and theoretical guarantees for several models of practical instances of clustering (including planar graphs, low-dimensional Euclidean space, and structured instances) and low-dimensional Euclidean instances of TSP and Steiner tree.

Beyond the complexity results, proving strong guarantees for a heuristic widely used in practice for the (theoretical) characterizations of real-world instances suggests that those characterizations are meaningful: they capture some of the structural properties of the real-world instances that make local search a good heuristic.

One of the main ingredients of our proofs is the existence of "cheap" separators that split the instance into two sub-instances of roughly equal size. The notion of separator has been mainly used to derive algorithmic results via the use of dynamic programming and divide-and-conquer. Here, we exhibit cases where the existence of cheap separators implies that any local optimum is close to a global optimum for both clustering and network design problems[2].

In Part II, we have designed a more structured separator, tailored for network design problems and we have shown that it yields a general approach for obtaining PTASs for various network design problems in planar graphs. This gives the first PTAS for connected

[1]The running time could be improved though, see Open Problem 5.

[2]See [148] for other connections between cheap separators and local search.

dominating set, connected vertex cover, tour and tree cover and feedback vertex set in weighted planar graphs.

Perspectives on network design problems. As we have seen in Chapter 8, the standard local search algorithm cannot achieve better than a 2-approximation for low-dimensional Euclidean TSP in the worst-case. This seems to be due to the rather non-local constraint that the solution has to be a tour. Indeed, for low-dimensional Steiner tree – which only induces a connectivity constraint – there is no lower bound known so far. Thus, two natural questions are the following.

Open Problem 1. *Is there a refinement of local search that achieves better bounds for Euclidean TSP?*

And concerning Steiner tree,

Open Problem 2. *What is the performance of local search for low-dimensional Euclidean Steiner tree?*

In Chapter 9.1, we have designed a general framework to obtain approximation schemes for a variety of network design problems. It is natural to ask about an extension of this work, in the flavor of the bidimensionality framework of [66].

Open Problem 3. *Is it possible to extend the ubiquity framework,*

- *to more general classes of graphs (e.g., graphs excluding a fixed minor)?*
- *to more general network design problems (e.g., problems that asks for a i-connected network, $i > 1$)?*

Perspectives on clustering problems. Very recent results show that, for some problems, *non-oblivious* versions of local search[1] allow to avoid trivial local optimum. As we have seen, the standard local search algorithm can get stuck in a local optimum of cost greater than 3 times the optimal. Non-oblivious version of local search could avoid this particularly bad local optimum.

Open Problem 4. *Is there a non-oblivious local search algorithm that achieves a better than 3-approximation for general instances of k-median?*

[1]We say that local search is non-oblivious if the objective of the algorithm is not the same than the objective of the problem.

Perspectives on local search. A natural direction consists in designing more efficient local search algorithms. For example, improving the running time of Algorithm 1 to $O(f(s) \cdot n \cdot \text{polylog}(n))$ would allow to address big datasets (*e.g.,* social networks) for which quadratic-time algorithms are not suitable[1].

Open Problem 5. *Is it possible to improve the running time of local search while preserving provable guarantees?*

We finally conclude with a broader and challenging open problem. Our results on clustering in planar graphs and low-dimensional Euclidean space show that combining center-based clustering objectives and cheap separators lead local search to be efficient. At a more general level, it is interesting to determine what are the ingredients that ensure that local search has provable performance guarantees.

Open Problem 6. *Is there a characterization of (1) the problems and (2) the instances for which local search is efficient (i.e., an $O(1)$-approximation or a PTAS)?*

From practice to theory continued... We highlight two main open question raised by our experiments in Chapter 6. We have seen that local search achieves a much better approximation in practice than what we showed in theory. More concretely, for neighborhood of size 1, our bounds yield an approximation guarantee of at least 2. In the experiments, we observed that the approximation obtained is less than 1.2.

Open Problem 7. *Is it possible to tighten the bounds on the approximation guarantee of local search for low-dimensional or stable instances?*

One step toward this question would be to improve the bounds for instances generated from a mixture of Gaussians.

Finally, we recall the more challenging question of the appropriateness of the k-means clustering. Indeed, we observed that, in several cases, the optimal value w.r.t. the k-means objective can be significantly smaller than the value of the clustering that we would like to compute.

Open Problem 8. *Is it possible to characterize the instances for which the optimal value w.r.t. the k-means objective corresponds to the value of the "natural" clustering?*

[1] We also note that there is no FPTAS for the k-median problem unless P = NP, even when the inputs consist of planar graphs (see Lemma A.10).

Appendix

APPENDIX A

Notations and Definitions

In this chapter we formally define the problems studied in this thesis and describe the notion of separator for both the graph and Euclidean settings. The reader might choose to skip this section and come back to it whenever a particular concept is needed.

The basic definitions of graph theory, including the description of the main graph classes studied in this thesis and their properties are postponed to Section A.3. The standard notations and definitions of graph theory and computational complexity can also be found in Sections A.3 and A.4. We refer to the books of Knuth [120, 121, 122] and Cormen et al. [173] for more details on the elementary concepts of computational complexity and algorithms.

A.1 Problem Definitions

The problems tackled in this thesis are of two main flavors. On one hand, we consider *clustering* problems which consists in partitioning a set of points lying in a metric space into *clusters* while optimizing a certain objective function. Various objective functions have been studied, each of them defining a specific clustering problem. We proceed at a more abstract level and define a single problem (see Definition A.1) that generalizes the classic clustering problems.

On the other hand, we consider *network design* problems. We say that a problem is a network design problem if it asks for a graph satisfying some structural properties (*e.g.,* the traveling salesman problem asks for a graph that is a cycle). We address network design problems of different natures, from classic ones like TSP and Steiner tree to more constrained ones like connected dominating set.

A.1.1 Clustering Problems

Partitioning points in metric spaces according to similarity arise in a variety of contexts. Thus, several clustering problems have been defined thus far. We consider a clustering problem that generalizes some classic clustering problems like k-median and k-means. In the following, let $(\mathcal{A}, \text{dist})$ denote a metric space and assume that $\text{cost}(a, b) = \text{dist}(a, b)^p$ for some fixed constant $p \geqslant 1$.

Definition A.1 (k-Clustering). *Given a finite set of* clients $A \subseteq \mathcal{A}$*, a set of* candidate centers $F \subseteq \mathcal{A}$*, two positive integers k and p, the* k-clustering problem *asks for a set of* centers $S \subseteq F$*, of cardinality at most k, that minimizes*

$$\textit{cost}(S) = \sum_{x \in A} \min_{c \in S} \textit{dist}(x, c)^p.$$

For a particular clustering $C = \{C_1, \ldots, C_k\}$ and for each client c, we define $\text{cost}(c, C_i) = \text{dist}(c, C_i)^p$. Given an instance I of the k-clustering problem and a solution S to I, a partition $C = \{C_1, \ldots C_k\}$ of A is a clustering of A *induced by* S if for any $i \in \{1, \ldots, k\}$, for any $a \in C_i$, $\text{cost}(a, c_i) = \min_{s \in S} \text{cost}(x, c)\}$.

The cases where $p = 1$ and $p = 2$ are well-studied problems known as the k*-median* (or k-clustering in [150]) and k*-means* problems respectively. A slight relaxation of the problem is called the facility location problem.

Definition A.2 (Uncapacitated Uniform Facility Location). *Given a finite set of* clients $A \subseteq \mathcal{A}$*, a set of* candidate centers $F \subseteq \mathcal{A}$*, an* facility cost f*, and a function cost* : $A \times F \mapsto \mathbb{R}_+$*, the* Uncapacitated Uniform Facility Location *asks for a set of centers $S \subseteq F$,that minimizes*

$$\textit{cost}(S) = |S| \cdot f + \sum_{x \in A} \min_{c \in S} \textit{cost}(x, c).$$

If $p \to \infty$, the problem is known as the k-center problem.

Definition A.3 (k-Center). *Given a set of* clients $A \subseteq \mathcal{A}$*, a set of* candidate centers $F \subseteq \mathcal{A}$*, a non-negative integer k, and a function cost* : $A \times F \mapsto \mathbb{R}_+$*, the* k-center problem *asks for a set of centers $S \subseteq F$, of cardinality at most k, that minimizes*

$$\textit{dist}(S) = \max_{x \in A} \min_{c \in S} \textit{cost}(x, c).$$

A.1.2 Network Design Problems

There is a variety of network design problems arising from practical applications. Here we focus on four classical network design problems. Given a set of elements A and a cost function $\text{cost} : A \times A \mapsto \mathbb{R}_+$, a *tour* of A is a graph $T = (A, E)$ that is a simple cycle. The cost of tour T is the sum of the cost of the edges of E, *i.e.,* $\sum_{(u,v) \in E} \text{cost}(u, v)$. Similarly a tree of A is a graph $T = (A, E)$ that is a tree. Its cost is also $\sum_{(u,v)\in E} \mathrm{cost}(u, v)$. The *minimum spanning tree* of A is a tree of A of minimum cost. Let $(\mathcal{A}, \mathrm{dist})$ be a metric space.

We now turn to the definition of two classical problems: the traveling salesman problem and the Steiner tree problem.

Definition A.4 (Traveling Salesman)**.** *Given a set of elements A and a cost function cost* : $A \times A \mapsto \mathbb{R}_+$*, the* traveling salesman problem *asks for a tour of A of minimum cost.*

The definition of the Steiner tree problem is the following.

Definition A.5 (Steiner Tree)**.** *Given a set of elements A, a set of* candidate Steiner points *S and a cost function cost* : $(A \cup S) \times (A \cup S) \mapsto \mathbb{R}_+$*, the* Steiner tree problem *asks for a subset $S_0 \subset S$ of* Steiner points *such that the minimum spanning tree of $A \cup S_0$ is of minimum cost.*

We also consider the connected dominating set problem. This problem is of slightly different flavor since the value of a solution is a set of vertices. This comes from the fact that this problem is often used to model wireless networks and so, the cost of the network is a function of the number of nodes rather than the number of edges (it is more relevant to optimize the cost of powering the wireless stations rather than the cost of installing cables).

Definition A.6 (Connected Dominating Set)**.** *Given a graph $G = (V, E)$, the* connected dominating set problem *asks for a subset S of V such that*

- *$G[S]$ is connected,*
- *for any $v \in V$, we have that $v \in S$ or v is adjacent to a vertex of S and,*

S is of minimum size.

In the *node-weighted* version of the problem, in addition to G we are given a function $w : V \mapsto \mathbb{R}_+$. Moreover S is required to be minimum with respect to the sum of the weights of the vertices it contains, *i.e.,* S is minimizing $\sum_{v\in S} w(v)$.

A.2 Separators

We aim at providing a short overview on the separation property and its algorithmic implications. There is a rather long history (see [165]) of finding “good” ways to separate an instance into sub-instances of significantly smaller sizes, sometimes depending on the nature of the problem. We start with reviewing separators in graphs before presenting separators in low-dimensional Euclidean spaces.

A.2.1 The Separation Property in Graphs

For a graph $G = (V, E)$, we define a *separator* of G to be a subset of vertices S such that $G - S$ results in a graph which has at least two connected components. When solving an algorithmic problem, the different connected components of a graph can often be dealt with separately. Thus, a separator breaks the problem into smaller problems, allowing us to apply dynamic programming or divide-and-conquer techniques or, as we will see, ensure that local search algorithms perform well. The size of a separator is often defined by the number of vertices it contains.

To ensure that the sub-problems are much smaller we are mainly interested in balanced separators of small size. This ensures some properties that resemble the isoperimetric inequality of the Euclidean space. The isoperimetric inequality in the plane states that for any closed curve of perimeter L enclosing an area A, $4\pi A \leqslant L^2$. When the curve is a circle, the equality holds: $4\pi A = L^2$. Then, the important notion here is that the area enclosed is much bigger than the perimeter of the curve used to enclose it. This can be generalized to higher dimensions. In dimension 3 for example, for a given area A, the sphere is the simple closed surface of area A that encloses a region of maximum volume.

In the context of graphs, a separator that has size $o(n)$ and that separates the graphs into at least two connected components each of size $\Omega(n)$ yields some isoperimetric inequality for graphs. With this in mind, we say that a separator is *balanced* if it yields a graph such that each connected components has size at most cn for some constant $c < 1$.

Separators in Bounded Treewidth Graphs. The existence of small separators in graphs of bounded treewidth follow from the definition of tree-decomposition. The definitions and notations related to treewidth and graph of bounded treewidth are given in Appendix A.3. Given a tree-decomposition $(\mathcal{B}, \mathcal{V})$ of width w, for any two adjacent nodes N_i, N_j, $\mathcal{V}_i \cap \mathcal{V}_j$ is a separator of the graph. Since each bag contains at most $w + 1$ vertices, we have that the graph contains a separator of size at most $w + 1$ (even if we assume that no bag is a subset of another bag). It is known that one can find a balanced separator in this class of graph by moving along the tree and finding two adjacent nodes whose intersection forms a balanced separator.

The fact that for any two adjacent nodes N_i, N_j, $\mathcal{V}_i \cap \mathcal{V}_j$ is a separator of the graph can be shown as follows. Consider the two subtrees $\mathcal{T}_i$ and $\mathcal{T}_j$ resulting from the deletion of nodes N_i and N_j from $\mathcal{T}$. Assume toward contradiction that $\mathcal{V}_i \cap \mathcal{V}_j$ is not a separator and that no bag is contained into another. Thus there exist $u, v \notin \mathcal{V}_i \cap \mathcal{V}_j$ such that there is an edge (u, v) and u appears in a bag associated with a node of $\mathcal{T}_i$ and v appears in a bag associated with a node if $\mathcal{T}_j$. Since (u, v) is an edge there is a bag containing both of them. Thus, by the definition u or v appear in $\mathcal{V}_i \cap \mathcal{V}_j$, a contradiction.

Separators in Planar Graphs. In the following, we assume that each vertex is assign a *mass*. We say that a separator is balanced if its removal yields a graph such that each of its connected component has total mass at most 1/3. Moreover, we say that a mass assignment is *proper* if the mass of each vertex is in $[0, 1/3]$. In the following we will always assume that the mass assignment are proper.

The first separator theorem for planar graphs is due to Ungar [175] who proved that each planar graph with n vertices contains a balanced separator with $O(\sqrt{n \log n})$ vertices. Later, Lipton and Tarjan [133] improved the bound to $O(\sqrt{n})$ and gave a linear-time algorithm to compute it. The constant was later improved by Djidjev [71]. Note that this bound is tight up to a constant factor. There is a variety of algorithmic applications of the separator theorem. In a nutshell, this allows to apply a divide-and-conquer approach to the graph. It has lead to *e.g.,* efficient algorithms for finding shortest paths (see [104]), or approximation algorithms for the independent set problem (see [134]).

However, as we will see in Chapter 9, some problems require more structured balanced separators. Miller [143] was the first to introduced the notion of "cycle separator". He showed that one can find in any planar graph a balanced separator that forms a cycle in the graph. Later, Djidjev and Venkatesan [72] presented a linear-time algorithm that, given a 2-connected planar graph G with n vertices, returns a subgraph of G that is both a cycle and a separator of G of size at most $4\sqrt{n}$.

Separators in H-Minor Free Graphs. The tremendous implications of separator theorems for planar graphs have led researchers to look for separator theorems for more general classes of graphs. In 1990, Alon, Seymour, and Thomas [7] proved a separator theorem for the family of graphs excluding a fixed graph as a minor. They showed that given a graph H with h vertices there exists an algorithm that for any graph G that excludes H as a minor, any proper mass assignment to the faces, returns a balanced separator of G of size at most $\sqrt{nh^3}$.

Since for any surface Σ, the family of graphs that can be embedded on Σ excludes a finite set of graphs as minors, a separator theorem for graphs embedded on a surface can be deduced from the theorem of Alon, Seymour, and Thomas.

Thus, many algorithms for planar graphs can be applied directly to graphs excluding a fixed minor. However, some more structured separators, for example Miller's cycle separator theorem, have no equivalent in graphs excluding a fixed minor[1].

Separator theorems exist also for even more general classes of graphs such that graphs of bounded expansion. In this thesis we mainly focus on classes of graph excluding a fixed minor.

A.2.2 The Separation Property in Euclidean Space

We now aim at defining separators in the Euclidean setting. At an informal level, a separator splits a set of objects lying in a Euclidean space into two parts of roughly equal size. In

[1]However, there is no proof that none can exist so far.

the plane for example, a closed curve containing a constant fraction of the points in both its interior and exterior forms a separator in a way that is similar to the planar case: Making use of the isoperimetric inequality, one can find a curve whose length is sublinear with respect to the area it encloses, yielding a bound that resemble the bound of the planar graph separator.

Dissections of $\mathbb{R}^d$. One of the first use of Euclidean separators for algorithmic purposes is due to Karp [114], back in 1977. He introduced a way to partition a set of points in a Euclidean space of fixed dimension in order to provide an approximation algorithm to the traveling salesman problem (TSP) via a dynamic programming approach. The algorithm recursively finds a curve separating the instance into two sub-instances, solves the two sub-problems separately and combine them by paying (at most) the length of the separating curve. The approximation guarantee directly depends on the isoperimetric inequality: if the length of the curve is small compared to the length of the two sub-tours, the final tour is almost optimal.

In the 90s, Arora [9] and Mitchell [144] independently showed how to build more structured separators for TSP in order to improve the approximation ratio. They showed that it is possible to recursively divide a Euclidean space of fixed dimension using separators that are crossed only a few number of times by a near-optimal solution. This reduces the interface between two sub-problems induced by a separator to a constant number of bits of information, allowing the dynamic programming approach to efficiently compute the near-optimal solution.

Local Search and Separators. Separators were used more recently to show that local search algorithms perform well for some geometric problems. For example, given a set of points and disks in the plane, the problem of hitting all the disks using the minimum number of points can be well approximated using a simple local search algorithm (see [148]). The analysis relies on a planar graph that represents some of the possible local exchanges. Then it uses planar graph separators to partition the graph into locally-optimal regions and provides an analysis of the solution region by region.

Separators for Voronoi Diagrams. The connection between the isoperimetric inequality in the Euclidean setting and graphs separators looks even tighter in the light of the recent results of Bhattiprolu and Har-Peled [34]. Given a set of points P in $\mathbb{R}^d$, they show how to efficiently compute a set of points Z in $\mathbb{R}^d$ such that P can be partitioned into two sets P_1 and P_2 of roughly equal sizes satisfying the following condition: in the Voronoi diagram of $P \cup Z$, the cells of the points in P_1 do not touch the cells of the points in P_2. Thus, Z separates the points in P_1 from the points in P_2 in the Voronoi diagram. Moreover, they show that there exists such a set Z of size $|P|^{1-1/d}$.

A.3 Graph Theory

We use the standard notations of Diestel [69]. A *graph* $G = (V, E)$ is a pair (V, E) where $V \subseteq \{1, \dots, n\}$ is the set of *vertices* of G, and E is a multiset of (unordered or ordered) vertex pairs called *edges* of G. We usually denote by n the number of vertices of a graph G. Throughout this thesis we mainly consider undirected graphs.

For an edge $e = (u, v)$, we call u and v the *endpoints*. If $e = (u, u)$ we called it a *loop*. Two or more edges corresponding to one pair of vertices are called *parallel* edges. An edge $e = (u, v)$ is said to be *incident* to vertices u and v and u and v are said to be *adjacent* and *neighbors* of each other. The *degree* of vertex v is the number of edges incident to v A *path* is a graph $P = (V, E)$ of the form $V = \{x_0, x_1, \dots, x_k\}$ and $E = \{(x_0, x_1), (x_1, x_2), \dots, (x_{k-1}, x_k)\}$. Its *endpoints* are x_0 and x_k. A *cycle* is a graph $C = (V, E)$ of the form $V = \{x_0, x_1, \dots, x_k, x_0\}$ and $E = \{ (x_0, x_1), (x_1, x_2), \dots, (x_{k-1}, x_k), (x_k, x_0)\}$. Paths and cycles are said to be simple if the x_i are distinct. If the number of vertices of the path is ℓ, we say that the path has size ℓ.

The graph consisting of n vertices and that has an edge between every pair of vertices is called the *complete* graph and denoted K_n. A graph is bipartite if it does not contain any cycle of odd size. The vertices of a bipartite graph can be partitioned into two parts such that no edge connects vertices of the same part. The bipatite graph consisting of n vertices on one side and m vertices on the other and with an edge between each pair of vertices from diffrent sides is called the *complete bipartite* graph and denoted $K_{n,m}$.

A graph $H = (V', E')$ is a *subgraph* of a graph $G = (V, E)$ if both $V' \subseteq V$ and $E' \subseteq E$. In a graph $G = (V, E)$, vertices v and u are *connected* if there exists a subgraph of G that is path and such that v and u are its endpoints. In this case, we say that there exists a path between v and u. Connectivity induces an equivalency relationship on the vertices of G: A maximal set of vertices that are pairwise connected defines a *connected component* of G. If G has only one connected component we say that the graph is *connected*. More generally, if for any pair of vertices u, v there exists at least k disjoint paths from u to v, we say that G is k-connected.

For any graph $G = (V, E)$ and subgraph $H = (V', E')$, if E' contains all the edges (u, v) of E such that both $u, v \in V'$ then H is said to be an *induced* subgraph of G. In this case, we denote H by $G[V']$. We say that a graph $F = (V, E)$ is a *forest* if it does not contain any cycles as subgraph. A forest is said to be a *tree* if there is only one connected component. In a forest, we sometimes call a vertex a *node* and vertices of degree *leaves*. We say that a tree is binary if each non-leaf node has degree 3. We say that a tree is a binary rooted tree if each non-leaf node has degree 3 except for one non-leaf node of degree 2 (called the root). We say that a set of vertices $V' \subseteq V$ (resp. set of edges $E' \subseteq E$) is a cycle of G if there exists a subgraph $G' = (V'', E'')$ of G such that $V' = V''$ (resp. $E' = E''$) and G' is a cycle (and similarly for paths, trees, etc.). Given a graph $G = (V, E)$, we say that a tree $T = (V_t, E_t)$ is a *spanning-tree* if $V = V_t$ and $E_t \subseteq E$.

Throughout this thesis, we will be facing graphs together with *weight* functions associated with the edges of the graph. For each graph there will be a function $w : E \mapsto \mathbb{R}_+$. We slightly abuse notation and define $w(E')$ for any subset $E' \subseteq E$ as $w(E') = \sum_{e \in E'} w(e)$. When equipped with this weight function, we refer to the *length* of a path $P = (V', E')$ connecting two vertices by $w(E')$. We slightly abuse notation and refer to the length of P by $w(P)$ We then define a *distance* function dist : $V \times V \mapsto \mathbb{R}_+$ as follows. The distance between two vertices u and v is the minimum over all paths with endpoints u and v of the length of the path, *i.e.,* $\text{dist}(u, v) = \min_{\text{path } P \text{ between } u \text{ and } v} w(P)$.

In the case when no such path exists, the distance is said to be infinite. Otherwise the distance is finite since the weights are positive. For any pair of vertices u, v, each path of length $\text{dist}(u, v)$ is called a *shortest path* between u and v. We abuse notation and define $\text{dist}(U, S) = \min_{u \in U} \min_{v \in S} \text{dist}(u, v)$, for $U, S \subseteq V$.

We now define some operators on the edges of any graph $G = (V, E)$. The *contraction* of an edge $e = (u, v)$ consists in removing a vertices u, v and adding a new vertex v_e, which becomes adjacent to all the former neighbours of u and of v. The graph G where edge e is contracted is denote G/e. The contraction operation is commutative ($G/e/e' = G/e'/e$). We thus extend the notation to subsets: G/E' for $E' \subseteq E$ is the graph G where all the edges of E' are contracted.

The *deletion of an edge* e of graph $G = (V, E)$ results in a graph $G' = (V, E \backslash e)$. The *deletion of a vertex* v of graph $G = (V, E)$ results in a graph $G' = (V \backslash \{v\}, E')$ where E' is the set of edges of E that do not have v as their endpoints. We write $G - e$ and $G - v$ to denote the graphs resulting from the deletion of edge e and the deletion of vertex v respectively.

Given two graphs G and H, we say that H is a *minor* of G if it can be obtained from G by deleting zero or more vertices or edges and contracting zero or more edges.

A.3.1 Graph Classes

We introduce several classic families of graphs that appear throughout this thesis. Figure A.1 shows the inclusion relationships between the different classes.

A.3.1.1 Graphs of Bounded Treewidth

We now introduce a very useful parameter for graphs: the treewidth. Since algorithmic problems on graphs are often much more tractable when the input is assumed to be a tree, it is natural to define a parameter measuring how close a graph is to a tree. For example, the complete graph K_n is very far to be a tree and its treewidth is $n - 1$ whereas a tree or a cycle (which is a tree plus an edge) have treewidth 1 and 2 respectively.

We now give the formal definition of treewidth, as introduced by Robertson and Seymour [157, 158]. For this, we need to define a tree-decomposition of a graph.

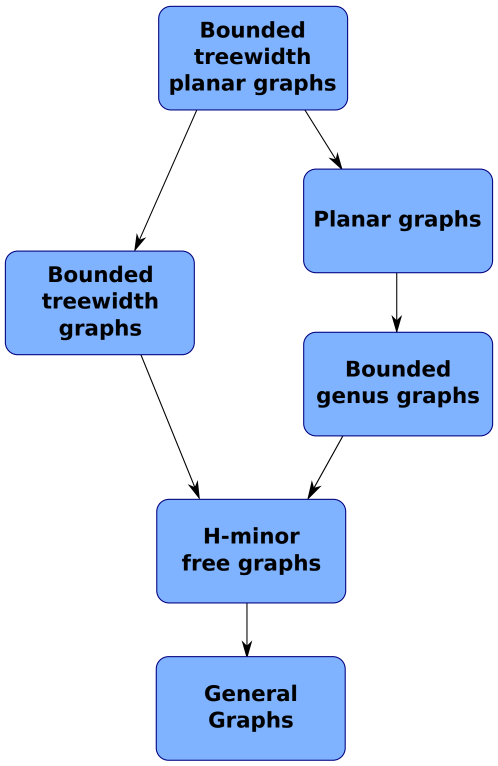


Figure A.1: The different graph classes appearing in this thesis. Arrows from class A to class B means that class A is included in class B.

Let $G = (V, E)$ be a graph. A *tree-decomposition* of G is a pair $(\mathcal{T}, \mathcal{B})$, where $\mathcal{B}$ is a collection of subsets of V called *bags* and $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is a tree, such that

- Each vertex appears in at least one bag. In other words, for all $v \in V$, $\exists B \in \mathcal{B}$ such that $v \in B$.
- For all edge (u, v) in E there exists a bag containing both u and v.
- For all i, the ith bag B_i of $\mathcal{B}$ is associated to the ith node N_i of $\mathcal{V}$ and we have that if two bags B_j, B_k contain vertex v, then each bag B_ℓ such that N_ℓ is in the path between N_j and N_k contains vertex v.

The width of the tree-decomposition is the cardinality of the largest bag minus 1. The treewidth of a graph is the minimum width over all tree-decompositions. In this thesis, we define a bounded treewidth graph as a graph whose treewidth is constant. Standard results show that there exists an efficient algorithm turning a minimum-width tree-decomposition to a tree-decomposition of same width whose tree is binary. When we refer to tree-decomposition, we will assume we will refer to a minimum-width tree-decomposition with binary tree.

Branchwidth. When dealing with some particular classes of graphs (for example graphs embedded on a surface), a useful decomposition that often encodes more structure than tree-decomposition (some topological structure in the context of graphs embedded on a surface) is the branch-decomposition of the graph.

To formally define the branchwidth, we need to define a branch-decomposition. Let $G = (V, E)$ be a graph. A *branch-decomposition* of G is an unrooted binary tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$ together with a bijection ϕ between the edges of G and the leaves of $\mathcal{T}$. For any edge $e \in \mathcal{E}$, define $\mathcal{T}_e^1 = (\mathcal{N}_e^1, \mathcal{E}_e^1)$ and $\mathcal{T}_e^2 = (\mathcal{N}_e^2, \mathcal{E}_e^2)$ the two subtrees of the forest $T - e$. Define $\text{mid}(e)$ to be the set of vertices that are the endpoints of both an edge in bijection with a leaf of $\mathcal{T}_e^1$ and an edge in bijection with a leaf to $\mathcal{T}_e^2$. More formally, $\text{mid}(e) = \{v \mid \exists (v, u), (v, w) \in E \text{ s.t } \phi((v, u)) \in \mathcal{N}_e^1 \text{ and } \phi((v, w)) \in \mathcal{N}_e^2\}$.

The width of a branch-decomposition $\mathcal{B} = (\mathcal{T}, \phi)$, denoted $w(\mathcal{B})$ is $\max_{e \in \mathcal{T}} |\text{mid}(e)|$. The *branchwidth* of a graph G is the minimum width of all the branch-decompositions. The branchwidth is closely related to the treewidth: Robertson and Seymour [160] showed that for any graph G, $bw(G) \leqslant tw(G) \leqslant 3bw(G)/2 - 1$.

A.3.1.2 Planar Graphs

Planar graphs is the class of graphs that can be drawn in the plane in a such a way that no two edges intersect in a point other than a common endpoint. It is a fairly natural class of graphs that dates back to at least Euler and his famous "seven bridges of Königsberg" problem. Indeed, since 2-dimensional objects are often used to represent our environment, practical instances of several algorithmic graph problems consists of planar graphs. For example, in the context of designing a network, the input is sometimes a graph whose edges are streets and vertices are intersections, yielding a planar graph.

In order to formally define planar graphs, we introduce the notion of *embedding*. An embedding of a graph $G = (V, E)$ on a surface Σ is a continuous injective mapping from the vertices and edges of E, considered as a topological space, to the surface Σ. Considering V, E as a topological leads the mapping to map vertices to distinc points and edges to disjoint paths, intersecting only at their endpoints. Every definition in this section can be generalized to the case of graphs embedded on a surface. A *combinatorial map* of an embedded graph is the combinatorial description of its embedding, namely the cyclic ordering of the edges around each vertex.

A graph is said *planar* if there it can be embedded into $\mathbb{R}^2$. We define the *faces* of G to be the set of components of $\mathbb{R}^2 \backslash G$. We extend the notion of adjacency and incidence to faces: Vertices and edges of G are said to be incident to a face f if they are part of the cycle of G forming the boundary of f. Two faces are adjacent if their boundaries share an edge. We say that the degree or length of a face is the number of edges in its boundary. Observe that there is a unique infinite face. We say that a planar graphs is triangulated if each face has degree three.

A classical notion formula by Euler relates the number of faces, vertices, edges and connected components of a planar graph. For any planar graph G, let f, n, e, c denote the number of faces, vertices, edges and connected components of G, we have

$$n + f - e - c = 1 \qquad \textit{(Euler's formula)}$$

We now consider planar graphs with no parallel edges and no loop and with at least three vertices. Observe that each edge appears in at most two faces (one for each component of $\mathbb{R}^2$ that it is the boundary of). Thus we have that the sum of the degrees (length of the boundaries) of the faces is at most $2e$. Moreover, each face has degree at least three in this context. Thus we have that this sum is at least $3f$, and so, $2e \geqslant 3f$. Replacing in Euler's formula yields

$$e \leqslant 3v - 6. \tag{A.1}$$

Moreover, observing that the sum of the degrees is at most $2e$, it is easy to deduce that for any planar graph with no parallel edges and no loop, there exists a vertex of degree at most 5.

We now turn to a crucial notion in planar graphs that we will use to levarage the topological properties of embedded planar graphs. The *dual* $G^* = (V^*, E^*)$ of an embedded planar graph $G = (V, E)$ is the graph whose vertex set consists of the set of faces of G, and such that for each edge e in E incident to faces f_1, f_2, E^* contains an edge connecting the vertices of V^* corresponding to faces f_1 and f_2. Note that when $f_1 = f_2$ this results in a loop. The dual G^* is planar, and the dual of G^* is isomorphic to G if G is connected. Throughout this thesis, when we refer to planar graphs we mean planar graphs without loop and parallel edges unless stated otherwise.

We finally consider the characterization of planar graphs. Perhaps surprisingly, there exists a beautiful combinatorial characterization of planar graphs dating back to Kuratowski in 1930 [128] using the definition of minor described above.

Theorem A.7 (Kuratowski [128]). *A graph is planar if and only if it does not have $K_{3,3}$ or K_5 as a minor.*

Given a planar graph G, the *medial graph* M_G is the embedded graph obtained by placing a vertex v_e for every edge e of G, and connecting the vertices v_e and v_e' with an edge whenever e and e' are adjacent on a face of G. The *barycentric subdivision* of an embedded graph G is the embedded graph obtained by adding a vertex on each edge and on each face and an edge between every such face-vertex and its adjacent (original) vertices and edge vertices.

A.3.1.3 Bounded Genus Graphs

Planar graphs are commonly used to model a wide array of discrete structures, and in many cases it is necessary to consider embeddings into surfaces instead of the plane or the sphere. For example, many instances of network design actually feature some crossings, coming from tunnels or overpasses, which are appropriately modeled by a surface of small genus. In other settings, such as in computer graphics or computer-aided design, we are looking for a discrete model for objects which inherently display a non-trivial topology (e.g., holes), and graphs embedded on surfaces are the natural tool for that. From a more theoretical point of view, the graph structure theorem of Robertson and Seymour showcases a very strong connection between graphs embedded on surfaces and minor-closed families of graphs.

We will use the classic notions on graphs embedded on surfaces, for more formal definitions and background see the textbook of Mohar and Thomassen [145].

We focus on compact surfaces without boundary. Let Σ be such a surface. Euler's formula can be adapted to graphs embedded on surfaces in order to provide define the *Euler genus* of Σ: For any connected graph G with n vertices and e edges that is embedded in Σ with f facial walks, the number $g = 2 - n + e - f$ is independent of G and is called the Euler genus of Σ. A graph is planar if and only if it has Euler genus zero.

In this setting we rather consider graphs with loops and parallel edges since the topological properties of parallel edges and loops might be of interest for our problems (for example, a loop might be non-contractible and so, be used in the solution some network design problems).

We say that an embedding in *cellular* if each face is homeomorphic to a disk. We define a *noose* to be an embedding of the circle $\mathbb{S}^1$ on Σ which intersects G only at its vertices.

An embedding of a graph G on a surface is said to be *polyhedral* if G is 3-connected and the smallest length of a non-contractible noose is at least 3 or if G is a clique and it has at most 3 vertices. In particular, a polyhedral embedding is cellular. If G is a graph embedded on Σ, the surface Σ' obtained by *cutting* Σ along G is the disjoint union of the faces of G, it is a (a priori disconnected) surface with boundary. When we cut a surface along a set of nooses, viewed as a graph, the resulting connected components will be called *regions*.

We now introduce the generalization of Kuratowski's theorem for graphs of bounded genus, due to Robertson and Seymour [159]

Theorem A.8 (Robertson and Seymour [159])**.** *For any surface Σ, there exists a finite set of graphs $T(\Sigma)$ such that any graph can be embedded on Σ if and only if it does not have any graph in $T(\Sigma)$ as a minor.*

The dual graph, medial graph, and barycentric subdivision generalize to graphs embedded on surface (see [145] for more details).

We remark that planar graphs and graphs of bounded genus do not have bounded treewidth and that graphs of bounded treewidth are not necessarily planar.

A.3.1.4 H-Minor Free Graphs

We turn to a class of graphs generalizing both bounded genus graphs and graphs of bounded treewidth.

A (not necessarily finite) family of graphs is said *minor-closed* if for any graph G of the family, any minor of G also belongs to the family. A celebrated theorem by Robertson and Seymour [162] shows that there is a so-called forbidden minor characterization for any minor-closed family of graphs. In other words, like in the case of graphs of bounded genus, for any minor-closed family of graphs there exists a finite set T of graphs such that any graph G is in the family if and only if it does not have any graph in T as a minor. We define the family of *H-minor free graphs* as the set of graphs that do not contain H as a minor. We will see in the next section the structural implications of the theorem of Robertson and Seymour for this family.

A.4 Computational Complexity

Throughout this thesis, we aim at designing approximation algorithms for optimization problems. Given a problem Π and an instance I of Π, we usually denote by OPT(Π, I) the value of an optimal solution for instance I of problem Π. When Π and I are clear from the context we just denote it OPT. We say that an algorithm is an *α-approximation* for a minimization problem Π if, for any instance I of Π, the algorithm returns a solution of cost at most αOPT(Π, I) and its running time is polynomial in the size of the instance. We say that an algorithm is *exact* if it returns a solution of cost OPT(Π, I)

In the classic memory model (as opposed to data streams considered in Chapter 5), no polynomial-time exact algorithm is known for any NP-Hard problem. Thus, since polynomial-time is often necessary for practical purposes it is natural to look for polynomial-time approximation algorithms.

When working on an NP-Hard problem, the best result one can hope for is a polynomial-time approximation scheme (PTAS). A *polynomial-time approximation scheme* for a problem Π is a collection of algorithms $\mathcal{A}$ such that for any positive constant ε, there exists an algorithm $A(\varepsilon) \in \mathcal{A}$ that is a $(1+\varepsilon)$-approximation for Π and runs in polynomial-time when ε is a fixed constant. If for any positive constant ε, the running time of $A(\varepsilon)$ is $O(f(\varepsilon)n^c)$ where c is a fixed constant independent of ε and f is any computable function then $\mathcal{A}$ is said to be an *efficient* polynomial-time approximation scheme (EPTAS).

However, among the NP-Hard problem, some of them are known to be also hard to approximate. For example, a problem Π is said to be *APX*-Hard if there exists an integer $a > 1$ such that there is no a-approximation algorithm for Π unless P = NP. Example of such problems are the k-median and k-means problems for general instances.

A.5 Hardness Result for an FPTAS for the Planar k-Clustering Problem

Definition A.9. *An approximation scheme is said to be a fully polynomial approximation scheme (FPTAS) if for any constant $\varepsilon > 0$, it achieves a $(1+\varepsilon)$-approximation in time poly(ε) poly(n).*

Lemma A.10. *There is no FPTAS for k-clustering problem even if the input consists of planar graphs unless P = NP.*

Proof. Meggido and Supowit [140] showed that the k-median problem is NP-hard even when the inputs consist of planar graphs[1] and the distances are integers in $\{1, 2, ..., M\}$, where $M = \text{poly}(n)$ and n is the number of vertices of the planar graph (thus the size of the instance is $\text{poly}(n)$).

We show the lemma by contradiction. Assume toward contradiction that an FPTAS exists. Given any $\varepsilon > 0$, it computes a $(1+\varepsilon)$-approximate solution to the problem in time poly$(\varepsilon)poly(n)$.

Now, observe that the value of any optimal solution, is at most $n \cdot M$ (since the objective function is the sum of the distances from each client to its closest center, which is at most M). Moreover, the cost difference between two solutions is at least 1 (because the distances are integers).

Now, running the FPTAS for $\varepsilon < n \cdot M$ yields a solution of cost at most $(1+\varepsilon)\text{OPT} < \text{OPT} + \text{OPT}/(nM) \leqslant \text{OPT} + 1$, from the above observation. Thus, the cost of the solution obtained by the FPTAS is less than OPT $+ 1$ and so at most OPT (and thus exactly OPT) since the cost difference between two solutions is at least 1.

Finally, since $n \cdot M$ is polynomial in the size of the input, we have an exact algorithm (an algorithm computing an optimal solution) running in polynomial time, a contradiction unless P = NP. □

[1]in fact for the even less general case of rectilinear inputs.

Bibliography

[1] *IBM ILOG CPLEX Optimizer*. urlhttp://www-01.ibm.com/software/integration/optimization/cplex-optimizer/, 2010. Cited on page 97

[2] E. AARTS AND J. K. LENSTRA, eds., *Local Search in Combinatorial Optimization*, John Wiley & Sons, Inc., New York, NY, USA, 1st ed., 1997. Cited on page 7, 29

[3] D. ACHLIOPTAS AND F. MCSHERRY, *On spectral learning of mixtures of distributions*, in Learning Theory, 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005, Proceedings, 2005, pp. 458–469. Cited on page 29

[4] P. K. AGARWAL, J. MATOUSEK, AND S. SURI, *Farthest neighbors, maximum spanning trees and related problems in higher dimensions*, Comput. Geom., 1 (1991), pp. 189–201. Cited on page 29

[5] P. K. AGARWAL AND R. SHARATHKUMAR, *Streaming algorithms for extent problems in high dimensions*, Algorithmica, 72 (2015), pp. 83–98. Cited on page 29

[6] A. A. AGEEV, *An approximation scheme for the uncapacitated facility location problem on planar graphs*, in Proceedings of the 12th International Baikal Workshop, 2001, pp. 9–13. Cited on page 9, 20

[7] N. ALON, P. D. SEYMOUR, AND R. THOMAS, *A separator theorem for graphs with an excluded minor and its applications*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA, 1990, pp. 293–299. Cited on page 41, 165

[8] E. M. ARKIN, M. M. HALLDÓRSSON, AND R. HASSIN, *Approximating the tree and tour covers of a graph*, Inf. Process. Lett., 47 (1993), pp. 275–282. Cited on page 151

[9] S. Arora, *Polynomial time approximation schemes for euclidean TSP and other geometric problems*, in 37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996, 1996, pp. 2–11. Cited on page 113, 117, 166

[10] ——, *Nearly linear time approximation schemes for euclidean TSP and other geometric problems*, in 38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997, 1997, pp. 554–563. Cited on page 117

[11] S. Arora, P. Raghavan, and S. Rao, *Approximation schemes for euclidean* k*-medians and related problems*, in Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, 1998, pp. 106–113. Cited on page 10

[12] ——, *Approximation schemes for Euclidean* k*-medians and related problems*, in Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, 1998, pp. 106–113. Cited on page 28, 114

[13] D. Arthur, B. Manthey, and H. Röglin, *Smoothed analysis of the k-means method*, J. ACM, 58 (2011), p. 19. Cited on page 28

[14] D. Arthur and S. Vassilvitskii, *k-means++: the advantages of careful seeding*, in Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007, 2007, pp. 1027–1035. Cited on page 28

[15] ——, *Worst-case and smoothed analysis of the ICP algorithm, with an application to the k-means method*, SIAM J. Comput., 39 (2009), pp. 766–782. Cited on page 28

[16] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, *Local search heuristics for k-median and facility location problems*, SIAM J. Comput., 33 (2004), pp. 544–562. Cited on page 7, 20, 29, 34, 44, 45, 65

[17] P. Awasthi, A. Blum, and O. Sheffet, *Stability yields a PTAS for k-median and k-means clustering*, in 51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA, 2010, pp. 309–318. Cited on page 9, 10, 23, 24, 28, 66, 67, 76

[18] ——, *Center-based clustering under perturbation stability*, Inf. Process. Lett., 112 (2012), pp. 49–54. Cited on page 24, 28

[19] P. Awasthi, M. Charikar, R. Krishnaswamy, and A. K. Sinop, *The hardness of approximation of Euclidean k-means*, in 31st International Symposium on

Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands, 2015, pp. 754–767. Cited on page 9, 28, 44

[20] P. AWASTHI AND O. SHEFFET, *Improved spectral-norm bounds for clustering*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings, 2012, pp. 37–49. Cited on page 10, 24, 29, 80

[21] B. BABCOCK, M. DATAR, R. MOTWANI, AND L. O'CALLAGHAN, *Maintaining variance and k-medians over data stream windows*, in Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA, 2003, pp. 234–243. Cited on page 30

[22] V. BAFNA, P. BERMAN, AND T. FUJITO, *Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs*, in Algorithms and Computations, Springer, 1995, pp. 142–151. Cited on page 12, 118, 153

[23] B. BAHMANI, B. MOSELEY, A. VATTANI, R. KUMAR, AND S. VASSILVITSKII, *Scalable k-means++*, PVLDB, 5 (2012), pp. 622–633. Cited on page 97

[24] B. BAKER, *Approximation algorithms for NP-complete problems on planar graphs*, J. of the ACM, 41 (1994), pp. 153–180. Cited on page 11, 20, 114, 119

[25] M. BALCAN, A. BLUM, AND A. GUPTA, *Approximate clustering without the approximation*, in Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009, 2009, pp. 1068–1077. Cited on page 6, 24

[26] ——, *Clustering under approximation stability*, J. ACM, 60 (2013), p. 8. Cited on page 24

[27] M. BALCAN AND Y. LIANG, *Clustering under perturbation resilience*, SIAM J. Comput., 45 (2016), pp. 102–155. Cited on page 10, 24, 25, 28

[28] S. BANDYAPADHYAY AND K. R. VARADARAJAN, *On variants of k-means clustering*, CoRR, abs/1512.02985 (2015). Cited on page 29

[29] R. BAR-YEHUDA, D. GEIGER, J. NAOR, AND R. M. ROTH, *Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference*, SIAM J. on Computing, 27 (1998), pp. 942–959. Cited on page 153

[30] M. Bateni, A. Bhaskara, S. Lattanzi, and V. S. Mirrokni, *Distributed balanced clustering via mapping coresets*, in Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, 2014, pp. 2591–2599. Cited on page 29

[31] J. Beardwood, J. H. Halton, and J. M. Hammersley, *The shortest path through many points*, Mathematical Proceedings of the Cambridge Philosophical Society, 55 (1959), pp. 299–327. Cited on page 124

[32] S. Ben-David and L. Reyzin, *Data stability in clustering: A closer look*, Theor. Comput. Sci., 558 (2014), pp. 51–61. Cited on page 28

[33] P. Berman and G. Yaroslavtsev, *Primal-dual approximation algorithms for node-weighted network design in planar graphs*, in Approximation, Randomization, and Combinatorial Optimization. Alg. and Tech., Springer, 2012, pp. 50–60. Cited on page 150

[34] V. V. S. P. Bhattiprolu and S. Har-Peled, *Separating a voronoi diagram*, CoRR, abs/1401.0174 (2014). Cited on page 23, 42, 166

[35] Y. Bilu, A. Daniely, N. Linial, and M. E. Saks, *On the practically interesting instances of MAXCUT*, in 30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany, 2013, pp. 526–537. Cited on page 24, 28

[36] Y. Bilu and N. Linial, *Are stable instances easy?*, Combinatorics, Probability & Computing, 21 (2012), pp. 643–660. Cited on page 6, 9, 24, 28

[37] D. Binkele-Raible and H. Fernau, *A faster exact algorithm for the directed maximum leaf spanning tree problem*, in Computer Science–Theory and Applications, Springer, 2010, pp. 328–339. Cited on page 151

[38] G. E. Blelloch and K. Tangwongsan, *Parallel approximation algorithms for facility-location problems*, in SPAA 2010: Proceedings of the 22nd Annual ACM Symposium on Parallelism in Algorithms and Architectures, Thira, Santorini, Greece, June 13-15, 2010, 2010, pp. 315–324. Cited on page 29

[39] H. L. Bodlaender, *On linear time minor tests and depth first search*, in W. on Algorithms and Data Structures, WADS, 1989, pp. 577–590. Cited on page 151

[40] H. L. Bodlaender, A. Grigoriev, and A. M. C. A. Koster, *Treewidth lower bounds with brambles*, Algorithmica, 51 (2008), pp. 81–98. Cited on page 136

[41] H. L. BODLAENDER AND D. M. THILIKOS, *Constructive linear time algorithms for branchwidth*, in Automata, Languages and Programming, Springer, 1997, pp. 627–637. Cited on page 131

[42] G. BORRADAILE, E. D. DEMAINE, AND S. TAZARI, *Polynomial-time approximation schemes for subset-connectivity problems in bounded-genus graphs*, Algorithmica, 68 (2014), pp. 287–311. Cited on page 119

[43] V. BRAVERMAN, H. LANG, K. LEVIN, AND M. MONEMIZADEH, *Clustering on sliding windows in polylogarithmic space*, in 35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India, 2015, pp. 350–364. Cited on page 30

[44] ——, *Clustering problems on sliding windows*, in Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, 2016, pp. 1374–1390. Cited on page 30

[45] V. BRAVERMAN, A. MEYERSON, R. OSTROVSKY, A. ROYTMAN, M. SHINDLER, AND B. TAGIKU, *Streaming k-means on well-clusterable data*, in Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011, 2011, pp. 26–40. Cited on page 28

[46] J. BYRKA, T. PENSYL, B. RYBICKI, A. SRINIVASAN, AND K. 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 '15, Philadelphia, PA, USA, 2015, Society for Industrial and Applied Mathematics, pp. 737–756. Cited on page 10, 28

[47] S. CABELLO, É. COLIN DE VERDIÈRE, AND F. LAZARUS, *Algorithms for the edge-width of an embedded graph*, Computational Geometry: Theory and Applications, 45 (2012), pp. 215–224. Cited on page 144

[48] E. W. CHAMBERS, É. COLIN DE VERDIÈRE, J. ERICKSON, F. LAZARUS, AND K. WHITTLESEY, *Splitting (complicated) surfaces is hard*, Comput. Geom., (2008), pp. 94–110. Cited on page 142

[49] T. M. CHAN AND S. HAR-PELED, *Approximation algorithms for maximum independent set of pseudo-disks*, Discrete & Computational Geometry, 48 (2012), pp. 373–392. Cited on page 40

[50] T. M. CHAN AND B. S. SADJAD, *Geometric optimization problems over sliding windows*, Int. J. Comput. Geometry Appl., 16 (2006), pp. 145–158. Cited on page 27, 30

[51] B. CHANDRA, H. J. KARLOFF, AND C. A. TOVEY, *New results on the old k-opt algorithm for the TSP*, in Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. 23-25 January 1994, Arlington, Virginia., 1994, pp. 150–159. Cited on page 118

[52] M. CHARIKAR, C. CHEKURI, T. FEDER, AND R. MOTWANI, *Incremental clustering and dynamic information retrieval*, in Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997, 1997, pp. 626–635. Cited on page 30

[53] M. CHARIKAR AND S. GUHA, *Improved combinatorial algorithms for facility location problems*, SIAM J. Comput., 34 (2005), pp. 803–824. Cited on page 29

[54] M. CHARIKAR, L. O'CALLAGHAN, AND R. PANIGRAHY, *Better streaming algorithms for clustering problems*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA, 2003, pp. 30–39. Cited on page 30

[55] C. CHEKURI, S. KHANNA, AND B. SHEPHERD, *Edge-disjoint paths in planar graphs*, in Symp. on Foundations of Computer Science, 2004, pp. 71–80. Cited on page 136

[56] J. CHEN, F. V. FOMIN, Y. LIU, S. LU, AND Y. VILLANGER, *Improved algorithms for feedback vertex set problems*, J. of Comput. and Syst. Sci., 74 (2008), pp. 1188–1198. Cited on page 153

[57] X. CHENG, X. HUANG, D. LI, W. WU, AND D.-Z. DU, *A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks*, Networks, 42 (2003), pp. 202–208. Cited on page 148

[58] N. CHRISTOFIDES, *Worst-case analysis of a new heuristic for the travelling salesman problem*, tech. rep., DTIC Document, 1976. Cited on page 117

[59] M. B. COHEN, S. ELDER, C. MUSCO, C. MUSCO, AND M. 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. Cited on page 76, 77

[60] A. COJA-OGHLAN, *Graph partitioning via adaptive spectral techniques*, Combinatorics, Probability & Computing, 19 (2010), pp. 227–284. Cited on page 29

[61] W. J. COOK, *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*, Princeton University Press, 2012. Cited on page 5

[62] G. CROES, *A method for solving traveling salesman problems*, Operations Research, 6 (1958), pp. 791–812. Cited on page 117

[63] M. CYGAN, J. NEDERLOF, M. PILIPCZUK, M. PILIPCZUK, J. M. VAN ROOIJ, AND J. O. WOJTASZCZYK, *Solving connectivity problems parameterized by treewidth in single exponential time*, in Symp. on Foundations of Computer Science, 2011, pp. 150–159. Cited on page 148

[64] G. DANTZIG, R. FULKERSON, AND S. JOHNSON, *Solution of a large-scale traveling-salesman problem*, Journal of the operations research society of America, 2 (1954), pp. 393–410. Cited on page 113

[65] S. DASGUPTA AND Y. FREUND, *Random projection trees for vector quantization*, IEEE Trans. Information Theory, 55 (2009), pp. 3229–3242. Cited on page 28

[66] E. D. DEMAINE AND M. HAJIAGHAYI, *Bidimensionality: new connections between FPT algorithms and PTASs*, in ACM-SIAM Symp. on Discrete Algorithms, 2005, pp. 590–601. Cited on page 11, 12, 114, 116, 118, 119, 129, 148, 152, 153, 156

[67] E. D. DEMAINE, M. T. HAJIAGHAYI, AND P. N. KLEIN, *Node-weighted Steiner tree and group Steiner tree in planar graphs*, ACM Tr. on Algorithms, 10 (2014), pp. 1–20. Cited on page 150

[68] D. B. DIAS, R. C. MADEO, T. ROCHA, H. H. BÍSCARO, AND S. M. PERES, *Hand movement recognition for brazilian sign language: a study using distance-based neural networks*, in Neural Networks, 2009. IJCNN 2009. International Joint Conference on, IEEE, 2009, pp. 697–704. Cited on page 98

[69] R. DIESTEL, *Graph Theory*, Electronic library of mathematics, Springer, 2006. Cited on page 167

[70] DIMACS, *8th dimacs implementation challenge: The traveling salesman problem*, 2000. Cited on page 5, 8, 114

[71] H. N. DJIDJEV, *On the problem of partitioning planar graphs*, SIAM Journal on Algebraic Discrete Methods, 3 (1982), pp. 229–240. Cited on page 165

[72] N. H. DJIDJEV AND M. S. VENKATESAN, *Reduced constants for simple cycle graph separation*, Acta Informatica, 34 (1997), pp. 231–243. Cited on page 165

[73] F. DORN, E. PENNINKX, H. L. BODLAENDER, AND F. V. FOMIN, *Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions*, Algorithmica, 58 (2010), pp. 790–810. Cited on page 131, 132

[74] R. J. DOUGLAS, *NP-completeness and degree restricted spanning trees*, Discrete Mathematics, (1992), pp. 41 – 47. Cited on page 151

[75] R. G. DOWNEY AND M. R. FELLOWS, *Parameterized computational feasibility*, Springer, 1995. Cited on page 151

[76] R. O. DUDA, P. E. HART, ET AL., *Pattern classification and scene analysis*, vol. 3, Wiley New York, 1973. Cited on page 98

[77] D. EISENSTAT, P. N. KLEIN, AND C. 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, 2014, pp. 617–627. Cited on page 9, 20

[78] D. EPPSTEIN, *Diameter and treewidth in minor-closed graph families*, Algorithmica, 27 (2000), pp. 275–291. Cited on page 130

[79] D. EPPSTEIN, *Dynamic generators of topologically embedded graphs*, in ACM-SIAM Symp. on Discrete Algorithms, 2003, pp. 599–608. Cited on page 143

[80] J. ERICKSON, *Graph separators*, 2009. Available at `jeffe.cs.illinois.edu/teaching/comptop/2009/notes/separators.pdf`. Cited on page 143

[81] J. ERICKSON AND K. WHITTLESEY, *Greedy optimal homotopy and homology generators*, in ACM-SIAM Symp. on Discrete Algorithms, 2005, pp. 1038–1046. Cited on page 143

[82] J. FEIGENBAUM, S. KANNAN, AND J. ZHANG, *Computing diameter in the streaming and sliding-window models*, Algorithmica, 41 (2004), pp. 25–41. Cited on page 27, 29, 83

[83] D. FELDMAN AND M. 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. Cited on page 28

[84] B. FISCHER, J. SCHUMANN, W. BUNTINE, AND A. G. GRAY, *Automatic derivation of statistical algorithms: The em family and beyond*, in Advances in Neural Information Processing Systems, 2002, pp. 673–680. Cited on page 98

[85] R. A. FISHER, *The use of multiple measurements in taxonomic problems*, Annals of eugenics, 7 (1936), pp. 179–188. Cited on page 98

[86] G. N. FREDERICKSON, *Fast algorithms for shortest paths in planar graphs, with applications*, SIAM J. Comput., 16 (1987), pp. 1004–1022. Cited on page 22, 41, 146

[87] C. B. FREY AND M. A. OSBORNE, *The future of employment: how susceptible are jobs to computerisation*, Retrieved September, 7 (2013), p. 2013. Cited on page 3

[88] A. FRIEZE AND W. PEGDEN, *Separating subadditive euclidean functionals*, in Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, New York, NY, USA, 2016, ACM, pp. 22–35. Cited on page 124

[89] Z. FRIGGSTAD, M. REZAPOUR, AND M. R. SALAVATIPOUR, *Local search yields a ptas for k-means in doubling metrics*, CoRR, (2016). Cited on page 29

[90] T. FUJITO, *On approximability of the independent/connected edge dominating set problems*, Inf. Process. Lett., 79 (2001), pp. 261–266. Cited on page 12, 118, 152

[91] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1979. Cited on page 148

[92] M. X. GOEMANS AND D. P. WILLIAMSON, *Primal-dual approximation algorithms for feedback problems in planar graphs*, Combinatorica, 18 (1998), pp. 37–59. Cited on page 153

[93] S. GUHA, *Tight results for clustering and summarizing data streams*, in Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings, 2009, pp. 268–275. Cited on page 29, 30, 90

[94] S. GUHA AND S. KHULLER, *Approximation algorithms for connected dominating sets*, Algorithmica, 20 (1998), pp. 374–387. Cited on page 148

[95] ——, *Greedy strikes back: Improved facility location algorithms*, J. Algorithms, 31 (1999), pp. 228–248. Cited on page 28

[96] ——, *Improved methods for approximating node weighted Steiner trees and connected dominating sets*, Inf. Comput., 150 (1999), pp. 57–74. Cited on page 12, 118, 149

[97] S. GUHA, A. MEYERSON, N. MISHRA, R. MOTWANI, AND L. O'CALLAGHAN, *Clustering data streams: Theory and practice*, IEEE Trans. Knowl. Data Eng., 15 (2003), pp. 515–528. Cited on page 29

[98] J. GUO, R. NIEDERMEIER, AND S. WERNICKE, *Parameterized complexity of vertex cover variants*, Theory of Computing Systems, 41 (2007), pp. 501–520. Cited on page 151, 152

[99] A. GUPTA AND K. TANGWONGSAN, *Simpler analyses of local search algorithms for facility location*, CoRR, abs/0809.2554 (2008). Cited on page 29, 33, 34

[100] V. GURUSWAMI AND P. INDYK, *Embeddings and non-approximability of geometric problems*, in Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA., 2003, pp. 537–538. Cited on page 28

[101] S. HAR-PELED AND A. KUSHAL, *Smaller coresets for k-median and k-means clustering*, Discrete & Computational Geometry, 37 (2007), pp. 3–19. Cited on page 28

[102] S. HAR-PELED AND S. 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. Cited on page 28

[103] S. HAR-PELED AND K. QUANRUD, *Approximation algorithms for polynomial-expansion and low-density graphs*, in Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings, 2015, pp. 717–728. Cited on page 40

[104] M. R. HENZINGER, PH. N. KLEIN, S. RAO, AND S. SUBRAMANIAN, *Faster shortest-path algorithms for planar graphs*, J. of Comput. and Syst. Sci., 55 (1997), pp. 3–23. Cited on page 136, 165

[105] K. JAIN, M. MAHDIAN, AND A. SABERI, *A new greedy approach for facility location problems*, in Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada, 2002, pp. 731–740. Cited on page 28

[106] K. JAIN AND V. VAZIRANI, *Approximation algorithms for metric facility location and* k*-median problems using the primal-dual schema and Lagrangian relaxation*, J. ACM, 48 (2001), pp. 274–296. Cited on page 28

[107] R. JAISWAL AND N. GARG, *Analysis of k-means++ for separable data*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Work-

shop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings, 2012, pp. 591–602. Cited on page 24, 28

[108] D. S. JOHNSON AND L. A. MCGEOCH, *The traveling salesman problem : A case study in local optimization*, Local Search in Combinatorial Optimization, 1 (1997), pp. 215–310. Cited on page 95, 118

[109] D. S. JOHNSON AND L. A. MCGEOCH, *The Traveling Salesman Problem and Its Variations*, Springer US, Boston, MA, 2007, ch. Experimental Analysis of Heuristics for the STSP, pp. 369–443. Cited on page 6, 113, 114

[110] W. B. JOHNSON AND J. LINDENSTRAUSS, *Extensions of Lipschitz mapping into Hilbert space*, in Conf. in modern analysis and probability, vol. 26 of Contemporary Mathematics, American Mathematical Society, 1984, pp. 189–206. Cited on page 74

[111] T. KANUNGO, D. M. MOUNT, N. S. NETANYAHU, C. D. PIATKO, R. SILVERMAN, AND A. Y. WU, *An efficient k-means clustering algorithm: Analysis and implementation*, IEEE TPAMI, 24 (2002), pp. 881–892. Cited on page 97

[112] ——, *A local search approximation algorithm for k-means clustering*, Comput. Geom., 28 (2004), pp. 89–112. Cited on page 7, 10, 20, 21, 29, 44, 97, 102

[113] ——, *Efficient algorithms for k-means clustering*, 2008. Cited on page 97

[114] R. M. KARP, *Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane*, Mathematics of Operations Research, 2 (1977), pp. 209–224. Cited on page 115, 118, 125, 127, 166

[115] M. KERBER AND S. RAGHVENDRA, *Approximation and streaming algorithms for projective clustering via random projections*, in Proceedings of the 27th Canadian Conference on Computational Geometry, CCCG 2015, Kingston, Ontario, Canada, August 10-12, 2015, 2015. Cited on page 75

[116] S. KIM AND H. AHN, *An improved data stream algorithm for clustering*, Comput. Geom., 48 (2015), pp. 635–645. Cited on page 30

[117] P. N. KLEIN, *A linear-time approximation scheme for planar weighted TSP*, in Symp. on Foundations of Computer Science, 2005, pp. 647–656. Cited on page 11, 114, 119, 130

[118] ——, *A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights*, SIAM J. on Computing, 37 (2008), pp. 1926–1952. Cited on page 114, 119, 130

[119] P. N. KLEIN, S. MOZES, AND C. SOMMER, *Structured recursive separator decompositions for planar graphs in linear time*, in ACM Symp. on Theory of Computing, 2013, pp. 505–514. Cited on page 138, 146

[120] D. E. KNUTH, *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997. Cited on page 161

[121] ——, *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997. Cited on page 161

[122] ——, *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998. Cited on page 161

[123] S. G. KOLLIOPOULOS AND S. RAO, *A nearly linear-time approximation scheme for the euclidean k-median problem*, SIAM J. Comput., 37 (2007), pp. 757–782. Cited on page 10, 28

[124] J. KÖNEMANN, G. KONJEVOD, O. PAREKH, AND A. SINHA, *Improved approximations for tour and tree covers*, in Approximation Algorithms for Combinatorial Optimization, Springer, 2000, pp. 184–193. Cited on page 12, 118, 151

[125] M. R. KORUPOLU, C. G. PLAXTON, AND R. RAJARAMAN, *Analysis of a local search heuristic for facility location problems*, J. Algorithms, 37 (2000), pp. 146–188. Cited on page 29

[126] A. KUMAR AND R. 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. Cited on page 9, 24, 25, 26, 29, 80

[127] A. KUMAR, Y. SABHARWAL, AND S. SEN, *Linear-time approximation schemes for clustering problems in any dimensions*, J. ACM, 57 (2010). Cited on page 28

[128] C. KURATOWSKI, *Sur le problème des courbes gauches en topologie*, Fundamenta Mathematicae, 15 (1930), pp. 271–283. Cited on page 171

[129] S. LI AND O. SVENSSON, *Approximating k-median via pseudo-approximation*, in Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013, 2013, pp. 901–910. Cited on page 10, 28

[130] M. LICHMAN, *UCI machine learning repository*, 2013. Cited on page 98

[131] S. LIN, *Computer solutions of the traveling salesman problem*, Bell System Technical Journal, The, 44 (1965), pp. 2245–2269. Cited on page 117

[132] S. LIN AND B. W. KERNIGHAN, *An effective heuristic algorithm for the traveling-salesman problem*, Operations research, 21 (1973), pp. 498–516. Cited on page 117, 118

[133] R. J. LIPTON AND R. E. TARJAN, *A separator theorem for planar graphs*, SIAM J. on Applied Mathematics, 36 (1979), pp. 177–189. Cited on page 11, 139, 165

[134] ——, *Applications of a planar separator theorem*, SIAM J. on Computing, 9 (1980), pp. 615–627. Cited on page 10, 165

[135] D. LOKSHTANOV, M. MNICH, AND S. SAURABH, *Linear kernel for planar connected dominating set*, in Theory and Applications of Models of Comput., Springer, 2009, pp. 281–290. Cited on page 148

[136] H.-I. LU AND R. RAVI, *Approximating maximum leaf spanning trees in almost linear time*, J. of Algorithms, 29 (1998), pp. 132–141. Cited on page 151

[137] M. MAHAJAN, P. NIMBHORKAR, AND K. R. VARADARAJAN, *The planar k-means problem is NP-hard*, Theor. Comput. Sci., 442 (2012), pp. 13–21. Cited on page 8, 28

[138] J. MATOUSEK, *On approximate geometric k-clustering*, Discrete & Computational Geometry, 24 (2000), pp. 61–84. Cited on page 22, 29, 73

[139] R. M. MCCUTCHEN AND S. KHULLER, *Streaming algorithms for k-center clustering with outliers and with anonymity*, in Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings, 2008, pp. 165–178. Cited on page 30

[140] N. MEGIDDO AND K. J. SUPOWIT, *On the complexity of some common geometric location problems*, SIAM J. Comput., 13 (1984), pp. 182–196. Cited on page 8, 28, 174

[141] O. MERSMANN, B. BISCHL, J. BOSSEK, H. TRAUTMANN, M. WAGNER, AND F. NEUMANN, *Local search and the traveling salesman problem: A feature-based characterization of problem hardness*, in Learning and Intelligent Optimization - 6th International Conference, LION 6, Paris, France, January 16-20, 2012, Revised Selected Papers, 2012, pp. 115–129. Cited on page 118

[142] R. R. METTU AND C. G. PLAXTON, *The online median problem*, SIAM J. Comput., 32 (2003), pp. 816–832. Cited on page 28

[143] G. L. MILLER, *Finding small simple cycle separators for 2-connected planar graphs*, J. Comput. Syst. Sci., 32 (1986), pp. 265–279. Cited on page 138, 165

[144] J. S. B. MITCHELL, *Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric tsp, k-mst, and related problems*, SIAM J. Comput., 28 (1999), pp. 1298–1309. Cited on page 113, 114, 117, 166

[145] B. MOHAR AND C. THOMASSEN, *Graphs on Surfaces*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, 2001. Cited on page 172

[146] A. MOITRA AND G. VALIANT, *Settling the polynomial learnability of mixtures of gaussians*, in Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on, IEEE, 2010, pp. 93–102. Cited on page 95

[147] D. MÖLLE, S. RICHTER, AND P. ROSSMANITH, *Enumerate and expand: Improved algorithms for connected vertex cover and tree cover*, Theory of Computing Systems, 43 (2008), pp. 234–253. Cited on page 151, 152

[148] N. H. MUSTAFA AND S. RAY, *PTAS for geometric hitting set problems via local search*, in Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, June 8-10, 2009, 2009, pp. 17–22. Cited on page 155, 166

[149] V. H. NGUYEN, *Approximation algorithms for metric tree cover and generalized tour and tree covers*, RAIRO - Operations Research, 41 (2007), pp. 305–315. Cited on page 12, 118, 151, 152

[150] R. OSTROVSKY AND Y. RABANI, *Polynomial time approximation schemes for geometric k-clustering*, in Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on, IEEE, 2000, pp. 349–358. Cited on page 162

[151] R. OSTROVSKY, Y. RABANI, L. J. SCHULMAN, AND C. SWAMY, *The effectiveness of Lloyd-type methods for the k-means problem*, J. ACM, 59 (2012), p. 28. Cited on page 6, 23, 24, 28

[152] S. PAUL AND R. E. MILLER, *Locating faults in a systematic manner in a large heterogeneous network*, in Int. Conf. on Computer Com., 1995, pp. 522–529. Cited on page 148

[153] F. PEDREGOSA, G. VAROQUAUX, A. GRAMFORT, V. MICHEL, B. THIRION, O. GRISEL, M. BLONDEL, P. PRETTENHOFER, R. WEISS, V. DUBOURG, J. VANDERPLAS, A. PASSOS, D. COURNAPEAU, M. BRUCHER, M. PERROT, AND E. DUCHESNAY, *Scikit-learn: Machine learning in python*, J. Mach. Learn. Res., 12 (2011), pp. 2825–2830. Cited on page 98

[154] G. PISIER, *The volume of convex bodies and Banach space geometry.*, Cambridge Tracts in Mathematics. 94, 1999. Cited on page 73

[155] V. RAMAN, S. SAURABH, AND C. SUBRAMANIAN, *Faster fixed parameter tractable algorithms for finding feedback vertex sets*, ACM Tr. on Algorithms, 2 (2006), pp. 403–415. Cited on page 153

[156] S. RAO AND W. D. SMITH, *Approximating geometrical graphs via "spanners" and "banyans"*, in Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, 1998, pp. 540–550. Cited on page 117

[157] N. ROBERTSON AND P. D. SEYMOUR, *Graph minors. iii. planar tree-width*, Journal of Combinatorial Theory, Series B, 36 (1984), pp. 49–64. Cited on page 168

[158] ——, *Graph minors. ii. algorithmic aspects of tree-width*, Journal of algorithms, 7 (1986), pp. 309–322. Cited on page 168

[159] ——, *Graph minors. viii. a kuratowski theorem for general surfaces*, J. Comb. Theory Ser. B, 48 (1990), pp. 255–288. Cited on page 172

[160] ——, *Graph minors. x. obstructions to tree-decomposition*, Journal of Combinatorial Theory, Series B, 52 (1991), pp. 153–190. Cited on page 170

[161] ——, *Graph minors. xi. circuits on a surface*, J. of Combinatorial Theory, Series B, 60 (1994), pp. 72 – 106. Cited on page 141

[162] ——, *Graph minors. xx. wagner's conjecture*, Journal of Combinatorial Theory, Series B, 92 (2004), pp. 325–357. Cited on page 173

[163] N. ROBERTSON, P. D. SEYMOUR, AND R. THOMAS, *Quickly excluding a planar graph*, J. of Combinatorial Theory, Series B, 62 (1994), pp. 323–348. Cited on page 136

[164] G. ROBINS AND A. ZELIKOVSKY, *Improved steiner tree approximation in graphs*, in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '00, Philadelphia, PA, USA, 2000, Society for Industrial and Applied Mathematics, pp. 770–779. Cited on page 119

[165] A. L. ROSENBERG AND L. S. HEATH, *Graph separators, with applications*, Springer Science & Business Media, 2001. Cited on page 163

[166] D. J. ROSENKRANTZ, R. E. STEARNS, AND P. M. L. II, *An analysis of several heuristics for the traveling salesman problem*, SIAM J. Comput., 6 (1977), pp. 563–581. Cited on page 118

[167] J. Rué, I. Sau, and D. M. Thilikos, *Dynamic programming for graphs on surfaces*, ACM Tr. on Algorithms, 10 (2014), p. 8. Cited on page 148

[168] C. D. Savage, *Depth-first search and the vertex cover problem*, Inf. Process. Lett., 14 (1982), pp. 233–237. Cited on page 152

[169] L. J. Schulman, *Clustering for edge-cost minimization (extended abstract)*, in Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA, 2000, pp. 547–555. Cited on page 28

[170] A. Sebö and A. van Zuylen, *The salesman's improved paths: 3/2+1/34 integrality gap and approximation ratio*, CoRR, abs/1604.02486 (2016). Cited on page 113

[171] P. D. Seymour and R. Thomas, *Call routing and the ratcatcher*, Combinatorica, 14 (1994), pp. 217–241. Cited on page 131, 132

[172] E. Snelson, C. E. Rasmussen, and Z. Ghahramani, *Warped gaussian processes*, Advances in neural information processing systems, 16 (2004), pp. 337–344. Cited on page 98

[173] C. H. Thomas, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, vol. 6, MIT press Cambridge, 2001. Cited on page 161

[174] W. T. Tutte, *On hamiltonian circuits*, Journal of the London Mathematical Society, s1-21 (1946), pp. 98–101. Cited on page 113

[175] P. Ungar, *A theorem on planar graphs*, Journal of the London Mathematical Society, s1-26 (1951), pp. 256–262. Cited on page 165

[176] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, *The numpy array: A structure for efficient numerical computation*, Computing in Science & Engineering, 13 (2011), pp. 22–30. Cited on page 97

[177] H. Zarrabi-Zadeh, *Core-preserving algorithms*, in Proceedings of the 20th Annual Canadian Conference on Computational Geometry, Montréal, Canada, August 13-15, 2008, 2008. Cited on page 30

[178] Z. Zhang, X. Gao, and W. Wu, *PTAS for connected vertex cover in unit disk graphs*, Theoretical Computer Science, 410 (2009), pp. 5398–5402. Cited on page 152

Abstract

What are the performance guarantees of the algorithms used in practice for clustering and network design problems? We answer this question by showing that the standard local search algorithm returns a nearly-optimal solution for low-dimensional Euclidean instances of the traveling salesman problem, Steiner tree, k-median and k-means. The result also extends to the case of graphs excluding a fixed minor for k-median and k-means.

The problem of finding a polynomial-time approximation scheme for instances of the k-means problem consisting of points lying in a low-dimensional Euclidean space or graphs excluding a fixed minor is a 15-year old open problem. In this work, we solve this problem by showing that local search is a polynomial-time approximation scheme for those instances. Furthermore, we show that for three recent characterizations of clustering instances stemming from machine learning and data analysis, local search returns a nearly-optimal solution.

One of the key ingredients that makes local search efficient is the existence of "cheap separators" in the instances. In the last part of this work, we show how to compute a more structured separator tailored for network design problems. This yields a general framework for obtaining approximation schemes for connectivity and domination problems in weighted planar graphs. It implies the first polynomial-time approximation schemes for weighted versions of tree and tour cover, connected vertex cover, connected dominating set, and feedback vertex set in planar graphs.