Theoretical Analysis of Hierarchical Clustering and the Shadow Vertex Algorithm

Anna-Klara Großwendt
geboren in Wesel

Dissertation
zur Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

Bonn 2019

1. Gutachter: Prof. Dr. Heiko Röglin
2. Gutachterin: Jun.-Prof. Dr. Melanie Schmidt
Tag der mündlichen Prüfung: 13.03.2020
Erscheinungsjahr: 2020

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Abstract

Agglomerative clustering (**AC**) is a very popular greedy method for computing hierarchical clusterings in practice, yet its theoretical properties have been studied relatively little. We consider **AC** with respect to the most popular objective functions, especially the diameter function, the radius function and the k-means function. Given a finite set $P \subseteq \mathbb{R}^d$ of points, **AC** starts with each point from P in a cluster of its own and then iteratively merges two clusters from the current clustering that minimize the respective objective function when merged into a single cluster.

We study the problem of partitioning P into k clusters such that the largest diameter of the clusters is minimized and we prove that **AC** computes an $O(1)$-approximation for this problem for any metric that is induced by a norm, assuming that the dimension d is a constant. This improves the best previously known bound of $O(\log k)$ due to Ackermann et al. [2]. Our bound also carries over to the k-center and the continuous k-center problem.

Moreover we study the behavior of agglomerative clustering for the hierarchical k-means problem. We show that **AC** computes a 2-approximation with respect to the k-means objective function if the optimal k-clustering is well separated. If additionally the optimal clustering also satisfies a balance condition, then **AC** fully recovers the optimum solution. These results hold in arbitrary dimension. We accompany our positive results with a lower bound of $\Omega((3/2)^d)$ for data sets in $\mathbb{R}^d$ that holds if no separation is guaranteed, and with lower bounds when the guaranteed separation is not sufficiently strong. Finally, we show that **AC** produces an $\mathcal{O}(1)$-approximative clustering for one-dimensional data sets.

Apart from **AC** we provide improved and in some cases new general upper and lower bounds on the existence of hierarchical clusterings. For the objective function discrete radius we provide a new lower bound of 2 and improve the upper bound of 4. For the k-means objective function we state a lower bound of 32 on the existence of hierarchical clusterings. This improves the best previously known bound of 576.

The simplex algorithm is probably the most popular algorithm for solving linear programs in practice. It is determined by so called pivot rules. The shadow vertex simplex algorithm is a popular pivot rule which has gained attention in recent years because it was shown to have polynomial running time in the model of smoothed complexity. In the second part of the dissertation we show that the shadow vertex simplex algorithm can be used to solve linear programs in strongly polynomial time with respect to the number n of variables, the number m of constraints, and $1/\delta$, where δ is a parameter that measures the flatness of the vertices of the polyhedron. This extends a previous result that the shadow vertex algorithm finds paths of polynomial length (w.r.t. n, m, and $1/\delta$) between two given vertices of a polyhedron [17].

Our result also complements a result due to Eisenbrand and Vempala [25] who have shown that a certain version of the random edge pivot rule solves linear programs with a running time that is strongly polynomial in the number of variables n and $1/\delta$, but independent of the number m of constraints. Even though the running time of our algorithm depends on m, it is significantly faster for the important special case of totally unimodular linear programs, for which $1/\delta \leq n$ and which have only $O(n^2)$ constraints.

Acknowledgments

I would briefly like to thank a few people without whom my thesis would not have been possible.

Above all, I would like to thank my supervisor Heiko Röglin for all of his support. I thank you for the interesting research ideas and the many things I have learned in the field of computer science. Especially that I was able to choose my research topics freely as well as the good and trend-setting advice and comments to lead them to a goal. I also thank you a lot for all the support far away from the scientific area, which I did not take for granted. I am specifically grateful that you made it always possible for me to combine my research and my job at the university with my two sons coming to life during my doctoral studies.

I am very grateful to be part of this working group. Over the last few years I met a lot of nice people, getting lots of ideas for my research and having great and interesting conversations, not only about computer science. In particular, I would like to thank Melanie Schmidt, with whom I did a lot of interesting research on clustering after she came to Bonn. The work was always fun for me. Particularly I thank you for your patience when our research was only possible via chat because of my parental leave.

Finally, I want to thank my family for their love and support. I especially thank Tim, Frederik and Alexander who make each day of my life valuable for me.

Contents

Chapter 1

Introduction

Since its first definition in 1971 in a paper by Stephen A. Cook [19], the complexity class NP has decisively shaped the scientific work in the field of computer science. Cook showed in his paper that every problem contained in NP can be reduced to the SAT-problem in polynomial time - this makes SAT the hardest type of problem in NP. Karp followed in 1972 with an extensive work [32], in which he uses polynomial time reductions to prove for 21 problems, including famous problems like Vertex Cover or Clique, that they are contained in the group of hardest problems in NP. Since then, research revolves around the question whether there is an efficient solution for such problems. By efficient we mean that a solution can be found in polynomial time. Such problems are summarized in the class P, which is a subclass of NP.

As a consequence one has to deal with the question how to proceed with hard optimization problems, which are of particular interest in practice and where a good solution has to be computed quickly. A crucial and trend-setting observation is, that we search in most of the cases for a good solution but not necessarily for an optimum solution. This introduces the large field of approximation algorithms. An approximation algorithm searches for a value which differs from the optimal solution by a factor of at most α. In general one allows α to be a function dependent on parameters like the input size of the instance.

One popular example for that type of optimization problems is the k-clustering problem where a finite set of points P shall be divided into a fixed number of clusters. Clustering is well known to be NP-hard for a large number of objective functions, including the diameter- or radius-function such as k-means. Since it is indispensable in numerous tasks in practice one has to find a way to compute good clusterings. Many approximation algorithms have been developed over the years. In practice greedy methods are very popular. They are often easy to implement with a small running time and behave naturally and locally optimal. In terms of clustering a very natural heuristic is to start with P and proceed with the cheapest possible merge steps one after another until a suitable number of clusters is obtained. This results in a k-clustering for each integer $k \in [|P|]$[1] where clusterings are refinements of subsequent clusterings. A set of clusterings with these properties is called a hierarchical clustering. Nevertheless for the most popular objective functions there are very few results known about approximation guarantees or

[1]We use the abbreviation $[i] = \{1, \ldots, i\}$ for $i \in \mathbb{N}$.

lower bounds for this greedy heuristic. In Chapter 2 we analyze the existence of such hierarchical clusterings in general and moreover analyze the greedy heuristic with respect to the most popular objective functions.

Besides there exist very established problems for which it is still not known if they are solvable in polynomial time or to belong to the group of NP-hard problems. A very popular example is the graph isomorphism problem. Another problem for which the complexity class was unknown for a long time is linear programming. The problem is to maximize a linear function under a number of linear constraints on the variables. This turns over to the problem of finding a certain vertex in a polyhedron. The simplex algorithm was invented by Dantzig in 1947 [21] and walks along a path of neighbored vertices until it reaches the target vertex. How the next neighbored edge is chosen in the path depends on so called pivot rules. There are many pivot rules which are very popular and perform well in practice, though there exist instances where the running time is known to be exponential for the most of them. Apart from the simplex method Leonid Khachiyan proved in 1979 [33] that linear programming is contained in P. He uses an extension of the ellipsoid method which can be used to solve linear programs in polynomial running time. Nevertheless the simplex method is still significant in practice. It is still unknown if there exists a pivot rule which leads to a polynomial running time of the algorithm. One very popular example for pivot rules is the shadow vertex pivot rule. The shadow vertex pivot rule visits vertices along the shape of the polyhedron projected into a 2 dimensional plane from a start vertex to the target vertex. In 2004 Spielman and Teng proved that the shadow vertex pivot rule has polynomial running time in the model of smoothed complexity [49] which justifies its relevance in practice. In Chapter 3 we provide a randomized algorithm based on the shadow vertex pivot rule which has strongly polynomial running time but depends on a parameter which represents the flatness of the polyhedron defined by the linear program.

1.1 Hierarchical Clustering

In a typical clustering problem, the goal is to partition a given set of objects into clusters such that similar objects belong to the same cluster while dissimilar objects belong to different clusters. Clustering is ubiquitous in computer science with applications ranging from biology to information retrieval and data compression. As an unsupervised learning method, it provides an easy way to gain insight into the structure of data without the need for expert knowledge to start with. A *k-clustering $\mathcal{C}$ of P* is a partition of P into k non-empty sets $C_1, \dots, C_k$. There exist a lot of popular objective functions to measure the quality of a given clustering $\mathcal{C}$ starting with geometric functions like diameter or (discrete) radius or stochastic functions like k-median and k-means also known under the name *sum of squared errors.*

Diameter and discrete radius Let (M, d) be a metric space and $P \subseteq M$ denote a finite set of points. Geometric objective functions consider properties of the convex hull of the clusters like their diameter but are not interested in properties like the number of points per cluster. We consider two common variants to measure the quality of a k-clustering $\mathcal{C}$, which lead to different optimization problems.

- **diameter k-clustering problem**: Find a k-clustering $\mathcal{C}$ with minimum *diameter*. The diameter $\operatorname{diam}(\mathcal{C})$ of $\mathcal{C}$ is given by the maximal diameter $\max_i \operatorname{diam}(C_i)$ of one of its clusters, where the diameter of a set $C \subseteq P$ is defined as $\operatorname{diam}(C) := \max_{x,y \in C} \operatorname{dist}(x, y)$.

- **k-center problem**: Find a k-clustering $\mathcal{C}$ with minimum *discrete radius*. The discrete radius $\operatorname{drad}(\mathcal{C})$ of $\mathcal{C}$ is given by the maximal discrete radius $\max_i \operatorname{drad}(C_i)$ of one of its clusters, where the discrete radius of a set $C \subseteq P$ is defined as $\operatorname{drad}(C) := \min_{y \in C} \max_{x \in C} \operatorname{dist}(x, y)$.

In some cases, when the metric is non-discrete, it is crucial to allow any point in M to be a cluster center. In that case we distinguish between discrete radius and radius as objective functions. We name the problem of finding a clustering with respect to the objective function radius as *continuous* k-center problem.

The approximability of clustering problems is well understood. In general we know that the k-center problem is NP-hard and it is even NP-hard to find a $(2 - \varepsilon)$-approximation for any $\varepsilon > 0$ [28]. The same bounds hold for the diameter k-clustering problem. Feder and Greene [26] proved that for the Euclidean metric the k-center problem and the diameter k-clustering problem cannot be approximated better than a factor of 1.822 and 1.969, respectively.

k-means One of the most popular clustering objectives is k-means: Given a set P of points in the Euclidean space $\mathbb{R}^d$, find k centers that minimize the sum of the squared distances of each point in P to its closest center. The objective is also called *sum of squared errors*, since the centers can serve as representatives, and then the sum of the squared distances becomes the squared error of this representation.

Theory has focused on metric objective functions for a long time: Facility location or k-median are very well understood, with upper and lower bounds on the best possible approximation guarantee slowly approaching one another. The k-means cost function is arguably more popular in practice, yet its theoretical properties were long not the topic of much analysis. In the last decade, considerable efforts have been made to close this gap.

We now know that k-means is NP-hard, even in the plane [41] and also even for two centers [4]. The problem is also APX-hard [7], and the currently best approximation algorithm achieves an approximation ratio of 6.357 [3]. The best lower bound, though, is only 1.0013 [37]. A seminal paper on k-means is the derivation of a practical approximation algorithm, k-means++, which is as fast as the most popular heuristic for the problem (the local search algorithm due to Lloyd [39]), has an upper bound of $\mathcal{O}(\log k)$ on the expected approximation ratio, and has proven to significantly improve the performance on actual data [5]. Due to its simplicity and superior performance, it (or variants of it) can now be seen as the de facto standard initialization for Lloyd's method.

Existence of good hierarchical clusterings From a practical point of view, however, there is still one major drawback of using k-means++ and Lloyd's method, and this has nothing to do with its approximation ratio or speed. Before using any method that strives to optimize k-means, one has to determine the number k of clusters. If one knows very little about the data at hand, then even this might pose a challenge. However, there is a simple and popular method available: hierarchical clustering. Instead of computing clusterings for several different numbers of clusters and comparing them, one computes one hierarchical clustering, which contains a clustering for every value of k. A hierarchical clustering $\mathcal{C}$ of a set P of n objects is a sequence $\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_n$, where $\mathcal{C}_i$ is a clustering of P into i non-empty clusters and $\mathcal{C}_{i-1}$ results from $\mathcal{C}_i$ by merging two clusters in $\mathcal{C}_i$. Besides the advantage that the number of clusters does not have to be specified in advance, hierarchical clusterings are also appealing because they help to understand the hereditary properties of the data and they provide information at different levels of granularity. A hierarchical clustering is apparently something very desirable, but the question is: Can the solutions be good for all values of k? Do we lose much by forcing the hierarchical structure?

Dasgupta and Long [23] were the first to give positive and negative answers to this question. Their analysis evolves around the k-center problem. They compare the k-center cost on each level of a hierarchical clustering to an optimal clustering with the best possible radius with the same number of clusters and look for the level with the worst factor. It turns out that popular heuristics for hierarchical clustering can be off by a factor of $\log k$ or even k compared to an optimal clustering. Dasgupta and Long also propose a clever adaption of the 2-approximation for k-center due to González [28], which results in a hierarchical clustering algorithm. For this algorithm, they can guarantee that the solution is an 8-approximation of the optimum on every level of the hierarchy simultaneously. We improve the known upper bound of 8 for the k-center problem to an upper bound of 4, though our argument is non-constructive.

Theorem 1.1. *For each finite point set P and each metric (M, d) with $P \subseteq M$ there exists a hierarchical clustering with approximation factor* 4 *on each level of granularity for the k-center problem.*

In a series of works, Mettu, and Plaxton [43], Plaxton [45] and finally Lin, Nagarajan, Rajaraman, and Williamson [38] develop and refine algorithms for the hierarchical k-median problem, which can be seen as the metric cousin of the hierarchical k-means problem. It consists of minimizing the sum of the distances of every point to its closest center, and is usually studied in metric spaces. The best known approximation guarantee is 20.06. However, the quality guarantee vastly deteriorates for k-means: An $\mathcal{O}(1)$-approximation for the hierarchical k-means problem follows from [45, 43] as well as from [38], but the approximation ratios range between 961 and 3662. Nevertheless the analysis of Lin, Nagarajan, Rajaraman, and Williamson [38] includes non-constructive upper bounds for the existence of a hierarchical clustering with respect to k-median and k-means of 24 and 576, respectively. We improve their upper bounds by providing a better subroutine. The following theorem improves the upper bound for the k-means problem. We mention that together with the algorithm in [38] the subroutine can be used to efficiently implement an approximation algorithm for the hierarchical k-means problem with approximation ratio 32α, given an α-approximation for the k-means problem for arbitrary k. This also improves the recently known upper bound of 576α.

Theorem 1.2. *For each finite point set $P \in \mathbb{R}^d$ there exists a hierarchical clustering with approximation factor* 32 *on each level of granularity for the k-means problem. Moreover given an α-approximation algorithm for the k-means problem, a hierarchical clustering with approximation ratio 32α can be computed.*

When talking about upper bounds it automatically raises the question what it the best hierarchical clustering we can hope for? Das and Kenyon-Mathieu state an instance for the diameter k-clustering problem in [22] where no hierarchical clustering exists which is better than a 2-approximation for each level of granularity. We provide a family of instances $(T_m)_{m\in\mathbb{N},m\geq 0}$ for the k-center problem where no hierarchical clustering has an approximation factor better than $2 - 1/2^m$.

Theorem 1.3. *For each $\epsilon > 0$ there exists a metric space (M, d) and a finite point set $P \subseteq M$ where the minimum hierarchical clustering for the k-center problem on P has approximation factor larger than $2 - \epsilon$.*

	k-center	diameter	k-means
Constructive Upper Bound	8 [23]	8 [23]	32α (576α [38])
Upper Bound	4 (8 [23])	8 [23]	32 (576 [38])
Lower Bound	2 (-)	2 [22]	-

Table 1.1: The table states currently known upper and lower bounds on the hierarchical versions of the three clustering problems. Whenever we improved one of the bounds we state the previous known bound in brackets right behind.

The agglomorative clustering greedy Hierarchical clustering algorithms are classified as *divisive* or *agglomerative*. Divisive algorithms work top-down, starting with P as the first clustering and subsequently dividing it. Agglomerative algorithms work bottom-up, starting with singletons clusters and subsequently merging them. Agglomerative methods are more popular because they are usually faster. The Agglomerative Clustering Greedy (**AC**) starts with the clustering $\mathcal{C}_n$, in which every object belongs to its own cluster. Then it iteratively merges the two clusters from the current clustering $\mathcal{C}_{i+1}$ with the smallest distance to obtain the next clustering $\mathcal{C}_i$. This is a locally optimal choice only, since the optimal merge in one operation may prove to be a poor choice with respect to a later level of the hierarchy. Depending on how the distance between two clusters is defined, different agglomerative methods can be obtained.

A common variant is the *complete-linkage method* in which the distance between two clusters A and B is defined as the diameter or the discrete radius of $A \cup B$, assuming some distance measure on the objects from P is given. Which clusters are merged in an iteration depends on the optimization problem we consider. For the diameter k-clustering problem, the complete-linkage method chooses two clusters A and B from $\mathcal{C}_{i+1}$ such that $\mathrm{diam}(A\cup B)$ is minimized. Similarly, for the k-center problem and the continuous k-center problem it chooses two clusters A and B from $\mathcal{C}_{i+1}$ such that $\mathrm{drad}(A \cup B)$ or $\mathrm{rad}(A \cup B)$ is minimized, respectively. Hence, every objective function gives rise to a different variant of the complete-linkage method. When it is not clear from the context which variant is meant, we will use the notation $\mathbf{CL}^{\mathrm{drad}}$, $\mathbf{CL}^{\mathrm{rad}}$, and $\mathbf{CL}^{\mathrm{diam}}$ to make the variant clear.

The complete-linkage method is very popular and successful in a wide variety of applications. To name just a few of many examples, Rieck et al. [46] have used it for automatic malware detection, Ghaemmaghami et al. [27] have used it to design a speaker attribution system, and Cole et al. [18] use it as part of the Ribosomal Database Project. Yet the complete-linkage method is not fully understood in theory and there is still a considerable gap between the known upper and lower bounds for its approximation guarantee.

Ackermann et al. [2] proved that the complete-linkage method yields an $O(\log k)$-approximation for any metric that is induced by a norm and constant dimension d. The analysis of Ackermann et al. proceeds in two phases. The first phase ends when $2k$ clusters are left and the second phase consists of the last k merge operations. In the first phase a factor depending only on d but not on k is incurred. Our analysis begins at the end of the first phase. We prove that the approximation factor in the last k steps increases by at most a constant factor. This leads to an improved upper bound for the complete-linkage method.

Theorem 1.4. *For $d \in \mathbb{N}$ and a finite point set $P \subseteq \mathbb{R}^d$ the algorithm $\boldsymbol{CL}^{\mathrm{drad}}$ computes an $O(d)$-approximation for the k-center problem. The algorithm $\boldsymbol{CL}^{\mathrm{diam}}$ computes a $2^{O(d)^d}$-approximation for the diameter k-clustering problem.*

Using **AC** for k-means yields *Ward's method* [51]. Here the distance between two clusters A and B is defined as the k-means cost of the clustering $\mathcal{C}_i$ resulting from $\mathcal{C}_{i+1}$ by merging A and B. To the best of our knowledge, the worst-case quality of Ward's method has not been studied before. In particular, it was not known whether the algorithm can be used to compute constant-factor approximations. We answer this question negatively by giving a family of examples with increasing k and d where the approximation factor of Ward is $\Omega((3/2)^d)$.

To explain the algorithms popularity, we then proceed to study it under different *clusterability* assumptions. Clustering problems are usually NP-hard and even APX-hard, yet clustering is routinely solved in practical applications. This discrepancy has led to the hypothesis that data sets are either easy to cluster, or they have little interesting structure to begin with. 'Well-clusterable data sets are computationally easy to cluster' [12] and 'Clustering is difficult only when it does not matter' [20] are two slogans summarizing this idea. Following it, many notions have been developed that strive to capture how well a data set is clusterable. One such notion is *center separation* [13]: A data set $P \subset \mathbb{R}^d$ is δ-center separated for some number k of clusters if the distance between any pair of clusters in the optimal clustering is at least δ times the maximal radius of one of the clusters. It satisfies the similar *α-center proximity* [6] for k if in the optimum k-clustering the distance of each data point to any center except for its own is larger by a factor of at least α than the distance to its own center. We apply these notions to hierarchical clustering by showing that if there is a well-separated optimum solution for a level, then the clustering computed by Ward on this level is a 2-approximation.

Theorem 1.5. *Let $P \subset \mathbb{R}^d$ be an instance that satisfies weak $(2+2\sqrt{2}+\epsilon)$-center separation or $(3+2\sqrt{2}+\epsilon)$-center proximity for some $k \in [|P|]$ and $\epsilon > 0$. Then Ward computes a 2-approximation on P for that k.*

This means that Ward finds good clusterings for all levels of granularity that have a meaningful clustering; and these good clusterings have a hierarchical structure. For levels on which the sizes of the optimal clusters are additionally to some extend balanced, we prove that Ward even computes the optimum clustering.

Theorem 1.6. *Let $P \subset \mathbb{R}^d$ be an instance with optimal k-means clustering $O_1, \ldots, O_k$ with centers $c_1^*, \ldots, c_k^* \in \mathbb{R}^d$. Assume that P satisfies $(2 + 2\sqrt{2\nu} + \epsilon)$-center separation for some $\epsilon > 0$, where $\nu = \max_{i,j \in [k]} \frac{|O_i|}{|O_j|}$ is the largest factor between the sizes of any two optimum clusters. Then Ward computes the optimal k-means clustering $O_1, \ldots, O_k$.*

Related Work

Let $P \subseteq \mathbb{R}^d$ and a metric dist on P be given and consider an objective function $f \in \{\text{drad}, \text{rad}, \text{diam}, \text{k-median}, \text{k-means}\}$. Let $\mathcal{O}_k^f$ be an optimal k-clustering of P with respect to f. For each of these problems, it is easy to find examples where no hierarchical clustering $\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_{|P|})$ exists such that $\mathcal{C}_k$ is an optimal k-clustering for every k. We say that a hierarchical clustering $\mathcal{C}$ is an *α-approximate hierarchical clustering* with respect to f if $f(\mathcal{C}_k) \leq \alpha \cdot f(\mathcal{O}_k^f)$ holds for every k. In general, we also allow α to be a function of k and d.

Upper and lower bounds on the existence of hierarchical clusterings The design of hierarchical clustering algorithms that satisfy per-level guarantees started with the paper by Dasgupta and Long [23] who gave an efficient algorithm that computes 8-approximate hierarchical clusterings for the diameter k-clustering problem and the k-center problem, thereby giving a constructive proof of the existence of such hierarchical clusterings. Their result holds true for arbitrary metrics on $\mathbb{R}^d$ and it can even be improved to an expected approximation factor of $2e \approx 5.44$ by a randomized algorithm. Their method turns González' algorithm [28] into a hierarchical clustering algorithm. González' algorithm is a 2-approximation not only for k-center, but also for the *incremental* k-center problem: Find an ordering of all points, such that for all k, the first k points in the ordering approximately minimize the k-center cost. The idea to make an algorithm for incremental clustering hierarchical was picked up by Plaxton [45], who proves that this approach leads to a constant factor approximation for the hierarchical k-median problem. He uses an incremental k-median algorithm due to Mettu and Plaxton [43]. Finally, Lin, Nagarajan, Rajaraman and Williamson [38] propose a general framework for approximating incremental problems that also works for incremental variants of MST, vertex cover, and set cover. They also cast hierarchical k-median and k-means into their framework for incremental approximation. They get a randomized/deterministic 20.06/41.42-approximation for hierarchical k-median and a randomized/deterministic $151.1\alpha/576\alpha$-approximation for k-means, where α is the approximation ratio of a k-means approximation algorithm. The results include a non constructive proof of the existence of a 576-hierarchical clustering for k-means clustering.

Das and Kenyon-Mathieu [22] provide a lower bound of 2 for the existence of a hierarchical clustering for the diameter k-clustering problem. They state an instance I and prove that each hierarchical clustering on I has at least approximation factor 2. To our best knowledge their exist no further lower bounds with respect to other objective functions.

Complete linkage Dasgupta and Long also studied in [23] the performance of the complete-linkage method and presented an artificial metric on $\mathbb{R}^2$ for which its approximation factor is only $\Omega(\log k)$ for the diameter k-clustering and the k-center problem.

Ackermann et al. [2] showed for the diameter k-clustering and the discrete k-center problem a lower bound of $\Omega(\sqrt[p]{\log k})$ for the ℓ_p-metric for every $p \in \mathbb{N}$, assuming $d = \Omega(k)$.

Ackermann et al. [2] also showed that the complete-linkage method yields an $O(\log k)$-approximation for any metric that is induced by a norm, assuming that d is a constant. Here the constant in the big O notation depends on the dimension d. For the k-center problem the dependence on d is only linear and additive. For the continuous k-center problem the dependence is multiplicative and exponential in d, while for the diameter k-clustering problem it is multiplicative and even doubly exponential in d. The analysis of Ackermann et al. proceeds in two phases. The first phase ends when $2k$ clusters are left and the second phase consists of the last k merge operations. In the first phase a factor depending only on d but not on k is incurred. To make this precise, let $\mathcal{C}_{2k}^{\mathrm{drad}}$, $\mathcal{C}_{2k}^{\mathrm{rad}}$, and $\mathcal{C}_{2k}^{\mathrm{diam}}$ denote the $2k$-clusterings computed by the corresponding variants of **CL**. Ackermann et al. prove that for each objective $F \in \{\mathrm{drad}, \mathrm{rad}, \mathrm{diam}\}$ there exists a function κ_F such that

$$F(\mathcal{C}_{2k}^{F}) \leq \kappa_F(d) \cdot F(\mathcal{O}_k^F). \tag{1.1}$$

The function κ_{drad} is linear in d, the function κ_{rad} is exponential in d, and the function κ_{diam} is doubly exponential in d. The factor $O(\log k)$ is only incurred in the last k merge operations. Let $\mathcal{C}_{k}^{\mathrm{drad}}$, $\mathcal{C}_{k}^{\mathrm{rad}}$, and $\mathcal{C}_{k}^{\mathrm{diam}}$ denote the k-clusterings computed by the corresponding variants of **CL**. Ackermann et al. show that for each objective $F \in \{\mathrm{drad}, \mathrm{rad}, \mathrm{diam}\}$, it holds

$$F(\mathcal{C}_k^F) \leq O(\log k) \cdot F(\mathcal{C}_{2k}^F),$$

where the constant in the big O notation depends again on the dimension d. Additionally, Ackermann et al. [2] studied the case $d = 1$ separately and proved that the complete-linkage method computes 3-approximate hierarchical clusterings for the diameter k-clustering problem and the k-center problem for $d = 1$.

Ward's algorithm Balcan, Liang, and Gupta [11] observe that Ward's method cannot be used to recover a given target clustering. We discuss their example and the question whether Ward can find a *specific* target clustering, namely the optimum clustering, in Section 2.4.5. To our best knowledge, the quality in terms of approximation ratio of Ward's method has not been analyzed theoretically at all previous to our work.

Clusterability assumptions There is a vast body of literature on clusterability assumptions, i.e., assumptions on the input that make clustering easier either in the sense that a target clustering can be (partially) recovered or that a good approximation of an objective function can be computed efficiently. A survey of work in this area can be found in [12]. Particularly relevant for our results are the notions of δ-center separation [13] and α-center proximity [6] mentioned above. There are several papers showing that under these assumptions it is possible to recover the target/optimal clustering if δ and α are sufficiently large [6, 10, 36, 42]. Other notions include the *strict separation property* of Balcan, Blum, and Vempala [9], the *ϵ-separation property* of Ostrovsky et al. [44], and the weaker version of the proximity condition due to Kumar and Kannan [35] which Awasthi and Sheffet [8] proposed (it is based on the spectral norm of a matrix whose rows are the difference vectors between the points in the data set and their centers). For all these notions of clusterability, algorithms are developed that (partially) recover the target/optimal clustering.

Our Contribution

Upper and lower bounds on the existence of hierarchical clusterings We prove that there exists a family of instances for the k-center problem such that for each $\varepsilon > 0$ there is some instance on which no $2 - \varepsilon$-hierarchical clustering exists. This implies a lower bound of 2 for the existence of hierarchical clusterings for the k-center problem. On the other side we introduce new augmentation routines which extend and in the case of k-means replace the augmentation routines used by Lin et al. in [38]. This leads to significantly better upper bounds for the k-center problem where the upper bound of Dasgupta and Long decreases to 4 and k-means clustering where we turn the bound of 576 for the existence of a hierarchical clustering into an upper bound of 32.

Ward's algorithm We show that, in general, Ward's method does not achieve a constant approximation factor. We present a family of instances $(P_d)_{d\in\mathbb{N}}$ with $P_d \subset \mathbb{R}^d$ on which the cost of the 2^d-clustering computed by Ward is larger than the cost of the optimal 2^d-means clustering of P_d by a factor of $\Omega((3/2)^d)$. Then we observe that the family of instances used for this lower bound satisfy the *strict separation property* of Balcan, Blum, and Vempala [9], the ϵ-separation property of Ostrovsky et al. [44] for any $\epsilon > 0$, and the separation condition from Awasthi and Sheffet [8]. Hence, none of these three notions of clusterability helps Ward's method to avoid that the approximation factor grows exponentially with the dimension.

Moreover we show that the approximation ratio of Ward's method on one-dimensional inputs is $\mathcal{O}(1)$. The one-dimensional case turns out to be more tricky than one would expect, and our analysis is quite complex and technically challenging.

Finally, we analyze the approximation factor of Ward's method on data sets that satisfy different well-known clusterability notions. It turns out that the assumption that the input satisfies a high *δ-center separation* [13] or *α-center proximity* [6] implies a very good bound on the approximation guarantee of Ward's method. We show that Ward's method computes a 2-approximation for all values of k for which the input data set satisfies $(2+2\sqrt{2})$-center separation or $(3+2\sqrt{2})$-center proximity. We also show that on instances that satisfy $(2 + 2\sqrt{2\nu})$-center separation and for which all clusters O_i and O_j in the optimal clustering satisfy $|O_j| \geq |O_i|/\nu$, Ward even recovers the optimal clustering.

Complete linkage As a part of this thesis we prove that the complete-linkage method yields an $O(1)$-approximation for the k-center problem, the continuous k-center problem and the diameter k-clustering problem for any metric on $\mathbb{R}^d$ that is induced by a norm, assuming that d is a constant. This does not contradict the lower bound of Ackermann et al. because this lower bound assumes that the dimension depends linearly on k. In light of our result, the dependence of this lower bound on k is somewhat misleading and it could also be expressed as $\Omega(\sqrt[p]{\log d})$.

In order to obtain our result, we improve the second phase of the analysis of Ackermann et al. [2] and we prove that for each objective $F \in \{\mathrm{drad}, \mathrm{rad}, \mathrm{diam}\}$,

$$F(\mathcal{C}_k^F) \leq O(1) \cdot F(\mathcal{C}_{2k}^F).$$

The constant in the big O notation depends neither on d nor on k. It is 43, 19, and 17 for the discrete k-center problem, the k-center problem, and the diameter k-clustering problem, respectively. Together with (1.1) this yields the desired bound for the approximation factor.

1.2 Shadow Vertex Algorithm

Linear Programming (LP) describes an optimization problem where one searches for the maximum of a linear function while a set of linear constraints is preserved by the solution vector. LP is probably the most important optimization model. It has an immense influence on modeling in economics and there exist extensive applications in industrial and military areas. Besides it plays a major role in the design of approximation algorithms when translating a hard problem into a suitable Integer Program and using LP to solve a relaxation to find a good estimation for the optimum value.

For a given matrix $A = [a_1, \ldots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ the standard form of a linear program is given by $\max\{c^{\mathrm{T}}x \mid Ax \leq b\}$. The set $\{x \mid Ax \leq b\}$ of linear constraints builds an n-dimensional polyhedron. Since the objective function is linear, it is geometrically clear that a vertex of the polyhedron takes the optimum value in the case where the optimum value does not tend to infinity.

Simplex method In 1974 Dantzig introduced the simplex method as a first algorithm to solve linear programs [21]. His idea was to start at a vertex of the polyhedron which has to be identified in a first phase and then walk along the shape of the polyhedron over a path of pairwise neighbored vertices in the direction of the objective function $c^T x$ until an optimal vertex or an unbounded ray is found. Up to now the simplex method is one of the most important methods for solving linear programs and is still widely used in practice. It is a bit misleading to talk about the simplex method as one algorithm. In fact, the simplex method is mainly determined by how the next vertex in the path is selected among all vertices that improve the objective function $c^T x$. As a first pivot rule, Dantzig proposed to choose the vertex which improves the target function the most. In the further course, many popular pivot rules have been depeloved which lead to different behaviors of the path from the start vertex to the optimum vertex and lead to advantages or disadvantages depending on the applications. A big advantage of the simplex method is that when additional constraints are added subsequently, a so-called warm start can be performed instead of calculate a solution right from the beginning. In 1970 Klee and Minty stated an instance in form of a unit hypercube of variable dimension whose corners have been perturbed and on which the simplex method in the original form as presented by Dantzig visits an exponential number of corners before reaching an optimal vertex. The Klee Minty cube is for many pivot rules an example for their non-polynomial running time and there are a lot of modifications and further hard instances concerning a large number of pivot rules. Up to now, there is no pivot rule for which a polynomial running time has been proven, although many of them belong to the fastest alternatives in practical applications.

Shadow vertex algorithm The shadow vertex algorithm is a well-known pivoting rule for the simplex method. The idea is that linear programming is easy in the two dimensions. Assume we have a start vertex x_0 and an objective function $c^T x$. First a vector u is computed such that x_0 minimizes the function $u^T x$. Then the polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ of feasible solutions is projected onto the plane by the projection vectors u and v. Then one walks along the shape of the shadow from x_0 to the optimal vertex. The shadow vertex algorithm is known to have an exponential running time in the worst case. Nevertheless it performs well in practical applications. It has gained attention in recent years because it was shown to have polynomial running time in the model of smoothed analysis [49] which justifies its relevance in practice. Brunsch and Röglin observed that it can also be used to find short paths between given vertices of a polyhedron [17]. Here short means that the path length is $O(\frac{mn^2}{\delta^2})$, where n denotes the number of variables, m denotes the number of constraints, and δ is a parameter of the polyhedron that we will define shortly.

The result left open the question whether or not it is also possible to solve linear programs in polynomial time with respect to n, m, and $1/\delta$ by the shadow vertex simplex algorithm. We resolve this question and introduce a variant of the shadow vertex simplex algorithm that solves linear programs in strongly polynomial time with respect to these parameters.

For a given matrix $A = [a_1, \dots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^m$ and $c_0 \in \mathbb{R}^n$ our goal is to solve the linear program $\max\{{c_0}^{\mathrm{T}} x \mid Ax \leq b\}$. We assume without loss of generality that $\|c_0\| = 1$ and $\|a_i\| = 1$ for every row a_i of the constraint matrix.

Definition 1.7. *The matrix A satisfies the* δ-distance property *if the following condition holds: For any $I \subseteq \{1, \dots, m\}$ and any $j \in \{1, \dots, m\}$, if $a_j \notin \operatorname{span}\{a_i \mid i \in I\}$ then $\operatorname{dist}(a_j, \operatorname{span}\{a_i \mid i \in I\}) \geq \delta$. In other words, if a_j does not lie in the subspace spanned by the a_i, $i \in I$, then its distance to this subspace is at least δ.*

We present a variant of the shadow vertex simplex algorithm that solves linear programs in strongly polynomial time with respect to n, m, and $1/\delta$, where δ denotes the largest δ' for which the constraint matrix of the linear program satisfies the δ'-distance property. (In the following theorems, we assume $m \geq n$. If this is not the case, we use the method from Section 3.7.1 to add irrelevant constraints so that A has rank n. Hence, for instances that have fewer constraints than variables, the parameter m should be replaced by n in all bounds.)

Theorem 1.8. *There exists a randomized variant of the shadow vertex simplex algorithm (described in Section 3.3) that solves linear programs with n variables and m constraints satisfying the δ-distance property using $O(\frac{mn^3}{\delta^2} \cdot \log(\frac{1}{\delta}))$ pivots in expectation if a basic feasible solution is given. A basic feasible solution can be found using $O(\frac{m^5}{\delta^2} \cdot \log(\frac{1}{\delta}))$ pivots in expectation.*

We stress that the algorithm can be implemented without knowing the parameter δ. From the theorem it follows that the running time of the algorithm is strongly polynomial with respect to the number n of variables, the number m of constraints, and $1/\delta$ because every pivot can be performed in time $O(mn)$ in the arithmetic model of computation (see Section 3.5).[2]

Let $A \in \mathbb{Z}^{m\times n}$ be an integer matrix and let $A' \in \mathbb{R}^{m\times n}$ be the matrix that arises from A by scaling each row such that its norm equals 1. If Δ denotes an upper bound for the absolute value of any sub-determinant of A, then A' satisfies the δ-distance property for $\delta = 1/(\Delta^2 n)$ [17]. For such matrices A Phase 1 of the simplex method can be implemented more efficiently and we obtain the following result.

Theorem 1.9. *For integer matrices $A \in \mathbb{Z}^{m\times n}$, there exists a randomized variant of the shadow vertex simplex algorithm (described in Section 3.3) that solves linear programs with n variables and m constraints using $O(mn^5\Delta^4 \log(\Delta + 1))$ pivots in expectation if a basic feasible solution is given, where Δ denotes an upper bound for the absolute value of any sub-determinant of A. A basic feasible solution can be found using $O(m^6\Delta^4 \log(\Delta + 1))$ pivots in expectation.*

Theorem 1.9 implies in particular that totally unimodular linear programs can be solved by our algorithm with $O(mn^5)$ pivots in expectation if a basic feasible solution is given and with $O(m^6)$ pivots in expectation otherwise.

Besides totally unimodular matrices there exist also other classes of matrices for which $1/\delta$ is polynomially bounded in n. Eisenbrand and Vempala [25] observed, for example, that $\delta = \Omega(1/\sqrt{n})$ for edge-node incidence matrices of undirected graphs with n vertices. One can also argue that δ can be interpreted as a condition number of the matrix A in the following sense: If $1/\delta$ is large then there must be an $(n \times n)$-submatrix of A of rank n that is almost singular.

Related Work

Shadow vertex simplex algorithm We will briefly explain the geometric intuition behind the shadow vertex simplex algorithm. For a complete and more formal description, we refer the reader to [15] or [49]. Let us consider the linear program $\max\{c_0{}^{\mathrm{T}}x \mid Ax \le b\}$ and let $P = \{x \in \mathbb{R}^n \mid Ax \le b\}$ denote the polyhedron of feasible solutions. Assume that an initial vertex x_1 of P is known and assume, for the sake of simplicity, that there is a unique optimal vertex $x^\star$ of P that maximizes the objective function $c_0{}^{\mathrm{T}}x$. The shadow vertex pivot rule first computes a vector $w \in \mathbb{R}^n$ such that the vertex x_1 minimizes the objective function $w^{\mathrm{T}}x$ subject to $x \in P$. Again for the sake of simplicity, let us assume that the vectors c_0 and w are linearly independent.

In the second step, the polyhedron P is projected onto the plane spanned by the vectors c_0 and w. The resulting projection is a (possibly open) polygon P' and one can show that the projections of both the initial vertex x_1 and the optimal vertex $x^\star$ are vertices of this polygon. Additionally, every edge between two vertices x and y of P' corresponds

[2]By *strongly polynomial with respect to n, m, and* $1/\delta$ we mean that the number of steps in the arithmetic model of computation is bounded polynomially in n, m, and $1/\delta$ and the size of the numbers occurring during the algorithm is polynomially bounded in the encoding size of the input.

to an edge of P between two vertices that are projected onto x and y, respectively. Due to these properties a path from the projection of x_1 to the projection of $x^\star$ along the edges of P' corresponds to a path from x_1 to $x^\star$ along the edges of P.

This way, the problem of finding a path from x_1 to $x^\star$ on the polyhedron P is reduced to finding a path between two vertices of a polygon. There are at most two such paths and the shadow vertex pivot rule chooses the one along which the objective ${c_0}^{\mathrm{T}}x$ improves.

Finding short paths In [17] Brunsch and Röglin considered the problem of finding a short path between two given vertices x_1 and x_2 of the polyhedron P along the edges of P. Their algorithm is the following variant of the shadow vertex algorithm: Choose two vectors $w_1, w_2 \in \mathbb{R}^n$ such that x_1 uniquely minimizes ${w_1}^{\mathrm{T}}x$ subject to $x \in P$ and x_2 uniquely maximizes ${w_2}^{\mathrm{T}}x$ subject to $x \in P$. Then project the polyhedron P onto the plane spanned by w_1 and w_2 in order to obtain a polygon P'. Let us call the projection π. By the same arguments as above, it follows that $\pi(x_1)$ and $\pi(x_2)$ are vertices of P' and that a path from $\pi(x_1)$ to $\pi(x_2)$ along the edges of P' can be translated into a path from x_1 to x_2 along the edges of P. Hence, it suffices to compute such a path to solve the problem. Again computing such a path is easy because P' is a two-dimensional polygon.

The vectors w_1 and w_2 are not uniquely determined, but they can be chosen from cones that are determined by the vertices x_1 and x_2 and the polyhedron P. Brunsch and Röglin proved in [17] that the expected path length is $O(\frac{mn^2}{\delta^2})$ if w_1 and w_2 are chosen randomly from these cones. For totally unimodular matrices this implies that the diameter of the polyhedron is bounded by $O(mn^4)$, which improved a previous result by Dyer and Frieze [24] who showed that for this special case paths of length $O(m^3n^{16}\log(mn))$ can be computed efficiently.

Additionally, Bonifas et al. [14] proved that in a polyhedron defined by an integer matrix A between any pair of vertices there exists a path of length $O(\Delta^2 n^4 \log(n\Delta))$ where Δ is the largest absolute value of any sub-determinant of A. For the special case that A is a totally unimodular matrix, this bound simplifies to $O(n^4 \log n)$. Their proof is non-constructive, however.

Geometric random edge Eisenbrand and Vempala [25] have presented an algorithm that solves a linear program $\max\{{c_0}^{\mathrm{T}}x | Ax \leq b\}$ in strongly polynomial time with respect to the parameters n and $1/\delta$. Remarkably the running time of their algorithm does not depend on the number m of constraints. Their algorithm is based on a variant of the random edge pivoting rule. The algorithm performs a random walk on the vertices of the polyhedron whose transition probabilities are chosen such that it quickly attains a distribution close to its stationary distribution.

In the stationary distribution the random walk is likely at a vertex x_c that optimizes an objective function $c^{\mathrm{T}}x$ with $\|c_0 - c\| < \frac{\delta}{2n}$. The δ-distance property guarantees that x_c and the optimal vertex $x^\star$ with respect to the objective function ${c_0}^{\mathrm{T}}x$ lie on a common facet. This facet is then identified and the algorithm is run again in one dimension lower. This is repeated at most n times until all facets of the optimal vertex $x^\star$ are identified. The number of pivots to identify one facet of $x^\star$ is proven to be $O(n^{10}/\delta^8)$. A single pivot can be performed in polynomial time but determining the right transition probabilities is rather sophisticated and requires to approximately integrate a certain function over a convex body.

Let us point out that the number of pivots of our algorithm depends on the number m of constraints. However, Heller showed that for the important special case of totally unimodular linear programs $m = O(n^2)$ [30]. Using this observation we also obtain a bound that depends polynomially only on n for totally unimodular matrices.

Combinatorial linear programs Éva Tardos has proved in 1986 that combinatorial linear programs can be solved in strongly polynomial time [50]. Here combinatorial means that A is an integer matrix whose largest entry is polynomially bounded in n. Her result implies in particular that totally unimodular linear programs can be solved in strongly polynomial time, which is also implied by Theorem 1.9. However, the proof and the techniques used to prove Theorem 1.9 are completely different from those in [50].

Our Contribution

We replace the random walk in the algorithm of Eisenbrand and Vempala by the shadow vertex algorithm. Given a vertex x_0 of the polyhedron P we choose an objective function $w^{\mathrm{T}}x$ for which x_0 is an optimal solution. As in [17] we choose w uniformly at random from the cone determined by x_0. Then we randomly perturb each coefficient in the given objective function ${c_0}^{\mathrm{T}}x$ by a small amount. We denote by $c^{\mathrm{T}}x$ the perturbed objective function. As in [17] we prove that the projection of the polyhedron P onto the plane spanned by w and c has $O(\frac{mn^2}{\delta^2})$ edges in expectation. If the perturbation is so small that $\|c_0 - c\| < \frac{\delta}{2n}$, then the shadow vertex algorithm yields with $O(\frac{mn^2}{\delta^2})$ pivots a solution that has a common facet with the optimal solution $x^\star$. We follow the same approach as Eisenbrand and Vempala and identify the facets of $x^\star$ one by one with at most n calls of the shadow vertex algorithm.

The analysis in [17] exploits that the two objective functions possess the same type of randomness (both are chosen uniformly at random from some cones). This is not the case anymore because every component of c is chosen independently uniformly at random from some interval. This changes the analysis significantly and introduces technical difficulties that we address in our analysis.

The problem when running the simplex method is that a feasible solution needs to be given upfront. Usually, such a solution is determined in Phase 1 by solving a modified linear program with a constraint matrix A' for which a feasible solution is known and whose optimal solution is feasible for the linear program one actually wants to solve. There are several common constructions for this modified linear program, it is, however, not clear how the parameter δ is affected by modifying the linear program. To solve this problem, Eisenbrand and Vempala [25] have suggested a method for Phase 1 for which the modified constraint matrix A' satisfies the δ-distance property for the same δ as the matrix A. However, their method is very different from usual textbook methods and needs to solve m different linear programs to find an initial feasible solution for the given linear program. We show that also one of the usual textbook methods can be applied. We argue that $1/\delta$ increases by a factor of at most $\sqrt{m}$ and that Δ, the absolute value of any sub-determinant of A, does not change at all in case one considers integer matrices. In this construction, the number of variables increases from n to $n + m$.

1.3 Bibliographical Notes

Preliminary versions of the results concerning the analysis of Complete Linkage and Ward's algorithm in Chapter 2 such as the Shadow Vertex Method in Chapter 3 have been published in conference proceedings and in a journal:

- Anna Großwendt, Heiko Röglin, and Melanie Schmidt. Analysis of Ward's Method. In Proc. of the 30th SODA, pp. 2939–2957, 2019.
- Anna Großwendt and Heiko Röglin. Improved Analysis of Complete-Linkage Clustering.
  In Proc. of the 23rd ESA, pp. 656-667, 2015.
  Also appeared in Algorithmica, Volume 78, Issue 4, pp. 1131–1150, 2017.
- Tobias Brunsch, Anna Großwendt, and Heiko Röglin. Solving Totally Unimodular LPs with the Shadow Vertex Algorithm.
  In Proc. of the 32nd STACS, pp. 171-183, 2015.

Chapter 2

Hierarchical Clustering

In this chapter we deal with the quality of hierarchical clusterings and analyze a popular greedy heuristic which computes hierarchical clusterings bottom up. Remember that a hierarchical clustering is a set $\mathcal{C}$ of clusterings of a finite point set P which contains exactly one k-clustering for each $k \in [|P|]$. Moreover each k-clustering for $k \in [|P|]$ is a refinement of the $(k-1)$-clustering contained in $\mathcal{C}$. We mentioned in Chapter 1.1 that in general there does not exist a hierarchical clustering which is optimal on each level of granularity.

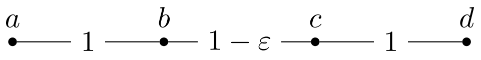


Figure 2.1: The optimal 3-clustering has diameter $1-\varepsilon$ while the optimal 2-clustering has diameter 1. There does not exist a hierarchical clustering which is optimal on each level of granularity.

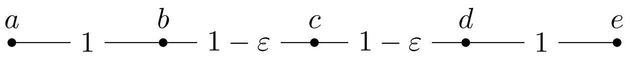


Figure 2.2: An analogous example for the k-center problem. The optimal 3-clustering has discrete radius $1-\varepsilon$ while the optimal 2-clustering has discrete radius 1.

Figure 2.1 and Figure 2.2 show simple examples where no hierarchical clustering exists which is optimal on each level of granularity for the diameter k-clustering problem and the k-center problem.

Definition 2.1. *Let* (M, dist) *be an arbitrary metric space,* $P \subseteq M$ *a finite point set and* cost *an arbitrary objective function. We call a hierarchical clustering* $\mathcal{C}$ *of* P *an* α-hierarchical clustering *if for each integer* $k \in [|P|]$ *and for the* k*-clustering* $\mathcal{C}_k \in \mathcal{C}$ *it holds that* $\text{cost}(\mathcal{C}_k) \leq \alpha \cdot \text{cost}(\mathcal{O}_k)$*, where* $\mathcal{O}_k$ *is an optimal* k*-clustering on* P *with respect to* cost. *We call a hierarchical clustering* minimum hierarchical clustering *if no other hierarchical clustering with smaller approximation factor exists on that instance.*

We deal with the natural question what is the minimum α for which an α-hierarchical clustering exists. Therefore we provide in Section 2.3 a first non-trivial lower bound for the k-center problem. Moreover we derive upper bounds for the most popular objective functions diameter, discrete radius and k-means. In addition to that we analyze in Section 2.4 and Section 2.5 the popular agglomerative greedy algorithm known as Complete Linkage or Ward depending on the considered objective function.

2.1 Outline of the Analysis

We start with a brief overview about the ideas and main techniques which are used in this chapter to prove the main results.

2.1.1 Bounds on the Existence of Hierarchical Clusterings

We start with general lower and upper bounds on the existence of hierarchical clusterings. Remember that a hierarchical clustering $\mathcal{C}$ has approximation factor α on a finite point set P if on each level on granularity $k \in [|P|]$ we have that $\operatorname{cost}(\mathcal{C}_k) \leq \alpha \operatorname{cost}(\mathcal{O}_k)$ where $\mathcal{C}_k \in \mathcal{C}$ is the k-clustering in $\mathcal{C}$ and $\mathcal{O}_k$ is an optimal k-clustering of P. We start with a lower bound for the k-center problem. Therefore we just state a suitable set of instances. Then we prove that on a certain set of granularity levels there exists only one unique hierarchical clustering with approximation factor smaller than two. Moreover this factor tends to two for large instances. It follows that there does not exist a hierarchical clustering with a lower approximation factor on all levels on granularity.

We provide improved upper bounds for the k-center problem and the k-means clustering problem. Therefore we use the algorithm IncApprox(γ, δ) stated in [38]. The algorithm requires an augmentation routine which computes for a given k'-clustering $\mathcal{C}_{k'}$ and an optimal k-clustering $\mathcal{O}_k$ (where $k' > k$) a k-clustering $\mathcal{C}_k$ which is hierarchically compatible with $\mathcal{C}_{k'}$ and the costs of $\mathcal{C}_k$ have an upper bound of $\operatorname{cost}(\mathcal{C}_k) \leq \gamma \operatorname{cost}(\mathcal{C}_{k'}) + \delta \operatorname{cost}(\mathcal{O}_k)$ for real values δ and γ with $\delta, \gamma \geq 1$. Based on that IncApprox(γ, δ) computes a hierarchical clustering with approximation factor $4\gamma\delta$. We provide a simple but new $(1, 1)$-augmentation for the k-center problem and a $(2, 1)$-augmentation for the diameter k-clustering problem and the continuous k-center problem. Moreover we improve the $(18, 8)$-augmentation with respect to k-means given in [38] to a $(4, 2)$-augmentation.

2.1.2 Approximation Guarantees for AC

For the rest of the chapter we analyze the Agglomerative Clustering Greedy **AC** with respect to different objective functions. The algorithm **AC** computes a hierarchical clustering starting with $|P|$ singleton clusters. Now iteratively **AC** merges these two clusters, which minimize the increase of the objective function. Notice that whenever **AC** makes a decision, it is optimal for the clustering in the next step. Where does its error lie? The problem is that every merge forces the points of two clusters to be in the same cluster for any clustering to come. In later clusterings, the condition to cluster certain points together may induce error. The main challenge is now to relate the cost of the k-clustering $\mathcal{C}_k$ computed by **AC** to the cost of an optimal k-clustering $\mathcal{O}_k$ without any information about the structure or geometrical properties of the optimal clustering. The only information given by $\mathcal{O}_k$ is which points may be merged together within costs $\operatorname{cost}(\mathcal{O}_k)$. We use this information to provide **AC** a set $\mathcal{M}$ of possible merge steps at the beginning which costs are related to the cost of $\mathcal{O}_k$. Whenever **AC** merges two clusters $(A, B) \notin \mathcal{M}$ although there exists a suitable merge step $(C, D) \in \mathcal{M}$ (here suitable means that the clusters C and D are available for **AC** in the current step) then we know that $\mathrm{cost}(A, B) \leq \mathrm{cost}(C, D)$ holds because **AC** behaves optimally in each single step. After the merge step A and B are no longer available thus we delete all merges related to A or B from $\mathcal{M}$. Moreover we delete the merge step (C, D) from $\mathcal{M}$ which has been used for an upper bound on $\mathrm{cost}(A, B)$. We use this strategy to give an upper bound on the costs of the performed merge steps and thus on the cost of $\mathcal{C}_k$. However, the objective functions diameter, discrete radius and radius have a clear contrast in its behavior to k-median and k-means as objective functions. The reason is that the first three objectives measure the size of a cluster with respect to the maximum cluster in a k-clustering while the k-means objective is defined as the sum of the squared distances of each point to its center of all clusters in the clustering. Especially this means that the sets of merges which are crucial to determine the objective values differ considerably. Therefore we discuss our idea in the following exemplarily for the objective functions diameter and k-means.

k-means In terms of k-means clustering **AC** is well-known as Ward's algorithm. Therefore we name the algorithm Ward in the following. The k-means objective function considers for each cluster the sum of the squared distances of each point in the cluster to its cluster center. Then it sums up these values over all clusters. Especially that means that we can decompose the costs of a hierarchical clustering $\mathcal{C}$ into costs of single steps as follows: In each step $k \in [|P| - 1]$ we perform exactly one merge step (A, B) which modifies a $(k+1)$-clustering C_{k+1} into a k-clustering C_k. By that the k-means cost of the clustering increase and we denote the amount by $D(A, B)$. If our hierarchical clustering is represented by the sequence of merge steps $m_1, \ldots, m_{|P|-1}$ it directly follows that the k-means cost of $\mathcal{C}_k$ are given by $\mathcal{C}_k = \sum_{i=1}^{|P|-k} D(m_i)$. Now we apply the strategy described above and try to provide upper bounds for all merges $m_1, \ldots, m_{|P|}$ in $\mathcal{M}$. Our main technical idea is that we can assemble the optimal k-clustering $\mathcal{O}_k$ from a sequence of merge steps and each merge step has some cost. Thereby we add some hierarchical structure to $\mathcal{O}_k$. Analogously to $\mathcal{C}_k$ the cost of $\mathcal{O}_k$ are then just given by the sum of the costs of all merge steps in the sequence.

In the following we give some technical details how $\mathcal{M}$ is chosen. Whenever Ward merges two clusters which are contained in a common cluster in $\mathcal{O}_k$ we know that the cost of this merge are included in $\mathrm{opt}_k = \mathrm{cost}(\mathcal{O}_k)$. Thus we are especially interested in situations where Ward differs in its behavior which means Ward merges subclusters from different optimal clusters. To focus on that situation we introduce an equivalence relation where two points p_1, p_2 in P are equivalent if there exists a cluster C with $p_1, p_2 \in C$ constructed by Ward which is contained in some optimum cluster of $\mathcal{O}_k$. We call these clusters inner clusters. The equivalence classes are then given by the largest inner clusters formed by Ward, right before Ward acts differently and merges equivalence classes from different optimal clusters. We know that we can build up $\mathcal{O}_k$ from that equivalence classes by defining an arbitrary path for each cluster O_i in $\mathcal{O}_k$ containing all equivalence classes which are subclusters of O_i. We save this paths in the so called potential graph. The vertices are the equivalence classes while the edges save merge costs for the merges of incident vertices. Note that the sum of all edge weights does not equal opt_k since the optimal clustering proceeds the merges in a path one after another which means in a 2-path of clusters A, B, C the optimal clustering has cost $D(A, B) + D(A \cup B, C)$ while our edge weights are given by $D(A, B)$ and $D(B, C)$ to enable Ward a more generous choice of possible merges.

To find an upper bound for the sum of the edge weights we decompose an optimal k-clustering. Consider an arbitrary cluster $O_j \in \mathcal{O}_k$ and let $P_1^j, \ldots, P_{n_j}^j$ be the equivalence classes O_j consists of. We pretend O_j to result from a hierarchical decomposition. Especially we use two different hierarchical decompositions. Consider the set $\mathcal{M}_j = \{\{P_1^j, P_2^j\}, \{P_2^j, P_3^j\}, \ldots, \{P_{n_j-1}^j, P_{n_j}^j\}\}$ of merges. Observe that the merges in $\mathcal{M}_j$ cannot be applied one after another because after the first merge $\{P_1^j, P_2^j\}$ the singleton class P_2^j is gone, which is to be merged in the second merge $\{P_2^j, P_3^j\}$. But it is possible to do every second merge of $\mathcal{M}_j$, obtaining sets of clusters $\mathcal{M}_1 = \{\{P_1^j, P_2^j\}, \{P_3^j, P_4^j\}, \ldots,\}$ and $\mathcal{M}_2 = \{\{P_2^j, P_3^j\}, \{P_4^j, P_5^j\}, \ldots,\}$. Both sets can be hierarchically completed to O_j and thus all merges in $\mathcal{M}_j$ together cost at most $2\Delta(O_j)$. Now let $\mathcal{M} = \cup_j \mathcal{M}_j$. Then all merges in $\mathcal{M}$ together cost at most $2\,\mathrm{opt}_k$. Together this implies that Ward computes a clustering with cost at most $2\,\mathrm{opt}_k$.

Finally we search for a bijection between the steps of Ward and the edges in the graph. This bijection has the property that every merge of Ward is at most as expensive as the edge assigned to it. Since Ward and the optimal clustering proceed the same number of steps this can only be guaranteed if every merge of Ward decreases the number of available edges by only 1. One can show that this follows from separation assumptions defined in Section 2.4.5.

For the one-dimensional case, the basic approach is similar. The main difference is that without separation, we can no longer guarantee that the number of available merges decreases by only 1 with every step of Ward. Indeed, the original set $\mathcal{M}$ of good merges may be empty after $n - 2k$ merges. To bound the cost of the remaining merge steps, we find a new set of (relatively) good merges, i.e., a set of merges whose costs can be bounded by a constant times opt_k. Again, this set may run dry, and we have to start again. Essentially, we show that after a constant number of *phases* (Ward merges that are charged against a specific set of good merges), Ward has obtained a k-clustering. Although the basic idea is similar, the technical implementation of the proof for $d = 1$ is very different from our proof for well-clusterable data. Every time that Ward does not merge in a way compatible to the optimum clustering, we have to account for all possible consequences. Techniques like reordering help us to organize the proof. We also simplify the instance before the actual proof.

Diameter, Radius and discrete Radius Now we consider exemplarily the objective function diameter. In literature **AC** became popular with respect to the diameter k-clustering problem and the k-center problem under the name Complete Linkage (**CL**). The diameter function behaves very differently in comparison with k-means. The reason is that the objective function diameter considers at most two points in each cluster to calculate the objective value while k-means takes each point in each cluster into account. The most conspicuous geometrical consequence is that while k-means takes rather an outlier than two different dense points in a cluster the diameter objective is not negatively affected by that. A theoretical consequence is that separation assumptions lead already with very small separation parameters to optimal results and are no longer of theoretical interest. Instead, we derive a general upper bound for all instances depending on the dimension d as a parameter.

Therefore we base our analysis on results of Ackermann et al. stated in [2]. As written in the introduction their analysis in [2] proceeds in two phases. The first phase ends when $2k$ clusters are left and the second phase consists of the last k merge operations. In the first phase an approximation factor depending only on d is incurred (even though the factor is doubly exponential in d). Based on the first phase we give an improved analysis of the second phase. Let $\mathcal{C}'$ be the $2k$-clustering computed by **CL**. We know that in phase two only k merge steps are left and again provide a set $\mathcal{M}$ of possible merge steps. Unlike in the case of k-means we are not interested in summing up the costs of all steps in $\mathcal{M}$. Indeed the diameter of a given k-clustering $\mathcal{C}$ refers only to the distance of two specific points contained in one cluster. Especially the costs of merge steps applied before and related to other clusters in $\mathcal{C}$ are not of special interest for the objective function. Due to that we define a threshold t related to the cost of $\mathcal{O}_k$ such that an adequate number of merges can be applied to $\mathcal{C}'$ leading to a diameter of at most t. Let $\text{opt}_k = \text{diam}(\mathcal{O}_k)$ be the cost of an optimal k-clustering. Note that each point in P is contained in some optimal cluster. It follows that each cluster in $\mathcal{C}'$ intersects with at least one optimal cluster. Moreover there exists an optimal cluster which has common points with at least two clusters from $\mathcal{C}'$. Using the triangle inequality it follows that two such clusters can be merged within diameter cost of $2 \cdot \text{diam}(\mathcal{C}') + \text{opt}_k$. To organize such potential merge steps we introduce the concept of *clustering intersection graphs* (CI). The clusters in $\mathcal{C}'$ build the set of vertices of a CI-graph G while optimal clusters from $\mathcal{O}_k$ form hyperedges which connect all clusters in $\mathcal{C}'$ which have a common point with the optimal cluster related to that edge. Then we set $t = 2 \cdot \text{diam}(\mathcal{C}') + \text{opt}_k$ and add k pairs of such incident clusters in G from $\mathcal{C}'$ to $\mathcal{M}$ such that each cluster of $\mathcal{C}'$ is contained in a merge step from $\mathcal{M}$. Note that the merge steps in $\mathcal{M}$ define a subforest F on G.

Unfortunately **CL** could merge clusters which do not intersect in a common optimal cluster and in that case two merge steps are deleted from $\mathcal{M}$ (the two merge steps which contain A or B). We have to deal with the problem that the set of possible merges $\mathcal{M}$ runs empty whereas **CL** did not terminate. We will fix this by filling up $\mathcal{M}$ with new possible merge steps remembering the number of different fill-up processes (note that each fill-up process increases the threshold and therefore the diameter of **CL**). We represent this process in F. Whenever two clusters are merged we contract the corresponding vertices in F. When $\mathcal{M}$ runs empty for a certain threshold d we know that it may be filled up with all pairs of vertices neighbored in F for the new threshold $2d + \text{opt}_k$. To decrease the threshold to $d + \text{opt}_k + \text{diam}(\mathcal{C}')$ we proceed more carefully and add only merges of neighbored vertices to $\mathcal{M}$ where one of the vertices is a leaf. One can argue that the corresponding cluster is contained in $\mathcal{C}'$. Finally we argue that we only need 3 fill-up processes until F has at most k vertices and **CL** terminates. We obtain our results by carefully exploiting the structural properties of F. At the beginning each connected component is a tree and especially each connected component contains a leaf which means there is some merge contained in $\mathcal{M}$. We argue that after two fill-up processes for each connected component either there is a cycle contained or the component is involved in two merge steps in the next period. Finally it follows from counting arguments that after a third fill-up process each component contains a cycle and by that the number of vertices is at most the number of edges which is k.

2.2 Preliminaries

Our instances are given by a finite point set $P \subseteq M$ which is a subset from a metric space (M, dist). We denote by $\text{opt}_k(P)$ the value of a k-clustering that minimizes a specific objective function.

k-means We consider inputs in the Euclidean space $\mathbb{R}^d$. The Euclidean distance of $x_1, x_2 \in \mathbb{R}^d$ is denoted by $||x_1 - x_2|| = ||x_1 - x_2||_2$. Let $P \subset \mathbb{R}^d$ be a finite set of points. For any center $c \in \mathbb{R}^d$, we denote the sum of the squared distances of each point in P to c by $\Delta(P, c) = \sum_{p \in P} ||p - c||^2$. This sum is minimized when the center is the *centroid* $\mu(P) := \frac{1}{|P|} \sum_{p \in P} p$ of P. We set $\Delta(P) := \Delta(P, \mu(P))$. For any set of k centers $C \subset \mathbb{R}^d$, the *k-means objective cost* is $\Delta(P, C) = \sum_{p \in P} \min_{c \in C} ||p - c||^2$. The 1-means cost of P is $\Delta(P)$. If P is weighted with a weight function $w : P \to \mathbb{N}_{\geq 1}$, then we denote the *total weight* by $w(P) := \sum_{x \in P} w(x)$ and extend the above notations by $\mu(P) = \frac{1}{w(P)} \sum_{x \in P} w(x)x$, $\Delta(P, c) = \sum_{x \in P} w(x)||x - c||^2$, and $\Delta(P) = \Delta(P, \mu(P))$. The weighted k-means objective is $\Delta(P, C) = \sum_{x \in P} \min_{c \in C} w(x)||x - c||^2$. We denote the costs of the k-clustering computed by Ward's method on data set P by $\text{Ward}_k(P)$. The following two facts are well known.

Lemma 2.2 (Relaxed triangle inequality). *For all $x, y, z \in \mathbb{R}^d$, $||x - y||^2 \leq 2(||x - z||^2 + ||z - y||^2)$.*

Lemma 2.3. *For any finite point set $P \subset \mathbb{R}^d$ and any $c \in \mathbb{R}^d$, $\Delta(P, c) = \Delta(P) + |P| \cdot ||c - \mu(P)||^2$.*

Proof. See for example Lemma 2.1 in [31] or Lemma 2.4.1 in [47]. □

Lemma 2.3 has the following important consequence. Whenever a set of points P' is clustered *together*, i.e., all points in it are assigned to the same center in a given solution, then the cost for this assignment can be computed by knowing only the centroid of the point set and $\Delta(P')$. Indeed, $\Delta(P')$ is always part of the cost when the points in P' end up in the same cluster, no matter where they are assigned. Consider any bottom-up hierarchical clustering strategy. As soon as a set of points P' has been merged by the strategy, it is clear that all of the points in P' will be assigned together in all clusterings to come. Thus, we can treat such a P' as one weighted point with some additional constant cost. This view is very helpful to simplify the analysis of agglomerative hierarchical clustering strategies and we will adopt it whenever convenient.

Diameter and (discrete) Radius If the metric is not specified we consider general metric spaces (M, dist). Moreover an instance is given by a finite point set $P \subseteq M$. Especially our upper bounds on the existence of α-hierarchical clusterings in Section 2.3.2 are true for arbitrary metric spaces (M, dist). Also our analysis for **CL** in Section 2.5 works for arbitrary metric spaces. Though one should mention that Lemma 2.77, Lemma 2.79

as well as Lemma 2.81 are based on results from [2] which hold true for all metrics which are induced by a norm. For a k-clustering $\mathcal{C}$ we denote the diameter of $\mathcal{C}$ by $\operatorname{diam}(C) := \max_{x,y\in C} \operatorname{dist}(x,y)$. The radius of $\mathcal{C}$ is given by $\operatorname{rad}(C) := \min_{y\in\mathbb{R}^d} \max_{x\in C} \operatorname{dist}(x,y)$. Finally the discrete radius of $\mathcal{C}$ is defined by $\operatorname{drad}(C) := \min_{y\in C} \max_{x\in C} \operatorname{dist}(x,y)$. When it is not clear from the context which variant is meant, we will use the notation $\mathbf{CL}^{\text{drad}}$, $\mathbf{CL}^{\text{rad}}$, and $\mathbf{CL}^{\text{diam}}$ to make the variant clear.

2.3 Existence of Hierarchical Clusterings

In general it is not possible to find a hierarchical clustering which is optimal on every level of granularity. This fact automatically raises the question which is the minimum approximation factor we can guarantee for a hierarchical clustering at each level of granularity.

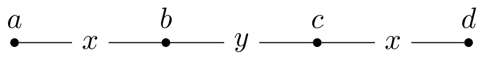


Figure 2.3: Figure 2.1 with more general distances.

Let us consider Figure 2.1 again but try to achieve an approximation factor of the hierarchical clustering as large as possible. As seen above there exists no hierarchical clustering which is optimal on each level if $y < x$. If the hierarchical clustering merges b and c first it is optimal in the first step but realizes an approximation factor of $(x+y)/x$ in the next merge. The other way round when the hierarchical clustering starts with merging a and b it can become an optimal 2-clustering but the approximation factor of the resulting 3-clustering is given by x/y. Note that the minimal approximation factor $\min\{x/y, (x+y)/x\}$ is maximized by the the golden cut which is given by $(\sqrt{5}-1)/2 \approx 0.618$. It follows an approximation factor of approximately 1.618 for the diameter k-clustering problem. Analyzing Figure 2.2 in a similar way we achieve an even smaller maximal approximation factor for the k-center problem of $\sqrt{2} \approx 1.42$ for $a = 1$ and $b = 1/\sqrt{2}$. The reason is that the center may be chosen in an optimal way for the objective function which leads to more conditions on the choice of x and y.

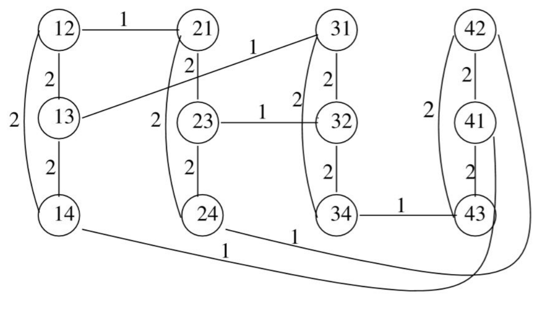


Figure 2.4: An optimal 6-clustering has diameter 1 while the optimal 4 clustering has diameter 2. The hierarchical clustering either starts with an optimal 6-clustering and merges two further clusters (but this leads to a diameter of 4) or starts with another 6-clustering but including an edge of weight 2.

Aparna Das and Claire Kenyon state some instance in [22] where no α-hierarchical clustering with α smaller than 2 exists. This instance is the first non-trivial lower bound for the diameter k-clustering problem and is shown in Figure 2.4. In this section we introduce the first non-trivial lower bound for the k-center problem. We prove that also in the case of the objective function discrete radius in general no hierarchical clustering with approximation factor smaller than 2 exists. In a second step we extend the work of Lin et al. in [38] which improves several upper bounds on the existence of hierarchical clusterings.

2.3.1 A Lower Bound of 2 for the Hierarchical k-Center Problem

Our family of instances consists of a finite point set P_m with 3^{m+1} points for $m \geq 0$ where the metric on P_m is given by a shortest-path metric induced by a tree T_m with vertex set $V(T_m) = P_m$. Especially the distance between two points $v, w \in P_m$ is given by the length of the unique $v - w$-path in T_m. It is helpful to spend some time on general properties of clusterings on tree metrics before we start with the analysis.

Definition 2.4. *We call a clustering on a tree metric convex, if each vertex x on a v, w-path in T, where v and w are part of a common cluster C, is also contained in the cluster C.*

Now we consider a convex k-clustering on a tree. If we build the union of all paths between all pairs of vertices in a cluster we obtain a subtree T' by the above definition and it holds $C = V(T')$. This implies that we can identify clusterings in a tree by sets of edges instead of vertices.

Lemma 2.5. *Each forest with m edges on $V(T) = P$ induces a convex $|P| - m$-clustering on P: Each of the $|P| - m$ connected components defines a cluster, and the points in these clusters are just the points of the related connected component. Vice versa, each convex $|P| - m$ clustering can be represented by a forest in G by connecting all points in the same cluster.*

In each tree metric T and for each $k \in [|V(T)|]$ there exists a convex optimal solution for the k-center problem. Unfortunately this property does not turn over to hierarchical clusterings. A simple example is given in Figure 2.5.

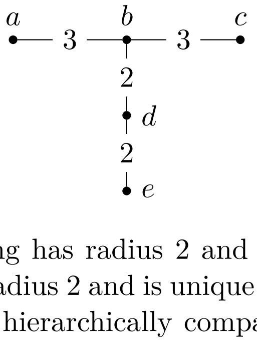


Figure 2.5: An optimal 4-clustering has radius 2 and is given by either $\{b, d\}$ or $\{d, e\}$. The optimal 3-clustering has also radius 2 and is unique with the clusters $\{a\}, \{c\}, \{b, d, e\}$. The optimal 2-clustering which is hierarchically compatible is also unique with radius 3 and given by $\{a, c\}, \{b, d, e\}$. Note that it is not convex.

Nevertheless we use connected components of a forest as representatives for our clusters. We will justify in the analysis, that the minimum hierarchical clustering on a given set of granularity levels is convex in our family of instances.

The Family of Instances $\{T_m\}_{m\geq 0}$ We define $\{T_m\}_{m\geq 0}$ with $P_m = V(T_m)$ recursively as follows: Let T_0 be a tree with three vertices connected by a path with two edges of weight $2^0 = 1$. Then T_m for $m > 0$ starts with the edge- and vertex set of T_{m-1} and adds for each vertex $v \in V(T_{m-1})$ two copies v' and v'' to $V(T_m)$. Moreover we add two edges (v'', v') and (v', v) with weight 2^m to $E(T_m)$. We call the edge (v'', v') outer edge of weight 2^m and the edge (v', v) inner edge of weight 2^m. Moreover we call v' a *designated center on level m*. Figure 2.6 depicts T_2 as the left picture.

Note that all edge weights are positive and thus the tree induces a metric on its vertices. Moreover the most expensive edges in T_m have weight 2^m. The graph T_0 has 3 vertices and in each step the number of vertices increases by a factor of 3. Especially the graph T_m consists of $|V(T_m)| = 3^{m+1}$ vertices. We denote by K_i the set of edges with weight exactly 2^i for $i = 0, \ldots, m$. Let k_i be the number of edges in K_i. Then, $k_0 = 2$. Since for each vertex in T_{m-1} we add two edges with weight 2^m, $k_i = 2 \cdot |V(T_{i-1})| = 2 \cdot 3^i$ for $i = 0, \ldots, m$.

Theorem 2.6. *For the instance $P_m = V(T_m)$ the optimal hierarchical clustering has an approximation factor of at least*

$$\Phi = \frac{2^{m+1} - 1}{2^m} = 2 - \frac{1}{2^m},$$

i.e., for any $\epsilon > 0$, there is an m where the factor is $> 2 - \epsilon$.

Proof. Let $n = |P_m|$ denote the number of points for a fixed parameter $m \geq 0$.

Optimal Clusterings on T_m First we consider the costs of optimal clusterings with $(n - k_i)$ clusters for $i \in [m]$. We argue that an optimal $(n - k_i)$-clustering for $i \in [m]$ has cost 2^i. We have to choose $k_i = 2 \cdot 3^i$ edges in G to obtain an $(n - k_i)$-clustering. We can just choose all edges from K_i and obtain a clustering with cost 2^i, where the centers are the designated centers on level i, and each connected component is a path with two edges of weight 2^i. Thus each cluster has discrete diameter 2^i. We argue that there is no cheaper clustering with $(n - k_i)$ clusters. Each forest including an edge with weight at least 2^i has discrete radius at least 2^i. Thus, to obtain a cheaper $(n - k_i)$-clustering we have to choose edges with weight smaller than 2^i. Note that all edges with weight smaller than 2^i are contained in T_{i-1} and moreover T_{i-1} is a tree. Thus there exist only $|V(T_{i-1})| - 1 = 3^i - 1 < 2 \cdot 3^i = k_i$ such edges. It follows that an optimal $(n - k_i)$-clustering in T_m has cost 2^i.

A Hierarchical Clustering $\mathcal{T}$ on T_m We recursively define a hierarchical clustering $\mathcal{T}$ on T_m which has an approximation factor of $2 - 1/2^m$. For each level of granularity $n - k_0, \ldots, n - k_m$ we denote the $(n - k_i)$-clustering from $\mathcal{T}$ by $\mathcal{T}_i$. Then we argue that $\mathcal{T}$ is basically the only possible hierarchical clustering with a factor smaller than two on the specified granularity levels (it is unique up to symmetry).

The $(n-2)$-clustering $\mathcal{T}_0$ is given by the forest containing the two edges from K_0. Note that this is an optimal $(n-2)$-clustering on T_m. Now assume we have a hierarchical $(n-k_i)$-clustering $\mathcal{T}_0, \ldots, \mathcal{T}_i$ for $i \in [m-1]$. We define the $(n-k_{i+1})$-clustering $\mathcal{T}_{i+1}$ which is hierarchically compatible as follows: First we add all outer edges with weight 2^{i+1}. Our forest then contains $k_i + k_{i+1}/2 = 2 \cdot 3^i + 3^{i+1}$ edges. Thus, we have to choose 3^i further edges to obtain a forest with k_{i+1} edges. Notice that it holds $3^i = k_i/2$. The set K_i of edges with weight 2^i decomposes into $k_i/2$ many 2-paths consisting of an inner and outer edge. Thus it is sufficient to choose one further edge from K_{i+1} for each 2-path in K_i. Let $v - v' - v''$ be a 2-path in K_i. Then its designated center v' is incident to some inner edge of weight 2^{i+1} by construction of T_m. We add this edge to the hierarchical clustering $\mathcal{T}_{i+1}$ (note that these edges are not yet part of the clustering since we only chose outer edges with weight 2^{i+1} or edges with smaller weight in the recursive step before). We choose the designated centers on level $i+1$ as the centers of $\mathcal{T}_{i+1}$. Note that this choice is mandatory for clusters in $\mathcal{T}_{i+1}$ which contain the inner and outer edge with weight 2^{i+1}. Each other choice would lead to a path of length 2^{i+2} and thus to an approximation factor of at least two.

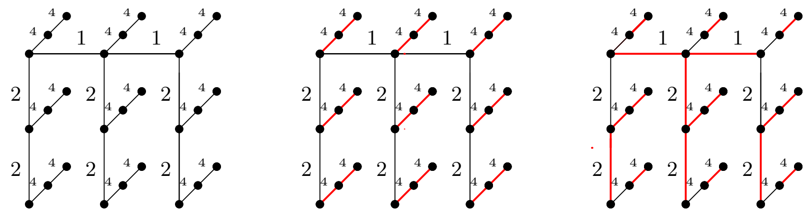


Figure 2.6: The picture shows T_2, an optimal 9-clustering and the hierarchical clustering $\mathcal{T}_2$.

The Approximation Factor We claim that for each $i \in [m]$ the hierarchical clustering $\mathcal{T}_i$ is the only one with approximation factor smaller than two on each level $n - k_j$ for $j \in [i]$ up to isomorphism. Moreover the cluster centers are the designated centers of level i and the discrete radius is given by a path with weight $\sum_{j=0}^{i} 2^j$ containing one edge of each different weight smaller or equal to 2^i outgoing from the cluster centers.
We prove the claim by induction. Furthermore we ensure in each step that the following property is fulfilled by $\mathcal{T}_i$:

($\star$): Each connected component in $\mathcal{T}_i$ restricted to T_i contains either a 2-path of K_i or the component consists of exactly one outer edge of weight 2^i.

Base Case: The optimal $(n-k_0) = (n-2)$-clustering has cost $2^0 = 1$. We have to proceed two merge steps which lead to a clustering with maximum radius smaller than two to achieve an approximation factor smaller than two. There are only two paths between points with suitable length which are each given by one of the two edges of weight 1. Thus there exists only one unique $(n-k_0)$ clustering with the required approximation factor, namely $\mathcal{T}_0$, which is obtained by choosing the edges of weight 1. Moreover setting the center to the designated center on level 1 leads to a radius of 1. Obviously ($\star$) is fulfilled.

Thus, the claim holds for $i = 0$.

Inductive Step: Assume the claim holds for all j where $j \leq i$. We prove that the claim follows for $i+1$. In the i-th step we considered as cluster centers the designated centers of level i. Now we switch to the designated centers on level $i+1$ as centers. We start with an upper bound for the radius of $\mathcal{T}_{i+1}$ which is given by the longest outgoing path from the cluster center.

Claim 2.7. *The $(n-k_{i+1})$-clustering $\mathcal{T}_{i+1}$ has discrete radius $\sum_{j=0}^{i} 2^j$.*

Proof. Consider an arbitrary cluster from $\mathcal{T}_{i+1}$. There are two cases: Either the cluster only consists of the outer edge of K_{i+1}. But then the longest path consists of one edge of weight 2^{i+1}. Note that $(\star)$ is fulfilled in that case. In the second case the cluster contains both, the inner and the outer edge of weight 2^{i+1} (also in that case $(\star)$ is fulfilled). By construction the inner edge is then incident to some designated center on level i. By induction the longest outgoing path of an i-th designated center has length $\sum_{j=0}^{i} 2^j$ containing one edge of each different weight smaller or equal to 2^i. Now it becomes longer by the inner edge with weight 2^{i+1} to a path of length $\sum_{j=0}^{i+1} 2^j$. All other paths increase also by an additive factor of 2^{i+1} and thus cannot become longer than the longest path. All in all we have an upper bound of $\sum_{j=0}^{i+1} 2^j$ on the length of a longest path. One should mention vice versa that by construction for all designated centers of level i the neighbored inner edge of weight 2^{i+1} is part of $\mathcal{T}_{i+1}$. Because of $(\star)$ each cluster contains an edge of weight 2^i which is incident to a designated center. Especially each of the $n-k_i$ clusters gets a new edge of weight 2^{i+1} in $\mathcal{T}_{i+1}$. Thus $(\star)$ is also true for $\mathcal{T}_{i+1}$ restricted to T_{i+1}. Especially the designated center of level i with the outgoing path of length $\sum_{j=0}^{i} 2^j$ becomes a new incident edge of weight 2^{i+1} which is incident to the new cluster center c (the designated center of level 2^{i+1}). It follows that c has an outgoing path of lenght $\sum_{j=0}^{i+1} 2^j$ in $\mathcal{T}_{i+1}$. $\square$

Since the optimal $(n-k_{i+1})$-clustering has cost 2^{i+1}, the approximation factor is given by

$$\frac{\sum_{j=0}^{i+1} 2^j}{2^{i+1}} = \frac{2^{i+2}-1}{2^{i+1}} < 2.$$

It remains to show that the hierarchical clustering stays unique with this property up to isomorphism. In that context the isomorphism appears when we have a connected component which consists for example of a single outer edge of weight 2^i. It is not mandatory to choose the inner edge with weight 2^{i+1} incident to its designated center on level i. We could just choose the inner edge incident to the outer vertex and obtain an isomorphic version of our clustering. By induction, the hierarchical clustering is unique up to isomorphism on all previous steps. To prove that the clustering stays unique, it is sufficient to show that there is no other hierarchically compatible clustering with $\mathcal{T}_i$ which has an approximation factor smaller than two. For the rest of the proof we may restrict our attention to the subtree T_{i+1}, since larger edges are not valid for the construction of the clustering as argued above. By that we shrink the number of points from 3^{m+1} to 3^{i+2}. That means we lose a point set of size $3^{m+1} - 3^{i+2}$ and we know that each point in that point set builds necessarily a singleton cluster in $\mathcal{T}_{i+1}$. Thus in the following we search for $n - k_{i+1} - (3^{m+1} - 3^{i+2}) = 3^{i+2} - 2 \cdot 3^{i+1} = 3^{i+1}$ clusters in T_{i+1}.

Claim 2.8. *Consider a 2-path v, v', v'' in K_{i+1} where v is contained in T_i. Then v' and v'' are contained in the same cluster of $\mathcal{T}_{i+1}$.*

Proof. First note that two different outer vertices (which are incident to outer edges in K_{i+1} and have degree one in T_{i+1}) may not be contained in the same cluster. Whenever a cluster contains such a vertex w the cluster center has to be chosen as the designated center on level $i+1$ which has a common edge with w. Otherwise the cluster would have a path of length at least 2^{i+2}. Since the optimal $(n - k_{i+1})$-clustering has radius 2^{i+1} this leads to an approximation factor of at least two. But we may only choose one center and therefore we cannot have two vertices of that type in one cluster. But since there are 3^{i+1} many of that points and we search for 3^{i+1} clusters each cluster contains exactly one of the outer vertices in K_{i+1}. Now we argue that v' and v'' are contained in the same cluster. In fact assume v' is contained in another cluster C. But C contains also an outer vertex w and moreover the cluster center is the designated center on level $i+1$ which has a common outer edge with w. Note that each path from the cluster center passes an inner edge of weight 2^{i+1} incident to the cluster center. But then the path from the cluster center to v' contains two edges of weight 2^{i+1} which leads to an approximation factor of at least two. It follows that all outer edges of weight $i+1$ are necessarily contained in $\mathcal{T}_{i+1}$. $\square$

Now we show that based on this fact there is essentially only one way to construct $\mathcal{T}_{i+1}$. From Claim 2.8 it follows that each of the 3^{i+1} outer edges in K_{i+1} has to be contained in its own cluster in $\mathcal{T}_{i+1}$ without another outer edge of weight 2^{i+1}. Since we search for a 3^{i+1}-clustering we have to deal with the question how the clusters in $\mathcal{T}_i \mid_{T_i}$ are allocated to the outer edges from K_{i+1}. Especially it holds that if a cluster $C \in \mathcal{T}_i \mid_{T_i}$ is contained in a cluster $C' \in \mathcal{T}_{i+1}$, then C' also contains some outer edge from K_{i+1}.

Claim 2.9. *Assume $C \in \mathcal{T}_i \mid_{T_i}$ and $e_{out} \in K_{i+1}$ are contained in a common cluster $C' \in \mathcal{T}_{i+1}$. Let e_{in} be the corresponding inner edge in K_{i+1}. Then e_{in} is incident to a point v contained in C.*

Proof. We argued above that necessarily the designated center c on level $i+1$ incident to e_{out} builds the cluster center of C'. Let e_{in} be the inner edge incident to c. Note that each path from c to a point in C contains e_{in} as an edge. By ($\star$) we know that each connected component of $\mathcal{T}_i \mid_{T_i}$ contains an edge of weight 2^i. We consider two cases: Either a pair of an inner and outer edge of level 2^i is contained in C. But then the path from the cluster center must pass an edge incident to the designated center of level i to avoid two edges of weight 2^i. Otherwise there exists a path from the cluster center which contains an edge of weight 2^{i+1} and two edges of weight 2^i which leads to a radius of at least $2^{i+1} + 2 \cdot 2^i = 2^{i+2}$ and with other words an approximation factor of at least two. But by construction of the instance there are only inner edges from K_{i+1} incident to the designated center. Since we may not pass two edges of weight 2^{i+1} this edge equals e_{in}. The other case is when C contains only an outer edge of weight 2^i. Let $e'_{out} = (x, y)$ be that outer edge of weight 2^i. If e_{in} is not incident to x or y, then the path from the cluster center to e'_{out}'s outer vertex passes e_{in} of weight 2^{i+1}. Then it passes the inner edge of weight 2^i which lies on the

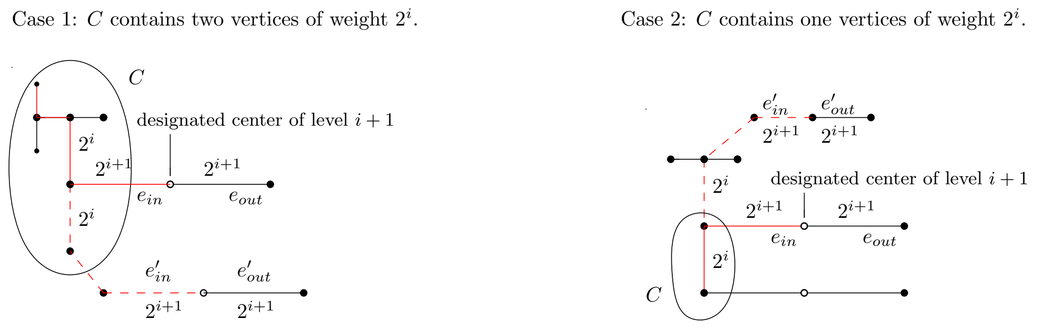


Figure 2.7: The two cases specify the position of the outer and inner edge with respect to a merged cluster from $C \in \mathcal{T}_i |_{T_i}$. Here e_{in} and e_{out} were suitable candidates for outer edges merged with C while e'_{in} and e'_{out} are other candidates. The dotted red edges are paths from the cluster center of other candidates which lead to a path length of 2^{2+1}. Thus these edges may not be merged with C to avoid a too large approximation factor.

2-path of e'_{out} in K_i and finally e'_{out}. But then the path contains again an edge of weight 2^{i+1} and two edges of weight 2^i. Thus e_{in} is incident to x or y. $\square$

Claim 2.9 provides a lot of information how $\mathcal{T}_{i+1}$ has to be built. For a cluster $C \in \mathcal{T}_i |_{T_i}$ we know exactly which outer edge from K_{i+1} is merged with C. First we know that the corresponding inner edge from K_{i+1} is incident to a vertex in C. This provides the information, that the merge step can be represented by adding an inner edge from K_{i+1} to $\mathcal{T}_i$. Moreover the proof gives us the exact information which inner edge e_{in} must be chosen. If C is not given by a single outer edge then e_{in} is given by the inner edge incident to the designated center on level i contained in C. If C is given by a single outer edge from K_i one can choose between two incident edges from K_{i+1}. We mention that in both cases the resulting component consists of a path of length 3 which starts with an edge of weight 2^i followed by two edges of weight 2^{i+1}. Thus both versions are clearly isomorphic. Without loss of generality we may assume that the inner and outer edge from K_{i+1} are incident to the designated center of level i. Otherwise we use the isomorphism to map further merge steps extending the forest later into an isormorphic version with that property. In fact this is possible because the instance is a symmetrical construction.

Finally a clustering is a partition and thus $v \in e_{in}$ is only contained in one cluster from $\mathcal{T}_i$. Thus no two clusters from $\mathcal{T}_i |_{T_i}$ are merged in $\mathcal{T}_{i+1}$. Up to isomorphism there remains only one possible $(n - k_{i+1})$-clustering which is hierarchically compatible to $\mathcal{T}_i$ and has approximation factor smaller than 2.

$\square$

Theorem 1.3. *For each $\epsilon > 0$ there exists a metric space (M, d) and a finite point set $P \subseteq M$ where the minimum hierarchical clustering for the k-center problem on P has approximation factor larger than $2 - \epsilon$.*

2.3.2 Upper Bounds on the Existence of a Hierarchical Clustering

In this section we give upper bounds on the existence of hierarchical clusterings. Therefore we improve results stated by Lin et al. in [38] by providing a better subroutine of their algorithm and extend their results to the k-center problem. We start by restating the essential framework in terms of clustering.

Definition 2.10 ((δ, γ)-augmentation)**.** *We say that the (δ, γ)-augmentation property holds for reals $\gamma, \delta \geq 0$ if for each k'-clustering $\mathcal{C}'$ of a point set P and for each integer $k' > k$ there exists an augmented k-clustering $\mathcal{C}$ such that $\mathcal{C}'$ is a refinement of $\mathcal{C}$ and for the costs of $\mathcal{C}$ it holds:*

$$\operatorname{cost}(\mathcal{C}) \leq \gamma \operatorname{cost}(\mathcal{C}') + \delta(\operatorname{cost}(\mathcal{O}_k)).$$

Let Augment$(\mathcal{C}', k, \gamma, \delta)$ *denote a subroutine that computes such an augmentation.*

We state now the algorithm from [38] which computes a hierarchical clustering. Again for the sake of simplicity we reformulate the algorithm for the special case of clustering.

Algorithm 1 IncApprox(γ, δ).

```
1: Initialization: Let $\mathcal{C}_0$ be a $k$-clustering, where $k'$ is the smallest integer for which
   $\operatorname{cost}(\mathcal{C}_0) = 0$. Set $i = 0$.
2: Iteration $i$: $\mathcal{C}_{i+1} = \text{Augment}(\mathcal{C}_i, k, \gamma, \delta)$, where $k$ is the smallest integer for which
   $\operatorname{cost}(\mathcal{C}_{i+1})$ is at most $(2\gamma)^{i+1}$.
3: Termination: If $|C_{i+1}| > 1$ set $i = i + 1$ and go to Step 2; otherwise return sequence
   $\mathcal{C}_0, \ldots, \mathcal{C}_{i+1}$.
```

We mention that IncApprox(γ, δ) does not necessarily compute a complete hierarchical clustering, which means that there may be some k where no clustering in computed. To build a complete hierarchical clustering for every level of granularity it is sufficient to chose the smallest integer k' with $k' > k$ and the largest integer k^* with $k > k^*$ for which IncApprox(γ, δ) computed a k'-clustering $\mathcal{C}'$ and a k^*-clustering $\mathcal{C}^*$, respectively. Then one may chose an arbitrary hierarchical sequence which transforms C' into C^*. Note that this is possible since C' is a refinement of C^*. The approximation guarantees of the following theorem remain valid for all such clusterings in between.

Theorem 2.11 (Theorem 2.3 of [38])**.** *If (δ, γ)-augmentation holds for reals $\gamma \geq 1, \delta \geq 1$, then IncApprox$(\gamma, \delta)$ computes a hierarchical clustering with approximation ratio $4\gamma\delta$.*

One should say that these results are a priori only of theoretical interest. The augmentation routine takes as an input an optimal k-clustering which is clearly unknown. Lin et. al. provide therefore a variant of their algorithm, which calculates a $4\gamma\delta\alpha$-approximation, for the case where an α-approximation for the optimal clustering is known for each $k \in [|P|]$. Here we are only interested in theoretically bounds on the existence of hierarchical clusterings. Thus in the following we derive (δ, γ)-augmentations for the objective functions (discrete) radius, diameter and k-means.

$(1,1)$-Augmentation for discrete Radius

We start with a simple $(1,1)$-augmentation for the case of k-center clustering which directly leads to the following theorem.

Theorem 1.1. *For each finite point set P and each metric (M,d) with $P \subseteq M$ there exists a hierarchical clustering with approximation factor 4 on each level of granularity for the k-center problem.*

Proof. By Theorem 2.11 it is sufficient to prove that a $(1,1)$-augmentation exists. Therefore assume we have a k-clustering $\mathcal{C}$ and an integer $k' < k$. Let $\mathcal{O}_{k'} = \{O_1, \ldots, O_{k'}\}$ be an optimal k'-clustering of P. For each cluster $C \in \mathcal{C}$ let x_C be its center. Note that x_C is also contained in an optimal cluster, say O_i with its center x_{O_i}. Because of triangle inequality we know that for each point $x \in C$ we have $\operatorname{dist}(x, x_{O_i}) \leq \operatorname{dist}(x, x_C) + \operatorname{dist}(x_C, x_{O_i}) \leq \operatorname{cost}(\mathcal{C}) + \operatorname{cost}(\mathcal{O}_{k'})$. Thus merging all clusters from $\mathcal{C}$ whose centers lie in a common optimal cluster of $\mathcal{O}_{k'}$ leads to a $(1,1)$-augmentation. $\square$

$(2,1)$-Augmentation for Radius and Diameter

Also there is a natural $(2,1)$-augmentation for the objective functions radius and diameter.

Theorem 2.12. *For each finite point set P and each metric (M,d) with $P \subseteq M$ there exists a hierarchical clustering with approximation factor 8 on each level of granularity for the continuous k-center problem and the diameter k-clustering problem.*

Proof. By Theorem 2.11 it is sufficient to prove the existence of a $(2,1)$-augmentation. Therefore assume we have a k-clustering $\mathcal{C}$ and an integer $k' < k$. Let $\mathcal{O}_{k'} = \{O_1, \ldots, O_{k'}\}$ be an optimal k'-clustering of P. For each cluster $C \in \mathcal{C}$ let x_C be an arbitrary but fixed point in C. Now merge all clusters from $\mathcal{C}$ whose labeled points x_C lie in a common optimal cluster of $\mathcal{O}_{k'}$. Denote the resulting k'-clustering by $\mathcal{C}'$. To calculate the objective let $C' \in \mathcal{C}'$ be an arbitrary cluster. If the objective function is given by the diameter of C' chose two points $x, y \in C'$ which maximize the distance under all pairs of points in C'. By definition of C' there exist points p_x and p_y which are elements in the same optimal cluster $O \in \mathcal{O}_{k'}$ and we have that $d(x, p_x) \leq \operatorname{diam}(\mathcal{C})$ such as $d(y, p_y) \leq \operatorname{diam}(\mathcal{C})$. Moreover $d(p_x, p_y) \leq \operatorname{diam}(\mathcal{O}_{k'})$. Using the triangle inequality we obtain $\operatorname{diam}(C') \leq \operatorname{diam}(\mathcal{O}_{k'}) + 2\operatorname{diam}(\mathcal{C})$ which proves the claim. $\square$

$(4,2)$-Augmentation for k-means

We define a natural (δ, γ)-augmentation for the k-means clustering problem as follows. Assume we have a k'-clustering $\mathcal{C}'$ such as centers $C = c_1, \ldots, c_k$ of an optimal k-clustering $\mathcal{O}_k$. Note that $\operatorname{opt}_k = \operatorname{cost}(\mathcal{O}_k) = \Delta(P, C)$. We allocate each cluster in $\mathcal{C}'$ to the optimal center which minimizes the 1-means cost of the cluster with respect to that center. The following Lemma relates the costs of the allocation to the costs of $\mathcal{C}'$ and $\mathcal{O}_k$.

Lemma 2.13. *Let $C = \{c_1, \ldots c_k\} \subset \mathbb{R}^d$ be any set of k centers, and let M be any subset of the points $P \subseteq \mathbb{R}^d$.*

$$\min_{i \in [k]} \Delta(M, c_i) \leq 4\Delta(M) + 2\Delta(M, C).$$

Proof. Define

$$M_1 = \{x \in P \mid ||c_1 - x|| \leq ||c_j - x|| \forall j \in \{2, \dots, k\}\}$$

and

$$M_i = \{x \in P \backslash (M_1, \dots, M_{i-1}) \mid ||c_1 - x|| \leq ||c_j - x|| \forall j \in \{2, \dots, k\}\}$$

for $i \in \{2, \dots, k\}$. Abbreviate $n_i = |M_i|$, $\mu_i = \mu(M_i)$ and $\mu = \mu(M)$. Observe that by Lemma 2.3,

$$\min_{i \in [k]} \Delta(M, c_i) = \Delta(M) + \min_{i \in [k]} |M| \cdot ||\mu - c_i||^2.$$

Thus it is optimal to cluster M with the center c_i which is closest to $\mu = \mu(M)$. Without loss of generality, assume that the numbering is such that this center is c_1. The optimal assignment cost thus is $\Delta(M, c_1)$. We rewrite this cost by splitting it up for the points in the k different subsets M_j, again using Lemma 2.3:

$$\begin{aligned}
\Delta(M, c_1) &= \sum_{j=1}^{k} \left(\Delta(M_i) + |M_i| \cdot ||\mu_i - c_1||^2 \right) \\
&= \sum_{j=1}^{k} \Delta(M_i) + \sum_{j=1}^{k} |M_i| (||\mu_i - \mu|| + ||\mu - c_1||)^2 \\
&\leq \sum_{j=1}^{k} \Delta(M_i) + \sum_{j=1}^{k} |M_i| (||\mu_i - \mu|| + ||\mu - c_1||)^2 \\
&\leq \sum_{j=1}^{k} \Delta(M_i) + \sum_{j=1}^{k} |M_i| (||\mu_i - \mu|| + ||\mu - c_i||)^2 \\
&\leq \sum_{j=1}^{k} \Delta(M_i) + \sum_{j=1}^{k} |M_i| (2 \cdot ||\mu_i - \mu|| + ||\mu_i - c_i||)^2 \\
&\leq \sum_{j=1}^{k} \Delta(M_i) + 4 \sum_{j=1}^{k} |M_i| \cdot ||\mu_i - \mu||^2 + 2 \sum_{j=1}^{k} |M_i| \cdot ||\mu_i - c_i||^2
\end{aligned}$$

The second inequality holds because c_1 is the closest center to μ, the other first and third inequalities are due to the triangle inequality. We know that

$$\Delta(M) = \sum_{i=1}^{k} \left(\Delta(M_i) + |M_i| \cdot ||\mu_i - \mu||^2 \right) = \sum_{j=1}^{k} \Delta(M_i) + \sum_{i=1}^{k} |M_i| \cdot ||\mu_i - \mu||^2.$$

Furthermore,

$$\Delta(M, C) = \sum_{i=1}^{k} \Delta(M_i, c_i) = \sum_{i=1}^{k} \Delta(M_i) + \sum_{i=1}^{k} |M_i| \cdot ||\mu_i - c_i||^2.$$

Thus, we get that

$$\Delta(M, c_1) \leq 4\Delta(M) + 2\Delta(M, C).$$

□

Corollary 2.14. *Let $\mathcal{C}' = C_1, \ldots, C_{k'}$ be any k'-clustering of $P \subset \mathbb{R}^d$ and $C = \{c_1, \ldots c_k\} \subset \mathbb{R}^d$ be an optimal k-means solution. Then*

$$\sum_{i=1}^{k'} \min_{j \in \{1,\ldots,k\}} \Delta(C_i, c_j) \leq 4 \sum_{i=1}^{k'} \Delta(C_i) + 2 \cdot \Delta(P, C) = 4 \operatorname{cost}(\mathcal{C}') + 2 \operatorname{cost}(\mathcal{O}_k).$$

Proof. We apply Lemma 2.13 to each C_i separately to obtain that $\min_{j \in [k]} \Delta(C_i, c_j) \leq 4\Delta(C_i) + 2\Delta(C_i, C)$, and then add up the bounds. □

Note that Corollary 2.14 derives an upper bound for the costs when we allocate each cluster in $\mathcal{C}'$ to the nearest center of the optimal solution. It follows that the resulting k-clustering is a $(4, 2)$-augmentation. We mention that together with the results in [38] this directly turns over into a 32α-approximation algorithm for the hierarchical k-center problem.

Theorem 1.2. *For each finite point set $P \in \mathbb{R}^d$ there exists a hierarchical clustering with approximation factor 32 on each level of granularity for the k-means problem. Moreover given an α-approximation algorithm for the k-means problem, a hierarchical clustering with approximation ratio 32α can be computed.*

2.4 Ward's Algorithm

We come to the analysis of Ward's Algorithm which computes a solution for the k-means clustering problem. We motivated above that we will analyze the costs of the algorithm as the sum of the costs of each single step. Therefore we define the increase of the costs per step $D(A, B)$ in the next section. After that we state a family of instances of increasing dimension d where Ward computes for some number $k = k(d)$ of clusters a k-clustering that costs $\Omega((3/2)^d \operatorname{opt}_k)$ in Section 2.4.3. In Section 2.4.4 we analyze Ward for 1-dimensional instances. Finally in Section 2.4.6 and Section 2.4.7 we analyze the behavior of Ward's Algorithm on well-separated instances with respect to different separation assumptions in arbitrary dimensions.

2.4.1 Cost of one step

To describe Ward's method, the easiest way is to define the following quantity that describes how much the sum of the 1-means costs increases when merging two clusters.

Definition 2.15. *Let $A, B \subset \mathbb{R}^d$ be two finite point sets. We define $D(A, B) = \Delta(A \cup B) - \Delta(A) - \Delta(B)$. If a set contains only one point, e.g., $A = \{a\}$, we slightly abuse notation and write $D(a, B) = D(\{a\}, B)$ (similarly, if $A = \{a\}$ and $B = \{b\}$, we write $D(a, b) = D(\{a\}, \{b\})$).*

The value $D(A, B)$ plays a central role in the analysis of Ward's method. By using Lemma 2.3, it is easy to show that $D(A, B)$ does not depend on $\Delta(A)$ or $\Delta(B)$. The following lemma gives an explicit formula, which leads to convenient upper and lower bounds. These bounds say that the cost of merging two clusters is roughly equivalent to assigning the points of the smaller cluster to the centroid of the larger cluster.

Lemma 2.16. *Let A and B be two clusters. Then $D(A,B) = \frac{|A||B|}{|A|+|B|} \cdot ||\mu_A - \mu_B||^2$. Furthermore, $\frac{1}{2} \cdot \min\{|A|,|B|\} \cdot ||\mu_A - \mu_B||^2 \leq D(A,B) \leq \min\{|A|,|B|\} \cdot ||\mu_A - \mu_B||^2$. The left hand side is attained for $|A| = |B|$, and the right hand side for $\frac{\max\{|A|,|B|\}}{\min\{|A|,|B|\}} \to \infty$.*

Proof. Notice that $\mu(A \cup B) = \frac{|A|\mu(A)+|B|\mu(B)}{|A|+|B|}$ and thus $\mu(A) - \mu(A \cup B) = \frac{|B|}{|A|+|B|}(\mu(A) - \mu(B))$ and $\mu(B) - \mu(A \cup B) = \frac{|A|}{|A|+|B|}(\mu(B) - \mu(A))$, respectively. By Lemma 2.3, we get that

$$\begin{aligned}
&D(A,B) = \Delta(A \cup B) - \Delta(A) - \Delta(B) \\
=&\Delta(A, \mu(A \cup B)) + \Delta(B, \mu(A \cup B)) - \Delta(A) - \Delta(B) \\
=&\Delta(A) + \Delta(B) + |A| \cdot ||\mu(A) - \mu(A \cup B)||^2 + |B| \cdot ||\mu(B) - \mu(A, \cup B)||^2 - \Delta(A) - \Delta(B) \\
=&|A| \cdot \frac{|B|^2}{(|A|+|B|)^2}||\mu(A) - \mu(B)||^2 + |B| \cdot \frac{|A|^2}{(|A|+|B|)^2}||\mu(A) - \mu(B)||^2 \\
=&\frac{|A||B|}{|A|+|B|}||\mu(A) - \mu(B)||^2 \cdot \left(\frac{|B|}{|A|+|B|} + \frac{|A|}{|A|+|B|}\right),
\end{aligned}$$

which implies the first statement of the lemma. Now we estimate $D(A,B)$. Observe that

$$\frac{|A||B|}{|A|+|B|} = \frac{\min\{|A|,|B|\}}{1 + \frac{\min\{|A|,|B|\}}{\max\{|A|,|B|\}}}.$$

Since $1 \leq 1 + \frac{\min\{|A|,|B|\}}{\max\{|A|,|B|\}} \leq 2$, we get that

$$\frac{1}{2} \cdot \min\{|A|,|B|\} \cdot ||\mu_A - \mu_B||^2 \leq D(A,B) \leq \min\{|A|,|B|\} \cdot ||\mu_A - \mu_B||^2.$$

The left hand side is attained when $|A| = |B|$. The right hand side is not attained for finite $|A|$ and $|B|$, but when $\max\{|A|,|B|\}/\min\{|A|,|B\}$ goes to infinity, $D(A,B)$ approaches the right hand side. □

2.4.2 Monotonicity

Notice that performing arbitrary merge operations is not monotone: Say that $a < b < c$ are one-dimensional points such that the centroid of a and c is b. Then merging a and c first results in a point set where merging with b costs nothing; clearly, this is not monotone. It is natural to assume that costs of the merges of Ward's method are monotonically increasing, and it is indeed true. The main technical hurdle to show that Ward is monotone is contained in the proof of the following decomposition lemma for $D(A,B)$.

Lemma 2.17. *Let A, B, and C be clusters. Then*

$$D(A \cup B, C) = \frac{|A|+|C|}{|A|+|B|+|C|}D(A,C) + \frac{|B|+|C|}{|A|+|B|+|C|}D(B,C) - \frac{|C|}{|A|+|B|+|C|}D(A,B).$$

Proof. In the following, we use the abbreviations $a = |A|$, $b = |B|$, $c = |C|$, $\mu_{AB} = \mu(A \cup B)$, $\mu_A = \mu(A)$ and $\mu_B = \mu(B)$. In the following calculation, we use the bilinearity of the inner product and the fact that $\mu_{A\cup B} = \frac{a}{a+b}\mu_A + \frac{b}{a+b}\mu_B$:

$$\begin{aligned}
||\mu_{AB} - \mu_C||^2 =& \langle \mu_{AB} - \mu_C, \mu_{AB} - \mu_C\rangle = \langle \mu_{AB}, \mu_{AB}\rangle - 2\langle \mu_{AB}, \mu_C\rangle + \langle \mu_C, \mu_C\rangle \\
=& \frac{a^2}{(a+b)^2}\langle \mu_A, \mu_A\rangle + 2\frac{ab}{(a+b)^2}\langle \mu_A, \mu_B\rangle + \frac{b^2}{(a+b)^2}\langle \mu_B, \mu_B\rangle \\
& - 2\frac{a}{a+b}\langle \mu_A, \mu_C\rangle - 2\frac{b}{a+b}\langle \mu_B, \mu_C\rangle + \langle \mu_C, \mu_C\rangle \\
=& \frac{a}{a+b}(\langle \mu_A, \mu_A\rangle - 2\langle \mu_A, \mu_C\rangle + \langle \mu_C, \mu_C\rangle) \\
& + \frac{b}{a+b}(\langle \mu_B, \mu_B\rangle - 2\langle \mu_B, \mu_C\rangle + \langle \mu_C, \mu_C\rangle) \\
& - \frac{ab}{(a+b)^2}(\langle \mu_A, \mu_A\rangle - 2\langle \mu_A, \mu_B\rangle + \langle \mu_B, \mu_B\rangle) \\
=& \frac{a}{a+b}\langle \mu_A - \mu_C, \mu_A - \mu_C\rangle + \frac{b}{a+b}\langle \mu_B - \mu_C, \mu_B - \mu_C\rangle \\
& - \frac{ab}{(a+b)^2}\langle \mu_A - \mu_B, \mu_A - \mu_B\rangle \\
=& \frac{a}{a+b}||\mu_A - \mu_C||^2 + \frac{b}{a+b}||\mu_B - \mu_C||^2 - \frac{ab}{(a+b)^2}||\mu_A - \mu_B||^2.
\end{aligned}$$

It follows from Lemma 2.16 that

$$\begin{aligned}
D(A\cup B, C) &= \frac{(a+b)c}{a+b+c}||\mu_{AB} - \mu_C||^2 \\
&= \frac{ac}{a+b+c}||\mu_A - \mu_C||^2 + \frac{bc}{a+b+c}||\mu_B - \mu_C||^2 - \frac{abc}{(a+b)(a+b+c)}||\mu_A - \mu_B||^2 \\
&= \frac{a+c}{a+b+c}D(A,C) + \frac{b+c}{a+b+c}D(B,C) - \frac{c}{a+b+c}D(A,B). \qquad \square
\end{aligned}$$

Corollary 2.18. *[Monotonicity of Ward's method] Let D_i be the increase of the objective function in the i-th step of Ward's method. Then $D_i \le D_j$ for $i \le j$.*

Proof. Assume this is not true. Then there exists an i such that $D_i > D_{i+1}$. Let A and B be the clusters merged in the i-th step. In the $(i+1)$th step $A \cup B$ has to be merged with another cluster C: If the merge in the $(i+1)$th step merges to clusters unrelated to A and B, then this merge could be done in the ith step, and Ward's method would have chosen it. Hence, $D_i = D(A,B)$ and $D_{i+1} = D(A\cup B, C)$ for clusters A, B, C. Now observe that

$$\begin{aligned}
D(A\cup B, C) &= \frac{|A|+|C|}{|A|+|B|+|C|}D(A,C) + \frac{|B|+|C|}{|A|+|B|+|C|}D(B,C) - \frac{|C|}{|A|+|B|+|C|}D(A,B) \\
&\ge \frac{|A|+|C|}{|A|+|B|+|C|}D(A,B) + \frac{|B|+|C|}{|A|+|B|+|C|}D(A,B) - \frac{|C|}{|A|+|B|+|C|}D(A,B) \\
&= D(A,B),
\end{aligned}$$

where the inequality follows from the choice of Ward's method. We get a contradiction. $\square$

Monotonicity is a very helpful property. It allows us to analyze the costs of possible merge steps at some certain point of the algorithm. Then we can directly derive an upper bound for the costs of previous performed merge steps. For example in the argument discussed in Section 2.4.6 we use, e.g., that all merges that are possible in the final k-clustering computed by Ward's method are at least as expensive as all merges that are performed before by Ward's method to obtain the k-clustering.

k-median as an unexpected example

It turns out that different objective functions may not behave monotonously with respect to agglomerative clustering. Unfortunately k-median is a very famous example of this type.

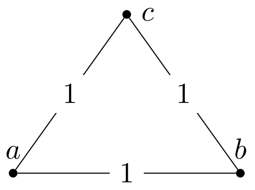


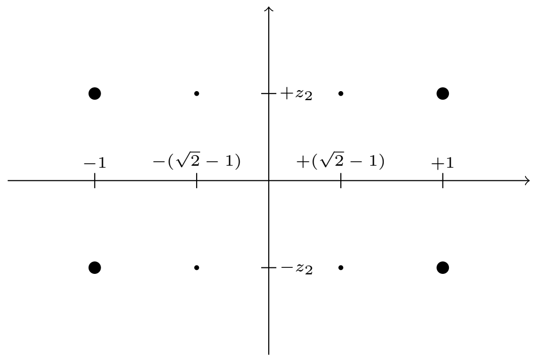
Figure 2.8: Merging two arbitrary points of $\{a, b, c\}$ causes an increase of the costs by 1. Merging the resulting 2 clusters in a second step increases the k-median costs by an amount smaller than 1.

Figure 2.8 shows a simple example from [34] where Ward does not behave monotonously. It follows that we cannot use our techniques to give a 2-approximation for well-separated instances with respect to k-median and also the proof of the one dimensional case cannot be transferred into terms of k-median. However the property that Ward works exact on balanced, well-separated instances is also true for k-median, which is also derivated in [34].

2.4.3 Exponential Lower Bound in High Dimension

In the following, we describe a family of instances of increasing dimension d where Ward computes for some number $k = k(d)$ of clusters a k-clustering that costs $\Omega((3/2)^d \operatorname{opt}_k)$. Here and in all other worst-case examples, we assume that given a choice between equally expensive merges, Ward chooses the action that leads to a worse outcome. This is without loss of generality because we can always slightly move the points to ensure the outcome we want. However, it greatly simplifies the exposition.

To further simplify the exposition, we start by giving an instance containing points of infinite weight and assume that the optimal cluster centers coincide with these infinite weight points. For any finite realization of the example, that is not the case. To ensure that Ward actually behaves like described in the following, we have to move the high weight points by an infinitesimal distance. Notice that merging a cluster H of infinite weight with a cluster A of finite weight costs $|A| \cdot ||\mu(A) - \mu(H)||^2$ by Lemma 2.16.



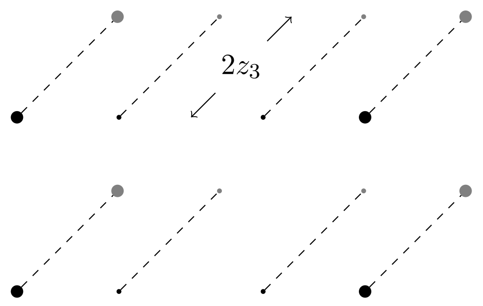


Figure 2.9: Point set P_d from the family of worst-case examples, drawn for $d = 2$ and $d = 3$. The heavy points are drawn larger.

Lower Bound with Infinite Weights

Let d be given. We construct an instance $P_d \subseteq \mathbb{R}^d$ with 2^{d+1} points. For $i \geq 2$ let $z_i^2 = \frac{3^{i-2}}{2^{i-1}}$ and define

$$P_d = \{(x_1, \ldots, x_d) \mid x_1 \in \{-1, -(\sqrt{2}-1), \sqrt{2}-1, 1\}, x_i \in \{-z_i, z_i\} \ \forall i \in \{2, \ldots, d\}\}.$$

All points from P_d whose first coordinate is -1 or 1 have weight ∞ (we call these *heavy points*). All other points have weight 1 (we call these *light points*). For an illustration of P_2 and P_3, see Figure 2.9.

We show the following theorem.

Theorem 2.19. *The family of point sets $(P_d)_{d\in\mathbb{N}}$ satisfies* $\mathrm{Ward}_k(P_d) \in \Omega((3/2)^d \cdot \mathrm{opt}_k(P_d))$ *for $k = 2^d$.*

In the theorem, we use $k = k(d) = 2^d$, i.e., we are interested in finding a 2^d-clustering of P_d. Observe that in the optimal 2^d-clustering of P_d, the heavy points are in separate clusters. Due to their infinite weight, they also determine the cluster centers. Hence, in the optimal solution each light point is in the same cluster as its closest heavy point. Since each light point is within distance $2-\sqrt{2}$ of a heavy point, the cost of the optimal solution is

$$\mathrm{opt}_k(P_d) = 2^d \cdot (2-\sqrt{2})^2.$$

Now we look at a run of Ward's method on P_d. We say that phase 1 lasts as long as there is at least one light point that forms its own cluster. We prove by induction that during phase 1 the only clusters that occur are singleton clusters consisting of one light or one heavy point and clusters that consist of two light points that differ only in the first coordinate. We call the latter *pair clusters.* At the beginning this is clearly the case. Now assume that the induction hypothesis holds at some point of time in phase 1. Merging two heavy points has infinite cost and merging a heavy point with a light point or a pair cluster has cost at least $(2-\sqrt{2})^2 \approx 0.343$ because $2-\sqrt{2}$ is the minimum distance between a light and a heavy point. Merging two singleton light points that differ only in the first coordinate costs $\frac{1}{2} \cdot (2\sqrt{2}-2)^2 = (2-\sqrt{2})^2$ (observe that the induction hypothesis guarantees that for any singleton light point the light point that differs only in the first coordinate is also a singleton point). Merging two singleton light points that differ in any other coordinate costs at least $\frac{1}{1+1} \cdot (2z_2)^2 = 1$, merging a singleton light point with a pair cluster costs at least $\frac{1\cdot 2}{1+2} \cdot (2z_2)^2 = \frac{4}{3}$, and merging two pair clusters costs at least $\frac{2\cdot 2}{2+2} \cdot (2z_2)^2 = 2$. Hence, we can assume that Ward merges two singleton light points that differ only in the first coordinate. After that the induction hypothesis is still true. Hence, in phase 1 all 2^{d-1} pairs of points of the form $(-(\sqrt{2}-1), x_2, \dots, x_d)$ and $(\sqrt{2}-1, x_2, \dots, x_d)$ will be merged. We call the clusters that consist of these points the $(*, x_2, \dots, x_d)$-clusters in the following.

Then phase 2 starts. Phase 2 lasts as long as there are pair clusters. We show by induction that the only clusters that occur in phase 2 are singleton heavy points, pair clusters, and clusters with four points that result from merging two pair clusters that differ only in the second coordinate. We call the latter *quadruple clusters*. Merging two pair clusters of the form $(*, -z_2, x_3, \dots, x_d)$ and $(*, z_2, x_3, \dots, x_d)$ to form a quadruple cluster costs $\frac{2\cdot 2}{2+2}(2z_2)^2 = 2$. Merging two pair clusters that differ in any other coordinate than the second is more expensive because their centers are further apart than $2z_2$. Merging the $(*, x_2, \dots, x_d)$-cluster with a heavy point costs at least 2 because the center of this cluster is $(0, x_2, \dots, x_d)$, which is at distance 1 from the heavy points. Similarly merging a quadruple cluster (whose center is $(0, 0, x_3, \dots, x_d)$) with a heavy point costs at least $2 + z_2^2 \geq 2$. Merging a quadruple cluster with a pair cluster costs at least $\frac{2\cdot 4}{2+4}(2z_3)^3 > 2$ and merging two quadruple clusters costs at least $\frac{4\cdot 4}{4+4}(2z_3)^3 > 2$. Hence, we can assume that Ward merges two pair clusters that differ only in the second coordinate. After that the induction hypothesis is still true. Hence, in phase 2 all 2^{d-2} pairs of clusters of the form $(*, -z_2, x_3, \dots, x_d)$ and $(*, z_2, x_3, \dots, x_d)$ will be merged. We call the clusters that consist of these points the $(*, *, x_3, \dots, x_d)$-clusters in the following.

At the beginning of phase $i \geq 2$, there are 2^d singleton heavy points and 2^{d-i+1} clusters of the form $(*, \dots, *, x_i, \dots, x_d)$ with 2^{i-1} points each. Phase i ends when there is no cluster of the form $(*, \dots, *, x_i, \dots, x_d)$ left. One can show again by induction that Ward merges in phase i all pairs of clusters of the form $(*, \dots, *, -z_i, x_{i+1}, \dots, x_d)$ and $(*, \dots, *, z_i, x_{i+1}, \dots, x_d)$. The center of the cluster $(*, \dots, *, x_i, \dots, x_d)$ is given by $(0, \dots, 0, x_i, \dots, x_d)$, which is at distance $\sqrt{1 + z_2^2 + \ldots + z_{i-1}^2}$ from the heavy points. Hence, merging such a cluster with a heavy point costs at least $2^{i-1} \cdot (1 + z_2^2 + \ldots + z_{i-1}^2) = 2^i z_i^2$, where the equation follows from the following observation.

Observation 2.20. *It holds that* $1 + z_2^2 + \ldots + z_{i-1}^2 = 2z_i^2$.

Proof. It holds that

$$\begin{aligned}
1 + z_2^2 + \ldots + z_{i-1}^2 &= 1 + \sum_{j=2}^{i-1} \frac{3^{j-2}}{2^{j-1}} = 1 + \frac{1}{3}\sum_{j=2}^{i-1} \frac{3^{j-1}}{2^{j-1}} = 1 + \frac{1}{3}\left(\sum_{j=0}^{i-2} \frac{3^j}{2^j} - 1\right) \\
&= \frac{2}{3} + \frac{1}{3}\sum_{j=0}^{i-2} \frac{3^j}{2^j} = \frac{2}{3} + \frac{1}{3} \cdot \frac{1-(3/2)^{i-1}}{1-(3/2)} \\
&= \frac{2}{3} + \frac{1}{3} \cdot \left(\frac{3^{i-1}}{2^{i-2}} - 2\right) = \frac{2}{3} + \frac{3^{i-2}}{2^{i-2}} - \frac{2}{3}, = 2z_i^2
\end{aligned}$$

which proves the observation. □

Merging the clusters $(*,\dots,-z_i,x_{i+1},\dots,x_d)$ and $(*,\dots,z_i,x_{i+1},\dots,x_d)$ costs

$$\frac{2^{i-1}\cdot 2^{i-1}}{2^{i-1}+2^{i-1}}\cdot(2z_i)^2 = 2^i z_i^2.$$

Merging two clusters that differ in one of the $d-i$ last coordinates costs at least

$$\frac{2^{i-1}\cdot 2^{i-1}}{2^{i-1}+2^{i-1}}(2z_{i+1})^2 = 2^i\cdot z_{i+1}^2 > 2^i z_i^2.$$

As a consequence, in phase i all 2^{d-i} pairs of clusters of the form $(*,\dots,*,-z_i,x_{i+1},\dots,x_d)$ and $(*,\dots,*,z_i,x_{i+1},\dots,x_d)$ will merge, which costs in total $2^{d-i}\cdot 2^i z_i^2$.

Phases 2 until d together cost $\sum_{i=2}^d 2^{d-i}\cdot 2^i z_i^2 = 2^d\cdot(2z_{d+1}^2-1) = 2\cdot 3^{d-1}-2^d$, where we used Observation 2.20. After phase d, all light points will be in the same cluster. Then the number of clusters is 2^d+1 and in the last step the cluster of light points, whose center is the origin, will be merged with one heavy point. This costs

$$2^d\cdot(1+z_2^2+\ldots+z_d^2) = 2^{d+1}\cdot z_{d+1}^2 = 2\cdot 3^{d-1}.$$

Phase 1 costs in total $2^{d-1}(2-\sqrt{2})^2$. Thus, the overall cost of Ward's solution is

$$\mathrm{Ward}_k(P_d) = 2^{d-1}(2-\sqrt{2})^2 + 2\cdot 3^{d-1} + 2\cdot 3^{d-1} - 2^d = 4\cdot 3^{d-1} + 2^{d-1}(2-\sqrt{2})^2 - 2^d.$$

This implies

$$\begin{aligned}\frac{\mathrm{Ward}_k(P_d)}{\mathrm{opt}_k(P_d)} &= \frac{4\cdot 3^{d-1}+2^{d-1}(2-\sqrt{2})^2-2^d}{2^d\cdot(2-\sqrt{2})^2}\\ &= \frac{4}{3(2-\sqrt{2})^2}\cdot\left(\frac{3}{2}\right)^d+\frac{1}{2}-\frac{1}{(2-\sqrt{2})^2}\in\Omega\left(\left(\frac{3}{2}\right)^d\right).\end{aligned}$$

Lower Bound with Finite Weights

In this section, we present a version of the lower bound from Section 2.4.3 in which the heavy points have a large finite weight m (to be determined later) instead of an infinite weight. In order to not change the behavior of Ward's method by this adaption of the weights, we have to move the heavy points slightly further to the outside. For given d, we construct the set

$$P'_d = \{(x_1,\dots,x_d) \mid x_1\in\{-(1+\epsilon),-(\sqrt{2}-1),\sqrt{2}-1,1+\epsilon\}, x_i\in\{-z_i,z_i\}\ \forall i\in\{2,\dots,d\}\}$$

where $\epsilon := \frac{2^d}{m}\frac{3^{d-2}}{2^{d-1}}$. All points from P'_d whose first coordinate is $-(1+\epsilon)$ or $1+\epsilon$ have weight m (we call these *heavy points*). All other points have weight 1 (we call these *light points*). We set $k=k(d)=2^d$, i.e., we are interested in finding a 2^d-clustering of P'_d.

We will now argue that the behavior of Ward's method on the input P'_d is exactly the same as on the input P_d with infinite weights. Observe that the costs for merging clusters that do not contain heavy points are the same in P_d and P'_d because the light points are at the same location in both these point sets. We use the same inductive argument as for P_d. For phase 1 we only need to verify that merging a light point with its closest heavy point (which is at distance $(2-\sqrt{2}+\epsilon)$) still costs at least $(2-\sqrt{2})^2$. This follows with Lemma 2.16 because

$$\begin{aligned}\frac{m}{m+1}\cdot(2-\sqrt{2}+\epsilon)^2 &> \frac{m}{m+1}\cdot(2-\sqrt{2})^2+\frac{m}{m+1}\cdot 2\cdot(2-\sqrt{2})\cdot\epsilon\\ &= \frac{m}{m+1}\cdot(2-\sqrt{2})^2+\frac{1}{m+1}\cdot 2\cdot(2-\sqrt{2})\cdot 2^d\cdot\frac{3^{d-2}}{2^{d-1}} > (2-\sqrt{2})^2.\end{aligned}$$

In phase $i \geq 2$ we start with 2^{d-i+1} clusters of the form $(*,\ldots,*,x_i,\ldots,x_d)$ and the heavy points in singleton clusters. We show that merging a heavy point with a cluster of the form $(*,\ldots,*,x_i,\ldots,x_d)$ still costs at least $2^i z_i^2$. The center of such a cluster is $(0,\ldots,0,x_i,\ldots,x_d)$. Observe that this is at distance $\sqrt{(1+\epsilon)^2+z_2^2+\ldots+z_{i-1}^2}$ from the closest heavy point. Hence, the inequality follows again with Lemma 2.16 and Observation 2.20 because

$$\begin{aligned}\frac{m\cdot 2^{i-1}}{m+2^{i-1}}\cdot((1+\epsilon)^2+z_2^2+\ldots+z_{i-1}^2) &= \frac{m\cdot 2^{i-1}}{m+2^{i-1}}(2z_i^2+2\epsilon+\epsilon^2)\\ &> \frac{m}{m+2^{i-1}}\cdot 2^i z_i^2+\frac{m}{m+2^{i-1}}\cdot 2^i\frac{2^d}{m}\cdot\frac{3^{d-2}}{2^{d-1}}\\ &\geq \frac{m}{m+2^{i-1}}\cdot 2^i z_i^2+\frac{2^{i-1}}{m+2^{i-1}}2^i z_i^2 = 2^i z_i^2.\end{aligned}$$

This proves that Ward's method behaves identically on P_d and P'_d. Next we calculate the cost of an optimal k-clustering and the cost of the k-clustering computed by Ward's method. In the optimal 2^d-clustering of P'_d, the heavy points are in separate clusters, and every light point is paired with its closest heavy point. Each of these clusters costs

$$\frac{m}{m+1}\cdot((1+\epsilon)-(\sqrt{2}-1))^2.$$

Thus the optimal solution has a cost of

$$\mathrm{opt}_k(P'_d) = 2^d\cdot\frac{m}{m+1}\cdot(2-\sqrt{2}+\epsilon)^2 < 2^d\cdot(2-\sqrt{2})^2+\frac{m}{m+1}\cdot 2^d\cdot(2\epsilon+\epsilon^2).$$

Now we come to the costs of the k-clustering computed by Ward's method. After phase d, all light points will be in the same cluster. Then the number of clusters is 2^d+1 and in the last step the cluster of light points will be merged with one heavy point. For $m \geq 2^d$ this costs

$$\begin{aligned}\frac{m\cdot 2^d}{m+2^d}\cdot((1+\epsilon)^2+z_2^2+\ldots+z_d^2) &= \frac{m\cdot 2^d}{m+2^d}\cdot(2z_{d+1}^2+2\epsilon+\epsilon^2)\\ &\geq \frac{2^d}{2}\cdot(2z_{d+1}^2+2\epsilon+\epsilon^2)\\ &= 2^{d-1}\cdot(2\cdot\frac{3^{d-1}}{2^d}+2\epsilon+\epsilon^2)\\ &= 3^{d-1}+2^{d-1}\cdot(2\epsilon+\epsilon^2).\end{aligned}$$

Since in the first d phases only light points are involved, the costs for these phases are the same on P'_d as on P_d. We have seen in Section 2.4.3 that phase 1 costs $2^{d-1}(2-\sqrt{2})^2$

and that phases 2 until d together cost $\sum_{i=2}^{d} 2^{d-i} \cdot 2^i z_i^2 = 2^d \cdot (2z_{d+1}^2 - 1) = 2 \cdot 3^{d-1} - 2^d$. Hence, the total costs of Ward's method can be written as follows:

$$\begin{aligned}\mathrm{Ward}_k(P'_d) &\geq 2^{d-1}(2-\sqrt{2})^2 + 2 \cdot 3^{d-1} - 2^d + 3^{d-1} + 2^{d-1}(2\epsilon + \epsilon^2) \\ &= 3 \cdot 3^{d-1} + 2^{d-1}(2-\sqrt{2})^2 - 2^d + 2^{d-1}(2\epsilon + \epsilon^2) \\ &= 3^d - 2^{d+1}(\sqrt{2}-1) + 2^{d-1}(2\epsilon + \epsilon^2).\end{aligned}$$

This implies

$$\frac{\mathrm{Ward}_k(P'_d)}{\mathrm{opt}_k(P'_d)} \geq \frac{3^d - 2^{d+1}(\sqrt{2}-1) + 2^{d-1}(2\epsilon+\epsilon^2)}{2^d \cdot (2-\sqrt{2})^2 + \frac{m}{m+1} \cdot 2^d \cdot (2\epsilon + \epsilon^2)} \geq \frac{3^d - 2^{d+1}(\sqrt{2}-1)}{2^d \cdot (2-\sqrt{2})^2 + \frac{m}{m+1} \cdot 2^d \cdot (2\epsilon + \epsilon^2)}.$$

Observe that

$$\frac{m}{m+1} \cdot 2^d \cdot (2\epsilon + \epsilon^2) \leq \frac{m}{m+1} \cdot 2^d \cdot 3\epsilon \leq \frac{1}{m+1} \cdot 2^d \cdot 3 \cdot 2 \cdot 3^{d-2}.$$

Thus, by assuming that $m+1 \geq 4 \cdot 6^{d-1}$, we can make sure that $\frac{m}{m+1} \cdot 2^d \cdot (2\epsilon + \epsilon^2) \leq 1$, and then we have

$$\frac{\mathrm{Ward}_d}{\mathrm{opt}_d} \geq \frac{3^d - 2^{d+1}(\sqrt{2}-1)}{2^d \cdot (2-\sqrt{2})^2 + 1} \in \Omega\left((3/2)^d\right).$$

2.4.4 Ward's Method in Dimension One

In this section, we discuss the approximation ratio of Ward's method for inputs $P \subset \mathbb{R}^1$ and show the following theorem.

Theorem 2.21. *Let $P \subset \mathbb{R}$ be an arbitrary one-dimensional instance. Then, for every k, the k-clustering computed by Ward on P is an $\mathcal{O}(1)$-approximation with respect to the k-means objective function.*

For the purpose of analyzing the worst-case behavior of Ward's method, an instance sometimes also contains an integer $k \in \mathbb{N}$ in addition to P (even though Ward itself only takes P as the input). If we specify P and k, then we are interested in the quality of the k-clustering produced by Ward on P.

We will usually denote the hierarchical clustering computed by Ward on P by $\mathcal{W} = (\mathcal{W}_0, \ldots, \mathcal{W}_{n-1})$. Ward's method always chooses greedily a cheapest merge to perform. We say that a merge is a *greedy merge* if it is a cheapest merge; if all merges are greedy, we call $\mathcal{W}$ greedy. Ward's method computes a greedy hierarchical clustering, and every greedy hierarchical clustering can be the output of Ward's method.

Prelude: Reordering

The following statements only hold for $d = 1$. First we observe that Ward satisfies the following convexity property.

Lemma 2.22 (Convexity in $\mathbb{R}^1$)**.** *For any three finite convex clusters $A, B, C \subset \mathbb{R}^1$ with $\mu(A) < \mu(C) < \mu(B)$, we have $D(A, C) < D(A, B)$ or $D(B, C) < D(A, B)$.*

Proof. We observe that $D(A, B) = \Delta(A \cup B) - \Delta(A) - \Delta(B) = |A| \cdot ||\mu(A) - \mu(A \cup B)||^2 + |B| \cdot ||\mu(B) - \mu(A \cup B)||^2 > |B| \cdot ||\mu(B) - \mu(A \cup B)||^2$. Now first assume that $\mu(A \cup B) \leq \mu(C)$. Then we conclude

$$D(A, B) > |B| \cdot ||\mu(B) - \mu(A \cup B)||^2 \geq |B| \cdot ||\mu(B) - \mu(C)||^2 \geq D(B, C),$$

where the second inequality follows since $\mu(A \cup B) \leq \mu(C)$. If $\mu(A \cup B) > \mu(C)$, then we get $D(A, C) < D(A, B)$ in the same manner. □

Lemma 2.22 means that Ward will always merge A and C or B and C, and never A and B. This gives us a convexity property: If Ward forms a cluster M, then no other point or cluster lies within the convex hull of M. Clusters can thus also never overlap, and we get a concept of neighbors on the line. Thus, the clusterings $\mathcal{W}_i$ consist of non-overlapping clusters, which we can thus view as ordered by their position on the line. Ward's method always merges neighbors on the line. We will combine it with the following useful corollary of Lemma 2.16. It gives a condition under which merging a cluster A with a subcluster $B' \subset B$ is cheaper than merging A with B. Notice that without the condition, the statement is not true: Imagine that A and B have the same centroid (merging them is free), but $\mu(B') \neq \mu(B)$. Then clearly, merging A with B' is more expensive than merging A and B.

Corollary 2.23. *Assume we have two finite clusters $B' \subsetneq B \subset \mathbb{R}^d$ and a third finite cluster $A \subset \mathbb{R}^d$ such that $||\mu(A) - \mu(B')||^2 \leq ||\mu(A) - \mu(B)||^2$. Then $D(A, B') \leq D(A, B)$.*

Proof. The statement follows from Lemma 2.16 since $|B'| < |B|$ and $||\mu(A) - \mu(B')||^2 \leq ||\mu(A) - \mu(B)||^2$:

$$\begin{aligned} D(A, B') = \frac{|A| \cdot |B'|}{|A| + |B'|} ||\mu(A) - \mu(B')||^2 \leq & \frac{|A| \cdot |B'|}{|A| + |B'|} ||\mu(A) - \mu(B)||^2 \\ \leq & \frac{|A| \cdot |B|}{|A| + |B|} ||\mu(A) - \mu(B)||^2 = D(A, B). \end{aligned}$$ □

Corollary 2.23 holds in arbitrary dimension. However, for $d = 1$, it is much easier to benefit from it. We get a very convenient tool that we call *reordering*. Say that Ward at some point merges two clusters A and B. By Lemma 2.22, that means that $\mu(A)$ and $\mu(B)$ are neighbors on the line (at the time of the merge). Now assume that A and B are present for a while before they are merged. Then during all this time, they are neighbors. Notice that this means that merging A and B will result in a centroid $\mu(A \cup B)$ which is further away from any other cluster than $\mu(A)$ and $\mu(B)$ are. So, clusters that did not want to merge with A or B would also not merge with $A \cup B$ by Corollary 2.23. Thus, we could perform the merge (A, B) *earlier* without distorting Ward's course of action at all (except that the merge (A, B) is at the wrong position). Lemma 2.24 below formulates this idea.

Recall that a hierarchical clustering can also be described by the $n - 1$ merge operations that produce it. We usually denote the sequence of merges by $(A, B)(\mathcal{W}) = ((A_1, B_1), \ldots, (A_{n-1}, B_{n-1}))$. We say that a cluster $Q \subset P$ *exists* in $\mathcal{W}$ after merge t if $Q \in \mathcal{W}_t$. If Q is the result of the merge (A_i, B_i) (i.e., $Q = A_i \cup B_i$), and it is later merged with another cluster in merge (A_j, B_j) (i.e., $A_j = Q$ or $B_j = Q$), then Q exists as long as merge i has happened and merge j has not yet happened. All singleton clusters exist in $\mathcal{W}_0$. After merge $n-1$, P is the only remaining existing cluster.

Lemma 2.24 (Reordering Lemma)**.** *Let $P \subset \mathcal{R}^d$ be an input for which Ward computes the clustering $\mathcal{W}$ with merge operations $(A,B)(\mathcal{W})$. Consider the merge (A_t, B_t) for $t \in [n-1]$. If both A_t and B_t exist after merge $s < t$, then*

1. *The sequence of merge operations $(A', B') = (A_1, B_1)$, …, (A_s, B_s),(A_t, B_t), $(A_{s+1}$,$B_{s+1})$,…, (A_{t-1}, B_{t-1}), (A_{t+1}, B_{t+1}), …, (A_{n-1}, B_{n-1}) results in a valid hierarchical clustering $\mathcal{W}'$.*
2. *$\mathcal{W}'_j = \mathcal{W}_j$ for all $j \geq t$.*
3. *All merges except the moved merge $(A'_{s+1}, B'_{s+1}) = (A_t, B_t)$ are greedy merges.*

Proof. (1) and (2) hold because performing merges in a different order does not change the resulting clustering, and after merge t, all deviations from the original order are done. For (3), we have to argue that inserting (A_t, B_t) as step $s+1$ does not create cheaper merges. For this, we observe that by Lemma 2.22, A_t and B_t are neighbors on the line. In the original sequence, no cluster was merged with A_t or B_t up to point t. The cluster $A_t \cup B_t$ is a superset of A_t and of B_t, and its centroid is further away from all other clusters than the centroids of A_t and B_t. Thus by Corollary 2.23, up to point t, merging with $A_t \cup B_t$ cannot be cheaper than the merges we do. However, after (A_{t-1}, B_{t-1}), the clustering is identical to $\mathcal{W}_t$ by (1), thus all remaining merges are also greedy merges. □

Lemma 2.24 a crucial observation to allow us to systematically analyze Ward's steps: We can sort them into steps that depend on each other, and then analyze them in batches / phases. Note that when we analyze the costs of a k-clustering computed by Ward we analyze the sum of the costs of the first $n-k$ steps proceeded by Ward. Using reordering we change the order of proceeded merge steps which may result in a different sequence of first $(n-k)$-merge steps. Using the monotonicity of Ward we ensure that reordering may only increase the sum of the costs of this sequence.

In $\mathbb{R}^d$ for $d > 1$, reordering does not work. Also, we cannot assume that there are no inner-cluster merges. This can easily be seen from the example in Figure 2.13: Here, Ward merges c and d, and then a and b, and a and b are input points from the same optimum cluster. However, moving the merge $(\{a\}, \{b\})$ to the front destroys the example; Ward will just compute the optimum solution then.

Prelude: No Inner-cluster Merges

Reordering also gives us a nice simplification tool. Assume that A and B are in fact singleton clusters, $A = \{a\}$ and $B = \{b\}$, and they are from the same optimum cluster. Then they are present from the start; we can reorder the merge (A, B) to be the first merge Ward does. Indeed, instead of actually doing this merge, we can also simply forget about it and replace a and b by a weighted point. How does this affect the approximation ratio? Both Ward's cost and the optimal cost decrease by $\Delta(\{a, b\})$, meaning that the approximation ratio can only get worse. We can now assume that there are no merges between inner clusters, since inner clusters arise from merging input points that belong to the same optimum cluster. We formalize our observation in Lemma 2.25.

We directly apply Lemma 2.24 in order to achieve a simplification method. Recall that (given an optimal k-clustering) we call a merge (A_i, B_i) an inner-cluster merge if A_i and B_i are inner clusters from the same optimum cluster. For a worst-case instance (P, k) we can always assume that such inner-cluster merges do not happen, as they are only helpful for Ward's method. We formally see this in the next lemma, where we relocate inner-cluster merges to the front of the hierarchical clustering and then eliminate them.

Recall that $\Delta_k(\mathcal{W}) = \sum_{Q \in \mathcal{W}_{n-k}} \Delta(Q)$ is the cost of the k-clustering contained in $\mathcal{W}$. For an instance (P, k) and Ward's resulting clustering $\mathcal{W}$, the approximation ratio of Ward's method is $\Delta_k(\mathcal{W}) / \operatorname{opt}_k(P)$.

Lemma 2.25. *Let (P, k) be an instance with $P \subset \mathbb{R}^d$ and $k \in \mathbb{N}$, for which $\mathcal{O} = \{O_1, \dots, O_k\}$ is an optimal k-clustering and for which Ward computes the hierarchical clustering $\mathcal{W}$ with merge operations $(A, B)(\mathcal{W})$. Then there exists a weighted point set P' and a hierarchical clustering $\mathcal{W}'$ for P' with merges $(A', B')(\mathcal{W}')$ with the following properties:*

1. *$\mathcal{W}'$ is greedy.*
2. *No (A'_i, B'_i) is an inner-cluster merge with respect to $\mathcal{O}$.*
3. *For some $\alpha \geq 0$, $\Delta_k(\mathcal{W}') = \Delta_k(\mathcal{W}) - \alpha$ and $\operatorname{opt}_k(P') \leq \operatorname{opt}_k(P) - \alpha$.*

Proof. Assume that P is weighted; this will be necessary to iterate the following process. Let $(\{x\}, \{y\})$ be a merge operation in $(A, B)(\mathcal{W})$ that merges two points $x, y \in O_j$ for $j \in [k]$, i.e., two points from the same cluster in the optimal solution. Let their weights be $w(x)$ and $w(y)$. By Lemma 2.24, we can move the merge $(\{x\}, \{y\})$ to the front. Then we replace x and y in P by one point $z = \frac{w(x)x + w(y)y}{w(x)+w(y)}$ with weight $w(z) := w(x) + w(y)$. By Lemma 2.16, z behaves identically to $\{x, y\}$ in Ward's method. Thus, we can adjust $\mathcal{W}'$ by removing the merge operation $(\{x\}, \{y\})$, and replacing x and y by z in all further merge operations of the cluster $\{x, y\}$. We see that (1) holds for the new hierarchical clustering. Our adjustment will change the cost by $\alpha := \Delta(\{x, y\})$. Similarly, we can replace x and y in O_j by z, which decreases the cost of the clustering induced by $O_1, \dots, O_k$ by α. Since this is still a possible clustering, the optimal clustering can cost at most $\operatorname{opt}_k(P) - \alpha$. Thus, (3) holds for the new clustering.

Observe that if (2) is not true, then there has to be a merge operation where two points from the same cluster in the optimum are merged. Thus, we can complete the proof by repeating the above process until we have removed all pairs with this property. Then (2) holds. □

Now if Ward performs inner-cluster merges on an instance, we apply Lemma 2.25. If this changes the optimum solution, we just apply Lemma 2.25 again, and repeat this until Ward does not do any inner-cluster merges. We explicitly note the following trivial corollary.

Corollary 2.26. *Assume that $\mathcal{W}'$ and $(A', B')(\mathcal{W}')$ result from applying Lemma 2.25 until Ward does not do inner cluster merges. If a merge (A'_i, B'_i) for $i \in [n-1]$ contains an inner cluster, then this inner cluster is a (weighted) input point.*

Proof. If A resulted from a previous merge, then this merge was an inner-cluster merge, which is a contradiction. $\square$

Corollary 2.26 implies that we can use the terms inner cluster and input point interchangeably.

Prelude: Clustering points together

Crucial in showing the approximation factors of the good merges is the following lemma. To see its usage, assume that A and B belong to one optimum cluster, and C and D belong to another. Then the lemma implies that if Ward has already merged B and C, but $\Delta(B \cup C)$ is small, say $\Delta(B \cup C) \leq c \cdot (\Delta(B) + \Delta(C))$, then we can still obtain a $7c$-approximation.

Lemma 2.27. *Let $A, B, C, D \subset \mathbb{R}^d$ be disjoint sets with $|A| \leq |B|$ and $|C| \geq |D|$. Then*

$$\Delta(A \cup B \cup C \cup D) \leq \Delta(A) + 3 \cdot \Delta(B \cup C) + \Delta(D) + 4 \cdot D(A, B) + 4 \cdot D(C, D) \text{ and}$$

$$D(A \cup B, C \cup D) \leq 3 \cdot \Delta(B \cup C) + 3 \cdot D(A, B) + 3 \cdot D(C, D) - \Delta(B) - \Delta(C).$$

Proof. We find an upper bound on $\Delta(A \cup B \cup C \cup D)$ by computing the cost of clustering all four clusters with the center of $B \cup C$. Then we decompose the cost and use Lemma 2.3:

$$\begin{aligned}
&\Delta(A \cup B \cup C \cup D) \\
\leq &\Delta(A \cup B \cup C \cup D, \mu(B \cup C)) \\
= &\Delta(A, \mu(B \cup C)) + \Delta(B \cup C, \mu(B \cup C)) + \Delta(D, \mu(B \cup C)) \\
= &\Delta(A) + |A| \cdot ||\mu(A) - \mu(B \cup C)||^2 + \Delta(B \cup C) + \Delta(D) + |D| \cdot ||\mu(D) - \mu(B \cup C)||^2.
\end{aligned}$$

Next, we apply the relaxed triangle inequality in Lemma 2.2 and use that $|A| \leq |B|$ to get

$$\begin{aligned}
|A| \cdot ||\mu(A) - \mu(B \cup C)||^2 \leq &2|A| \cdot (||\mu(A) - \mu(B)||^2 + ||\mu(B) - \mu(B \cup C)||^2) \\
\leq &2|A| \cdot ||\mu(A) - \mu(B)||^2 + 2|B| \cdot ||\mu(B) - \mu(B \cup C)||^2.
\end{aligned}$$

Similarly, we get that

$$|D| \cdot ||\mu(D) - \mu(B \cup C)||^2 \leq 2|D| \cdot ||\mu(D) - \mu(C)||^2 + 2|C| \cdot ||\mu(C) - \mu(B \cup C)||^2.$$

Using Lemma 2.16 and the fact that $|A| \leq |B|$, we observe that

$$\begin{aligned}
D(A, B) &= \frac{|A| \cdot |B|}{|A| + |B|} \cdot ||\mu(A) - \mu(B)||^2 \\
\Leftrightarrow |A| \cdot ||\mu(A) - \mu(B)||^2 &= \frac{|A| + |B|}{|B|} \cdot D(A, B) \leq 2D(A, B)
\end{aligned}$$

holds, and, similarly, $|D| \cdot ||\mu(D) - \mu(C)||^2 \leq 2D(C, D)$ since $|D| \leq |C|$. Thus, $2|A| \cdot ||\mu(A) - \mu(B)||^2 + 2|D| \cdot ||\mu(D) - \mu(C)||^2 \leq 4D(A, B) + 4D(C, D)$. Furthermore, $\Delta(B \cup C) \geq D(B, C) = |B| \cdot ||\mu(B) - \mu(B \cup C)||^2 + |C| \cdot ||\mu(C) - \mu(B \cup C)||^2$. Together, we get that

$$\begin{aligned}
&|A| \cdot ||\mu(A) - \mu(B \cup C)||^2 + |D| \cdot ||\mu(D) - \mu(B \cup C)||^2 \\
\leq &\ 4 \cdot D(A, B) + 4 \cdot D(C, D) + 2\Delta(B \cup C),
\end{aligned}$$

which implies that

$$\Delta(A \cup B \cup C \cup D) \leq \Delta(A) + \Delta(B \cup C) + \Delta(D) + 4 \cdot D(A, B) + 4 \cdot D(C, D) + 2\Delta(B \cup C)$$

and

$$\begin{aligned}
& D(A \cup B, C \cup D) \\
= \; & \Delta(A \cup B \cup C \cup D) - \Delta(A \cup B) - \Delta(C \cup D) \\
= \; & \Delta(A \cup B \cup C \cup D) - \Delta(A) - \Delta(B) - D(A, B) - \Delta(C) - \Delta(D) - D(C, D) \\
\leq \; & \Delta(A) + 3\Delta(B \cup C) + \Delta(D) + 3 \cdot D(A, B) + 3 \cdot D(C, D) \\
& - \Delta(A) - \Delta(B) - \Delta(C) - \Delta(D) \\
= \; & 3 \cdot \Delta(B \cup C) + 3 \cdot D(A, B) + 3 \cdot D(C, D) - \Delta(B) - \Delta(C) \qquad \square
\end{aligned}$$

The analysis

We now analyze the worst-case behavior of Ward's method. For this, we fix an arbitrary worst-case example that does not contain inner-cluster merges (we can assume the latter by Lemma 2.25).

The general plan is the following. Whenever Ward merges two clusters, it does so greedily, meaning that the cost of the merge is always bounded by the cost of any other merge. Thus, if we can find a merge with low cost, then the merge actually performed can only be cheaper. We can clearly find cheap merges in the beginning, however, Ward's decisions may lead us to a situation where we run out of the originally good options. The idea of the proof is to find a point during Ward's execution where

- We still know a bound on the costs produced so far.
- We know a set $\mathcal{S}$ of good merges that can still be performed and lead to a good k-clustering.
- We can ensure that no merge can possibly destroy two merges from $\mathcal{S}$.

At such a point in time, we can use $\mathcal{S}$ to charge the remaining merges that Ward does to compute a k-clustering. We find this point in time by sorting specific merges of Ward into the front, and bounding their cost. There will be five phases of merges which we need to pull forward and charge.

The phases We will use the reordering lemma (Lemma 2.24) to sort the merges into phases and then analyze the cost of the solution after each phase.

In the following, we call a cluster that contains points from more than one optimum cluster *composed*, more precisely, we call it an *ℓ-composed cluster* if it contains points from ℓ different optimum clusters. Most of the time, we are interested in 2-composed clusters, and we name such a cluster *2-composed cluster from O_j and O_{j+1}* if these are the involved optimum clusters.

The goal of the reordering is simple in nature; we want to collect all merges that create 2-composed clusters and that grow 2-composed clusters. We can think of the phases as different stages of development of 2-composed clusters. A 2-composed cluster may become

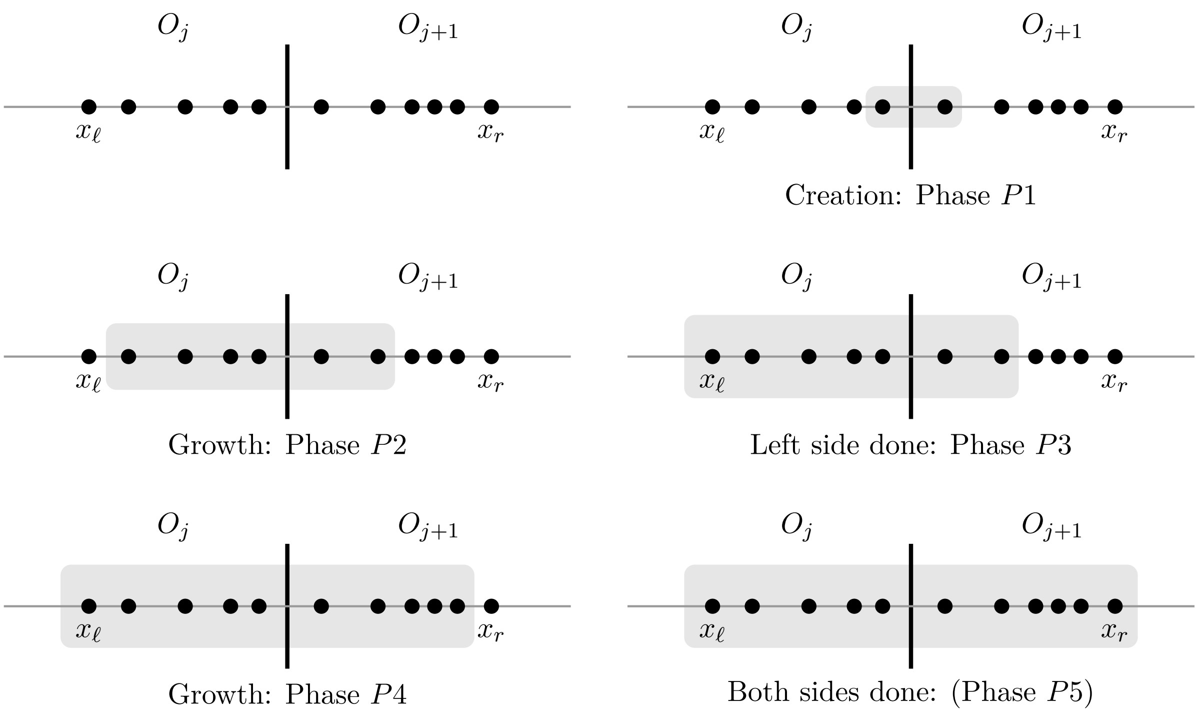


Figure 2.10: The pricipal phases of development of a 2-composed cluster.

part of the k-clustering computed by Ward's method, or it may at some point become i-composed for $i > 2$, at which time we are no longer interested in it. By the *final stage* of a 2-composed cluster we either mean how it looks in the k-clustering, or how it looked in the last step before it became more than 2-composed.

Consider the example in Figure 2.10, where we depict the development of a 2-composed cluster from O_j and O_{j+1} which in its final stage consists of the input points $x_\ell, \ldots, x_r$. It undergoes five principal phases: It is created by merging a point from O_j with a point from O_{j+1} (phase $P1$). Then it grows; it is merged with points left and right of itself (phase $P2$). We add extra phases for the last points on both sides. In phase $P3$, the first side is completed; in the example, it is the left side. This merge is again followed by a growth phase (phase $P4$). The final phase $P5$ consists of the final merge on the other side; the right side in the example. (We skip some merges in $P5$, the details of $P5$ are not discussed until much later in this proof).

So, we use reordering to pull the following phases of merges to the front.

P1 (Creation phase)
We create 2-composed clusters by collecting the merges $(\{a_i\}, \{b_i\})$ with $a_i \in O_j$, $b_i \in O_{j+1}$ for some $j \in [k]$. The collected merges constitute phase $P1$. For technical reasons, we make one exception. If the 2-composed cluster only consists of two input points in its final stage (i.e., the creating merge is also the last merge), then we defer the merge to phase $P5$.

P2 (Main growth phase)

We now grow the 2-composed clusters initialized during phase $P1$. For each 2-composed cluster, we move the growth merges to phase $P2$, preserving their original order. We stop right before one side of the 2-composed cluster is done. There may be many growth merges for a cluster, or none.

P3 (First side elimination phase)
This phase consists of at most one merge for each 2-composed cluster, and this merge is the last merge on the first side. After phase $P3$, every 2-composed cluster thus has one side where it will not be merged with further input points. Notice that a cluster may skip phase $P3$ if it only shares one point with O_j or O_{j+1} in its final stage, anyway.

P4 (Second growth phase)
This phase resembles phase $P2$, however, the growth is now one-sided. For each 2-composed cluster, we move the growth merges to phase $P4$, preserving their original order, and stopping right before the final merge.

P5 (Second side elimination phase)
The last phase consists of at most one merge for each cluster. If the final stage of a 2-composed cluster contains only two points, then the merging of these two points is done in phase $P5$. Otherwise, phase $P5$ may contain the last merge for the cluster, resulting in its final state. For technical reasons, we have to exclude some merges; we postpone the details to Definition 2.31.

We now analyze the sum of the 1-means costs of all clusters in the clustering after each phase. We start with phases $P1$ and $P2$.

Lemma 2.28. *Let $N = \{x_a, \ldots, x_b\}$ with $x_a, \ldots, x_m \in O_j$ and $x_{m+1}, \ldots, x_b \in O_{j+1}$ be a 2-composed cluster after phases $P1$ and $P2$. Then*

$$\Delta(N) \leq \sum_{h=a-1}^{m-1} D(x_h, x_{h+1}) + \sum_{h=m+1}^{b} D(x_h, x_{h+1}).$$

Furthermore, $D(N \cap O_j, N \cap O_{j+1}) \leq D(x_{a-1}, x_a) + D(x_b, x_{b+1})$.

Proof. We show the statement by induction on the number of points in N. The base case is the merge done in phase $P1$. By the way we defined phase $P1$, we know that when x_m and x_{m+1} were merged in the original order, either x_{m-1} or x_{m+2} were also present. Thus,

$$\Delta(x_m, x_{m+1}) \leq \max\{D(x_{m-1}, x_m), D(x_{m+1}, x_{m+2})\} \leq D(x_{m-1}, x_m) + D(x_{m+1}, x_{m+2}).$$

Now say that at some point, the cluster is $N' = \{x_c, \ldots, x_d\}$, and it is expanded by one point. Without loss of generality, say it is expanded by x_{c-1} by the merge $\{N', x_{c-1}\}$ (the other case follows symmetrically). By the induction hypothesis, $\Delta(N') \leq \sum_{h=c-1}^{m-1} D(x_h, x_{h+1}) + \sum_{h=m+1}^{d} D(x_h, x_{h+1})$. By the definition of phase $P2$, x_{c-2} is also present during the merge, meaning that $D(N', x_{c-1}) \leq D(x_{c-1}, x_{c-2})$, and that $\Delta(N' \cup \{x_{c-1}\}) \leq \sum_{h=c-2}^{m-1} D(x_h, x_{h+1}) + \sum_{h=m+1}^{d} D(x_h, x_{h+1})$, which proves the induction step.

The second statement follows since $\Delta(N \cap O_j) = \Delta(\{x_a, \ldots, x_m\}) \geq \sum_{h=a}^{m-1} D(x_h, x_{h+1})$, $\Delta(N \cap O_{j+1}) = \Delta(\{x_{m+1}, \ldots, x_b\}) \geq \sum_{h=m+1}^{b-1} D(x_h, x_{h+1})$ and $D(N \cap O_j, N \cap O_{j+1}) = \Delta(N) - \Delta(N \cap O_j) - \Delta(N \cap O_{j+1})$. □

In phase $P3$, Ward's method faces the first situation where it may run out of good merge options and has to resort to more expensive merges. Notice that by the definition of our phases, each cluster has one side where after phase $P2$, there is exactly one point left which has not been added to the cluster.

In the following, we will again and again use the following statement which follows directly from Lemma 2.27.

Corollary 2.29. *Let A, B, and C be three disjoint sets of points with $|A| \leq |B|$ (or $w(A) \leq w(B)$, for weighted sets). Then $\Delta(A \cup B \cup C) \leq \Delta(A) + 3 \cdot \Delta(B \cup C) + 4 \cdot D(A, B)$ and $D(A \cup B, C) \leq 3 \cdot \Delta(B \cup C) + 3 \cdot D(A, B) - \Delta(B) - \Delta(C)$.*

Proof. □

We need the following interpretation of Corollary 2.29. If we have a 2-composed cluster $M = A \cup B$ which consists of a lighter cluster $A \subseteq O'$ for an optimum cluster O' and a heavier cluster $B \subset O''$ for another optimum cluster O'', then merging $A \cup B$ with another cluster $C \subset O''$ basically costs as much as $A \subseteq O'$ and $B \cup C \subseteq O''$ cost individually, plus what merging A and B costed us already (up to constant factors). We now analyze the 1-means costs of the clusters after phase $P4$

Lemma 2.30. *Let $F = \{x_\ell, \ldots, x_r\}$ be the final state of a 2-composed cluster, with $x_\ell, \ldots, x_m \in O_j$ and $x_{m+1}, \ldots, x_r \in O_{j+1}$. The state of the cluster after phase $P4$ is either $N = \{x_\ell, \ldots, x_{r-1}\}$ or $N = \{x_{\ell-1}, \ldots, x_r\}$. In both cases,*

$$\Delta(N) \leq 8 \cdot (\Delta(\{x_\ell, \ldots, x_m\}) + \Delta(\{x_{m+1}, \ldots, x_r\})).$$

Proof. During phase $P2$, N grew to the penultimate point on one side, and is now merged with the last point on this side in phase $P3$. Without loss of generality, we assume that this is the left side. This means that $N = \{x_\ell, \ldots, x_{r-1}\}$ will be the state after $P4$. The state after $P2$ is $N' = \{x_{\ell+1}, \ldots, x_d\}$ for $d \in \{m+1, \ldots, r-1\}$. Phase $P3$ does the merge $\{N', \{x_\ell\}\}$.

There are two cases for how to charge this merge. First, assume that $w(N' \cap O_j) \geq w(N' \cap O_{j+1})$, i.e., x_ℓ lies on the heavier side of N. This allows us to use Corollary 2.29 with $A = N' \cap O_{j+1}$, $B = N' \cap O_j$ and $C = \{x_\ell\}$ to obtain

$$\begin{aligned}
\Delta(N' \cup \{x_\ell\}) &\leq \Delta(N' \cap O_{j+1}) + 3 \cdot \Delta((N' \cap O_j) \cup \{x_\ell\}) + 4 \cdot D(N' \cap O_j, N' \cap O_{j+1}) \\
&\leq \Delta(\{x_{m+1}, \ldots, x_d\}) + 3 \cdot \Delta(\{x_\ell, \ldots, x_m\}) + 4 \cdot (D(x_\ell, x_{\ell+1}) + D(x_d, x_{d+1})) \\
&\leq 4 \cdot (\Delta(\{x_\ell, \ldots, x_m\}) + \Delta(\{x_{m+1}, \ldots, x_{d+1}\}) + D(x_\ell, x_{\ell+1}) + D(x_d, x_{d+1})),
\end{aligned}$$

where the second inequality follows from Lemma 2.28.

Now assume $w(N' \cap O_j) \leq w(N' \cap O_{j+1})$. We can still apply Corollary 2.29, but with reversed roles. We know that x_{d+1} is still present by the definition of phase $P3$. Thus, we charge the merge $\{N', \{x_\ell\}\}$ to the merge $\{N', \{x_{d+1}\}\}$. By Corollary 2.29 with $A = N' \cap O_j$, $B = N' \cap O_{j+1}$ and $C = \{x_{d+1}\}$, we get

$$\Delta(N' \cup \{x_{d+1}\}) \leq \Delta(N' \cap O_j) + 3 \cdot \Delta((N' \cap O_{j+1}) \cup \{x_{d+1}\}) + 4 \cdot D(N' \cap O_j, N' \cap O_{j+1}).$$

Thus, $D(N', x_{d+1}) = \Delta(N' \cup \{x_{d+1}\}) - \Delta(N')$, which implies that

$$\begin{aligned}
&\Delta(N' \cup \{x_\ell\}) \\
=&\Delta(N') + D(N', \{x_\ell\}) \\
\leq&\Delta(N') + D(N', \{x_{d+1}\}) \\
\leq&\Delta(N' \cap O_j) + 3 \cdot \Delta((N' \cap O_{j+1}) \cup \{x_{d+1}\}) + 4 \cdot D(N' \cap O_j, N' \cap O_{j+1}) \\
\leq&\Delta(\{x_{\ell+1}, \dots, x_m\}) + 4 \cdot \Delta(\{x_{m+1}, \dots, x_{d+1}\}) + 4 \cdot (D(x_\ell, x_{\ell+1}) + D(x_d, x_{d+1})) \\
\leq&4 \cdot (\Delta(\{x_\ell, \dots, x_m\}) + \Delta(\{x_{m+1}, \dots, x_{d+1}\}) + D(x_\ell, x_{\ell+1}) + D(x_d, x_{d+1})).
\end{aligned}$$

We see that we bounded the cost by the same expression in both cases. After phase $P3$, there are possibly additional merges in phase $P4$, which extend $N' \cup \{x_\ell\}$ to the right. More precisely, $P4$ extends the cluster from $\{x_\ell, \dots, x_d\}$ to $\{x_\ell, \dots, x_{r-1}\}$. Similarly to Lemma 2.28, we can show that this extension increases the cost of $N' \cup \{x_\ell\}$ by

$$\sum_{h=d+1}^{r-1} D(x_h, x_{h+1}).$$

Finally, notice that

$$\begin{aligned}
&4 \cdot (\Delta(\{x_\ell, \dots, x_m\}) + \Delta(\{x_{m+1}, \dots, x_{d+1}\}) + D(x_\ell, x_{\ell+1}) + D(x_d, x_{d+1})) + \sum_{h=d+1}^{r-1} D(x_h, x_{h+1}) \\
\leq\ & 4 \cdot (\Delta(\{x_\ell, \dots, x_m\}) + \Delta(\{x_{m+1}, \dots, x_{d+1}\}) + D(x_\ell, x_{\ell+1})) + 4 \cdot \sum_{h=d}^{r-1} D(x_h, x_{h+1}) \\
\leq\ & 8 \cdot (\Delta(\{x_\ell, \dots, x_m\}) + \Delta(\{x_{m+1}, \dots, x_r\})),
\end{aligned}$$

which concludes the proof. □

Now we come to phase $P5$, which we haven't completely defined yet. The problem with phase $P5$ is that we can no longer charge all clusters 'internally'. To see what the issue is, first notice that we say that a 2-composed cluster F from O_j and O_{j+1} *points to cluster A* if

- $w(F \cap O_j) \geq w(F \cap O_{j+1})$ and A is the cluster left of F, or
- $w(F \cap O_j) \leq w(F \cap O_{j+1})$ and A is the cluster right of F.

We define a *lopsided cluster* to be a 2-composed cluster $F = \{x_\ell, \dots, x_r\}$ for which the last merge is $\{F \backslash \{x\}, \{x\}\}$, but at the time of this merge, $F' = F \backslash \{x\}$ does not point to $\{x\}$. This means that we cannot use Corollary 2.29 (directly) to charge this merge. As a technicality, we also call a 2-composed cluster lopsided if it only contains two points in its final state; again, we cannot use Corollary 2.29 in this case.

We have to pay attention to one more detail when defining phase $P5$. When charging 2-composed clusters internally, we could always be sure that the clusters that are involved are part of one of the two optimum clusters that the 2-composed cluster intersects. That is because the 2-composed cluster by definition only contains points from two optimum clusters, and we only dealt with points and subclusters of such a 2-composed cluster. However, in the following arguments, we will have to argue about clusters neighboring a 2-composed cluster. These may or may not belong to one of the optimum clusters. Let A and B be two clusters that are neighbors on the line such that A lies left of B. We say that there is an *opt change between A and B* if the last point in A and the first point in B belong to different optimum clusters.

Now we define phase $P5$. Let Y be the cluster that lies on the other side of F' than x *at the time of the merge* $\{F', \{x\}\}$. Let Z be the cluster that lies 'behind' x from the point of view of F' *at the time of the merge* $\{F', \{x\}\}$. By *behind from F's point of view* we mean that if x lies left of F, then Z lies left of x, and if x lies right of F', then Z lies right of x.

Definition 2.31 (Phase P5)**.** *Phase $P5$ contains the final merge $\{F', \{x\}\}$ of a cluster $F = F' \cup \{x\}$ if any of the following conditions applies.*

1. *F is not lopsided (phase $P5a$),*
2. *F is lopsided, there is no opt change between Y and F', and Y is an inner cluster (phase $P5b$),*
3. *F is lopsided, there is no opt change between $\{x\}$ and Z, and Z is an inner cluster (phase $P5c$),*
4. *F is lopsided, there is no opt change between $\{x\}$ and Z, Z is 2-composed, and points to $\{x\}$ (phase $P5d$).*

The next lemma deals with merges in $P5a$.

Lemma 2.32. *Let $F = \{x_\ell, \dots, x_r\}$ be the final state of a 2-composed cluster, with $x_\ell, \dots, x_m \in O_j$ and $x_{m+1}, \dots, x_r \in O_{j+1}$. Assume that F is not lopsided. Then*

$$\Delta(F) \leq 35 \cdot (\Delta(\{x_\ell, \dots, x_m\}) + \Delta(\{x_{m+1}, \dots, x_r\})).$$

Proof. Let $\{F \backslash \{x\}, \{x\}\}$ be the final merge in $P5$. Without loss of generality, assume that $x = x_r$, i.e., the final merge happens at the right end of F. Set $F' = F \backslash \{x\}$, so the final merge is $\{F', \{x_r\}\}$. By Lemma 2.30, we know that

$$D(F' \cap O_j, F' \cap O_{j+1}) \leq \Delta(F') \leq 8 \cdot (\Delta(\{x_\ell, \dots, x_m\}) + \Delta(\{x_{m+1}, \dots, x_r\}))$$

is true after $P4$. By our assumption that the final merge is not lopsided, we know that $F \backslash \{x\}$ points to $\{x\}$. So, we can apply Corollary 2.29 with $A = F' \cap O_j = F \cap O_j$, $B = F' \cap O_{j+1} = (F \cap O_{j+1}) \backslash \{x_r\}$ and $C = \{x_r\}$. We get:

$$\begin{aligned}
&\Delta(F' \cup \{x\}) \\
\leq &\Delta(F \cap O_j) + 3 \cdot \Delta(((F \cap O_{j+1}) \backslash \{x_r\}) \cup \{x_r\}) + 4 \cdot D(F' \cap O_j, F' \cap O_{j+1}) \\
\leq &3 \cdot (\Delta(\{x_\ell, \dots, x_m\}) + \Delta(\{x_{m+1}, \dots, x_r\})) + 4 \cdot (8 \cdot (\Delta(\{x_\ell, \dots, x_m\}) + \Delta(\{x_{m+1}, \dots, x_r\}))) \\
\leq &35 \cdot (\Delta(\{x_\ell, \dots, x_m\}) + \Delta(\{x_{m+1}, \dots, x_r\})). \qquad \square
\end{aligned}$$

Now we consider the merges in phase $P5b$.

Lemma 2.33. *Let $F = \{x_\ell, \ldots, x_r\}$ be the final state of a 2-composed cluster, with $x_\ell, \ldots, x_m \in O_j$ and $x_{m+1}, \ldots, x_r \in O_{j+1}$. Assume that F is lopsided. Assume that at the time of the merge $\{F \backslash \{x\}, \{x\}\}$, the cluster on the other side of $F' = F \backslash \{x\}$ is an inner cluster Y, and there is no opt change between F' and Y. Then if $x = x_\ell$, we have*

$$\Delta(F) \leq 35 \cdot (\Delta(\{x_\ell, \ldots, x_m\}) + \Delta(\{x_{m+1}, \ldots, x_{r+1}\})),$$

and if $x = x_r$, then

$$\Delta(F) \leq 35 \cdot (\Delta(\{x_{\ell-1}, \ldots, x_m\}) + \Delta(\{x_{m+1}, \ldots, x_r\})).$$

Proof. W.l.o.g. assume that $F = \{x_\ell, \ldots, x_r\}$, that $x = x_\ell$ lies left of F' and that the cluster Y is thus $\{x_{r+1}\}$. Then we can still use Corollary 2.29, with $A = F' \cap O_j$, $B = F \cap O_{j+1}$ and $C = \{x_{r+1}\}$ (notice that $w(F' \cap O_j) \leq w(F \cap O_{j+1})$ because F is lopsided), and obtain that $\Delta(F) \leq \Delta(F' \cup \{x_{r+1}\}) \leq \Delta(F' \cap O_j) + 3 \cdot \Delta((F \cap O_{j+1}) \cup \{x_{r+1}\}) + 4 \cdot D(F' \cap O_j, F \cap O_{j+1})$. Since F' results from $P4$, we know from Lemma 2.30 that $D(F' \cap O_j, F \cap O_{j+1}) \leq \Delta(F') \leq 8 \cdot (\Delta(\{x_\ell, \ldots, x_m\}) + \Delta(\{x_{m+1}, \ldots, x_r\}))$. That means that

$$\begin{aligned}
\Delta(F) \leq& \Delta(F' \cap O_j) + 3 \cdot \Delta((F \cap O_{j+1}) \cup \{x_{r+1}\}) + 4 \cdot D(F' \cap O_j, F \cap O_{j+1}) \\
\leq& \Delta(\{x_\ell, \ldots, x_m\}) + 3\Delta(\{x_{m+1}, \ldots, x_{r+1}\}) + 32(\Delta(\{x_\ell, \ldots, x_m\}) + \Delta(\{x_{m+1}, \ldots, x_r\})) \\
\leq& 35 \cdot (\Delta(\{x_\ell, \ldots, x_m\}) + \Delta(\{x_{m+1}, \ldots, x_{r+1}).
\end{aligned}$$

Notice that the upper bound involves x_{r+1}, which is not part of the final state F of the 2-composed cluster. In the symmetric case that $x = x_r$, the upper bound involves $x_{\ell-1}$ instead of x_{r+1}. □

We are ready to show the main lemma about the phases.

Lemma 2.34. *Let C_5 be the clustering after phase $P5$. Then*

$$\sum_{A \in C_5} \Delta(A) \leq \mathcal{O}(1) \cdot \mathrm{opt}_k .$$

Proof. Let C_5' be the clustering that arises from performing $P1$-$P4$ and finishing phases $P5a$ and $P5b$. By Lemma 2.32, we know that for any non-lopsided cluster in C_5', its 1-means cost is at most 35 times as much as the cost of its points in the optimal solution. By Lemma 2.33, we know that for any lopsided cluster in C_5' resulting from phase $P5b$, nearly the same holds, except that the upper bound involves one more point. However, by the definition of phase $P5b$, this point is in one of the two optimum clusters that the lopsided cluster intersects. All other 2-composed clusters in C_5' cost at most 8 times their optimum cost by Lemma 2.30. The inner clusters cost nothing (they are input points).

Now we consider phases $P5c$ and $P5d$. Let $F = \{x_\ell, \ldots, x_r\} = F' \cup \{x_r\}$ with $x_\ell, \ldots, x_m \in O_j$ and $x_{m+1}, \ldots, x_r \in O_{j+1}$ be lopsided with final merge $\{F', \{x_r\}\}$. We know that at the time of the final merge, the cluster M right of x_r is either an inner cluster or it is a 2-composed cluster that points to x_r. In the first case (phase $P5c$), we observe that the merge $\{\{x_r\}, \{x_{r+1}\}\}$ is available, which costs at most $D(x_r, x_{r+1})$. Thus, in this case $D(F', x_r) \leq D(x_r, x_{r+1})$. At this point, it is important that we assumed that there is no opt change between $\{x_r\}$ and its neighboring cluster, here $\{x_{r+1}\}$: Thus, x_r and x_{r+1} are in the same optimum cluster, and $D(x_r, x_{r+1})$ is part of the optimum cost. Notice that $D(x_r, x_{r+1})$ has not been charged before.

Now assume that M is 2-composed and points to x_r. Note that M is in its final state, since any merge of M with a possible additional inner cluster right of M would happen in Phase $P5b$. This is a tricky technical detail: We need that M, the cluster next to $\{x_r\}$ *at the time of the merge* $\{F', \{x\}\}$, is finished after phase $P5b$, such that we know that its cost is bounded and can use this cost bound in the following charging argument. Indeed, this is the sole purpose of phase $P5b$.

We now charge the merge $\{F', \{x_r\}\}$ to the possible merge $\{M, \{x_r\}\}$. By Corollary 2.29 with $A = M \cap O_{j+2}$, $B = M \cap O_{j+1}$ and $C = \{x_r\}$,

$$\begin{aligned} D(F', x_r) &\leq D(M, x_r) \leq \Delta(M \cup \{x_r\}) \\ &\leq \Delta(M \cap O_{j+2}) + 3\Delta((M \cap O_{j+1}) \cup \{x_r\}) + 4 \cdot D(M \cap O_{j+1}, M \cap O_{j+2}). \end{aligned}$$

Notice that even though we charge a merge with F' to a merge with M, $\Delta(F) + \Delta(M)$ increases by a factor of at most 7 compared to $\Delta(F') + \Delta(M)$. Since no M can be part of two such charge operations, we know that the overall sum of all 1-means costs of all clusters increases by a factor of at most 7 when we process all the remaining lopsided clusters. Thus, the final cost after phase $P5$ is at most $245 \cdot \mathrm{opt}_k$. $\square$

Good merges for the final analysis In general, the clustering of Ward after phase $P5$ has still more than k clusters. It remains to analyze the merges after phase $P5$ that reduce the number of clusters to k. For the final charging argument, we need four types of *good merges*. Good merges are not necessarily merges that Ward's method does, instead, it's a collection of merges that are possible and can be used for charging. Indeed, good merges include merges that would not be present anymore if Ward did them, since then we would move them to the phases. But if Ward never uses them, they may still be present for us to charge against.

The whole point of the phases is to ensure that any merge that Ward may still do does not destroy two good merges. The final arguments of the proof will be to count good merges and to show that no two good merges can be invalidated simultaneously by one of Ward's merges.

Recall that $W_1, \ldots, W_\ell$ is the current Ward solution, and $O_1, \ldots, O_k$ is a fixed optimal solution, numbered from left to right. The following merges are good merges in the sense that we can bound the increase in cost. Of course, the result of the merge only forms a cluster of low cost if the participating clusters had low cost beforehand.

Type 1: Two inner clusters W_i, W_{i+1} of the same optimal cluster O_j, i.e., $W_i, W_{i+1} \subset O_j$. This type of merge is never actually applied by Ward on simplified examples, but we need it for charging.

Type 2: A 2-composed cluster $W_i \subset O_j \cup O_{j+1}$ for some j and an inner cluster $W_{i+1} \subset O_{j+1}$, with the condition that W_{i+2} is an inner cluster of O_{j+1} as well. Also: The symmetric situation of a 2-composed cluster $W_i \subset O_j \cup O_{j+1}$ for some j and an inner cluster $W_{i-1} \subset O_j$ with the condition that $W_{i-2} \subset O_j$.

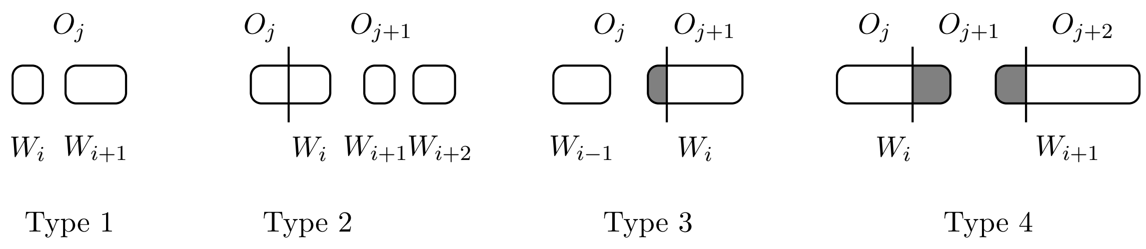


Figure 2.11: Different types of good merge situations. A part of a Ward cluster that is filled with gray contains more points than the white part of the same Ward cluster.

Type 3: A 2-composed cluster $W_i \subset O_j \cup O_{j+1}$ for some j and an inner cluster $W_{i-1} \subset O_j$, with the condition that W_i points to W_{i-1}. Also: The symmetric situation of a 2-composed cluster $W_i \subset O_j \cup O_{j+1}$ for some j and an inner cluster $W_{i+1} \subset O_{j+1}$ with the condition that W_i points to W_{i+1}.

Type 4: Two 2-composed clusters $W_i \subset O_j \cup O_{j+1}$ and $W_{i+1} \subset O_{j+1} \cup O_{j+2}$ that point at each other.

We already know Type 1 merges (inner-cluster merges), type 2 merges (growth phase and phase 5*c*) and type 3 merges (merges chargeable with Corollary 2.29). We know that applying them increases the cost by at most a constant factor. We also know that these merges cannot happen anymore: Type 1 merges are inner-cluster merges, which Ward does not do on our example. Type 2 merges happen either in the growth phase, or in phase 5*c*. Type 3 merges merge non-lopsided clusters, which happens in phase 5*a*.

Type 4 is a type of merge that we did not yet consider, and which Ward can still do. Indeed, to charge it, we need the general charging statement in the below Lemma 2.27 from which Corollary 2.29 follows.

Lemma 2.27. *Let $A, B, C, D \subset \mathbb{R}^d$ be disjoint sets with $|A| \leq |B|$ and $|C| \geq |D|$. Then*

$$\Delta(A \cup B \cup C \cup D) \leq \Delta(A) + 3 \cdot \Delta(B \cup C) + \Delta(D) + 4 \cdot D(A, B) + 4 \cdot D(C, D) \text{ and}$$

$$D(A \cup B, C \cup D) \leq 3 \cdot \Delta(B \cup C) + 3 \cdot D(A, B) + 3 \cdot D(C, D) - \Delta(B) - \Delta(C).$$

Let W_i and W_{i+1} constitute a type 4 merge as described above. Then Lemma 2.27 with $A = W_i \cap O_j$, $B = W_i \cap O_{j+1}$, $C = W_{i+1} \cap O_{j+1}$ and $D = W_{i+1} \cap O_{j+2}$ implies that

$$\begin{aligned} &\Delta(W_i \cup W_{i+1}) \\ \leq\ &\Delta(W_i \cap O_j) + 3\Delta(O_{j+1}) + \Delta(W_{i+1} \cap O_{j+2}) \\ &\qquad + 4D(W_i \cap O_j, W_i \cap O_{j+1}) + 4D(W_{i+1} \cap O_{j+1}, W_{i+1} \cap O_{j+2}). \end{aligned}$$

Thus, if $\Delta(W_i) + \Delta(W_{i+1})$ was bounded by a constant factor times the optimal cost of the points in $W_i \cup W_{i+1}$, then this is still true after the merge of W_i and W_{i+1} (with a higher factor).

Counting inner clusters Observe that the only merges that delete more than one inner cluster are the merges in phase $P1$. All other merges remove either exactly one inner cluster, or none at all. In phase $P2$-$P5$, every merge eliminates exactly one inner cluster. In the beginning, there are n inner clusters. So if phase $P1$ has n_1 merges and $P2$ until $P5$ together have n_r merges, then we have $n - 2n_1 - n_r$ inner clusters after phase $P5$, and we have n_1 2-composed clusters. The total number of all clusters is $n - n_1 - n_r$.

Consider the Ward clustering $W_1, \ldots, W_t$ after phase $P5$. We split the clustering into blocks, based on the inner clusters. More precisely, we get $n - 2n_1 - n_r - 1$ blocks that start with an inner cluster, possibly has some 2-composed clusters and ends with another inner cluster. The blocks overlap in the inner clusters.

We argue that there is at least one good merge in every block except for $k - n_1 - 1$ blocks. The exceptions are the blocks where the optimum cluster changes between start and end, but the change happens between the clusters (not in a 2-composed cluster). This can only happen $k - n_1 - 1$ times because n_1 of the $k - 1$ cluster borders are within 2-composed clusters. For the remaining blocks, we argue the following. If there are no 2-composed clusters in the block, then the two inner clusters are neighbored and form a type 1 merge. If there is only one 2-composed cluster in the block, then it has to point at an inner cluster and thus there is a type 2 or a type 3 merge. If there are multiple 2-composed clusters, we argue as follows. The first 2-composed cluster either points left and thus there is a type 2 or a type 3 merge, or it points to the right. Any further 2-composed cluster either points to the one before it, forming a type 4 merge, or it points to the right. This goes on until we either find a merge, or we find the last 2-composed cluster, which then has to point at the second inner cluster, forming a type 2 or 3 merge.

We collect one good merge from every block and call the resulting set of merges $\mathcal{S}$. Observe that the cost of all merges in $\mathcal{S}$ together is a constant factor of the cost that we have so far, so all merges together cost $\mathcal{O}(1) \operatorname{opt}_k$.

This argument alone is not enough. The main feature of $\mathcal{S}$ is that every merge that Ward actually performs can make at most one merge from our set invalid. This means that we can charge $n - 2n_1 - n_r - 1 - (k - n_1 - 1)$ merges to $\mathcal{S}$.

Notice that our merges are disjoint except for possible overlap at inner clusters. Assume that a merge of Ward invalidates two merges from our set. There are two ways how this can happen. Case one is that Ward's merge is one of the two good merges that are invalidated. Say this merge is called (A, B). Then the second merges involves either A or B, say it involves B. Thus, there is another cluster C next to B, and the merge (A, B) invalidates itself and (B, C). This in particular means that (A, B) is a good merge. Since Ward does not do inner-cluster merges, either A or B has to be 2-composed, since (A, B) is a merge of Ward. If they are both 2-composed clusters, then A and B are in the same block, thus (A, B) and (B, C) cannot both be in $\mathcal{S}$. Thus, one is 2-composed and the other is an inner cluster, i.e., they form a type 2 or 3 merge, since (A, B) is supposed to be a good merge. If it is a type 3 merge, then (A, B) is not lopsided, and would have happened in phase $P5a$. If it is a type 2 merge, then it is either not lopsided (phase $P5a$), or it is lopsided, but has an inner cluster behind its inner cluster (phase $P5c$). We conclude that a good merge (A, B) cannot invalidate another good merge.

Case two is that the two good merges are disjoint, and Ward does a merge that overlaps with both of them. Thus, we have two good merges (A, B) and (C, D), and Ward performs merge (B, C). Since Ward does not do inner-cluster merges, either B or C is 2-composed, w.l.o.g. say that C is 2-composed. If B is 2-composed as well, then (A, B) and (C, D) are in the same block, so they would not both be in $\mathcal{S}$. So B is an inner cluster. If C points to B, then (B, C) is not lopsided and would have happened in phase $P5a$. Thus, C points to D. If A is an inner cluster, then (B, C) is a type 2 merge and would have happened in phase $P5c$. So say that A is 2-composed. (A, B) is a good merge. It is not a type 2 merge since C is 2-composed. It has to be a type 3 merge, thus, A points to B. Thus, (B, C) would have happened in phase $P5d$: It is a lopsided merge with B left of C, and the 2-composed cluster left of B points to A.

We have seen that no merge of Ward can invalidate two merges from $\mathcal{S}$. Thus, we can now charge in the following way. The cost of the performed merge is bounded by the cost of any available merge. For each Ward step, we look whether it invalidates a merge from $\mathcal{S}$. If so, then we charge the performed merge to this good merge. If Ward's merge does not invalidate any merge from $\mathcal{S}$, we just arbitrarily charge a merge in $\mathcal{S}$ and mark it as invalid. In this manner, we can pay for $n-2n_1-n_r-(k-n_1-1)-1$ merges, i.e., we can pay until the number of clusters is reduced to $n-n_1-n_r-(n-2n_1-n_r-(k-n_1-1)-1) = k$. That completes the proof of Theorem 2.21.

2.4.5 Separation Conditions and Well-Clusterable Data

Clustering suffers from a general gap between theoretical study and practical application; clustering objectives are usually NP-hard to optimize, and even NP-hard to approximate to arbitrary precision. On the other hand, heuristics like Lloyd's algorithm, which can produce arbitrarily bad solutions, are known to work well or reasonably well in practice. One way of interpreting this situation is that data often has properties that make the problem computationally easier. Indeed, for clustering it is very natural to assume that the data has some structure – otherwise, what do we hope to achieve with our clustering? The challenge is to find good measures of structure that characterize what makes clustering easy (but non-trivial).

Many notions of *clusterability* have been introduced in the literature and there are also different ways to measure the quality of a clustering. While traditionally a clustering is evaluated on the basis of an objective function (e.g., the k-means objective function), there has been an increased interest recently to study which notions of clusterability make it feasible to recover (partially) a *target clustering*, some *true* clustering of the data. For this, the niceness conditions imposed on the input data are usually some form of separation condition on the clusters of the target clustering. We study the effect of five well-studied clusterability notions on the quality of the solution computed by Ward's method.

δ-center separation and α-center proximity First we study the notions of *δ-center separation* and *α-center proximity*, which have been introduced by Ben-David and Haghtalab [13] and Awasthi, Blum, and Sheffet [6], respectively.

Definition 2.35 ([13]). *An input $P \subset \mathbb{R}^d$ satisfies* δ-center separation *with respect to some target clustering $C_1, \ldots, C_k$ if there exist centers $c_1^*, \ldots, c_k^* \in \mathbb{R}^d$ such that $||c_j^* - c_i^*|| \geq \delta \cdot \max_{\ell \in k} \max_{x \in C_\ell} ||x - c_\ell^*||$ for all $i \neq j$. We say the input satisfies* weak δ-center separation *if for each cluster C_j with $j \in [k]$ and for all $i \neq j$, $||c_j^* - c_i^*|| \geq \delta \cdot \max_{x \in C_j} ||x - c_j^*||$.*

Kushagra, Samadi, and Ben-David [36] show that single linkage and a pruning technique are sufficient to find the target clustering under the condition that the data satisfies δ-center separation for $\delta \geq 3$.

While the goal of Ben-David and Haghtalab [13] is to recover a target clustering, we focus on approximating the k-means objective function. Hence, in the following we will always assume that the target clustering $C_1, \ldots, C_k$ is an optimal k-means clustering (which we usually denote by $O_1, \ldots, O_k$) and the centers $c_1^*, \ldots, c_k^* \in \mathbb{R}^d$ are the optimal k-means centers for this clustering. We will make this assumption also for all other notions of clusterability that are based on a target clustering and that we introduce in the following.

Definition 2.36 ([6])**.** *An instance P satisfies α-*center proximity *if there exists an optimal k-means clustering $O_1, \ldots, O_k$ with centers $c_1^*, \ldots, c_k^* \in \mathbb{R}^d$ such that for all $j \neq i, j \in [k]$ and for any point $x \in C_i$ it holds $||x - c_j^*|| \geq \alpha ||x - c_i^*||$.*

Awasthi, Blum, Sheffet [6] introduced the notion of *α-perturbation resilience* and showed that it implies α-center proximity. They show that for $\alpha \geq 3$, the optimal clustering can be recovered if the data is α-perturbation resilient. This was improved by Balcan and Liang [10] and finally by Makarychev and Makarychev [42], who show that it is possible to completely recover the optimal clustering for $\alpha = 2$. The latter paper shows that the results even hold for a weaker property called *metric perturbation resilience.* We show that for large enough δ and α, Ward's method computes a 2-approximation if the data satisfies δ-center separation or α-center proximity.

Theorem 2.37. *Let $P \subset \mathbb{R}^d$ be an instance that satisfies weak $(2 + 2\sqrt{2} + \epsilon)$-center separation or $(3 + 2\sqrt{2} + \epsilon)$-center proximity for some $\epsilon > 0$ and some number k of clusters. Then the k-clustering computed by Ward on P is a 2-approximation with respect to the k-means objective function.*

We also show that on instances that satisfy $(2 + 2\sqrt{2\nu} + \epsilon)$-center separation and for which all clusters O_i and O_j in the optimal clustering satisfy $|O_j| \geq |O_i|/\nu$, Ward even recovers the optimal clustering.

It is interesting to note that the example proposed by Arthur and Vassilvitskii [5] that shows that the famous k-means++ algorithm has an approximation ratio of $\Omega(\log k)$ satisfies δ-center separation and α-center proximity for large values of δ and α, and has balanced clusters, i.e., $\nu = 1$.

Observation 2.38. *There is a family of examples where k-means++ has an expected approximation ratio of $\Omega(\log k)$, while Ward computes an optimal solution.*

In contrast we will see that the instances that we use to prove our exponential lower bound on the approximation factor of Ward's method (Theorem 2.19) satisfy δ-center separation and α-center proximity for $\delta \leq 1 + \sqrt{2}$ and $\alpha \leq 1 + \sqrt{2}$. We will also see that even for arbitrary large δ and α there are instances that satisfy δ-center separation and α-center proximity and on which Ward's method does not compute an optimal solution.

Strict separation property Balcan, Blum, and Vempala [9] introduce the *strict separation property.* In [1], this property is defined as follows[1].

[1]In [9], the property is defined for similarity measures.

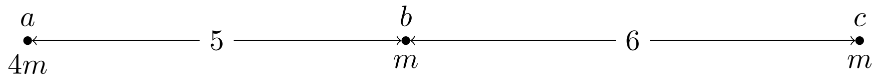


Figure 2.12: Redrawing of Figure 16 in [11]. There are three groups of points A, B and C at the locations a, b and c. The numbers indicate the sizes of the groups, the total number is $n = 6m$.

Definition 2.39 ([1, 9])**.** *An instance P with target clustering $C_1, \ldots, C_k$ satisfies the* strict separation property *if for all $x, y \in C_i, i \in [k]$, and every $z \in C_j$ for $j \in [k]$ with $j \neq i$, $||x - y|| < ||y - z||$. It satisfies ν-strict separation if there is a subset of P of size at least $(1 - \nu)|P|$ for which the property is satisfied.*

Balcan et al. show that if the clusters in the target clustering are of sufficient size and the instance is ν-strict separated, then one can correctly classify all but νn points (correctly with respect to the target clustering). Indeed, if all points satisfy the property, then the target clustering can be completely recovered.

Balcan, Liang, and Gupta [11] have already studied Ward's method on strictly separated instances. They present the instance shown in Figure 2.12 (In §C, Figure 16). Given the target clustering $A \cup B$, C, the example clearly satisfies strict separation for all points. However, Ward will compute the clustering A, $B \cup C$: It will start by merging all points at the same location, resulting in three weighted points p_a, p_b, p_c at a, b, and c. But then it will merge p_b and p_c because this is cheaper than merging p_a and p_b: It costs $\frac{m}{2} \cdot 6^2 = 18m$, while the alternative merge costs $\frac{4}{5}m \cdot 5^2 = 20m$. The resulting clustering is then judged to be very bad since it misclassifies $m = \frac{1}{6}n$ points.

Now one can argue that this judgment is overly critical because Ward actually achieves its design goal on this instance: it computes a solution with minimal k-means cost. Hence, in this thesis we study the behavior of Ward when the target clustering is an optimal k-means clustering. We will see that also in this case the strict separation property does not help Ward to compute a good clustering.

ORSS-separation Another line of work on niceness conditions for clustering investigates conditions that help to find a low-cost clustering with respect to the k-means objective function, usually a $(1 + \epsilon)$-approximation. For this area, conditions that ensure a cost separation between different solutions are helpful. We will see that the strongest among these conditions, namely ϵ-separation [44] (we will also use the term *ϵ-ORSS-separation*), does not help Ward to avoid the worst-case example from Theorem 2.19.

Definition 2.40 ([44])**.** *An instance P satisfies the* ϵ-ORSS-separation property *for some number of clusters k if* $\mathrm{opt}_k(P) / \mathrm{opt}_{k-1}(P) \leq \epsilon^2$*.*

Ostrovsky et al. [44] show that a variant of Lloyd's method with the right seeding computes a $(1 + \epsilon)$-approximation for the k-means problem on instances that satisfy ϵ-ORSS-separation.

AS-separation For the following separation condition, it is convenient to denote the point set by a matrix A, where row i contains point A_i. Let $C_1, \ldots, C_k$ be a target clustering for A and let $\mu_1, \ldots, \mu_k$ be the corresponding centroids, i.e., $\mu_i = \mu(C_i)$. Assume that $C \in \mathbb{R}^{n \times d}$ is a matrix where the ith row contains the centroid of the cluster that A_i belongs to. Then $||A - C||_F^2$ is the k-means cost of the clustering $C_1, \ldots, C_k$, where $|| \cdot ||_F$ is the Frobenius norm. Let $|| \cdot ||$ denote the spectral norm.

In a seminal paper, Kumar, and Kannan [35] defined a proximity condition and showed that if all points satisfy this condition, then the target clustering can be reconstructed (and if only a fraction satisfies it, then the target clustering can be mostly recovered). The proximity condition states that the projection of a point onto the line joining its cluster center μ_i with another cluster center μ_j is closer to μ_i than to μ_j by at least a value Δ_{ij}, where Δ_{ij} depends on the number of points in the two clusters, and on $k||A - C||$ (and a big constant). Here, we consider the weaker center-based condition due to Awasthi and Sheffet [8], which was developed in follow-up work. We call it *AS-center separation* to distinguish it from the above δ-center separation.

Definition 2.41 ([8]). *Let A and C be as defined above and define*

$$\Delta_i = \frac{1}{\sqrt{|C_i|}} \min\{\sqrt{k}||A - C||, ||A - C||_F\}.$$

Then the instance A satisfies AS-center separation *with respect to the target clustering $C_1, \ldots, C_k$ if for all $i \neq j$, $i, j \in [k]$, it holds that*

$$||\mu_i - \mu_j|| \geq c(\Delta_i + \Delta_j)$$

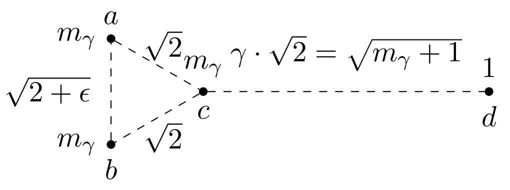
where c is a fixed constant.

Again, if all points satisfy AS-center separation, then the target clustering can be recovered [8]. We will see that the exponential lower bound instances satisfy AS-separation when the target clustering is the optimal k-means clustering.

Corollary 2.42. *For any $\epsilon > 0$, there is a family of point sets $(P_d)_{d \in \mathbb{N}}$ with $P_d \subset \mathbb{R}^d$ that are ϵ-separated and that satisfy $1 + \sqrt{2}$-center separation, $1 + \sqrt{2}$-center proximity, the strict separation property and the AS-center separation property where $\mathrm{Ward}_k(P_d) \in \Omega((3/2)^d \cdot \mathrm{opt}_k(P_d))$ for $k = 2^d$. Furthermore, for any $\delta > 1$ and any $\alpha > 1$, there exists a point set that satisfies δ-center separation and α-center proximity and for which Ward does not compute an optimal solution.*

2.4.6 Bounds for δ-center separation and α-center proximity

In this section, we analyze the behavior of Ward on δ-center separated instances and instances that satisfy α-center proximity for some number k of clusters. Before we will prove Theorem 2.37 we start by giving a simple example that shows that Ward does not necessarily compute an optimal solution for instances that satisfy δ-center separation and α-center proximity for arbitrary δ and α. After that we show that nevertheless Ward works well on such instances and even computes an optimal clustering if the clusters in the optimal clustering are balanced with respect to its size.



Coordinates
a: $(0, (\sqrt{2+\epsilon})/2)$
b:
$(0, -(\sqrt{2+\epsilon})/2)$
c:
$(\sqrt{3/2 - \epsilon/4}, 0)$
$d : c + (\gamma \cdot \sqrt{2}, 0)$

$\epsilon := 1/(m_\gamma + 1)$
$m_\gamma := 2\gamma^2 - 1$
$\Leftrightarrow$
$\gamma^2 = \dfrac{m_\gamma + 1}{2}$

Figure 2.13: Family of instances for $k = 2$ that shows that Ward does not necessarily compute an optimal solution for instances that satisfy δ-center separation and α-center proximity for arbitrary δ and α.

Center Separation and Center Proximity do not Guarantee Optimality

In this section we give an example that shows that not even for arbitrary large δ and α, δ-center separation and α-center proximity guarantee that Ward's method computes an optimal clustering. Figure 2.13 depicts a family of instances for $k = 2$. The idea of the example is that Ward's method merges the lone point d with the points at c, which is inconsistent with the optimum clustering. We first compute merge costs for different possible merges.

Lemma 2.43. *For all instances of the family in Figure 2.13, $D(a,b) = m_\gamma(1 + \frac{\epsilon}{2})$ and $D(a,c) = D(b,c) = D(c,d) = m_\gamma$. Furthermore, $D(a,cd) = D(b,cd) > D(a,b)$ and $D(ab,c) < m_\gamma$.*

Proof. By Lemma 2.16, $D(c,d) = \frac{m_\gamma}{m_\gamma+1} \cdot 2\gamma^2 = m_\gamma$, since the squared distance between c and d is $2\gamma^2$. The squared distance between a and b is $2 + \epsilon$, and the squared distance between a and c as well as between b and c is 2. Thus, Lemma 2.16 implies that $D(a,b) = \frac{m_\gamma^2}{2m_\gamma}(2+\epsilon) = m_\gamma(1 + \frac{\epsilon}{2})$ and that $D(a,c) = D(b,c) = \frac{m_\gamma^2}{2m_\gamma} \cdot 2 = m_\gamma$, and $D(c,d) = \frac{m_\gamma}{m_\gamma+1} \cdot \gamma^2 \cdot 2 = m_\gamma$.

Next, we show that $D(a,cd) = D(b,cd) > D(a,b)$. We have that

$$D(a,cd) = \frac{m_\gamma(m_\gamma+1)}{2m_\gamma+1} \cdot ||\mu_a - \mu_{cd}||^2 \geq \frac{1}{2} m_\gamma \cdot ||\mu_a - \mu_{cd}||^2.$$

Note that μ_{cd} is given by $\mu_{cd} = c + (\frac{1}{\sqrt{m_\gamma+1}}, 0)$. Using Pythagoras we obtain

$$\begin{aligned}
||\mu_a - \mu_{cd}||^2 &= ||\mu_a - (0,0)||^2 + ||(0,0) - \mu_{cd}||^2 \\
&> ||\mu_a - (0,0)||^2 + ||(0,0) - c||^2 + \frac{1}{m_\gamma + 1} \\
&= ||a - c||^2 + \frac{1}{m_\gamma + 1} \\
&= 2 + \epsilon.
\end{aligned}$$

Thus, $D(a, cd) > \frac{1}{2} m_\gamma \cdot (2+\epsilon)$ and we obtain $D(a, cd) > D(a, b)$ (and $D(b, cd) > D(a, b)$). Finally,

$$D(ab, c) = \frac{2}{3} m_\gamma \cdot ||c - (0,0)||^2 = \frac{2}{3} m_\gamma \cdot \left(\frac{3}{2} - \frac{\epsilon}{4}\right) < m_\gamma,$$

which completes the proof. □

As announced above, we assume that Ward's method chooses to merge c and d in the first step, which is one of the cheap merges. In the second step, it will then merge a and b, since $D(a, cd) = D(b, cd) > D(a, b)$. The resulting clustering $\{a, b\}, \{c, d\}$ costs $D(c, d) + D(a, b)$. This is strictly more than the cost of the clustering $\{a, b, c\}, \{d\}$, which costs $D(a, b) + D(ab, c) < D(a, b) + m_\gamma = D(a, b) + D(c, d)$. Thus, Ward's method does not compute an optimal solution.

Lemma 2.44. *All instances of the family in Figure 2.13 satisfy $\gamma\sqrt{2}$-center separation and $\gamma\sqrt{2}$-center proximity.*

Proof. The optimum 2-clustering for an instance from the family is $\{a, b, c\}, \{d\}$ with centers $\mu_1 = (\sqrt{\frac{1}{6} - \frac{\epsilon}{36}}, 0)$ and $\mu_2 = d$. The point d has distance 0 to its center μ_2, the point c has distance $\sqrt{\frac{3}{2} - \frac{\epsilon}{4}} - \sqrt{\frac{1}{6} - \frac{\epsilon}{36}} < \sqrt{\frac{3}{2}} - \sqrt{\frac{1}{6} - \frac{1}{36}} < 1$ to μ_1 and using Pythagoras again a and b have distance $\sqrt{\frac{2+\epsilon}{4} + \frac{1}{6} - \frac{\epsilon}{36}} \le \sqrt{\frac{1}{2} + \frac{1}{4} + \frac{1}{6}} < 1$ to μ_1. The distance between μ_1 and μ_2 is more than $\gamma\sqrt{2}$. Thus, the instance satisfies $\gamma\sqrt{2}$-center separation. It also satisfies $\gamma\sqrt{2}$-center proximity, since the distance between any point in $\{a, b, c\}$ and μ_2 is at least $\gamma\sqrt{2}$. □

Corollary 2.45. *For any $\delta > 0$ and any $\alpha > 0$, there is an instance with $k = 2$ that satisfies δ-center separation and α-center proximity and for which Ward does not find an optimum clustering.*

The approximation ratio between Ward's solution and the optimum solution in Figure 2.13 is relatively small. Notice, however, that this is a family for examples that shows that Ward is not optimal for any possible δ and α.

The upper Bound

We are only interested in the k-clustering computed by Ward. Hence, in the following we assume that k is fixed and that Ward stops as soon as it has obtained a k-clustering. First we prove that center proximity implies weak center separation. Hence, it suffices to study instances that satisfy weak center separation.

Lemma 2.46. *Let $P \subset \mathbb{R}^d$ be an instance that satisfies α-center proximity. Then P also satisfies weak $(\alpha - 1)$-center separation.*

Proof. Let $O_1, \dots, O_k$ be an optimal k-means clustering for P with cluster centers $c_1^*, \dots, c_k^*$. Fix arbitrary $j \in [k]$ and $i \in [k]$ with $i \neq j$. For $x \in O_j$ we have $||x - c_i^*|| \ge \alpha ||x - c_j^*||$. Moreover, we have that $||x - c_i^*|| \le ||x - c_j^*|| + ||c_i^* - c_j^*||$. Together this implies $(\alpha - 1)||x - c_j^*|| \le ||c_i^* - c_j^*||$. Since this is true for all $x \in O_j$, we get that $||c_i^* - c_j^*|| \ge (\alpha - 1) \cdot \max_{x \in O_j} ||x - c_j^*||$. □

In the following we call a cluster A that is formed by Ward an *inner cluster* if A is completely contained within an optimum cluster. We start our analysis with the following lemma, which states one very crucial property of Ward's behavior on well-separated data. It implies that Ward does not merge inner clusters from two different optimal clusters as long as there exists more than one inner cluster in both of these optimal clusters.

Lemma 2.47. *Let $P \subset \mathbb{R}^d$ be an instance that satisfies weak $(2+2\sqrt{2}+\epsilon)$-center separation for some $\epsilon > 0$. Assume we have two optimal clusters O_1 and O_2 and each of them contains at least two inner clusters A_1, B_1 and A_2, B_2, respectively, directly after the i-th step of Ward. Then, in step $i+1$, Ward will not merge an inner cluster of O_1 with an inner cluster of O_2.*

Proof. To prove the lemma, we assume w.l.o.g. that merging A_1 and A_2 is the merge operation with minimum increase under all operations containing exactly one inner cluster of O_1 and one inner cluster of O_2. We prove that $\min\{D(A_1, B_1), D(A_2, B_2)\} < D(A_1, A_2)$. Thus, Ward will not merge A_1 and A_2. Due to the choice of A_1 and A_2 this implies that Ward will not merge an inner cluster of O_1 with an inner cluster of O_2.

Let $r_i = \max_{x \in O_i} ||x - \mu(O_i)||$ be the radius of cluster O_i. Since A_i is contained in O_i, we have $||\mu(A_i) - \mu(O_i)|| \leq r_i$. From the triangle inequality and the weak $(2 + 2\sqrt{2} + \epsilon)$-center separation it follows that

$$\begin{aligned}
||\mu(A_1) - \mu(A_2)|| &\geq ||\mu(O_1) - \mu(O_2)|| - r_1 - r_2 \\
&\geq (2 + 2\sqrt{2} + \epsilon) \cdot \max\{r_1, r_2\} - r_1 - r_2 \\
&\geq (2 + 2\sqrt{2} + \epsilon) \cdot \max\{r_1, r_2\} - 2\max\{r_1, r_2\} \\
&> 2\sqrt{2}\max\{r_1, r_2\}.
\end{aligned}$$

For symmetry reasons we may assume $|A_1| \leq |A_2|$. Then with the above bound for $||\mu(A_1) - \mu(A_2)||$ we obtain the following lower bound for $D(A_1, A_2)$:

$$\begin{aligned}
D(A_1, A_2) &= \frac{|A_1| \cdot |A_2|}{|A_1| + |A_2|} \cdot ||\mu(A_1) - \mu(A_2)||^2 \\
&\geq \frac{|A_1| \cdot |A_2|}{|A_2| + |A_2|} \cdot ||\mu(A_1) - \mu(A_2)||^2 \\
&\geq \frac{|A_1|}{2} \cdot ||\mu(A_1) - \mu(A_2)||^2 \\
&> \frac{|A_1|}{2}\left(2\sqrt{2}\max\{r_1, r_2\}\right)^2 \\
&\geq 4|A_1| \cdot r_1^2.
\end{aligned}$$

Now we compare this to $D(A_1, B_1)$. Since A_1 and B_1 are both contained in O_1, we have $||\mu(A_1) - \mu(B_1)|| \leq 2r_1$. In accordance with Lemma 2.16, this implies

$$D(A_1, B_1) = \frac{|A_1| \cdot |B_1|}{|A_1| + |B_1|} \cdot ||\mu(A_1) - \mu(B_1)||^2 \leq |A_1| \cdot ||\mu(A_1) - \mu(B_1)||^2 \leq 4|A_1| \cdot r_1^2 < D(A_1, A_2).$$

$\square$

Inner-cluster merges In the following let $P \subset \mathbb{R}^d$ be an arbitrary instance and let $O_1, \dots, O_k$ be an optimal k-clustering of P with objective value $\mathrm{opt} = \mathrm{opt}_k(P)$. Our goal is to show that the k-clustering $W_1, \dots, W_k$ computed by Ward on P is worse by only a factor of at most 2 if P satisfies weak $(2 + 2\sqrt{2} + \epsilon)$-center separation for some $\epsilon > 0$.

Observe that Lemma 2.47 does not exclude the possibility that Ward performs inner-cluster merges on P, i.e., it might merge two inner clusters from the same optimum cluster at some point during its execution. While we will see that in the one-dimensional case one can assume that such inner-cluster merges do not happen, we cannot make this assumption in general (see Figure 2.13, where the counterexample crucially needs an inner-cluster merge). In our analysis, we bound the costs of the inner-cluster merges separately from the costs of the other merges, which we call *non-inner merges* in the following.

We define an equivalence relation r on P as follows: two points x_1 and $x_2 \in P$ are equivalent if and only if there exists an inner cluster C constructed by Ward at some point of time with $x_1, x_2 \in C$. We denote the equivalence classes of r by $P/r = \{C_1, \dots, C_m\}$. The following observation is immediate.

Observation 2.48. *If Ward merges in any step an inner cluster C with another cluster that is not an inner cluster of the same optimal cluster, then $C \in P/r$ is an equivalence class.*

This means that the equivalence classes represent inner clusters of Ward right before they are merged with points from outside their optimal cluster. With other words, if we perform all inner cluster merges that are performed by Ward and leave out all non-inner merges, we get the clustering represented by P/r.

Consider an arbitrary optimal cluster O_j and let $P_1^j, \dots, P_{n_j}^j$ denote the inner clusters of O_j in P/r. We assume that these inner clusters are indexed in the order in which they are merged with other clusters by Ward. To illustrate this definition, consider the step in which P_i^j is merged by Ward with some other cluster Q. Since $P_i^j \in P/r$, this step is a non-inner merge and in particular Q is not equal to any of the clusters $P_{i+1}^j, \dots, P_{n_j}^j$. At the time this merge happens, the indexing guarantees that the cluster P_{i+1}^j is either present or there exist multiple parts $C_1, \dots, C_\ell$ of P_{i+1}^j that are only later merged by inner-cluster merges to P_{i+1}^j. Since Ward merges P_i^j and Q, we know that $D(P_i^j, Q) \leq D(P_i^j, C_h)$ for any $h \in [\ell]$. We will use this fact to give an upper bound for the costs of the clustering $W_1, \dots, W_k$.

It might be that some inner clusters of O_j in P/r are not merged at all by Ward and contained in the clustering $W_1, \dots, W_k$. These inner clusters are the last in the ordering, i.e., they are $P_a^j, \dots, P_{n_j}^j$ where $n_j - a + 1$ is the number of such clusters.

Potential graph In order to bound the costs of the clustering $W_1, \dots, W_k$ produced by Ward we introduce the *potential graph* $G = (V, E)$ with vertex set $V = P/r$. The edges E of G are directed and there are only edges between inner clusters of the same optimal cluster. Consider an arbitrary optimal cluster O_j with $j \in [k]$ and let $P_1^j \dots P_{n_j}^j$ be the inner clusters of O_j in P/r indexed as above in the order in which they are merged with other clusters by Ward. Then for every $i \in [n_j - 1]$ the set E contains the edge (P_i^j, P_{i+1}^j). Both the vertices and the edges are weighted and we denote the sum of all vertex and edge weights by $w(G)$.

The weight of a vertex $Q \in P/r$ is defined as $w(Q) = \Delta(Q)$, i.e., the weight of vertex Q equals the costs of forming the inner cluster Q. We will now define weights for the edges such that the sum of all vertex and edge weights in the potential graph is at most $2\,\mathrm{opt}_k$. After that we prove that there is a one-to-one correspondence between the non-inner merges of Ward and the edges in the graph such that the costs of each non-inner merge of Ward are at most the weight of the associated edge. Together this proves that Ward computes a solution with costs at most $2\,\mathrm{opt}_k$.

To define the weight of the edge (P_i^j, P_{i+1}^j), we first consider the case that P_i^j is merged at some point of time with another cluster Q by Ward. Then let $C_1, \ldots, C_\ell$ again denote the parts of P_{i+1}^j that are present at that point of time. The edge weight $w(P_i^j, P_{i+1}^j)$ is defined as $\max_{h\in[\ell]} D(P_i^j, C_h)^2$. Observe that since Ward performs greedy merges, this definition guarantees that the merge of P_i^j and Q costs at most the edge weight $w(P_i^j, P_{i+1}^j)$. If P_i^j is not merged at all by Ward, we set the weight $w(P_i^j, P_{i+1}^j)$ to $D(P_i^j, P_{i+1}^j)$.

Lemma 2.49. *Let $P \subset \mathbb{R}^d$ be a finite point set and let $Q_1, \ldots, Q_\ell$ denote an arbitrary partition of P into pairwise disjoint parts. Then $\Delta(P) \geq \Delta(Q_1) + \ldots + \Delta(Q_\ell)$.*

Proof. The lemma follows from the following calculation:

$$\begin{aligned}
\Delta(P) = \Delta(P, \mu(P)) &= \sum_{x\in P} ||x - \mu(P)||^2 \\
&= \sum_{i=1}^{\ell} \sum_{x\in Q_i} ||x - \mu(P)||^2 \\
&= \sum_{i=1}^{\ell} \Delta(Q_i, \mu(P)) \\
&\geq \sum_{i=1}^{\ell} \Delta(Q_i, \mu(Q_i)) = \sum_{i=1}^{\ell} \Delta(Q_i). \qquad \square
\end{aligned}$$

Lemma 2.50. *The weights in the potential graph satisfy $w(G) \leq 2\,\mathrm{opt}_k$.*

Proof. Since there are no edges between inner clusters of different optimal clusters, we can analyze each optimal cluster separately. Let O_j be an arbitrary optimal cluster and let $P_1^j, \ldots, P_{n_j}^j$ denote the inner clusters of O_j in P/r. Then the graph G contains for each $i \in [n_j - 1]$ the edge (P_i^j, P_{i+1}^j). Let us denote the set of these edges by E_j. We partition E_j into two disjoint matchings E_j^{odd} and E_j^{even} where $E_j^{\mathrm{odd}} = \{(P_i^j, P_{i+1}^j) \mid i \text{ is odd}\}$ and $E_j^{\mathrm{even}} = E_j \setminus E_j^{\mathrm{odd}}$.

Let $i \in [n_j - 1]$. We first consider the case that P_i^j is merged by Ward at some point of time with some other cluster. We denote by $C_1, \ldots, C_\ell$ the parts of P_{i+1}^j that are present at that point of time. Let Q_{i+1}^j denote a part C_h of P_{i+1}^j for which $D(P_i^j, C_h)$ is maximal.

[2]When reading the proof the reader might notice that our definition of $w(P_i^j, P_{i+1}^j)$ is to some extend arbitrary. Instead of defining it as $\max_{h\in[\ell]} D(P_i^j, C_h)$, we could also define it as $\min_{h\in[\ell]} D(P_i^j, C_h)$ or as $D(P_i^j, C_h)$ for any h.

If P_i^j is not merged by Ward, we set $Q_{i+1}^j = P_{i+1}^j$. Then by the definition of the potential graph, in both cases, the edge (P_i^j, P_{i+1}^j) has weight $D(P_i^j, Q_{i+1}^j)$ and $Q_{i+1}^j \subseteq P_{i+1}^j$.

Let us first assume that n_j is even. Then we obtain with Lemma 2.49

$$\begin{aligned}
\Delta(O_j) &\geq \sum_{(P_i^j, P_{i+1}^j) \in E_j^{\text{odd}}} \Delta\Big(P_i^j \cup P_{i+1}^j\Big) \\
&\geq \sum_{(P_i^j, P_{i+1}^j) \in E_j^{\text{odd}}} \Delta\Big(P_i^j \cup Q_{i+1}^j\Big) \\
&= \sum_{(P_i^j, P_{i+1}^j) \in E_j^{\text{odd}}} \Delta\Big(P_i^j\Big) + \Delta\Big(Q_{i+1}^j\Big) + D\Big(P_i^j, Q_{i+1}^j\Big) \\
&\geq \sum_{(P_i^j, P_{i+1}^j) \in E_j^{\text{odd}}} w\Big(P_i^j\Big) + w\Big(P_i^j, P_{i+1}^j\Big).
\end{aligned}$$

An analogous bound holds true for E_j^{even}. In fact, since we assumed n_j to be even, the last vertex $P_{n_j}^j$ is not covered by E_j^{even}. This yields by the same reasoning as above the following slightly stronger inequality:

$$\begin{aligned}
\Delta(O_j) &\geq \Delta\Big(P_{n_j}^j\Big) + \sum_{(P_i^j, P_{i+1}^j) \in E_j^{\text{even}}} \Delta\Big(P_i^j \cup P_{i+1}^j\Big) \\
&\geq w\Big(P_{n_j}^j\Big) + \sum_{(P_i^j, P_{i+1}^j) \in E_j^{\text{even}}} w\Big(P_i^j\Big) + w\Big(P_i^j, P_{i+1}^j\Big).
\end{aligned}$$

Adding the inequalities for E_j^{odd} and E_j^{even} yields

$$2\Delta(O_j) \geq \sum_{j=1}^{n_j} w\Big(P_i^j\Big) + \sum_{(P_i^j, P_{i+1}^j) \in E_j} w\Big(P_i^j, P_{i+1}^j\Big). \tag{2.1}$$

In the case that n_j is odd, we obtain the same inequality by adding the last vertex $P_{n_j}^j$ to the inequality for E_j^{odd} instead of E_j^{even}. Observe that the right-hand side of (2.1) equals the sum of all vertex and edge weights in the component of the potential graph that corresponds to O_j. Adding up the inequalities for every j proves the lemma:

$$2\,\text{opt} = 2\Big(\sum_{j=1}^{k} \Delta(O_j)\Big) \geq \sum_{Q \in P/r} w(Q) + \sum_{(P,Q) \in E} w(P, Q). \qquad \square$$

Bijection between non-inner merges and edges We have seen that the total weight of the potential graph is at most $2\,\text{opt}_k$. Our goal is now to find a bijection between the non-inner merges of Ward and the edges of the potential graph such that the costs of any non-inner merge are bounded from above by the weight of the edge assigned to it in the bijection. The existence of such a bijection implies that also the costs of the solution $W_1, \ldots, W_k$ computed by Ward are at most $2\,\text{opt}_k$.

Now we construct this bijection. Let us first consider non-inner merges in which at least one of the clusters is an inner cluster contained in P/r. Let this be the inner cluster P_i^j of some optimal cluster O_j and assume further that $i < n_j$. Then P_i^j has an outgoing edge to P_{i+1}^j. We denote by Q the cluster with which P_i^j is merged and we assign the merge of P_i^j with Q to the edge (P_i^j, P_{i+1}^j) in the bijection.

Lemma 2.51. *Let $P \subset \mathbb{R}^d$ be an instance that satisfies weak $(2+2\sqrt{2}+\epsilon)$-center separation for some $\epsilon > 0$. Consider a non-inner merge of Ward between two inner clusters from P/r. Then at most one of these inner clusters has an outgoing edge in G.*

Proof. Let $P_{i_1}^{j_1}$ and $P_{i_2}^{j_2}$ be the two clusters from P/r that are merged. From the definition of P/r it follows that $j_1 \neq j_2$. Assume for contradiction that both $P_{i_1}^{j_1}$ and $P_{i_2}^{j_2}$ have outgoing edges in G. Then $i_1 < n_{j_1}$ and $i_2 < n_{j_2}$. Hence, when $P_{i_1}^{j_1}$ and $P_{i_2}^{j_2}$ are merged there exist two other inner clusters $P_{i_1+1}^{j_1}$ and $P_{i_2+1}^{j_2}$ of O_{j_1} and O_{j_2}, respectively. This is a contradiction to Lemma 2.47. □

Observe that it cannot happen that the same edge is assigned to two different merges by the construction described above because an edge (P_i^j, P_{i+1}^j) can only be assigned to a step in which P_i^j is merged with some other cluster and there can only be one such merge.

Let $L \subseteq E$ denote the set of edges that are not assigned to a step of Ward by the above construction. The potential graph G contains $|V| = |P/r|$ vertices and $|V| - k$ edges. Since the number of non-inner merges of Ward is also $|V| - k$, there are also $|L|$ non-inner merges that are not yet assigned to an edge. We finish the construction of the bijection by assigning the unassigned non-inner merges arbitrarily bijectively to L.

Lemma 2.52. *The costs of each non-inner merge of Ward are bounded from above by the weight of the assigned edge in the potential graph.*

Proof. First we consider steps in which one of the clusters is an inner cluster P_i^j with $i < n_j$ contained in P/r. Let Q denote the cluster with which P_i^j is merged. At the point of time at which this merge happens, let $C_1, \dots, C_\ell$ denote the parts of P_{i+1}^j that are present. The merge of P_i^j and Q is assigned to the edge (P_i^j, P_{i+1}^j) in the potential graph. The weight of this edge is defined as $\max_{h \in [\ell]} D(P_i^j, C_h)$. Since the merge of P_i^j and Q is a greedy merge, it must be $D(P_i^j, Q) \leq D(P_i^j, C_h)$ for all $h \in [\ell]$. Hence, the weight of the edge (P_i^j, P_{i+1}^j) is an upper bound for the costs of the merge of P_i^j and Q.

It remains to consider the steps in which no inner cluster P_i^j with $i < n_j$ is involved. These steps are assigned arbitrarily to the set L of unassigned edges at the end. For these steps we can use the monotonicity of Ward (Corollary 2.18). Observe that an edge (P_i^j, P_{i+1}^j) belongs to L if and only if the inner cluster P_i^j is not merged at all by Ward. Due to the ordering of the inner clusters this implies that also the cluster P_{i+1}^j is not merged by Ward. Hence, both P_i^j and P_{i+1}^j are clusters in the final clustering $W_1, \dots, W_k$. Hence, in this clustering the costs of a greedy merge are at most $D(P_i^j, P_{i+1}^j)$. Due to Corollary 2.18, this implies that all merges performed by Ward to obtain the Clustering $W_1, \dots, W_k$ have each costs at most $D(P_i^j, P_{i+1}^j)$. Hence, the weight of any edge in L is an upper bound for the costs of each merge of Ward. □

Now the following theorem follows easily.

Theorem 1.5. *Let $P \subset \mathbb{R}^d$ be an instance that satisfies weak $(2+2\sqrt{2}+\epsilon)$-center separation or $(3+2\sqrt{2}+\epsilon)$-center proximity for some $k \in [|P|]$ and $\epsilon > 0$. Then Ward computes a 2-approximation on P for that k.*

Proof. First we consider instances that satisfy weak $(2+2\sqrt{2}+\epsilon)$-center separation. The costs of the k-clustering $W_1, \dots, W_k$ computed by Ward equal the sum $\sum_{Q \in P/r} \Delta(Q) = \sum_{Q \in P/r} w(Q)$ plus the costs of all non-inner merges performed by Ward. In accordance with Lemma 2.52, the sum of the costs of the non-inner merges is bounded from above by the sum of edge weights in the potential graph. Hence, the costs of $W_1, \dots, W_k$ are upper bounded by the sum of vertex and edge weights in the potential graph. This sum is at most $2\,\mathrm{opt}$ due to Lemma 2.50.

For instances that satisfy $(3+2\sqrt{2}+\epsilon)$-center proximity the theorem follows from the first part of the theorem and Lemma 2.46. □

Theorem 1.6. *Let $P \subset \mathbb{R}^d$ be an instance with optimal k-means clustering $O_1, \dots, O_k$ with centers $c_1^*, \dots, c_k^* \in \mathbb{R}^d$. Assume that P satisfies $(2+2\sqrt{2\nu}+\epsilon)$-center separation for some $\epsilon > 0$, where $\nu = \max_{i,j \in [k]} \frac{|O_i|}{|O_j|}$ is the largest factor between the sizes of any two optimum clusters. Then Ward computes the optimal k-means clustering $O_1, \dots, O_k$.*

Proof. Assume that there are merges between inner clusters of different optimum clusters, and let (A_1, A_2) be the first such merge. That means that A_1 and A_2 are two inner clusters from O_i and O_j for some $i, j \in [k]$, $i \neq j$. They are merged by Ward's method, and before their merge, all merges were inner-cluster merges. Since the instance is $(2+2\sqrt{2\nu}+\epsilon)$-center separated, the triangle inequality implies $||\mu(A_1) - \mu(A_2)|| \geq (2\sqrt{2\nu}+\epsilon)r$ for $r = \max_{\ell \in [k]} \max_{x \in C_\ell} ||x - c_\ell^*||$ (cf. proof of Lemma 2.47). Hence, we get by Lemma 2.16 that

$$D(A_1, A_2) > \min\{|A_1|, |A_2|\} \cdot \frac{1}{2}(8\nu + \epsilon^2)r^2 > \min\{|A_1|, |A_2|\} \cdot 4\nu r^2.$$

If there are two inner clusters $B_1 \neq A_1$ and $B_2 \neq A_2$ with $B_1 \in O_i$ and $B_2 \in O_j$ at the time of the merge (A_1, A_2), then A_1 and A_2 will not be merged by the same argument as in the proof of Lemma 2.47. If only B_1 exists, but A_2 is the only inner cluster in O_j, then $|A_2| = |O_j| \geq |O_i|/\nu$. We know that

$$D(A_1, B_1) = \frac{|A_1| \cdot |B_1|}{|A_1| + |B_1|} \cdot ||c^*(A_1) - c^*(B_1)||^2 \leq \min\{|A_1|, |B_1|\} \cdot 4r^2.$$

If $\min\{|A_1|, |A_2|\} = |A_1|$, then $D(A_1, A_2) > |A_1| \cdot 4\nu r^2 \geq D(A_1, B_1)$. Furthermore, if $\min\{|A_1|, |A_2|\} = |A_2| \geq |O_i|/\nu$, then

$$D(A_1, A_2) > |O_i| \cdot \frac{1}{\nu} \cdot 4\nu r^2 > \min\{|A_1|, |B_1|\} \cdot 4r^2.$$

Thus, the merge (A_1, A_2) will not happen. Lastly, assume that both A_1 and A_2 are the last inner cluster. Then we either have only k clusters left, or there are two inner clusters C and D in some other optimum cluster O_ℓ. We also know that $|A_1| = |O_i| \geq |O_\ell|/\nu$ and $|A_2| = |O_j| \geq |O_\ell|/\nu$, implying that $D(A_1, A_2) > \frac{|O_\ell|/\nu}{2} \cdot 8\nu r^2 > \min\{|C|, |D|\} \cdot 4r^2 \geq D(C, D)$, and we get a contradiction to the assumption that A_1 and A_2 are merged. □

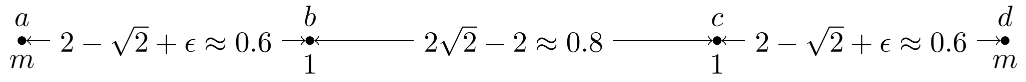


Figure 2.14: The left and right points (a and d) have weight m, while b and c have weight 1. This has the effect that Ward merges b and c (for $\epsilon = 1/\sqrt{m}$), and then ends with either $\{a, b, c\}, \{d\}$ or $\{a\}, \{b, c, d\}$. The optimum clustering is $\{a, b\}, \{c, d\}$, and the factor between the two clusterings converges to $2 + \sqrt{2} \approx 3.41$.

We conclude this section by showing that Theorem 1.5 does not hold for significantly smaller δ and α. Consider the one-dimensional example in Figure 2.14 from [48]. Ward may compute the clustering $\{a, b, c\}$, $\{d\}$, while the optimal clustering is $\{a, b\}$, $\{c, d\}$, and the approximation ratio of this example is $2 + \sqrt{2} \approx 3.41$. Notice that this example is $(3 + \sqrt{2})$-center separated (this is ≈ 0.414 smaller than the δ in our upper bound) and it satisfies $(1 + \sqrt{2})$-center proximity.

2.4.7 Exponential Lower Bound for Well-Clusterable Data

First we have a closer look at the construction of the lower bound in Theorem 2.19 and show that it satisfies the clusterability notions ORSS-separation, the strict separation property and AS-center separation. For the sake of simplicity we first look at the simplified version where the weight of the heavy points is infinite. After that we do a more careful calculation for the case that all weights are finite.

For the version with infinite weights ORSS-separation follows easily for any $\epsilon > 0$. For this observe that if the instance was to be clustered with $2^d - 1 = k - 1$ clusters, then at least two heavy points would need to be merged, which generates infinite costs, while the costs of the optimal k-clustering are finite.

Now we come to the strict separation property. Notice that the distance between a light point and its closest heavy point is $2 - \sqrt{2} \leq 0.59$. The minimal distance of two points from different optimal clusters is realized for clusters that differ only in the first coordinate. The two light points from two such clusters have a distance of $\sqrt{2} - 1 - (-\sqrt{2} - 1) \geq 0.82$. Points from two clusters that differ in any other coordinate than the first one have a distance of at least $2z_2 = \sqrt{2} \geq 1.4$. Thus, the exponential lower bound example satisfies the strict separation property.

Next we discuss the AS-center separation property for the instances from the lower bound in Theorem 2.19. Let the matrices A and C have the same meaning as in the definition of the AS-center separation property, where C encodes the optimal k-means clustering. In order to apply Definition 2.41, we replace each heavy point with $m \to \infty$ many unweighted points at the same location. To keep the description simple, we will also call these unweighted points heavy points. in the following. Since the number m of heavy points goes to infinity, the cluster centers coincide with the locations of the heavy points. Hence, there are only two types of rows in the matrix $A - C$: the rows for heavy points have only zero entries and the rows for light points have $\pm(2 - \sqrt{2})$ as first entry and zeros at all other positions. Let $a_1, \ldots, a_n \in \mathbb{R}$ denote the entries in the first columns of $A - C$. Then it follows from basic linear algebra that $||A - C||_F^2 = ||A - C||^2 = \sum_{i=1}^{n} a_i^2$. Furthermore, since $||A - C||_F^2$ coincides with the costs of the optimal k-means clustering C, we obtain$||A-C||_F^2 = 2^{d-1}(2-\sqrt{2})^2$. Every cluster i in the clustering $C_1,\dots,C_n$ contains $m+1$ points, one light and m heavy points. Hence in accordance with Definition 2.41, we obtain

$$\Delta_i = \frac{1}{\sqrt{|C_i|}}\min\{\sqrt{k}||A-C||, ||A-C||_F\}$$
$$= \frac{1}{\sqrt{m+1}}||A-C||_F = \frac{\sqrt{2^{d-1}(2-\sqrt{2})^2}}{\sqrt{m+1}}$$

for every $i \in [k]$. For $m \to \infty$, Δ_i becomes arbitrarily small. On the other hand, the distance between two different cluster centers is at least $2z_2 = \sqrt{2}$, which does not change with m. Hence, the condition $||\mu_i - \mu_j|| \geq c(\Delta_i + \Delta_j)$ for $i \neq j$ is satisfied for every constant c if m is chosen sufficiently large.

Corollary 2.53. *For any $\epsilon > 0$, there is a family of point sets $(P)_{d\in\mathbb{N}}$ with $P_d \subset \mathbb{R}^d$ which are ϵ-ORSS-separated, satisfy the strict separation property and the AS-center separation property, and where* $\mathrm{Ward}_k(P_d) \in \Omega((3/2)^d \cdot \mathrm{opt}_k(P_d))$ *for $k = 2^d$.*

It remains to study for which values of δ and α the lower bound construction satisfies δ-center separation and α-center proximity. For this we study again the idealized construction in which the heavy points have infinite weight and hence the centers of the optimal clustering coincide with the heavy points. The radius of each cluster is then the distance between its light and its heavy point, which is $2-\sqrt{2}$ for each cluster. The minimum distance between two centers is $2z_2 = \sqrt{2}$. Hence, the lower bound construction satisfies δ-center separation for any $\delta < \frac{\sqrt{2}}{2-\sqrt{2}} = 1+\sqrt{2}$. A short calculation shows that the minimum distance of a point from one cluster to a center of a different cluster is realized for clusters that differ only in the first coordinate and that it is $\sqrt{2}$. Hence, the lower bound construction also satisfies α-center separation for any $\alpha < \frac{\sqrt{2}}{2-\sqrt{2}} = 1+\sqrt{2}$.

Corollary 2.54. *There is a family of point sets $(P)_{d\in\mathbb{N}}$ with $P_d \subset \mathbb{R}^d$ that satisfy δ-center separation and α-center proximity for any $\delta < 1+\sqrt{2}$ and any $\alpha < 1+\sqrt{2}$ and where* $\mathrm{Ward}_k(P_d) \in \Omega((3/2)^d \cdot \mathrm{opt}_k(P_d))$ *for $k = 2^d$.*

Exponential Lower Bound with Finite Weights First we consider ORSS-separation. Assume we want to find a (2^d-1)-clustering. Then there exists a cluster containing two heavy points with weight m. The distance between two different heavy points is at least $2 \cdot z_2 = \sqrt{2}$. Thus, the 1-means cost of that cluster is at least $\frac{m \cdot m}{m+m} \cdot (\sqrt{2})^2 = m$. Hence, $\mathrm{opt}_{2^d-1} \geq m$. On the other side, we have seen in the case with infinite weights that $\mathrm{opt}_{2^d} \leq 2^d \cdot (2-\sqrt{2})^2 + 1$ if we choose $m+1 \geq 4 \cdot 6^{d-1}$. Hence, also in the case of finite weights, ORSS-separation follows for any $\epsilon > 0$ by choosing m larger by $m > \frac{2^d \cdot (2-\sqrt{2})^2+1}{\epsilon^2}$.

Now we come to the strict separation property. The distance between a light point and its closest heavy point is $2-\sqrt{2}+\epsilon \leq 0.59+\epsilon$ with $\epsilon = \frac{2^d}{m}\frac{3^{d-2}}{2^{d-1}} = \frac{2}{m}3^{d-2}$. Notice that ϵ tends to zero for large values of m. The minimal distance of two points from different optimal clusters is realized for clusters that differ only in the first coordinate. The two light points from two such clusters have a distance of $\sqrt{2}-1-(-\sqrt{2}-1) \geq 0.82$. Points from two clusters that differ in any other coordinate than the first one have a distance of at least $2z_2 = \sqrt{2} \geq 1.4$. Thus, the exponential lower bound example satisfies the strict separation property by choosing m large enough.

Last we consider the AS-center separation property for the lower bound instances with finite weights. We calculate more carefully the rows of the matrix $A - C$. The optimal clusters consist of one heavy point with weight m and one light point. The first coordinate of the centroid of the i-th optimal cluster is given by $\mu_{i,1} = \pm \frac{m\cdot(1+\epsilon)+(\sqrt{2}-1)}{m+1}$. The other coordinates equal the corresponding coordinates of the points in the corresponding optimal cluster. Thus, the rows of the heavy points have $\pm(1+\epsilon-\mu_{i,1})$ as first entry and zeros at all other positions. According to that the rows for the light points have $\pm(\mu_{i,1} - \sqrt{2} + 1)$ as first entry and zeros at all other positions. Let $a_1, \ldots, a_n \in \mathbb{R}$ denote the entries in the first columns of $A - C$. Since all other entries equal zero, we obtain like in the case with infinite weights that $||A-C||_F^2 = ||A-C||^2 = \sum_{i=1}^n a_i^2$. Again, since $||A-C||_F^2$ coincides with the costs of the optimal k-means clustering we get that $||A-C||_F^2 \leq 2^d \cdot (2-\sqrt{2})^2 + 1$ for $m+1 \geq 4 \cdot 6^{d-1}$. We obtain

$$\Delta_i = \frac{\sqrt{2^d \cdot (2-\sqrt{2})^2 + 1}}{\sqrt{m+1}}$$

for $i \in [k]$. If m tends to infinity, Δ_i becomes arbitrarily small. Like in the instance with infinite weights, the distance between two different cluster centers is at least $2z_2 = \sqrt{2}$ independently from m. Hence, the condition $||\mu_i - \mu_j|| \geq c(\Delta_i + \Delta_j)$ for $i \neq j$ is satisfied for every constant c if m is chosen sufficiently large.

2.5 Complete Linkage

In this section we analyze the approximation factor of the Complete Linkage Algorithm. Our analysis gives an upper bound how the approximation factor may increase by the execution of k-merge steps by **CL**. This matches a result in [2] which states an upper bound on the approximation factor or the first $n-2k$ steps. In Section 2.5.1 we introduce formally the concept of clustering intersection graphs and prove some elementary properties. In Section 2.5.2 we provide a short analysis of the one-dimensional case before we consider a more general case under the assumption of the existence of certain subgraphs in Section 2.5.3. In Section 2.5.3 we combine our analysis with the result of Ackermann et al. about the first phase to prove that the complete-linkage method yields an $O(1)$-approximation.

2.5.1 Clustering Intersection Graphs

Our analysis is based on studying the clustering intersection graph induced by **CL** at certain points of time. Before we introduce the concept of clustering intersection graphs formally, we will define these points of time. Let $P \subseteq \mathbb{R}^d$ be arbitrary but finite and let $\mathcal{O}_k$ denote some arbitrary optimal k-clustering of P (w.r.t. the chosen objective function diameter or (discrete) radius). By scaling our point set we may assume that the objective value of $\mathcal{O}_k$ equals 1. We define $\mathrm{t}_{\leq x}$ to be the last step before some cluster reaches an objective value (w.r.t. the chosen objective) of more than x and denote the clustering of **CL** at time $\mathrm{t}_{\leq x}$ by $\mathcal{A}_x$. The following lemma is crucial for our analysis.

Lemma 2.55. *Let $x > 0$. In $\mathcal{A}_x$ there do not exist two clusters a_1 and a_2 such that*

$$\operatorname{diam}(a_1) + \operatorname{dist}(a_1, a_2) + \operatorname{diam}(a_2) \le x, \text{ for } \boldsymbol{CL}^{\text{diam}},$$
$$\operatorname{rad}(a_1) + \operatorname{dist}(a_1, a_2) + 2\operatorname{rad}(a_2) \le x, \text{ for } \boldsymbol{CL}^{\text{rad}},$$
$$\operatorname{drad}(a_1) + \operatorname{dist}(a_1, a_2) + 2\operatorname{drad}(a_2) \le x, \text{ for } \boldsymbol{CL}^{\text{drad}},$$

where $\operatorname{dist}(a_1, a_2)$ *is defined as the minimum distance between two points $p_1 \in a_1$ and $p_2 \in a_2$.*

Proof. Let a_1 and a_2 be clusters that satisfy the stated inequality. Clearly we have $\operatorname{diam}(a_1 \cup a_2) \le \operatorname{diam}(a_1) + \operatorname{dist}(a_1, a_2) + \operatorname{diam}(a_2) \le x$ because of the triangle inequality. But that means $\mathbf{CL}^{\text{diam}}$ will not merge two clusters obtaining a cluster of diameter larger than x in the next step, contradicting the definitions of $\mathrm{t}_{\le x}$ and $\mathcal{A}_x$.

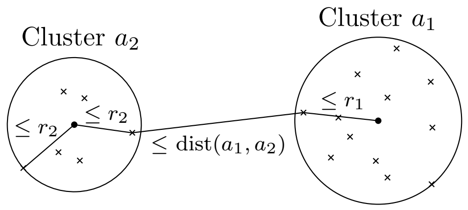


Figure 2.15: All points are contained in a cluster of radius at most $r_1 + \operatorname{dist}(a_1, a_2) + 2r_2$.

For the objective functions rad and drad we use an analogous argument based on the inequalities $\operatorname{rad}(a_1 \cup a_2) \le \operatorname{rad}(a_1) + \operatorname{dist}(a_1, a_2) + 2\operatorname{rad}(a_2)$ and $\operatorname{drad}(a_1 \cup a_2) \le \operatorname{drad}(a_1) + \operatorname{dist}(a_1, a_2) + 2\operatorname{drad}(a_2)$ (see Fig. 2.15). □

This implies that if we have at $\mathrm{t}_{\le x}$ two clusters $a_1, a_2 \in \mathcal{A}_x$ and some cluster $o \in \mathcal{O}_k$ with $a_1 \cap o \ne \emptyset$ and $a_2 \cap o \ne \emptyset$, then at $\mathrm{t}_{\le 2x+1}$ or $\mathrm{t}_{\le 3x+1}$ (depending on the objective function) not both a_1 and a_2 can be present anymore, i.e., either a_1 or a_2 or both were merged.

The fact that we can guarantee for certain pairs of clusters that one of them is merged at a certain point of time motivates us to define a clustering intersection graph (which is in general a hypergraph) with the clusters from $\mathcal{A}_x$ as vertices, where two vertices are neighbored if and only if there exists a cluster $o \in \mathcal{O}_k$ with which both have a non-empty intersection.

Definition 2.56. *Let $\mathcal{O}_k$ be an optimal k-clustering of some finite point set $P \subseteq \mathbb{R}^d$. Let $\mathcal{A}_x$ be the clustering of P computed by $\boldsymbol{CL}$ at time $\mathrm{t}_{\le x}$. We define the* clustering intersection graph (CI-graph) *$G_x = G_x(\mathcal{A}_x, \mathcal{O}_k)$ at point of time $\mathrm{t}_{\le x}$ as a hypergraph with vertex set $\mathcal{A}_x$. A set of vertices $N = \{v_1, \dots, v_\ell\}$ forms a hyperedge if there exists some cluster $o \in \mathcal{O}_k$ such that for each cluster v_i we have that $v_i \cap o \ne \emptyset$ and furthermore there does not exist a cluster $v \notin N$ with $v \cap o \ne \emptyset$.*

In general, the CI-graph is a hypergraph with exactly k edges and $|\mathcal{A}_x|$ vertices. Figure 2.16 gives an example of a CI-graph. If a statement holds for arbitrary points of time or the point of time is clear from context we omit the index x and just write G. Note that for each cluster $a \in \mathcal{A}_x$ each point $p \in a$ in the cluster is contained in some optimal

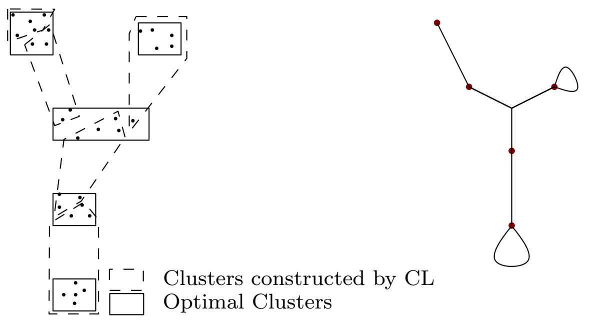


Figure 2.16: Example of a clustering instance with an optimal clustering and a clustering computed by **CL** (left side) and the corresponding CI-graph (right side).

cluster o. Thus, the CI-graph does not contain isolated vertices where isolated means that the vertex has no incident edge. We call a vertex ℓ a *leaf* if ℓ is incident to exactly one edge e and moreover ℓ is not the only vertex incident to e. Moreover an edge e is called a *loop* if e is only incident to one vertex. We define the *degree* of a vertex v to be number of non-loop edges that contain v plus twice the number of loops that consist of v.

The CI-graph has the crucial property that merging two clusters in $\mathcal{A}_x$ corresponds to contracting the corresponding vertices in the CI-graph. Assume that two clusters a_1 and a_2 are merged in a step of **CL**. Then all clusters $o \in \mathcal{O}$ that have a nonempty intersection with a_1 or a_2 clearly have a nonempty intersection with $a_1 \cup a_2$. Let G and G' denote the CI-graph before and after this merge operation, respectively. Then it is easy to see that G' is obtained from G by contracting the two vertices v_1 and v_2 corresponding to a_1 and a_2. The vertex that results from this contraction is incident to exactly those edges that were incident to v_1 or v_2 before. In particular, any edge $\{v_1, v_2\}$ becomes a loop that consists of the new vertex.

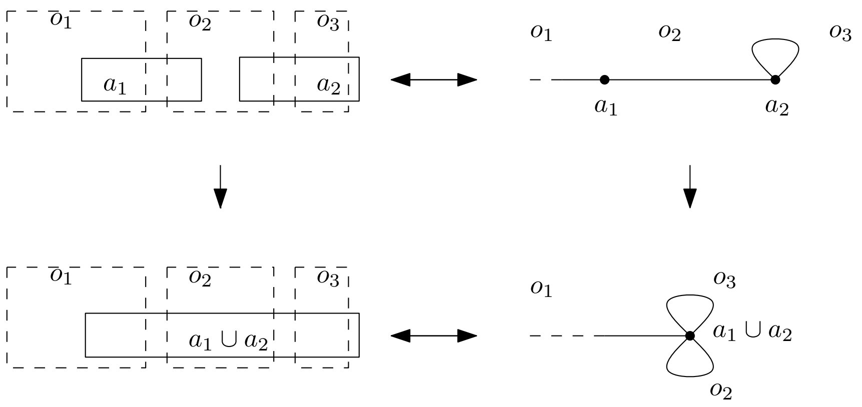


Figure 2.17: Example of a merge step computed by **CL** (left side) and the corresponding merge step of vertices in the CI-graph (right side).

To prove that the approximation factor of **CL** is at most x, it is sufficient to show that at time $\mathrm{t}_{\leq x}$ the CI-graph G_x contains at least as many edges as vertices. Clearly this is equivalent to $|\mathcal{A}_x| \leq k$, which means that **CL** has terminated.

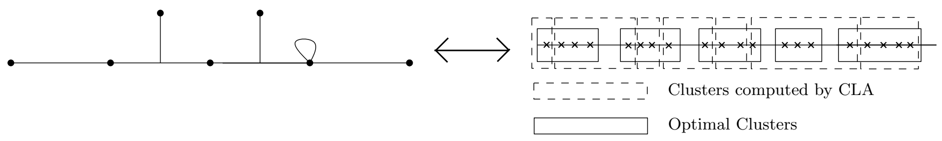


Figure 2.18: Each component of G_1 consists of vertices and edges of the above form.

2.5.2 The One-Dimensional Case

In this section we prove that **CL** yields a constant approximation factor for all finite point sets $P \subseteq \mathbb{R}$, all metrics $\mathrm{dist}\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ and all $k \in \mathbb{N}$ using the CI-graph. The result is known for the diameter k-clustering problem and the k-center problem [2]. Our result also holds for the discrete k-center problem and it demonstrates how the analysis of **CL** benefits from considering the CI-graph at certain points of time $\mathrm{t}_{\leq x}$.

Theorem 2.57. *For $d = 1$ and arbitrary k,*

$\boldsymbol{CL}^{\mathrm{diam}}$ computes a 3-approximation for the diameter k-clustering problem,

$\boldsymbol{CL}^{\mathrm{rad}}$ computes a 5-approximation for the k-center problem,

$\boldsymbol{CL}^{\mathrm{drad}}$ computes a 5-approximation for the discrete k-center problem.

Proof. At the beginning, we start with the CI-graph G_0 with vertex set $V = P$ and we have k hyperedges, each of which corresponds to a cluster $o \in \mathcal{O}_k$. Moreover in dimension one we have the additional property that at any point of time only clusters can be merged that are neighbored on the line. At $\mathrm{t}_{\leq 1}$ there do not exist two clusters $a_1, a_2 \in \mathcal{A}_x$ such that $a_1, a_2 \subseteq o$ for some $o \in \mathcal{O}_k$ because such clusters could be merged to form a new cluster with size at most 1, contradicting the definition of $\mathrm{t}_{\leq 1}$. Note that this is true for all objective functions. This implies the following claim.

Claim 2.58. *Each hyperedge in G_1 contains at most one leaf and hence, each hyperedge contains at most 3 vertices.*

Figure 2.18 shows an example that contains essentially all possibilities how hyperedges in G_1 can look like. In the following, we call hyperedges with 3 incident vertices *true hyperedges.* Observe that the number of incident vertices of the hyperedge that belongs to a certain cluster of $\mathcal{O}_k$ cannot increase during the course of **CL**.

Next we show the following claim.

Claim 2.59. *Let e_1 and e_2 be hyperedges that are incident to a common vertex and that contain each a unique leaf at $\mathrm{t}_{\leq 1}$. Then either e_1 or e_2 or both contain no leaves at $\mathrm{t}_{\leq 2}$ for the objective function* diam *and at $\mathrm{t}_{\leq 3}$ for the objective functions* rad *and* drad

Proof. Assume that there are two hyperedges e_1 and e_2 that are incident to a common vertex and that contain each a unique leaf ℓ_1 and ℓ_2, respectively, at $\mathrm{t}_{\leq 1}$. Then there exists a unique common vertex v. The cluster corresponding to v must be directly between the clusters corresponding to ℓ_1 and ℓ_2 on the line. This implies that as long as ℓ_1 and ℓ_2 are not merged, also v cannot be merged. Merging v and ℓ_i, for $i \in \{1, 2\}$, results in a cluster of diameter at most 2 because each vertex at $\mathrm{t}_{\leq 1}$ has diameter at most one and since leaves are fully contained in the incident optimal cluster and the optimal cluster also has at most diameter 1 (for an example see Figure 2.19).

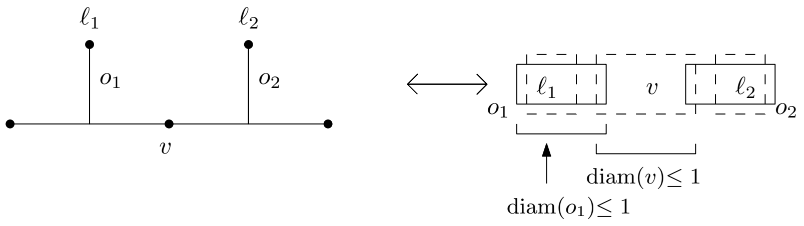


Figure 2.19: The cluster $\ell_1 \cup v$ has at most diameter 2.

Moreover we can argue that for the objective functions drad and rad merging v and ℓ_i, for $i \in \{1,2\}$, results in a cluster of (discrete) radius at most 3. Indeed, the clusters o_i can be covered by a ball with (discrete) radius 1. Moreover at $t_{\leq 1}$ also the cluster v can be covered by a ball with (discrete) radius 1 and thus has a diameter of at most 2. Additionally v has a common point with o_i. Thus, if we increase the radius of the ball covering o_i by 2 then v is also covered (for an example see Figure 2.20).

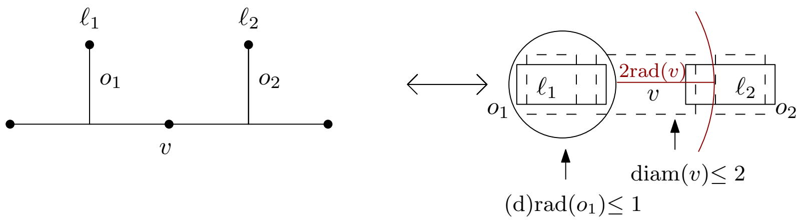
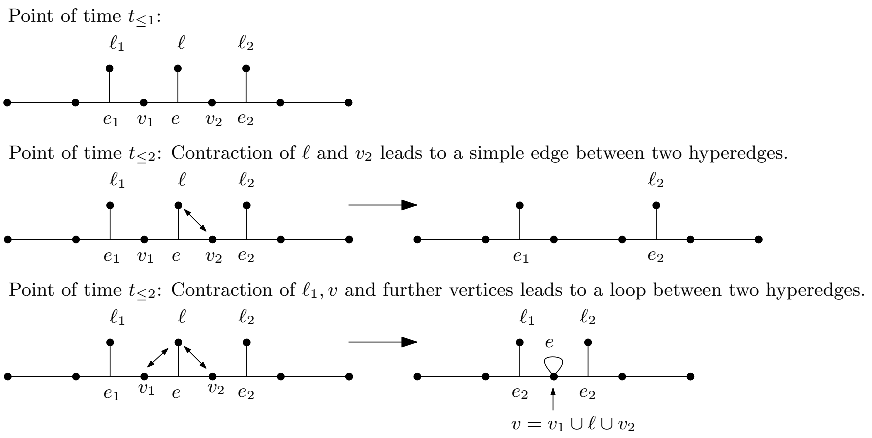


Figure 2.20: The cluster $\ell_1 \cup v$ has at most (discrete) radius 3.

From this observation we may conclude that either e_1 or e_2 or both contain no leaves at $\mathrm{t}_{\leq 2}$ for the objective function diam and at $\mathrm{t}_{\leq 3}$ for the objective functions rad and drad: If v is merged, then v is merged with ℓ_1 or ℓ_2. If w.l.o.g. v is merged with ℓ_1 then e_1 contains no longer a leaf and thus is either a simple edge or a loop. If v was not merged at $\mathrm{t}_{\leq 2}$ (or $\mathrm{t}_{\leq 3}$, respectively) then clearly ℓ_1 and ℓ_2 were both merged with other clusters and e_1 and e_2 do not contain leaves. □

Claim 2.60. *For* diam, *at* $\mathrm{t}_{\leq 2}$ *two hyperedges that each contain a leaf either cannot be neighbored or the common vertex is incident to a loop. The same is true for* rad *and* drad *at* $\mathrm{t}_{\leq 3}$ *instead of* $\mathrm{t}_{\leq 2}$.

Proof. For the objective function diam consider at time $\mathrm{t}_{\leq 2}$ two hyperedges e_1 and e_2 that each contain a unique leaf ℓ_1 and ℓ_2, respectively. If e_1 and e_2 contain a common vertex v at time $\mathrm{t}_{\leq 2}$, then from the previous argument it follows that v does not yet exist at time $\mathrm{t}_{\leq 1}$. It is created between $\mathrm{t}_{\leq 1}$ and $\mathrm{t}_{\leq 2}$ by some merge operations. In these merge operations there must be two vertices v_1 and v_2 involved that are contained in e_1 and e_2, respectively, at time $\mathrm{t}_{\leq 1}$. In particular v_1 and v_2 are no leaves, otherwise e_1 or e_2, respectively, would have contained 2 leaves at $\mathrm{t}_{\leq 1}$. Since all vertices on the line between v_1 and v_2 are contracted, their incident edges became a loop incident to v at $\mathrm{t}_{\leq 2}$. Hence, at $t_{\leq 2}$ two hyperedges containing a leaf either cannot be neighbored or the common vertex is incident to a loop (for an example see Figure 2.21). The same is true for rad and drad for $t_{\leq 3}$ instead of $t_{\leq 2}$. □



Point of time $t_{\leq 1}$:
ℓ_1 ℓ ℓ_2
e_1 v_1 e v_2 e_2
Point of time $t_{\leq 2}$: Contraction of ℓ and v_2 leads to a simple edge between two hyperedges.
ℓ_1 ℓ ℓ_2
e_1 v_1 e v_2 e_2
ℓ_2
e_1 e_2
Point of time $t_{\leq 2}$: Contraction of ℓ_1, v and further vertices leads to a loop between two hyperedges.
ℓ_1 ℓ ℓ_2
e_1 v_1 e v_2 e_2
ℓ_1 e ℓ_2
e_2 e_2
$v = v_1 \cup \ell \cup v_2$

Figure 2.21: Depending on how many vertices are contracted between $t_{\leq 1}$ and $t_{\leq 2}$ we have that at $t_{\leq 2}$ between two true hyperedges there exists at least a simple edge or a loop.

Claim 2.61. *For* diam *at* $t_{\leq 3}$ *all neighbors of leaves are incident to some loop. The same is true for* rad *and* drad *at* $t_{\leq 5}$ *instead of* $t_{\leq 3}$.

Proof. First consider the objective function diam. Assume that there exists some neighbor v_1 of a leaf ℓ_1 that is not incident to a loop at $t_{\leq 2}$. Since v_1 is no leaf it is incident to another hyperedge e_2 (otherwise e_1 contains 2 leaves which is a contradiction to $t_{\leq 1}$) and since v_1 is not incident to a loop at $t_{\leq 2}$ we know that e_2 contains no leaf by the previous claim. Using the argument shown in Figure 2.19 we know that at $t_{\leq 3}$ either ℓ_1 or v_1 has been merged. If ℓ_1 was merged, e_1 has no leaves at $t_{\leq 3}$. If ℓ_1 has not been merged at $t_{\leq 3}$ then v_2 has and therefore was merged with its other neighbor contained in e_2. But then a loop was built. The same is true for rad and drad for $t_{\leq 5}$ instead of $t_{\leq 3}$. □

Finally, we argue that at $t_{\leq 3}$ (or $t_{\leq 5}$) the number of vertices is bounded from above by k, which equals the number of edges in the hypergraph $G_3(G_5)$. We prove this bound for each connected component separately. Consider some arbitrary connected component C of $G_3(G_5)$. If C is a path we have clearly $|E(C)|+1$ vertices in C. For each true hyperedge in C, there exists one additional leaf. We will argue, that if we have j true hyperedges in C, then C contains $j+1$ loops. Thus, we have at least as many edges as vertices. Indeed, assume that there are j true hyperedges left at $t_{\leq 3}$ (or $t_{\leq 5}$). Each of them contains exactly one leaf. The right neighbor of this leaf is then incident to some loop at $t_{\leq 3}$ (or $t_{\leq 5}$) as argued above. Thus, there are at least j loops in the edge set. Moreover consider the leftmost hyperedge with its leaf ℓ. The left neighbor of ℓ is also incident to some loop. But then we have at least $j+1$ loops. Note that if $j=0$ the arguments above still hold for the leaf of the leftmost edge in a component. Thus, we have at least one loop at $t_{\leq 3}$ (or $t_{\leq 5}$).

□

2.5.3 The General Case

In this section, we prove that **CL** computes an $O(1)$-approximation for any constant dimension $d \in \mathbb{N}$. In the following, we assume that $x \geq 1$ is fixed arbitrarily and we analyze the CI-graph $G = G_x$ and how it changes over time. Later we will choose x such that G_x contains at most $2k$ nodes.

Completion of the CI-Graph

In the one-dimensional case one has the crucial property that all vertices of a CI-graph can be arranged in increasing order on a line such that only neighbored vertices on the line may be contracted. Additionally, it follows from Lemma 2.55 that at least one vertex of every neighbored pair must be contracted until a certain time step. This implies that each edge is incident to at most 3 vertices at $\mathrm{t}_{\leq 1}$, which is essential in the proof of Theorem 2.57. This property is not true anymore in higher dimensions.

Given a CI-graph $G = G_x$, we construct a weighted graph $\Gamma(G)$, which we call the *completion* of G. The graph $\Gamma(G)$ has the same vertex set as the CI-graph G. For every hyperedge $\{v_1, \dots, v_\ell\}$ in G, we introduce a clique with edge weights 1 in $\Gamma(G)$. In particular, for $\ell = 1$ we introduce a loop at node v_1 with weight 1.

Definition 2.62. *Let $G = (V, E)$ be a hypergraph and let $v, w \in V$ be two vertices in the same connected component of G. A* path *p between v and w is a sequence $v = v_0, \dots, v_\ell = w$ of vertices such that v_i and v_{i+1} are adjacent in G for all $i \in \{0, \dots, \ell - 1\}$. We call ℓ the* length *of path p. A* shortest *v-w-path in G is a v-w-path p_s such that no v-w-path p with length less than the length of p_s exists.*

For each pair of vertices v and w from the same connected component that are not adjacent, we add an edge (v, w) to $\Gamma(G)$. If p denotes the length of a shortest v-w-path in G then the weight of the edge (v, w) in $\Gamma(G)$ is set to $p + (p-1)x$ for the objective function diam and $p + (p-1)2x$ for the objective functions rad and drad. The next lemma shows that this construction ensures the following important property: the weight of every edge (v, w) in $\Gamma(G)$ is an upper bound for the distance of the corresponding clusters (remember that the distance of two clusters is defined as the smallest distance between any pair of points from these clusters).

Lemma 2.63. *Assume that a v-w-path in a CI-graph G has length p. Then the smallest distance between two points in v and w is at most $p + (p-1)x$ for the objective function* diam *and $p + (p-1)2x$ for the objective functions* rad *and* drad.

Proof. Let $P = v, v_1, \dots, v_{p-1}, w$ be a v-w-path of length p in the CI-graph G. Due to the definition of G, we can find a sequence of points (for a geometric intuition see Figure 2.22) $s_1, t_1, s_2, t_2, \dots, s_p, t_p$ with the properties

i) $s_1 \in v$ and $t_p \in w$,

ii) $t_i, s_{i+1} \in v_i$ for $1 \leq i \leq p-1$,

iii) $\mathrm{dist}(s_i, t_i) \leq 1$ for $1 \leq i \leq p$.

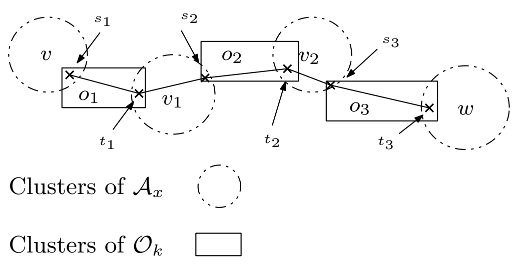


Figure 2.22: We can choose points in the clusters corresponding to the path in G.

For the objective function diam, we can additionally ensure

iv) $\operatorname{dist}(t_i, s_{i+1}) \le x$ for $1 \le i \le p-1$

while for the objective functions rad and drad, we can only additionally ensure

iv') $\operatorname{dist}(t_i, s_{i+1}) \le 2x$ for $1 \le i \le p-1$.

From this the claim follows by the triangle inequality. □ □

Our analysis is based on studying subgraphs of $\Gamma(G)$ that satisfy certain properties. The following lemma shows how subgraphs change during the course of **CL**. In order to state the lemma, let us define what we mean by contracting two nodes v and w in a multi-graph. It means that v and w get replaced by a new node and that all edges that were incident to v or w are now incident to the new node instead. In particular, all edges between v and w become loops. Hence, contraction operations do not change the total number of edges in the graph, and each such operation reduces the number of vertices by one.

Lemma 2.64. *Let G_x be a CI-graph at some point of time $\mathrm{t}_{\le x}$, and let H_x be a subgraph of $\Gamma(G_x)$ with $V(H_x) = V(G_x)$. Now consider the CI-graph $G_{x'}$ for some point of time $\mathrm{t}_{\le x'}$ with $x' > x$. Let $H_{x'}$ be the multi-graph that arises from H_x by performing the same contractions that are made between G_x and $G_{x'}$. Then $V(G_{x'}) = V(H_{x'})$ and moreover the weight of any edge (v, w) in $H_{x'}$ is an upper bound for the distance of the clusters corresponding to v and w.*

Proof. By definition of $H_{x'}$ it is clear that $V(H_{x'}) = V(G_{x'})$. Let $e = (v, w)$ be an arbitrary edge in $H_{x'}$. Then there exist vertices $v_1, \dots, v_{n_1} \in V(H_x)$ and $w_1, \dots, w_{n_2} \in V(H_x)$ that were contracted to v and w, respectively, between H_x and $H_{x'}$. Since $e = (v, w)$ is an edge in $H_{x'}$, there exist two vertices v_i and w_j such that the edge $e' = (v_i, w_j)$ is contained in H_x and has the same weight as the edge e. Hence, according to Lemma 2.63 the distance between v_i and w_j is at most the weight of edge e'. Since the distance between two clusters is defined as the smallest distance between any pair of points from these clusters, this implies that also the distance between v and w is bounded from above by the weight of edge e', which equals the weight of edge e. □ □

Subgraphs at Different Points of Time

Now we study CI-graphs that contain certain subgraphs. Assume that there exists a subgraph H_x of $\Gamma(G_x)$ that satisfies the following properties:

i) $V(H_x) = V(G_x)$,

ii) $|E(H_x)| \leq k$,

iii) no vertex in H_x is isolated (i.e., every vertex in H_x has at least one incident edge, which might also be a loop).

Let δ denote the largest edge weight in H_x. By Lemma 2.64, δ is an upper bound for the distance between any pair of clusters that are adjacent in H_x. For $i \in \mathbb{N}_0$, we will analyze time steps $\mathrm{t}_{\leq x+i(\delta+x)}$ for the diameter k-clustering problem and $\mathrm{t}_{\leq x+i(\delta+2x)}$ for the k-center and discrete k-center problem and denote them by t_i. In accordance to that, we define $x_i = x + i(\delta + x)$ for $\mathbf{CL}^{\mathrm{diam}}$ and $x_i = x + i(\delta + 2x)$ for $\mathbf{CL}^{\mathrm{rad}}$ and $\mathbf{CL}^{\mathrm{drad}}$, respectively.

Lemma 2.65. *If there exists a subgraph H_x of $\Gamma(G_x)$ that satisfies properties i), ii), and iii), then G_{t_4}, G_{t_7}, and G_{t_3} contain at most k nodes for $\boldsymbol{CL}^{\mathrm{diam}}$, $\boldsymbol{CL}^{\mathrm{drad}}$, and $\boldsymbol{CL}^{\mathrm{rad}}$, respectively.*

Under the assumption that a subgraph H_x with properties i), ii), and iii) exists, Lemma 2.65 implies that the approximation ratio of **CL** is bounded by x_4, x_7 and, x_3, respectively. Hence, it is constant if both x and δ are constant. In order to prove the lemma, we will prove that H_{t_4}, H_{t_6}, and H_{t_3}, respectively, contain at least as many edges as vertices. As the number of edges is at most k and $V(H_{t_i}) = V(G_{t_i})$, this proves the lemma.

In the following we denote $H_{x'}$ by H if the point of time is clear from context or if a statement holds for all $H_{x'}$ with $x' \geq x$. First note that H is a multi-graph. Multi-graphs have the crucial property that a connected component has at least as many edges as vertices if and only if it contains a cycle (where a loop is considered as a cycle).

Definition 2.66. *We call a connected component of H* tree-component *if the component is a tree.*

Observation 2.67. *If $H_{x'}$ has no tree-component, then $H_{x'}$ contains at most k nodes.*

Leaves of H and their neighbors play a key role in the analysis of the algorithm. We will show that between certain time steps either a leaf or its unique neighbor is merged.

Definition 2.68. *We call a vertex $p \in H$ in a tree-component of H a* leaf-parent *if p is the neighbor of some leaf and has at least degree* 2.

At the beginning of our analysis in $H_x = H_{x_0}$ there does not necessarily exist a leaf-parent in each tree-component because there could be tree-components that consist only of two vertices that are connected by an edge. These are the only possible tree-components without a leaf-parent (remember that in H there exist no isolated vertices by property iii); any connected component that consists of a single vertex must contain a loop). Furthermore it follows easily that any other tree-component does not only contain a leaf-parent but that the unique neighbor of every leaf is a leaf-parent. Analogously to dimension one we show that at point of time t_1 for each tree-component by **CL** either one vertex was merged with a vertex from another component and thereby some vertex with degree 2 is built or two vertices from one component were merged. The latter means that a cycle was built and the component is no longer a tree.

Lemma 2.69. *Each tree-component C of H that contains a vertex v of degree 2 contains at least one leaf-parent p. Furthermore H_{x_1} contains at least one leaf-parent in each tree-component.*

Proof. Since C is a tree-component, we know that C is a tree. If v is a leaf-parent itself then we are done. Otherwise v has no leaf as a neighbor. Thus $C \setminus \{v\}$ defines two trees T_1 and T_2 that each contain at least two vertices and one leaf. In particular, we can choose leaves $\ell_1 \in T_1$ and $\ell_2 \in T_2$ that are no neighbors of v. Let $\ell_1, p_1, \ldots, v, \ldots, p_2, \ell_2$ be the unique ℓ_1-ℓ_2-path in C. Then p_1 and p_2 have degree at least 2 and a neighbored leaf. Therefore both vertices are leaf-parents.

Using the same arguments as for the one-dimensional case, we can argue that at time $\mathrm{t}_{\leq 1}$ each tree-component contains a vertex with degree at least 2: We have argued above that the only tree-component for which this is not the case consists of two connected vertices and it must have been present at $\mathrm{t}_{\leq x}$ already. Hence the clusters corresponding to these vertices have both objective value at most x and their distance is at most δ. According to Lemma 2.55 and the definition of x_1 this implies that one of these clusters must have been merged at time t_1. □ □

The proof of Lemma 2.69 gives a hint that we have in most cases at least two leaf-parents in each tree-component while components with exactly one leaf-parent are of a special form. We will use this structure later on to prove that if each tree-component contains at least 2 leaf-parents then the algorithm terminates. For this we need some statement counting the number of remaining contractions depending on the number of leaf-parents. First, we need some statement how often contraction steps are performed in each component.

Lemma 2.70. *Let ℓ be some leaf in H_{x_i} at an arbitrary point of time t_i with $i \geq 0$. Then the leaf ℓ is also contained in H_{x_0} and it is not contracted between t_0 and t_i. Moreover between two steps of time t_i and t_{i+1} where $i \in \mathbb{N}$ we have that for each leaf ℓ either the leaf ℓ or its corresponding leaf-parent p_ℓ is contracted.*

Proof. We do not have any vertices with degree 0. Thus, a vertex is a leaf if and only if it has degree 1. Moreover by the contraction of two vertices the degree of the contracted vertex equals the sum of the degrees of the two vertices contracted. Since both vertices have degree at least 1 the contracted vertex has degree at least 2 and is therefore no leaf.

To prove the second claim we note that the distance between any leaf and its leaf-parent is at most δ because δ is an upper bound for the weight of any edge in H_{x_i} for any $i \in \mathbb{N}_0$. Since all leaves in H_{x_i} for any $i \in \mathbb{N}_0$ are already contained in H_{x_0}, they have an objective value of at most $x = x_0$. Moreover each leaf-parent p_ℓ has an objective value of at most x_i at t_i. But that means for a leaf ℓ and the corresponding leaf-parent p_ℓ we have for $\mathbf{CL}^{\text{diam}}$ that

$$\begin{aligned}\operatorname{diam}(p_\ell) + \delta + \operatorname{diam}(\ell) &\leq x_i + \delta + x \\ &\leq x + i(\delta + x) + \delta + x \\ &\leq x + (i+1)(\delta + x) = x_{i+1},\end{aligned}$$

and for $\mathbf{CL}^{\text{rad}}$ (analogously for $\mathbf{CL}^{\text{drad}}$) we get that

$$\begin{aligned}\operatorname{rad}(p_\ell) + \delta + 2\operatorname{rad}(\ell) &\leq x_i + \delta + 2x \\ &\leq x + i(\delta + 2x) + \delta + 2x \\ &\leq x + (i+1)(\delta + 2x) = x_{i+1}.\end{aligned}$$

Thus by Lemma 2.55, ℓ and p_ℓ cannot be both present at $\mathrm{t}_{\leq x_{i+1}}$ anymore. $\square$ $\square$

We denote the number of leaf-parents of H_{x_i} at time t_i for a connected component C by $n_{\ell p}(C)$. Since in each tree-component the number of leaf-parents is at most the number of leaves, we may conclude that the algorithm performs at least $n_{\ell p}/2$ contractions between t_i and t_{i+1} where $n_{\ell p} = \sum_{i=1}^{r} n_{\ell p}(C_i)$ is the sum over the number of leaf-parents in the tree-components. Now we count the number of leaf-parents contained in one tree-connected component. The idea is that if each tree-component contains at least two leaf-parents then we have at least as many contractions as tree-components and can conclude that the algorithm will terminate. Therefore we show that at a certain point of time every tree-component must contain at least two leaf-parents. First we will show that if the number of leaf-parents in a tree-component is at least two, then after contraction the number of leaf-parents does not decrease below two.

Lemma 2.71. *Assume that two vertices v_1 and v_2 from two different components C_1 and C_2 that contain each at least one leaf-parent are contracted in H. If the resulting component $C = C_1 \cup C_2$ is a tree then C has at least as many leaf-parents as the maximum of C_1 and C_2, i.e., $n_{\ell p}(C) \geq \max\{n_{\ell p}(C_1), n_{\ell p}(C_2)\}$.*

Proof. Assume w.l.o.g. that the number of leaf-parents in C_1 is larger than or equal to the number of leaf-parents in C_2. We claim that $n_{\ell p}(C) \geq n_{\ell p}(C_1)$. If v_1 is no leaf, then by contraction all leaves in C_1 are still leaves in C and the number of leaf-parents will not decrease in C. Thus $n_{\ell p}(C) \geq n_{\ell p}(C_1)$. We may assume that v_1 is a leaf and moreover it is the only leaf neighbored to its leaf-parent (otherwise the leaf-parent still remains a leaf-parent after the contraction). By that choice of v_1 we ensure that after contraction C contains $n_{\ell p}(C_1) - 1$ leaf-parents of C_1.

There are three possibilities for the choice of v_2 (leaf, leaf-parent, or an inner node). If v_2 is a leaf-parent or an inner node, then all leaves in C_2 remain leaves and thus all leaf-parents of C_2 are leaf-parents in C. But then we have $n_{\ell p}(C) = n_{\ell p}(C_1) - 1 + n_{\ell p}(C_2) \geq n_{\ell p}(C_1)$ by assumption. Finally, we analyze the case where v_1 and v_2 are leaves and their leaf-parents have only one leaf as a neighbor. Let p_2 denote the leaf-parent corresponding to v_2. Since p_2 has degree two there exists another neighbor v. By assumption v is no leaf (otherwise we are done). But in that case, there exists at least a second leaf-parent $\tilde{p}$ in C_2.

Thus, $\tilde{p}$ remains a leaf-parent in C and again we have $n_{\ell p}(C) = n_{\ell p}(C_1) - 1 + n_{\ell p}(C_2) - 1 \geq n_{\ell p}(C_1)$, which proves the claim. $\square$ $\square$

We may conclude that the only possibility to obtain a tree-component with just one leaf-parent is that we contract vertices from two different components that each contain only one leaf-parent. In particular for two such components C_1 and C_2, we have to contract the leaf-parents p_1 and p_2. If another vertex and therefore a leaf of C_1 is contracted another component $C_1 \cup C_2$ with at least two leaf-parents is built.

Lemma 2.72. *For $\boldsymbol{CL}^{\mathrm{diam}}$ each tree-component contains at least 2 leaf-parents at point of time t_3. For $\boldsymbol{CL}^{\mathrm{rad}}$ each tree-component contains at least 2 leaf-parents at t_2. For $\boldsymbol{CL}^{\mathrm{drad}}$ each tree-component contains at least 2 leaf-parents at t_6.*

Proof. We have proven in Lemma 2.69 that at point of time t_1 each tree-component contains at least one leaf-parent. By Lemma 2.71 this is also true for any point of time after t_1. Let $i \in \mathbb{N}$ and assume that there exists a tree-component C at time t_i that has only one leaf-parent p_C. Again from Lemma 2.71 it follows that C was either already present at t_1 or that it was created by merging the leaf-parents of components that contained exactly one leaf-parent at t_1. This implies that C must contain two leafs ℓ_1 and ℓ_2 that were already leaves at t_1 and that were furthermore contained in the same component C' at t_1. From the discussion above it follows that the component C' contains at t_1 exactly one leaf-parent $p_{C'}$. Hence we can bound the diameter of $\ell_1 \cup \ell_2$ from above by

$$\begin{aligned}
\mathrm{diam}(\ell_1 \cup \ell_2) &\leq \mathrm{diam}(\ell_1) + \mathrm{dist}(\ell_1, \ell_2) + \mathrm{diam}(\ell_2) \\
&\leq \mathrm{diam}(\ell_1) + \delta + \mathrm{diam}(p_{C'}) + \delta + \mathrm{diam}(l_2) \\
&\leq x + \delta + (x + (\delta + x)) + \delta + x \\
&= x + 3(\delta + x) = x_3.
\end{aligned}$$

Again by Lemma 2.55 we may conclude that at t_3 either ℓ_1, ℓ_2 or both leaves were merged by $\mathbf{CL}^{\mathrm{diam}}$, contradicting the existence of C at time t_3.
Analogous to $\mathbf{CL}^{\mathrm{diam}}$ we can bound the discrete radius of $\ell_1 \cup \ell_2$ from above by

$$\begin{aligned}
\mathrm{drad}(\ell_1 \cup \ell_2) &\leq \mathrm{drad}(\ell_1) + \mathrm{dist}(\ell_1, \ell_2) + 2\,\mathrm{drad}(\ell_2) \\
&\leq \mathrm{drad}(\ell_1) + \delta + 2\,\mathrm{drad}(p_{C'}) + \delta + 2\,\mathrm{drad}(l_2) \\
&\leq x + \delta + 2(x + (\delta + x)) + \delta + 2x \\
&= x + 6(\delta + x) = x_6.
\end{aligned}$$

In case of $\mathbf{CL}^{\mathrm{rad}}$ we find the following upper bound. For each of the leaves ℓ_1 and ℓ_2 it holds that $p_{C'} \cup \ell_i$ is contained in a ball with radius $\mathrm{rad}(p_{C'}) + \delta + 2\,\mathrm{rad}(\ell_i)$ with $i \in \{1, 2\}$ around the center of $p_{C'}$. Thus, we may bound the radius of $\ell_1 \cup \ell_2$ from above by

$$\begin{aligned}
\mathrm{rad}(\ell_1 \cup \ell_2) &\leq \mathrm{rad}(p_{C'}) + \delta + 2\max\{\mathrm{rad}(\ell_1), \mathrm{rad}(\ell_2)\} \\
&\leq x_1 + \delta + 2x \leq x + (\delta + 2x) + \delta + 2x = x_2.
\end{aligned}$$

We may conclude that at $\mathrm{t}_{\leq x_2} = t_2$ either ℓ_1 or ℓ_2 or both were merged by $\mathbf{CL}^{\mathrm{rad}}$. $\square$ $\square$

It remains to prove that **CL** terminates if each component contains at least two leaf-parents.

Lemma 2.73. *If at t_i each tree-component of H_{x_i} contains at least two leaf-parents then* ***CL*** *has terminated at t_{i+1} (i.e., $H_{x_{i+1}}$ contains at most k nodes).*

Proof. Assume that at t_i there exist j tree-components, each of them containing at least two distinct leaf-parents p_ℓ and p'_ℓ. Clearly the sets of corresponding leaves are disjoint. According to Lemma 2.70 that means that for each connected component at least two vertices will be contracted up to point of time t_{i+1}. In each tree-component the number of vertices equals the number of edges plus one. Thus if E is the set of edges in H we have at most $|E| + j$ vertices. Moreover $2j$ vertices will be contracted. This requires at least j contractions and by each contraction the number of vertices decreases by one. Finally, after j contractions we have at most $|E|$ vertices and the algorithms terminates. □ □

Now we are ready to prove Lemma 2.65.

Lemma 2.65. It follows from Lemma 2.72 that each tree-component contains at least 2 leaf-parents at point of time t_3, t_2, and t_6 for $\mathbf{CL}^{\text{diam}}$, $\mathbf{CL}^{\text{rad}}$, and $\mathbf{CL}^{\text{drad}}$, respectively. Now Lemma 2.73 implies that H_{x_4}, H_{x_3}, and H_{x_7}, respectively, contain at most k nodes. □

Subgraphs with Small Edge Weights

Our goal in this section is to find a subgraph H_x of $\Gamma(G_x)$ that satisfies properties i)-iii) and whose maximum edge weight is small. Note that properties i), ii), and iii) imply $|V(G_x)| = |V(H_x)| \leq 2|E(H_x)| \leq 2k = 2|E(G_x)|$, which means $|V(G_x)| \leq 2|E(G_x)|$ is a necessary condition to find a subgraph H_x.

In the following we will assume that $|V(G_x)| \leq 2|E(G_x)|$ and that G_x is connected. We will prove that, under this assumption, we can always find a subgraph H_x of $\Gamma(G_x)$ that satisfies properties i)-iii) and has the following additional property:

iv) For each edge $e' = (v, w) \in E(H_x)$, the vertices v and w have distance at most 2 in G_x, i.e., either there is an edge $e \in E(G_x)$ with $\{v, w\} \subseteq e$ or there are two edges $e_v \in E(G_x)$ and $e_w \in E(G_x)$ with $v \in e_v$, $w \in e_w$, and $e_v \cap e_w \neq \emptyset$.

In accordance with the definition of $\Gamma(G_x)$, property iv) implies that the maximum edge weight δ in H_x is bounded from above by $2 + x$ for $\mathbf{CL}^{\text{diam}}$ and by $2 + 2x$ for $\mathbf{CL}^{\text{rad}}$ and $\mathbf{CL}^{\text{drad}}$. Using this we will prove that **CL** terminates at time $\mathrm{t}_{\leq O(x)}$ if for each connected component C of the CI-graph G_x we have that $|V(C)| \leq 2|E(C)|$.

In order to find a subgraph H_x of $\Gamma(G_x)$ that satisfies properties i)-iv) we let T be a spanning tree of $\Gamma(G_x)$ that uses only edges of weight 1. Such a spanning tree is guaranteed to exist because we assumed G_x to be connected. Such a spanning tree satisfies all properties except for ii) because the number of edges in T is $|V(G_x)| - 1$ and $|V(G_x)|$ can be up to $2k$.

However, any perfect matching in the spanning tree T is a subgraph H that satisfies the properties i)-iv). If T does not contain a perfect matching, we show how to find a perfect 2-matching (according to the following definition).

Definition 2.74. *An α-matching in a graph G is a matching M in the complete graph with vertex set $V(G)$ such that for each matching edge $(v, w) \in M$ the distance of v and w in G is at most α. Moreover we call an α-matching perfect if M contains for edge vertex from $V(G)$ an incident edge.*

Lemma 2.75. *Each tree T with an even number $|V(T)| \geq 2$ of vertices has a perfect 2-matching.*

Proof. We prove the claim by induction on the height of the tree. Since the tree T contains at least 2 vertices, its height is at least 1. If the height is exactly 1, we have some root r with an odd number $v_1, \ldots, v_n$ of sons. Then clearly $\{(r, v_1), (v_2, v_3), \ldots, (v_{n-1}, v_n)\}$ is a 2-matching.

Now assume we have some tree of height j with an even number of vertices. For each vertex v in layer $j-1$ let $v_1, \ldots, v_n$ be its children. We distinguish 3 different cases:
Case 1: $n = 0$. That means v has no sons and we do nothing.
Case 2: n is even. In that case we add the edges $(v_1, v_2), \ldots, (v_{n-1}, v_n)$ to M and delete all children from T.
Case 3: n is odd. In that case we add the edges $(v, v_1), (v_2, v_3), \ldots, (v_{n-1}, v_n)$ to M and delete all children and v from T.
Thus we end up with a tree of height $j-1$. Moreover we deleted in each case an even number of vertices from T. Thus the number of vertices in T remains even and hence we can apply the induction hypothesis. This proves the claim. □ □

We construct a graph H_x that satisfies the properties i), ii), iii), and iv) as follows. First we compute an arbitrary spanning tree T of $\Gamma(G_x)$ that uses only edges of weight 1. If $|V(G_x)| = |V(H_x)|$ is even, then the graph H_x is chosen as a perfect 2-matching of T. Then the properties i), iii), and iv) are satisfied by construction and property ii) is satisfied because of $|E(H_x)| = |V(H_x)|/2 \leq k$. If $|V(G_x)|$ is odd, we choose some leaf v from the spanning tree T. Then we find a perfect 2-matching M in $T \setminus \{v\}$. Since $|V(G_x)| \leq 2|E(G_x)|$ we have that the matching contains at most $|E(G_x)| - 1$ edges. Thus we set H_x to M and may add the edge from T that is incident to v to H_x such that property iii) becomes true.

Now we have a graph H_x fulfilling properties i), ii), iii), and iv). Property iv) and Lemma 2.64 imply that $\delta \leq 2 + x$ for the objective function diam and $\delta \leq 2 + 2x$ for the objective functions rad and drad. We conclude with the following theorem.

Theorem 2.76. *Assume that the CI-graph G_x is connected and contains k edges and at most $2k$ vertices at some point of time $\mathrm{t}_{\leq x}$. Then $\boldsymbol{CL}^{\mathrm{diam}}$ computes a $9x+8$ approximation for the diameter k-clustering problem. Moreover $\boldsymbol{CL}^{\mathrm{rad}}$ computes a $13x+6$ approximation for the k-center problem and $\boldsymbol{CL}^{\mathrm{drad}}$ computes a $29x+14$ approximation for the discrete k-center problem.*

Proof. We conclude from Lemma 2.65 that $\mathbf{CL}^{\mathrm{diam}}$ has terminated at t_4. In addition to that we can bound the diameter of clusters at t_4 from above by

$$x + 4(\delta + x) \leq x + 4(2 + x + x) \leq 9x + 8.$$

For the objective function discrete radius $\mathbf{CL}^{\mathrm{drad}}$ has terminated at t_6. Again we can bound the discrete radius of clusters at t_7 from above by

$$x + 7(\delta + 2x) \leq x + 7(2 + 2x + 2x) \leq 29x + 14.$$

Finally for the objective function radius $\mathbf{CL}^{\mathrm{rad}}$ has terminated at t_3. We can bound the radius of clusters at t_3 from above by

$$x + 3(\delta + 2x) \leq x + 3(2 + 2x + 2x) \leq 13x + 6. \qquad \square$$

$\square$

Approximation Factor of CL

In this section we combine our analysis with the result of Ackermann et al. [2] for the first phase of **CL** (i.e., the steps until $2k$ clusters are left) in order to prove the main theorem. From the analysis of Ackermann et al. it follows that there is a function κ such that for $x = \kappa(d)$ the CI-graph G_x contains at most $2k$ vertices. We consider the completion $\Gamma(G_x)$ of G_x and assume that it is connected. This is not necessarily the case but we will see later that this assumption is without loss of generality because our analysis can be applied to each connected component separately. In fact, the result of Ackermann et al. implies that for each connected component of G_x the number of vertices is at most twice the number of edges.

Now for each version of the algorithm $\mathbf{CL}^{\mathrm{diam}}$, $\mathbf{CL}^{\mathrm{rad}}$, and $\mathbf{CL}^{\mathrm{drad}}$ we combine our analysis with the special result of [2] corresponding to each of the methods. We state the following lemma from [2] deriving an upper bound for a point of time x where $|V(G_x)| \leq 2k$.

Lemma 2.77 ([2])**.** *Let $P \subseteq \mathbb{R}^d$ be finite. Then, for all $k \in \mathbb{N}$ with $2k \leq |P|$, the partition $\mathcal{A}$ of P into $2k$ clusters computed by $\boldsymbol{CL}^{\mathrm{drad}}$ satisfies*

$$\max_{a \in \mathcal{A}} \mathrm{drad}(a) < 20d \cdot \mathrm{drad}(\mathcal{O}_k^{\mathrm{drad}}).$$

Combining this result with Theorem 2.76 yields the following theorem.

Theorem 2.78. *For $d \in \mathbb{N}$ and a finite point set $P \subseteq \mathbb{R}^d$ the algorithm $\boldsymbol{CL}^{\mathrm{drad}}$ computes an $O(d)$-approximation for the k-center problem.*

Proof. Define $x = 20d$ and consider the point of time $\mathrm{t}_{\leq x}$. Then either $\mathbf{CL}^{\mathrm{drad}}$ has terminated in case when the number of clusters $\mathcal{A}_x$ is at most k. But then the theorem is proven.

Otherwise assume we have a CI-graph with connected components $C_1, \ldots, C_r$. By Lemma 2.77 we have that for each component C_i with k_i edges there are at most $2k_i$ vertices. In fact this is true since each connected component at $\mathrm{t}_{\leq x}$ can be seen as a single clustering instance and then one can apply Lemma 2.77 to each instance separately. But then we have a CI-graph of the form claimed in Section 2.5.3 and $\mathbf{CL}^{\mathrm{drad}}$ terminates after at most $O(x)$ steps according to Theorem 2.76. $\square$ $\square$

Lemma 2.79 ([2])**.** *Let $P \subseteq \mathbb{R}^d$ be finite. Then, for all $k \in \mathbb{N}$ with $2k \leq |P|$, the partition $\mathcal{A}$ of P into $2k$ clusters computed by $\boldsymbol{CL}^{\mathrm{rad}}$ satisfies*

$$\max_{a \in \mathcal{A}} \mathrm{rad}(a) < 24d \cdot e^{24d} \cdot \mathrm{rad}(\mathcal{O}_k^{\mathrm{rad}}).$$

Combining this result with Theorem 2.76 yields the following theorem.

Theorem 2.80. *For $d \in \mathbb{N}$ and a finite point set $P \subseteq \mathbb{R}^d$ the algorithm $\boldsymbol{CL}^{\mathrm{rad}}$ computes an $e^{O(d)}$-approximation for the continuous k-center problem.*

Lemma 2.81 ([2])**.** *Let $P \subseteq \mathbb{R}^d$ be finite. Then, for all $k \in \mathbb{N}$ with $2k \leq |P|$, the partition $\mathcal{A}$ of P into $2k$ clusters computed by $\boldsymbol{CL}^{\mathrm{diam}}$ satisfies*

$$\max_{a \in \mathcal{A}} \mathrm{diam}(a) < 2^{3(42d)^d}(28d + 6) \cdot \mathrm{diam}(\mathcal{O}_k^{\mathrm{diam}}).$$

Analogously to $\mathbf{CL}^{\mathrm{drad}}$ and $\mathbf{CL}^{\mathrm{rad}}$ we can conclude the following theorem.

Theorem 2.82. *For $d \in \mathbb{N}$ and a finite point set $P \subseteq \mathbb{R}^d$ the algorithm $\boldsymbol{CL}^{\mathrm{diam}}$ computes a $2^{O(d)^d}$-approximation for the diameter k-clustering problem.*

Chapter 3

The Shadow Vertex Algorithm

The shadow vertex pivot rule is a popular pivoting rule for the simplex algorithm. Though it has no polynomial running time in general it is arguable fast. It was also shown to have polynomial running time in the model of smoothed complexity. In this chapter we provide a randomized variant of the shadow vertex algorithm which can be used to solve a linear program $\max\{c_0^T x | Ax \leq b\}$ in strongly polynomial time with respect to the dimension of the polyhedron, the number of constraints and a parameter $1/\delta$ where δ somehow measures the flatness of the polyhedron.

3.1 Outline of the Analysis

To analyze the shadow vertex algorithm remember that it is a modification of an algorithm by Eisenbrand and Vempala [25] which solves a linear program $\max\{c_0^T x | Ax \leq b\}$. For a feasible solution x_0 they perturb c_0 by a small amount. Let c be the perturbed objective and x_c be an optimal solution with respect to c. Then they determine a path along the edges of the polyhedron P of feasible solutions from x_0 to x_c. If the amount of perturbation is small enough they argue that x_c and x_{c_0} have a common facet which is then identified. Then they reduce the dimension of P by one and repeat their algorithm on the facet starting with x_c. We borrow their algorithm but replace the subroutine searching for a path from x_0 to x_c by a variant of the shadow vertex algorithm introduced by Brunsch and Röglin in [17].

The algorithm finds a path between two vertices, say x_1 and x_2, on a polytope by determining their cones and choosing a random vector w_1, respectively w_2, from each cone. Note that because of their choice the vertices x_1 and x_2 are optimal with respect to the objectives w_1 and w_2, respectively. We project the polyhedron P onto the plane spanned by w_1 and w_2 in order to reduce the problem to finding a path between vertices on a 2-dimensional polygon P', which is easy by walking along the edges. We want to adapt the algorithm to find an optimal solution x of P with respect to c_0, which is an element of x's cone. Anyway, to start the shadow vertex algorithm we need a randomly chosen vector. Therefore we perturb all entries of c_0 a little bit and denote the perturbed objective by c. Unfortunately the vectors w_1 and c are chosen by different types of randomness which changes the analysis in [17] significantly. Nevertheless, we adopt a lot of ideas and the main structure of the analysis. Notice that in general it may happen that by perturbation

c is no longer an element of x's cone. We denote by x_c an optimal solution with respect to the perturbed objective c.

From the description of the shadow vertex algorithm it is clear that the main step in proving Theorem 3.7 is to bound the expected number of edges on the path from $\pi(x_0)$ to $\pi(x_c)$ on the polygon P'. In order to do this, we look at the slopes of the edges on this path and prove that the slopes are pairwise distinct with probability one. This makes it possible to count the number of slopes in the interval $[0, \infty)$ instead of counting the number of vertices on the path from $\pi(x_0)$ to $\pi(x_c)$. Therefore we partition the interval $[0, n]$ into small subintervals such that with high probability, none of them contains more than one slope. Then it is sufficient to bound the probability for each interval $[t, t + \epsilon)$, that a slope is contained: By the choice of the projection vectors the sequence of slopes is monotonically decreasing and together with the above argument it follows that it is even strictly decreasing with probability one. Now we use a similar technique to the principle of deferred decisions motivated by [17]. We consider the unique leftmost vertex $\hat{p}$ on the path, which slope is bigger than t (see Figure 3.1). Note that this is the only chance for a slope in $[t, t + \epsilon)$. Since P is non-degenerated each two neighbors on the path differ in their basis elements by exactly one element. Let a_i be the element which is not part of the basis of $\hat{p}$ but in the basis of its left neighbor. It turns out that by switching w to an arbitrary vector on the ray $\{w - \gamma \cdot a_i \,|\, \gamma \geq 0\}$ the vertex $\hat{p}$ is invariant with that property. Now we split the random draw of the vectors w_1 and c in the shadow vertex algorithm into two steps. In the first step the vector c is completely revealed while instead of w_1 only an element $\tilde{w}_1$ from the ray $\{w_1 + \gamma \cdot a_i \,|\, \gamma \geq 0\}$ is revealed. By that we get enough information to identify the candidate $\hat{p}$. Even though $\hat{p}$ is determined in the first step, its slope is not. The only randomness left in the second step is the exact position of the vector w_2 on the ray $\{\tilde{w}_2 - \gamma \cdot a_i \,|\, \gamma \geq 0\}$, which suffices to bound the probability that the slope of e lies in the interval $(t, t + \epsilon]$.

To count all slopes between zero and infinity we use a trick and switch the order of the projection vectors w_1 and c. By that each slope of size m turns into a slope of size $1/m$. Again we count the expected number of slopes in the interval $[0, 1/n]$ and obtain the expected number of slopes in general which directly turns into the number of expected edges on the path.

We will describe our algorithm in Section 3.3.3 where we assume that the linear program in non-degenerate, that A has full rank n, and that the polyhedron P is bounded. It is already described in Section 3 of [17] that the linear program can be made non-degenerate by slightly perturbing the vector b. This does not affect the parameter δ because δ depends only on the matrix A. In Section 3.7 we discuss why we can assume that A has full rank and why P is bounded. There are, of course, textbook methods to transform a linear program into this form. However, we need to be careful that this transformation does not change δ. In Section 3.4 we analyze our algorithm and prove Theorem 1.8. In Section 3.5 we discuss the runtime of the algorithm and analyze the number of random bits necessary to run the shadow vertex method. In Section 3.6 we discuss how Phase 1 of the simplex method can be implemented. In Section 3.7 we argue how to cope with unbounded linear programs and linear programs without full column rank.

3.2 Preliminaries

We assume that we are given a linear program $\max\{{c_0}^{\mathrm{T}}x \mid Ax \leq b\}$ with vectors $b \in \mathbb{R}^m$ and $c_0 \in \mathbb{R}^n$ and a matrix $A = [a_1, \dots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$. Moreover, we assume that $\|c_0\| = \|a_i\| = 1$ for all $i \in [m]$, where $[m] := \{1, \dots, m\}$ and $\|\cdot\|$ denotes the Euclidean norm. This entails no loss of generality since any linear program can be brought into this form by scaling the objective function and the constraints appropriately. For a vector $x \in \mathbb{R}^n \setminus \{0^n\}$ we denote by $\mathcal{N}(x) = \frac{1}{\|x\|} \cdot x$ the normalization of vector x.

For a vertex v of the polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ we call the set of row indices $B_v = \{i \in \{1, \dots, m\} \mid a_i \cdot v = b_i\}$ *basis* of v. Then the *normal cone* C_v of v is given by the set

$$C_v = \left\{ \sum_{i \in B_v} \lambda_i a_i \mid \lambda_i \geq 0 \right\}.$$

Before we start with the algorithm we give an alternative and more general definition of δ and discuss some properties of this parameter.

3.2.1 The Parameter δ

In [17] Brunsch and Röglin introduced the parameter δ only for $m \times n$-matrices A with rank n. This was the only interesting case for the type of problem considered there. Here we cannot assume the constraint matrix to have full column rank. Hence, in Definition 1.7 we extended the definition of δ to arbitrary matrices (as Eisenbrand and Vempala [25]). We will now give a definition of δ that is equivalent to Definition 1.7 and allows to prove some important properties of δ.

Definition 3.1.

1. *Let $z_1, \dots, z_k \in \mathbb{R}^n$ be $k \geq 2$ linearly independent vectors and let $\varphi \in (0, \frac{\pi}{2}]$ be the angle between z_k and $\operatorname{span}\{z_1, \dots, z_{k-1}\}$. By $\hat{\delta}(\{z_1, \dots, z_{k-1}\}, z_k) = \sin\varphi$ we denote the sine of φ. Moreover, we set*

$$\delta(z_1, \dots, z_k) = \min_{\ell \in [k]} \hat{\delta}(\{z_i \mid i \in [k] \setminus \{\ell\}\}, z_\ell).$$

2. *Given a matrix $A = [a_1, \dots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$ with rank $r = \operatorname{rank}(A) \geq 2$, we set*

$$\delta(A) = \min\{\delta(a_{i_1}, \dots, a_{i_r}) \mid a_{i_1}, \dots, a_{i_r} \text{ linearly independent}\}.$$

Note that for the angle φ in Definition 3.1 we obtain the equation

$$\varphi = \min\{\angle(z_k, z) \mid z \in \operatorname{span}\{z_1, \dots, z_{k-1}\}\}.$$

Furthermore, the minimum is attained for the orthogonal projection of the vector z_k onto $\operatorname{span}\{z_1, \dots, z_{k-1}\}$ when we use the convention $\angle(x, 0) := \frac{\pi}{2}$ for any vector $x \in \mathbb{R}^n$. For this reason the sine is given by the length of the orthogonal projection divided by $\|z_k\|$. In the case where $\|z_k\|$ has length 1 this equals the length of the orthogonal projection and thus the δ-distance of z_k to $\operatorname{span}\{z_1, \dots, z_{k-1}\}$ as defined in Definition 1.7.

Lemma 3.2 (Lemma 5 of [17]). *Let $z_1, \dots, z_n \in \mathbb{R}^n$ be linearly independent vectors of length 1, let $A \in \mathbb{R}^{m \times n}$ be a matrix with $\operatorname{rank}(A) = n$, and let $\delta := \delta(A)$. Then the following properties hold:*

1. *If M is the inverse of $[z_1, \dots, z_n]^{\mathrm{T}}$, then*
$$\delta(z_1, \dots, z_n) = \frac{1}{\max_{k \in [n]} \|m_k\|} \leq \frac{\sqrt{n}}{\max_{k \in [n]} \|M_k\|}\,,$$
where $[m_1, \dots, m_n] = M$ and $[M_1, \dots, M_n] = M^{\mathrm{T}}$.

2. *If $Q \in \mathbb{R}^{n \times n}$ is an orthogonal matrix, then $\delta(Qz_1, \dots, Qz_n) = \delta(z_1, \dots, z_n)$.*

3. *Let y_1 and y_2 be two neighboring vertices of $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ and let ${a_i}^{\mathrm{T}}$ be a row of A. If ${a_i}^{\mathrm{T}} \cdot (y_2 - y_1) \neq 0$, then $|{a_i}^{\mathrm{T}} \cdot (y_2 - y_1)| \geq \delta \cdot \|y_2 - y_1\|$.*

4. *If A is an integral matrix, then $\frac{1}{\delta} \leq n\Delta_1\Delta_{n-1} \leq n\Delta^2$, where Δ, Δ_1, and Δ_{n-1} are the largest absolute values of any sub-determinant of A of arbitrary size, of size 1, and of size $n-1$, respectively.*

3.2.2 Some Probability Theory

In this section we state and formulate the corollary about linear combinations of random variables used in Section 3.4 and Appendix A. This theorem follows from Theorem 3.3 of [16] which we will recite here in a simplified variant.

Theorem 3.3 (cf. Theorem 3.3 of [16]). *Let $\varepsilon > 0$ and $\phi \geq 1$ be reals, let $I_1, \dots, I_n \subseteq [-1, 1]$ be intervals of length $1/\phi$, and let $X_1, \dots, X_n$ be independent random variables such that X_k is uniformly distributed on I_k for $k = 1, \dots, n$. Moreover, let $A \in \mathbb{R}^{n \times n}$ be an invertible matrix, let $(Y_1, \dots, Y_{n-1}, Z)^{\mathrm{T}} = A \cdot (X_1, \dots, X_n)^{\mathrm{T}}$ be the linear combinations of $X_1, \dots, X_n$ given by A, and let $I \colon \mathbb{R}^{n-1} \to \{[x, x + \varepsilon] \mid x \in \mathbb{R}\}$ be a function mapping a tuple $(y_1, \dots, y_{n-1}) \in \mathbb{R}^{n-1}$ to an interval $I(y_1, \dots, y_{n-1})$ of length ε. Then the probability that Z falls into the interval $I(Y_1, \dots, Y_{n-1})$ can be bounded by*
$$\mathbf{Pr}\left[Z \in I(Y_1, \dots, Y_{n-1})\right] \leq 2\varepsilon\phi \cdot \sum_{i=1}^{n} \frac{|\det A_{n,i}|}{|\det A|}\,,$$
where $A_{n,i}$ is the $(n-1) \times (n-1)$-submatrix of A obtained from A by removing row n and column i.

Now we can state

Corollary 3.4. *Let ε, ϕ, $X_1, \dots, X_n$, A, $Y_1, \dots, Y_{n-1}, Z$, and I be as in Theorem 3.3. Then the probability that Z falls into the interval $I(Y_1, \dots, Y_{n-1})$ can be bounded by*
$$\mathbf{Pr}\left[Z \in I(Y_1, \dots, Y_{n-1})\right] \leq \frac{2n\varepsilon\phi}{\delta(a_1, \dots, a_n) \cdot \min_{k \in [n]} \|a_k\|}\,,$$
where $a_1, \dots, a_n$ denote the columns of matrix A. Furthermore, if A is orthogonal, then even the stronger bound
$$\mathbf{Pr}\left[Z \in I(Y_1, \dots, Y_{n-1})\right] \leq 2\sqrt{n}\varepsilon\phi$$
holds.

Proof. In accordance with Theorem 3.3 it suffices to bound the sum $\sum_{i=1}^{n} \frac{|\det(A_{n,i})|}{|\det(A)|}$ from above. For this, consider the equation $Ax = e_n$, where $e_n = (0, \dots, 0, 1) \in \mathbb{R}^n$ denotes the nth unit vector. Following Cramer's rule and Laplace's formula, we obtain

$$|x_i| = \frac{|\det([a_1, \dots, a_{i-1}, e_n, a_{i+1}, \dots, a_n])|}{|\det(A)|} = \frac{|\det(A_{n,i})|}{|\det(A)|}\,.$$

Hence, applying Theorem 3.3 yields

$$\mathbf{Pr}\,[Z \in I(Y_1, \dots, Y_{n-1})] \le 2\varepsilon\phi \cdot \sum_{i=1}^{n} |x_i| = 2\varepsilon\phi \cdot \|x\|_1 \le 2\sqrt{n}\varepsilon\phi \cdot \|x\|\,.$$

Recall, that by $\|x\|$ we refer to the Euclidean norm $\|x\|_2$ of x. The claim for orthogonal matrices A follows immediately since $\|x\| = \|A^{-1}e_n\| = \|e_n\| = 1$ because $A^{-1} = A^{\mathrm{T}}$ is orthogonal as well.

For the general case we consider the equation $\hat{A}\hat{x} = e_n$, where $\hat{A} = [\mathcal{N}(a_1), \dots, \mathcal{N}(a_n)]$ consists of the normalized columns of matrix A. Vector $\hat{x} = \hat{A}^{-1}e_n$ is the nth column of the matrix $\hat{A}^{-1}$. Thus, we obtain

$$\|\hat{x}\| \le \max_{\substack{r \text{ column} \\ \text{of } \hat{A}^{-1}}} \|r\| \le \frac{\sqrt{n}}{\delta(a_1, \dots, a_n)}\,,$$

where second inequality is due to Claim 1 of Lemma 3.2. Due to $A = \hat{A}\cdot\mathrm{diag}(\|a_1\|, \dots, \|a_n\|)$, we have

$$x = A^{-1}e_n = \mathrm{diag}\left(\frac{1}{\|a_1\|}, \dots, \frac{1}{\|a_n\|}\right) \cdot \hat{A}^{-1}e_n = \mathrm{diag}\left(\frac{1}{\|a_1\|}, \dots, \frac{1}{\|a_n\|}\right) \cdot \hat{x}\,.$$

Consequently, $\|x\| \le \|\hat{x}\| / \min_{k \in [n]} \|a_k\|$ and, thus,

$$\mathbf{Pr}\,[Z \in I(Y_1, \dots, Y_{n-1})] \le 2\sqrt{n}\varepsilon\phi \cdot \frac{\|\hat{x}\|}{\min_{k\in[n]} \|a_k\|} \le \frac{2n\varepsilon\phi}{\delta(a_1, \dots, a_n) \cdot \min_{k\in[n]} \|a_k\|}\,. \qquad \square$$

3.3 Algorithm

Given a linear program $\max\{{c_0}^{\mathrm{T}}x \mid Ax \le b\}$ and a basic feasible solution x_0, our algorithm randomly perturbs each coefficient of the vector c_0 by at most $1/\phi$ for some parameter ϕ to be determined later. Let us call the resulting vector c. The next step is then to use the shadow vertex algorithm to compute a path from x_0 to a vertex x_c which maximizes the function $c^{\mathrm{T}}x$ for $x \in P$. For $\phi > \frac{2n^{3/2}}{\delta}$ one can argue that the solution x has a facet in common with the optimal solution $x^\star$ of the given linear program with objective function ${c_0}^{\mathrm{T}}x$. Then the algorithm is run again on this facet one dimension lower until all facets that define $x^\star$ are identified.

This section is organized as follows. In Section 3.3.1 we repeat a construction from [25] to project a facet of the polyhedron P into the space $\mathbb{R}^{n-1}$ without changing the parameter δ. This is crucial for being able to identify the facets that define $x^\star$ one after another. In Section 3.3.2 we also repeat an argument from [25] that shows how a common facet of x_c and $x^\star$ can be identified if x_c is given. Section 3.3.3 presents the shadow vertex algorithm, the main building block of our algorithm. Finally in Section 3.5 we discuss the running time of a single pivot step of the shadow vertex algorithm.

3.3.1 Reducing the Dimension

Assume that we have identified an element a_i, $i \in [m]$, of the optimal basis $x^\star$ (i.e., $a_i x^\star = b_i$). In [25] it is described how to reduce in this case the dimension of the linear program by one without changing the parameter δ. We repeat the details. Without loss of generality we may assume that a_1 is an element of the optimal basis. Let $Q \in \mathbb{R}^{n \times n}$ be an orthogonal matrix that rotates a_1 into the first unit vector e_1. Then the following linear programs are equivalent:

$$\max\{c_0^T x \mid x \in \mathbb{R}^n, Ax \leq b\} \tag{3.1}$$

and

$$\max\{c_0^T Qx \mid x \in \mathbb{R}^n, AQx \leq b\}.$$

In the latter linear program the first constraint is of the form $x_1 \leq b_1$. We set this constraint to equality and subtract this equation from the other constraints (i.e., we project each row into the orthogonal complement of e_1). Thus, we end up with a linear program of dimension $n-1$. Lemma 3.2 shows that the δ-distance does not change under multiplication with an orthogonal matrix. Furthermore, Lemma 3 of [25] ensures that the δ-distance property is not destroyed by the projection onto the orthogonal complement.

3.3.2 Identifying an Element of the Optimal Basis

In this section we repeat how an element of the optimal basis can be identified if an optimal solution x_c for an objective function $c^{\mathrm{T}}x$ with $\|c_0 - c\| < \delta/(2n)$ is given (see also [25]).

Lemma 3.5 (Lemma 2 of [25])**.** *Let $B \subseteq \{1, \dots, m\}$ be the optimal basis of the linear program* (3.1) *and let B' be an optimal basis of the linear program* (3.1) *with c_0 being replaced by c, where $\|c_0 - c\| < \delta/(2n)$ holds. Consider the conic combination*

$$c = \sum_{j \in B'} \mu_j a_j.$$

For $k \in B' \setminus B$, one has $\|c_0 - c\| \geq \delta \cdot \mu_k$.

The following corollary whose proof can also be found in [25] gives a constructive way to identify an element of the optimal basis.

Corollary 3.6. *Let $c \in \mathbb{R}^n$ be such that $\|c_0 - c\| < \delta/(2 \cdot n)$ and let μ_j, B, and B' be defined as in Lemma 3.5. There exists at least one coefficient μ_k with $\mu_k > 1/n \cdot (1 - \delta/(2 \cdot n))$ and any k with this property is an element of the optimal basis B (assuming $\|c_0\| = 1$).*

The corollary implies that given a solution x_c that is optimal for an objective function $c^{\mathrm{T}}x$ with $\|c_0 - c\| < \delta/(2n)$, one can identify an element of the optimal basis by solving the system of linear equations

$$[a'_1, \dots, a'_n] \cdot \mu = c,$$

where the a'_i denote the constraints that are tight in x_c.

3.3.3 The Shadow Vertex Method

In this section we assume that we are given a linear program of the form $\max\{{c_0}^{\mathrm{T}}x \mid x \in P\}$, where $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is a bounded polyhedron (i.e., a polytope), and a basic feasible solution $x_0 \in P$. We assume $\|c_0\| = \|a_i\| = 1$ for all rows a_i of A. Furthermore, we assume that the linear program is non-degenerate.

Due to the assumption $\|c_0\| = 1$ it holds $c_0 \in [-1,1]^n$. Our algorithm slightly perturbs the given objective function ${c_0}^{\mathrm{T}}x$ at random. For each component $(c_0)_i$ of c_0 it chooses an arbitrary interval $I_i \subseteq [-1,1]$ of length $1/\phi$ with $(c_0)_i \in I_i$, where ϕ denotes a parameter that will be given to the algorithm. Then a random vector $c \in [-1,1]^n$ is drawn as follows: Each component c_i of c is chosen independently uniformly at random from the interval I_i. We denote the resulting random vector by $\mathrm{pert}(c_0, \phi)$. Note that we can bound the norm of the difference $\|c_0 - c\|$ between the vectors c_0 and c from above by $\frac{\sqrt{n}}{\phi}$.

The shadow vertex algorithm is given as Algorithm 2. It is assumed that ϕ is given to the algorithm as a parameter. We will discuss later how we can run the algorithm without knowing this parameter. Let us remark that the Steps 5 and 6 in Algorithm 2 are actually not executed separately. Instead of computing the whole projection P' in advance, the edges of P' are computed on the fly one after another.

Algorithm 2 Shadow Vertex Algorithm

```
1: Generate a random perturbation $c = \mathrm{pert}(c_0, \phi)$ of $c_0$.
2: Determine $n$ linearly independent rows ${u_k}^{\mathrm{T}}$ of $A$ for which ${u_k}^{\mathrm{T}}x_0 = b_k$.
3: Draw a vector $\lambda \in (0,1]^n$ uniformly at random.
4: Set $w = -[u_1, \dots, u_n] \cdot \lambda$.
5: Use the function $\pi : x \mapsto (c^{\mathrm{T}}x, w^{\mathrm{T}}x)$ to project $P$ onto the Euclidean plane and obtain
   the shadow vertex polygon $P' = \pi(P)$.
6: Walk from $\pi(x_0)$ along the edges of $P'$ in increasing direction of the first coordinate
   until a rightmost vertex $\tilde{x}_c$ of $P'$ is found.
7: Output the vertex $x_c$ of $P$ that is projected onto $\tilde{x}_c$.
```

Note that

$$\|w\| \leq \sum_{k=1}^{n} \lambda_k \cdot \|u_k\| \leq \sum_{k=1}^{n} \lambda_k \leq n,$$

where the second inequality follows because all rows of A are assumed to have norm 1.

The Shadow Vertex Algorithm yields a path from the vertex x_0 to a vertex x_c that is optimal for the linear program $\max\{c^{\mathrm{T}}x \mid x \in P\}$ where $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. The following theorem (whose proof can be found in Section 3.4) bounds the expected length of this path, i.e., the number of pivots.

Theorem 3.7. *For any $\phi \geq \sqrt{n}$ the expected number of edges on the path output by Algorithm 2 is $O\Big(\frac{mn^2}{\delta^2} + \frac{m\sqrt{n}\phi}{\delta}\Big)$.*

Since $\|c_0 - c\| \leq \frac{\sqrt{n}}{\phi}$ choosing $\phi > \frac{2n^{3/2}}{\delta}$ suffices to ensure $\|c_0 - c\| < \frac{\delta}{2n}$. Hence, for such a choice of ϕ, by Corollary 3.6, the vertex x_c has a facet in common with the optimal solution of the linear program $\max\{{c_0}^{\mathrm{T}}x \mid x \in P\}$ and we can reduce the dimension of the linear program as discussed in Section 3.3.1. This step is repeated at most n times. It is important that we can start each repetition with a known feasible solution because the transformation in Section 3.3.1 maps the optimal solution of the linear program of repetition i onto a feasible solution with which repetition $i+1$ can be initialized. Together with Theorem 3.7 this implies that an optimal solution of the linear program (3.1) can be found by performing in expectation $O\Big(\frac{mn^3}{\delta^2}+\frac{mn^{3/2}\phi}{\delta}\Big)$ pivots if a basic feasible solution x_0 and the right choice of ϕ are given. We will refer to this algorithm as *repeated shadow vertex algorithm.*

Since δ is not known to the algorithm, the right choice for ϕ cannot easily be computed. Instead we will try values for ϕ until an optimal solution is found. For $i \in \mathbb{N}$ let $\phi_i = 2^i n^{3/2}$. First we run the repeated shadow vertex algorithm with $\phi = \phi_0$ and check whether the returned solution is an optimal solution for the linear program $\max\{{c_0}^{\mathrm{T}}x \mid x \in P\}$. If this is not the case, we run the repeated shadow vertex algorithm with $\phi = \phi_1$, and so on. We continue until an optimal solution is found. For $\phi = \phi_{i^\star}$ with $i^\star = \lceil \log_2(1/\delta) \rceil + 2$ this is the case because $\phi_{i^\star} > \frac{2n^{3/2}}{\delta}$.

Since $\phi_{i^\star} \le \frac{8n^{3/2}}{\delta}$, in accordance with Theorem 3.7, each of the at most $i^\star = O(\log(1/\delta))$ calls of the repeated shadow vertex algorithm uses in expectation

$$O\Big(\frac{mn^3}{\delta^2}+\frac{mn^{3/2}\phi_{i^\star}}{\delta}\Big) = O\Big(\frac{mn^3}{\delta^2}\Big).$$

pivots. Together this proves the first part of Theorem 1.8. The second part follows with Lemma 3.29, which states that Phase 1 can be realized with increasing $1/\delta$ by at most $\sqrt{m}$ and increasing the number of variables from n to $n+m \le 2m$. This implies that the expected number of pivots of each call of the repeated shadow vertex algorithm in Phase 1 is $O(m(n+m)^3\sqrt{m}^2/\delta^2) = O(m^5/\delta^2)$. Since $1/\delta$ can increase by a factor of $\sqrt{m}$, the argument above yields that we need to run the repeated shadow vertex algorithm at most $i^\star = O(\log(\sqrt{m}/\delta))$ times in Phase 1 to find a basic feasible solution. By setting $\phi_i = 2^i\sqrt{m}(n+m)^{3/2}$ instead of $\phi_i = 2^i(n+m)^{3/2}$ this number can be reduced to $i^\star = O(\log(1/\delta))$ again.

Theorem 1.9 follows from Theorem 1.8 using the following fact from [17]: Let $A \in \mathbb{Z}^{m\times n}$ be an integer matrix and let $A' \in \mathbb{R}^{m\times n}$ be the matrix that arises from A by scaling each row such that its norm equals 1. If Δ denotes an upper bound for the absolute value of any sub-determinant of A, then A' satisfies the δ-distance property for $\delta = 1/(\Delta^2 n)$. Additionally Lemma 3.30 states that Phase 1 can be realized without increasing Δ but with increasing the number of variables from n to $n+m \le 2m$. Substituting $1/\delta = \Delta^2 n$ in Theorem 1.8 almost yields Theorem 1.9 except for a factor $O(\log(\Delta^2 n))$ instead of $O(\log(\Delta+1))$. This factor results from the number $i^\star$ of calls of the repeated shadow vertex algorithm. The desired factor of $O(\log(\Delta+1))$ can be achieved by setting $\phi_i = 2^i n^{5/2}$ if a basic feasible solution is known and $\phi_i = 2^i(n+m)^{5/2}$ in Phase 1.

3.4 Analysis of the Shadow Vertex Algorithm

For given linear functions $L_1 \colon \mathbb{R}^n \to \mathbb{R}$ and $L_2 \colon \mathbb{R}^n \to \mathbb{R}$ we denote by $\pi = \pi_{L_1,L_2}$ the function $\pi \colon \mathbb{R}^n \to \mathbb{R}^2$, given by $\pi(x) = (L_1(x), L_2(x))$. Note that n-dimensional vectors

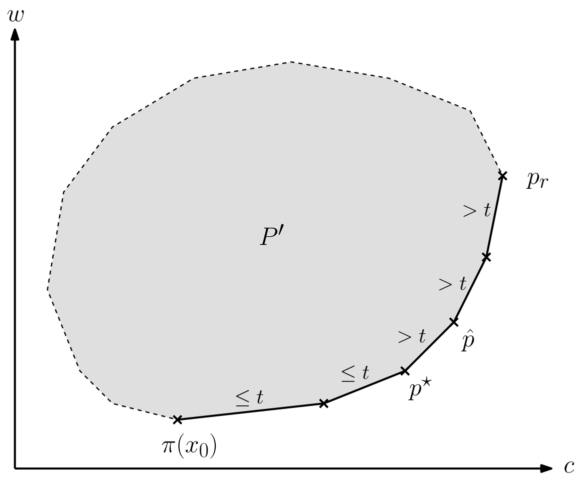


Figure 3.1: Slopes of the vertices of R

can be treated as linear functions. By $P' = P'_{L_1,L_2}$ we denote the projection $\pi(P)$ of the polytope P onto the Euclidean plane, and by $R = R_{L_1,L_2}$ we denote the path from the bottommost vertex of P' to the rightmost vertex of P' along the edges of the lower envelope of P'.

Our goal is to bound the expected number of edges of the path $R = R_{c,w}$, which is random since c and w are random. Each edge of R corresponds to a slope in $(0, \infty)$. These slopes are pairwise distinct with probability one (see Lemma 3.9). Hence, the number of edges of R equals the number of distinct slopes of R.

Definition 3.8. *For a real $\varepsilon > 0$ let $\mathcal{F}_\varepsilon$ denote the event that there are three pairwise distinct vertices z_1, z_2, z_3 of P such that z_1 and z_3 are neighbors of z_2 and such that*

$$\left| \frac{w^{\mathrm{T}} \cdot (z_2 - z_1)}{c^{\mathrm{T}} \cdot (z_2 - z_1)} - \frac{w^{\mathrm{T}} \cdot (z_3 - z_2)}{c^{\mathrm{T}} \cdot (z_3 - z_2)} \right| \leq \varepsilon \,.$$

Note that if event $\mathcal{F}_\varepsilon$ does not occur, then all slopes of R differ by more than ε. Particularly, all slopes are pairwise distinct. First of all we show that event $\mathcal{F}_\varepsilon$ is very unlikely to occur if ε is chosen sufficiently small. The proof of the following lemma is almost identical to the corresponding proof in [17] except that we need to adapt it to the different random model of c. The proof as well as the proofs of some other lemmas that are almost identical to their counterparts in [17] can be found in Appendix A for the sake of completeness. Proofs that are completely identical to [17] are omitted.

Lemma 3.9. *The probability of event $\mathcal{F}_\varepsilon$ tends to 0 for $\varepsilon \to 0$.*

Let p be a vertex of R, but not the bottommost vertex $\pi(x_0)$. We call the slope s of the edge incident to p to the left of p *the slope of p*. As a convention, we set the slope of $\pi(x_0)$ to 0 which is smaller than the slope of any other vertex p of R.

Let $t \geq 0$ be an arbitrary real, let $p^\star$ be the rightmost vertex of R whose slope is at most t, and let $\hat{p}$ be the right neighbor of $p^\star$, i.e., $\hat{p}$ is the leftmost vertex of R whose slope exceeds t (see Figure 3.1). Let $x^\star$ and $\hat{x}$ be the neighboring vertices of P with $\pi(x^\star) = p^\star$ and $\pi(\hat{x}) = \hat{p}$. Now let $i = i(x^\star, \hat{x}) \in [m]$ be the index for which ${a_i}^{\mathrm{T}} x^\star = b_i$ and for which $\hat{x}$ is the (unique) neighbor x of $x^\star$ for which ${a_i}^{\mathrm{T}} x < b_i$. This index is unique due to the non-degeneracy of the polytope P. For an arbitrary real $\gamma \geq 0$ we consider the vector $\tilde{w} := w - \gamma \cdot a_i$.

Lemma 3.10. *Let $\tilde{\pi} = \pi_{c,\tilde{w}}$ and let $\tilde{R} = R_{c,\tilde{w}}$ be the path from $\tilde{\pi}(x_0)$ to the rightmost vertex $\tilde{p}_r$ of the projection $\tilde{\pi}(P)$ of polytope P. Furthermore, let $\tilde{p}^\star$ be the rightmost vertex of $\tilde{R}$ whose slope does not exceed t. Then $\tilde{p}^\star = \tilde{\pi}(x^\star)$.*

Let us reformulate the statement of Lemma 3.10 as follows: The vertex $\tilde{p}^\star$ is defined for the path $\tilde{R}$ of polygon $\tilde{\pi}(R)$ with the same rules as used to define the vertex $p^\star$ of the original path R of polygon $\pi(P)$. Even though R and $\tilde{R}$ can be very different in shape, both vertices, $p^\star$ and $\tilde{p}^\star$, correspond to the same solution $x^\star$ in the polytope P, that is, $p^\star = \pi(x^\star)$ and $\tilde{p}^\star = \tilde{\pi}(x^\star)$.

Lemma 3.10 holds for any vector $\tilde{w}$ on the ray $\vec{r} = \{w - \gamma \cdot a_i \mid \gamma \geq 0\}$. As $\|w\| \leq n$ (see Section 3.3.3), we have $w \in [-n, n]^n$. Hence, ray $\vec{r}$ intersects the boundary of $[-n, n]^n$ in a unique point z. We choose $\tilde{w} = \tilde{w}(w, i) := z$ and obtain the following result.

Corollary 3.11. *Let $\tilde{\pi} = \pi_{c,\tilde{w}(w,i)}$ and let $\tilde{p}^\star$ be the rightmost vertex of path $\tilde{R} = R_{c,\tilde{w}(w,i)}$ whose slope does not exceed t. Then $\tilde{p}^\star = \tilde{\pi}(x^\star)$.*

Note that Corollary 3.11 only holds for the right choice of index $i = i(x^\star, \hat{x})$. However, the vector $\tilde{w}(w, i)$ can be defined for any vector $w \in [-n, n]^n$ and any index $i \in [m]$. In the remainder, index i is an arbitrary index from $[m]$.

We can now define the following event that is parameterized in i, t, and a real $\varepsilon > 0$ and that depends on c and w.

Definition 3.12. *For an index $i \in [m]$ and a real $t \geq 0$ let $\tilde{p}^\star$ be the rightmost vertex of $\tilde{R} = R_{c,\tilde{w}(w,i)}$ whose slope does not exceed t and let $y^\star$ be the corresponding vertex of P. For a real $\varepsilon > 0$ we denote by $E_{i,t,\varepsilon}$ the event that the conditions*

- ${a_i}^{\mathrm{T}} y^\star = b_i$ *and*
- $\frac{w^{\mathrm{T}}(\hat{y} - y^\star)}{c^{\mathrm{T}}(\hat{y} - y^\star)} \in (t, t + \varepsilon]$, *where $\hat{y}$ is the neighbor y of $y^\star$ for which ${a_i}^{\mathrm{T}} y < b_i$,*

are met. Note that the vertex $\hat{y}$ always exists and that it is unique since the polytope P is non-degenerate.

Let us remark that the vertices $y^\star$ and $\hat{y}$, which depend on the index i, equal $x^\star$ and $\hat{x}$ if we choose $i = i(x^\star, \hat{x})$. For other choices of i, this is, in general, not the case.

Observe that all possible realizations of w from the line $L := \{w + x \cdot a_i \mid x \in \mathbb{R}\}$ are mapped to the same vector $\tilde{w}(w, i)$. Consequently, if c is fixed and if we only consider realizations of λ for which $w \in L$, then vertex $\tilde{p}^\star$ and, hence, vertex $y^\star$ from Definition 3.12 are already determined. However, since w is not completely specified, we have some randomness left for event $E_{i,t,\varepsilon}$ to occur. This allows us to bound the probability of event $E_{i,t,\varepsilon}$ from above (see proof of Lemma 3.14). The next lemma shows why this probability matters.

Lemma 3.13 (Lemma 12 from [17])**.** *For any $t \geq 0$ and $\varepsilon > 0$ let $A_{t,\varepsilon}$ denote the event that the path $R = R_{c,w}$ has a slope in $(t, t+\varepsilon]$. Then, $A_{t,\varepsilon} \subseteq \bigcup_{i=1}^{m} E_{i,t,\varepsilon}$.*

With Lemma 3.13 we can now bound the probability of event $A_{t,\varepsilon}$. The proof of the next lemma is almost identical to the proof of Lemma 13 from [17]. We include it in the appendix for the sake of completeness. The only differences to Lemma 13 from [17] are that we can now use the stronger upper bound $\|c\| \leq 2$ instead of $\|c\| \leq n$ and that we have more carefully analyzed the case of large t.

Lemma 3.14. *For any $\phi \geq \sqrt{n}$, any $t \geq 0$, and any $\varepsilon > 0$ the probability of event $A_{t,\varepsilon}$ is bounded by*

$$\mathbf{Pr}\left[A_{t,\varepsilon}\right] \leq \frac{2mn^2\varepsilon}{\max\left\{\frac{n}{2}, t\right\} \cdot \delta^2} \leq \frac{4mn\varepsilon}{\delta^2}\,.$$

Lemma 3.15. *For any interval I let X_I denote the number of slopes of $R = R_{c,w}$ that lie in the interval I. Then, for any $\phi \geq \sqrt{n}$,*

$$\mathbf{E}\left[X_{(0,n]}\right] \leq \frac{4mn^2}{\delta^2}$$

Proof. For a real $\varepsilon > 0$ let $\mathcal{F}_\varepsilon$ denote the event from Definition 3.8. Recall that all slopes of R differ by more than ε if $\mathcal{F}_\varepsilon$ does not occur. For $t \in \mathbb{R}$ and $\varepsilon > 0$ let $Z_{t,\varepsilon}$ be the random variable that indicates whether R has a slope in the interval $(t, t+\varepsilon]$ or not, i.e., $Z_{t,\varepsilon} = 1$ if $X_{(t,t+\varepsilon]} > 0$ and $Z_{t,\varepsilon} = 0$ if $X_{(t,t+\varepsilon]} = 0$.

Let $k \geq 1$ be an arbitrary integer. We subdivide the interval $(0, n]$ into k subintervals. If none of them contains more than one slope then the number $X_{(0,n]}$ of slopes in the interval $(0, n]$ equals the number of subintervals for which the corresponding Z-variable equals 1. Formally

$$X_{(0,n]} \leq \begin{cases} \sum_{i=0}^{k-1} Z_{i\cdot\frac{n}{k},\frac{n}{k}} & \text{if } \mathcal{F}_{\frac{n}{k}} \text{ does not occur}\,, \\ m^n & \text{otherwise}\,. \end{cases}$$

This is true because $\binom{m}{n-1} \leq m^n$ is a worst-case bound on the number of edges of P and, hence, of the number of slopes of R. Consequently,

$$\begin{aligned} \mathbf{E}\left[X_{(0,n]}\right] &\leq \sum_{i=0}^{k-1} \mathbf{E}\left[Z_{i\cdot\frac{n}{k},\frac{n}{k}}\right] + \mathbf{Pr}\left[\mathcal{F}_{\frac{n}{k}}\right] \cdot m^n = \sum_{i=0}^{k-1} \mathbf{Pr}\left[A_{i\cdot\frac{n}{k},\frac{n}{k}}\right] + \mathbf{Pr}\left[\mathcal{F}_{\frac{n}{k}}\right] \cdot m^n \\ &\leq \sum_{i=0}^{k-1} \frac{2mn^2 \cdot \frac{n}{k}}{\frac{n}{2}\delta^2} + \mathbf{Pr}\left[\mathcal{F}_{\frac{n}{k}}\right] \cdot m^n = \frac{4mn^2}{\delta^2} + \mathbf{Pr}\left[\mathcal{F}_{\frac{n}{k}}\right] \cdot m^n\,. \end{aligned}$$

The second inequality stems from Lemma 3.14. Now the lemma follows because the bound on $\mathbf{E}\left[X_{(0,n]}\right]$ holds for any integer $k \geq 1$ and since $\mathbf{Pr}\left[\mathcal{F}_\varepsilon\right] \to 0$ for $\varepsilon \to 0$ in accordance with Lemma 3.9. □

In [17] Brunsch and Röglin only compute an upper bound for the expected value of $X_{(0,1]}$. Then they argue that the same upper bound also holds for the expected value of $X_{(1,\infty)}$. In order to see this, simply exchanged the order of the objective functions in the projection π. Then any edge with a slope of $s > 1$ becomes an edge with slope $\frac{1}{s} < 1$. Hence the number of slopes in $[1, \infty)$ equals the number of slopes in $(0, 1]$ in the scenario in which the objective functions are exchanged. Due to the symmetry in the choice of the objective functions in [17] the same analysis as before applies also to that scenario.

We will now also exchange the order of the objective functions $w^{\mathrm{T}}x$ and $c^{\mathrm{T}}x$ in the projection. Since these objective functions are not anymore generated by the same random experiment, a simple argument as in [17] is not possible anymore. Instead we have to go through the whole analysis again. We will use the superscript -1 to indicate that we are referring to the scenario in which the order of the objective functions is exchanged. In particular, we consider the events $\mathcal{F}_\varepsilon^{-1}$, $A_{t,\varepsilon}^{-1}$, and $E_{i,t,\varepsilon}^{-1}$ that are defined analogously to their counterparts without superscript except that the order of the objective functions is exchanged. The proof of the following lemma is analogous to the proof of Lemma 3.9.

Lemma 3.16. *The probability of event $\mathcal{F}_\varepsilon^{-1}$ tends to 0 for $\varepsilon \to 0$.*

Lemma 3.17. *For any $\phi \geq \sqrt{n}$, any $t \geq 0$, and any $\varepsilon > 0$ the probability of event $A_{t,\varepsilon}^{-1}$ is bounded by*

$$\mathbf{Pr}\left[A_{t,\varepsilon}^{-1}\right] \leq \frac{2mn^{3/2}\varepsilon\phi}{\max\left\{1, \frac{nt}{2}\right\} \cdot \delta} \leq \frac{2mn^{3/2}\varepsilon\phi}{\delta} .$$

Proof. Due to Lemma 3.13 (to be precise, due to its canonical adaption to the events with superscript -1) it suffices to show that

$$\mathbf{Pr}\left[E_{i,t,\varepsilon}^{-1}\right] \leq \frac{1}{m} \cdot \frac{2mn^{3/2}\varepsilon\phi}{\max\left\{1, \frac{nt}{2}\right\} \cdot \delta} = \frac{2n^{3/2}\varepsilon\phi}{\max\left\{1, \frac{nt}{2}\right\} \cdot \delta}$$

for any index $i \in [m]$.

We apply the principle of deferred decisions and assume that vector w is already fixed. Now we extend the normalized vector a_i to an orthonormal basis $\{q_1, \ldots, q_{n-1}, a_i\}$ of $\mathbb{R}^n$ and consider the random vector $(Y_1, \ldots, Y_{n-1}, Z)^{\mathrm{T}} = Q^{\mathrm{T}}c$ given by the matrix vector product of the transpose of the orthogonal matrix $Q = [q_1, \ldots, q_{n-1}, a_i]$ and the vector $c = (c_1, \ldots, c_n)^{\mathrm{T}}$. For fixed values $y_1, \ldots, y_{n-1}$ let us consider all realizations of c such that $(Y_1, \ldots, Y_{n-1}) = (y_1, \ldots, y_{n-1})$. Then, c is fixed up to the ray

$$c(Z) = Q \cdot (y_1, \ldots, y_{n-1}, Z)^{\mathrm{T}} = \sum_{j=1}^{n-1} y_j \cdot q_j + Z \cdot a_i = v + Z \cdot a_i$$

for $v = \sum_{j=1}^{n-1} y_j \cdot q_j$. All realizations of $c(Z)$ that are under consideration are mapped to the same value $\tilde{c}$ by the function $c \mapsto \tilde{c}(c, i)$, i.e., $\tilde{c}(c(Z), i) = \tilde{c}$ for any possible realization of Z. In other words, if $c = c(Z)$ is specified up to this ray, then the path $R_{\tilde{c}(c,i),w}$ and, hence, the vectors $y^\star$ and $\hat{y}$ from the definition of event $E_{i,t,\varepsilon}^{-1}$, are already determined.

Let us only consider the case that the first condition of event $E_{i,t,\varepsilon}^{-1}$ is fulfilled. Otherwise, event $E_{i,t,\varepsilon}$ cannot occur. Thus, event $E_{i,t,\varepsilon}^{-1}$ occurs iff

$$(t, t+\varepsilon] \ni \frac{c^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{w^{\mathrm{T}} \cdot (\hat{y} - y^\star)} = \underbrace{\frac{v^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{w^{\mathrm{T}} \cdot (\hat{y} - y^\star)}}_{=:\alpha} + Z \cdot \underbrace{\frac{{a_i}^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{w^{\mathrm{T}} \cdot (\hat{y} - y^\star)}}_{=:\beta} .$$

The next step in this proof will be to show that the inequality $|\beta| \geq \max\{1, \sqrt{n} \cdot t\} \cdot \frac{\delta}{n}$ is necessary for event $E^{-1}_{i,t,\varepsilon}$ to happen. For the sake of simplicity let us assume that $\|\hat{y} - y^\star\| = 1$ since β is invariant under scaling. If event $E^{-1}_{i,t,\varepsilon}$ occurs, then ${a_i}^{\mathrm{T}} y^\star = b_i$, $\hat{y}$ is a neighbor of $y^\star$, and ${a_i}^{\mathrm{T}} \hat{y} \neq b_i$. That is, by Lemma 3.2, Claim 3 we obtain $|{a_i}^{\mathrm{T}} \cdot (\hat{y} - y^\star)| \geq \delta \cdot \|\hat{y} - y^\star\| = \delta$ and, hence,

$$|\beta| = \left| \frac{{a_i}^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{w^{\mathrm{T}} \cdot (\hat{y} - y^\star)} \right| \geq \frac{\delta}{|w^{\mathrm{T}} \cdot (\hat{y} - y^\star)|} \,.$$

On the one hand we have $|w^{\mathrm{T}} \cdot (\hat{y} - y^\star)| \leq \|w\| \cdot \|\hat{y} - y^\star\| \leq \left(\sum_{i=1}^n \|u_i\| \right) \cdot 1 \leq n$. On the other hand, due to $\frac{c^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{w^{\mathrm{T}} \cdot (\hat{y} - y^\star)} \geq t$ we have

$$|w^{\mathrm{T}} \cdot (\hat{y} - y^\star)| \leq \frac{|c^{\mathrm{T}} \cdot (\hat{y} - y^\star)|}{t} \leq \frac{\|c\| \cdot \|\hat{y} - y^\star\|}{t} \leq \frac{\left(1 + \frac{\sqrt{n}}{\phi}\right)}{t} \leq \frac{2}{t} \,,$$

where the third inequality is due to the choice of c as perturbation of the unit vector c_0 and the fourth inequality is due to the assumption $\phi \geq \sqrt{n}$. Consequently,

$$|\beta| \geq \frac{\delta}{\min\left\{n, \frac{2}{t}\right\}} = \max\left\{1, \frac{nt}{2}\right\} \cdot \frac{\delta}{n} \,.$$

Summarizing the previous observations we can state that if event $E^{-1}_{i,t,\varepsilon}$ occurs, then $|\beta| \geq \max\left\{1, \frac{nt}{2}\right\} \cdot \frac{\delta}{n}$ and $\alpha + Z \cdot \beta \in (t, t + \varepsilon]$. Hence,

$$Z \cdot \beta \in (t, t + \varepsilon] - \alpha \,,$$

i.e., Z falls into an interval $I(y_1, \ldots, y_{n-1})$ of length at most $\varepsilon / (\max\left\{1, \frac{nt}{2}\right\} \cdot \delta / n) = n\varepsilon / (\max\left\{1, \frac{nt}{2}\right\} \cdot \delta)$ that only depends on the realizations $y_1, \ldots, y_{n-1}$ of $Y_1, \ldots, Y_{n-1}$. Let $B^{-1}_{i,t,\varepsilon}$ denote the event that Z falls into the interval $I(Y_1, \ldots, Y_{n-1})$. We showed that $E^{-1}_{i,t,\varepsilon} \subseteq B^{-1}_{i,t,\varepsilon}$. Consequently,

$$\mathbf{Pr}\left[E^{-1}_{i,t,\varepsilon}\right] \leq \mathbf{Pr}\left[B^{-1}_{i,t,\varepsilon}\right] \leq \frac{2\sqrt{n} n \varepsilon \phi}{\max\left\{1, \frac{nt}{2}\right\}} \leq \frac{2n^{3/2} \varepsilon \phi}{\max\left\{1, \frac{nt}{2}\right\} \cdot \delta} \,,$$

where the second inequality is due to Theorem 3.3 for the orthogonal matrix Q. □

Lemma 3.18. *For any interval I let X^{-1}_I denote the number of slopes of $R_{w,c}$ that lie in the interval I. Then*

$$\mathbf{E}\left[X^{-1}_{(0,1/n]}\right] \leq \frac{2m\sqrt{n}\phi}{\delta} \,.$$

Proof. As in the proof of Lemma 3.15 we define for $t \in \mathbb{R}$ and $\varepsilon > 0$ the random variable $Z^{-1}_{t,\varepsilon}$ that indicates whether $R_{w,c}$ has a slope in the interval $(t, t + \varepsilon]$ or not. For any integer $k \geq 1$ we obtain

$$\begin{aligned}
\mathbf{E}\left[X^{-1}_{\left(0,\frac{1}{n}\right]}\right] &\leq \sum_{i=0}^{k-1} \mathbf{E}\left[Z^{-1}_{i\cdot\frac{1}{kn},\frac{1}{kn}}\right] + \mathbf{Pr}\left[\mathcal{F}^{-1}_{\frac{1}{kn}}\right] \cdot m^n \\
&= \sum_{i=0}^{k-1} \mathbf{Pr}\left[A^{-1}_{i\cdot\frac{1}{kn},\frac{1}{kn}}\right] + \mathbf{Pr}\left[\mathcal{F}^{-1}_{\frac{1}{kn}}\right] \cdot m^n \\
&\leq \sum_{i=0}^{k-1} \frac{2mn^{3/2}\phi}{kn\delta} + \mathbf{Pr}\left[\mathcal{F}^{-1}_{\frac{1}{k2^\ell\sqrt{n}}}\right] \cdot m^n = \frac{2m\sqrt{n}\phi}{\delta} + \mathbf{Pr}\left[\mathcal{F}^{-1}_{\frac{1}{k2^\ell\sqrt{n}}}\right] \cdot m^n \,.
\end{aligned}$$

The second inequality stems from Lemma 3.17. Now the lemma follows because the bound holds for any integer $k \geq 1$ and $\mathbf{Pr}\left[\mathcal{F}^{-1}_\varepsilon\right] \to 0$ for $\varepsilon \to 0$ in accordance with Lemma 3.16. □

The following corollary directly implies Theorem 3.7.

Corollary 3.19. *The expected number of slopes of $R = R_{c,w}$ is*

$$\mathbf{E}\left[X_{(0,\infty)}\right] = \frac{4mn^2}{\delta^2} + \frac{2m\sqrt{n}\phi}{\delta} \,.$$

Proof. We divide the interval $(0,\infty)$ into the subintervals $(0,n]$ and (n,∞). Using Lemma 3.15, Lemma 3.18, and linearity of expectation we obtain

$$\begin{aligned}
\mathbf{E}\left[X_{(0,\infty)}\right] &= \mathbf{E}\left[X_{(0,n]}\right] + \mathbf{E}\left[X_{(n,\infty)}\right] = \mathbf{E}\left[X_{(0,n]}\right] + \mathbf{E}\left[X^{-1}_{\left(0,\frac{1}{n}\right]}\right] \\
&\leq \frac{4mn^2}{\delta^2} + \frac{2m\sqrt{n}\phi}{\delta} \,.
\end{aligned}$$

In the second step we have exploited that by definition $X_{(a,b)} = X^{-1}_{(1/b,1/a)}$ for any interval (a,b). □

3.5 Running Time

So far we have only discussed the number of pivots. In this section calculate the actual running time of our algorithm. For an initial basic feasible solution x_0 the repeated shadow vertex algorithm repeats the following three steps until an optimal solution is found. Initially let $P' = P$.

Step 1: Run the shadow vertex algorithm for the linear program $\max\{c^{\mathrm{T}}x \mid x \in P'\}$, where $c = \mathrm{pert}(c_0, \phi)$. We will denote this linear program by LP'.

Step 2: Let x_c denote the returned vertex in Step 1, which is optimal for the objective function $c^{\mathrm{T}}x$. Identify an element a'_i of x_c that is in common with the optimal basis.

Step 3: Calculate an orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ that rotates a'_i into the first unit vector e_1 as described in Section 3.3.1 and set LP' to the projection of the current LP' onto the orthogonal complement. Let P' denote the polyhedron of feasible solutions of LP'.

First note that the three steps are repeated at most n times during the algorithm. In Step 1 the shadow vertex algorithm is run once. Step 1 to Step 4 of Algorithm 2 can be performed in time $O(m)$ as we assumed P to be non-degenerate (this implies P' to be non-degenerate in each further step). Step 5 and Step 6 can be implemented with strongly polynomial running time in a tableau form, described in [15]. The tableau can be set up in time $O((m-d)d^3) = O(mn^3)$ where d is the dimension of P'. By Theorem 1 of [15] we can identify for a vertex on a path the row which leaves the basis and the row which is added to the basis in order to move to the next vertex in time $O(m)$ using the tableau. After that, the tableau has to be updated. This can be done in $O((m-d)d) = O(mn)$ steps. Using this and Theorem 3.7 we can compute the path from x_0 to x_c in expected time $O(mn^3 + mn \cdot (\frac{mn^2}{\delta^2} + \frac{m\sqrt{n}\phi}{\delta})) = O(\frac{m^2n^3}{\delta^2} + \frac{m^2n^{3/2}\phi}{\delta})$. Using that $\phi \leq \frac{8n^{3/2}}{\delta}$, as discussed above, yields a running time of $O(\frac{m^2n^3}{\delta^2})$.

Once we have calculated the basis of x_c we can easily compute the element a_i of the basis that is also an element of the optimal basis. Assume the rows $a'_1, \ldots, a'_n$ are the basis of x_c. As mentioned in Section 3.3.2 we can solve the system of linear equations $[a'_1, \ldots, a'_n]\mu = c$ and choose the row for which the coefficient μ_i is maximal. Then a'_i is part of the optimal basis. As a consequence, Step 2 can be performed in time $O(n^3)$. Moreover solving a system of linear equations is possible in strongly polynomial time using Gaussian elimination.

In Step 3, we compute an orthogonal matrix $Q \in \mathbb{R}^{d \times d}$ such that $e_1 Q = a_i$. Since Q is orthogonal we obtain the equation $e_1 = a_i Q^{\mathrm{T}}$. It is clear that the first row of Q is given by a_i. Thus, it is sufficient to compute an orthonormal basis including a_i. This is possible in strongly polynomial time $O(d^3) = O(n^3)$ using the Gram-Schmidt process.

Since all Steps are repeated in this order at most n times we obtain a running time $O(\frac{m^2n^4}{\delta^2})$ for the repeated shadow vertex algorithm.

Theorem 3.20. *The repeated shadow vertex algorithm has a running time of* $O(\frac{m^2n^4}{\delta^2})$.

The entries of both c and λ in Algorithm 2 are continuous random variables. In practice it is, however, more realistic to assume that we can draw a finite number of random bits. In Appendix 3.5.1 we will show that our algorithm only needs to draw $\mathrm{poly}(\log m, n, \log(1/\delta))$ random bits in order to obtain the expected running time stated in Theorem 1.8 if δ (or a good lower bound for it) is known. However, if the parameter δ is not known upfront and only discrete random variables with a finite precision can be drawn, we have to modify the shadow vertex algorithm. This will give us an additional factor of $O(n)$ in the expected running time.

3.5.1 An Upper Bound on the Number of Random Bits

For our analysis we assumed that we can draw continuous random variables. In practice it is, however, more realistic to assume that we can draw a finite number of random bits.

In this section we will show that our algorithm only needs to draw $\operatorname{poly}(\log m, n, \log(1/\delta))$ bits in order to obtain the expected running time stated in Theorem 1.8. However, if the parameter δ is not known to our algorithm, we have to modify the shadow vertex algorithm. This will give us an additional factor of $O(n)$ in the expected running time.

Let us assume that we want to approximate a uniform random draw X from the interval $[0,1)$ with k random bits $Y_1, \dots, Y_k \in \{0,1\}$. (A draw from an arbitrary interval $[a,b)$ can be simulated by drawing a random variable from $[0,1)$ and then applying the affine linear function $x \mapsto a + (b-a) \cdot x$.) We consider the random variable $Z = \sum_{\ell=1}^{k} Y_\ell \cdot 2^{-\ell}$. We observe that the random variable Z has the same distribution as the random variable $g(X)$, where $g(x) = \lfloor x \cdot 2^k \rfloor / 2^k$. Note that $|g(X) - X| \leq 2^{-k}$. Hence, instead of considering discrete variables and going through the whole analysis again, we will argue that, with high probability, the number of slopes of the shadow vertex polygon does not change if each random variable is perturbed by not more than a sufficiently small ε. If we have proven such a statement, this implies that we can approximate our continuous uniform random draws as discussed above by using $O\big(\log(1/\varepsilon)\big)$ bits for each draw. Recall that our algorithm draws two random vectors $\lambda \in (0,1]^n$ and $c \in [-1,1]^n$ that we have to deal with in this section.

For a vector $x \in \mathbb{R}^n$ and a real $\varepsilon > 0$ let $U_\varepsilon(x) \subseteq [-1,1]^n$ denote the set of vectors $x' \in [-1,1]^n$ for which $\|x' - x\|_\infty \leq \varepsilon$, that is, x' and x differ in each component by at most ε. In the remainder let us only consider values $\varepsilon \in (0,1]$.

Whenever a vector $c \in [-1,1]^n$ and a vector $\hat{c} \in U_\varepsilon(c)$ are defined, then by Δ_c we refer to the difference $\Delta_c := \hat{c} - c$. Observe that $\|\Delta_c\| \leq \sqrt{n}\varepsilon$. The same holds for the vectors $\lambda \in (0,1]^n$, $\hat{\lambda} \in U_\varepsilon(\lambda)$, and $\Delta_\lambda := \hat{\lambda} - \lambda$. When the vectors λ and $\hat{\lambda}$ are defined, then the vectors w and $\hat{w}$ are defined as $w := -[u_1, \dots, u_n] \cdot \lambda$ and $\hat{w} := -[u_1, \dots, u_n] \cdot \hat{\lambda}$ (cf. Algorithm 2). Furthermore, the vector Δ_w is defined as $\Delta_w := \hat{w} - w$. Note that $\|w\| = \|[u_1, \dots, u_n] \cdot \lambda\| \leq \sum_{\ell=1}^{n} \|u_\ell\| \leq n$ as the rows ${u_1}^{\mathrm{T}}, \dots, {u_n}^{\mathrm{T}}$ of matrix A are normalized. Similarly, $\|\hat{w}\| \leq n$ and $\|\Delta_w\| \leq n\varepsilon$. We will frequently make use of these inequalities without discussing their correctness again.

If P denotes the non-degenerate bounded polyhedron $\{x \in \mathbb{R}^n \mid Ax \leq b\}$, then we denote by $V_k(P)$ the set of all k-tuples $(z_1, \dots, z_k)$ of pairwise distinct vertices $z_1, \dots z_k$ of P such that for any $i = 1, \dots, k-1$ the vertices z_i and z_{i+1} are neighbors, that is, they share exactly $n-1$ tight constraints. In other words, $V_k(P)$ contains the set of all simple paths of length $k-1$ of the edge graph of P. Note that $|V_k(P)| \leq \binom{m}{n} \cdot n^{k-1} \leq m^n n^{k-2}$. For our analysis only $V_2(P)$ and $V_3(P)$ are relevant.

The following lemma is an adaption of Lemma A.1 for our needs in this section and follows from Lemma A.1.

Lemma 3.21. *The probability that there exist a pair $(z_1, z_2) \in V_2(P)$ and a vector $\hat{c} \in U_\varepsilon(c)$ for which $\hat{c}^{\mathrm{T}} \cdot (z_2 - z_1) = 0$ is bounded from above by $2m^n n^{3/2} \varepsilon \phi$.*

Proof. Let $c \in [-1,1]^n$ be a vector such that there exists a vector $\hat{c} \in U_\varepsilon(c)$ for which $\hat{c}^{\mathrm{T}} \cdot (z_2 - z_1) = 0$ for an appropriate pair $(z_1, z_2) \in V_2(P)$. Then

$$\begin{aligned} |c^{\mathrm{T}} \cdot (z_2 - z_1)| &= |\hat{c}^{\mathrm{T}} \cdot (z_2 - z_1) - {\Delta_c}^{\mathrm{T}} \cdot (z_2 - z_1)| \\ &\leq \|\Delta_c\| \cdot \|z_2 - z_1\| \\ &\leq \sqrt{n}\varepsilon \cdot \|z_2 - z_1\| \,. \end{aligned}$$

In accordance with Lemma A.1, the probability of this event is bounded from above by $2m^n n^{3/2}\varepsilon\phi$. □

A similar statement as Lemma 3.21 can be made for the objective w. However, for our purpose we need a slightly stronger statement.

Lemma 3.22. *The probability that there exist a pair $(z_1, z_2) \in V_2(P)$ and a vector $\hat{\lambda} \in U_\varepsilon(\lambda)$ for which $|\hat{w}^{\mathrm{T}} \cdot (z_2 - z_1)| \le n\varepsilon^{1/3} \cdot \|z_2 - z_1\|$, where $\hat{w} = -[u_1, \dots, u_n] \cdot \hat{\lambda}$ (cf. Algorithm 2), is bounded from above by $4m^n n^2 \varepsilon^{1/3}/\delta$.*

Proof. Fix a pair $(z_1, z_2) \in V_2(P)$ and let $\Delta_z := z_2 - z_1$. Without loss of generality let us assume that $\|\Delta_z\| = 1$. The event $\hat{w}^{\mathrm{T}}\Delta_z \in [-n\varepsilon^{1/3}, n\varepsilon^{1/3}]$ is equivalent to

$$w^{\mathrm{T}}\Delta_z \in [-n\varepsilon^{1/3}, n\varepsilon^{1/3}] - {\Delta_w}^{\mathrm{T}}\Delta_z \,.$$

This interval is a subinterval of $[-2n\varepsilon^{1/3}, 2n\varepsilon^{1/3}]$ as

$$|{\Delta_w}^{\mathrm{T}}\Delta_z| \le \|\Delta_w\| \cdot \|\Delta_z\| \le n\varepsilon \cdot 1 \le n\varepsilon^{1/3}$$

when recalling that $\varepsilon \le 1$. Since

$$\begin{aligned} w^{\mathrm{T}}\Delta_z \in [-2n\varepsilon^{1/3}, 2n\varepsilon^{1/3}] &\iff (U\lambda)^{\mathrm{T}}\Delta_z \in [-2n\varepsilon^{1/3}, 2n\varepsilon^{1/3}] \\ &\iff \lambda^{\mathrm{T}} y \in [-2n\varepsilon^{1/3}, 2n\varepsilon^{1/3}] \end{aligned}$$

for $U = [u_1, \dots, u_n]$ and $y = U^{\mathrm{T}}\Delta_z$, in the next part of this proof we will derive a lower bound for $\|y\|$. Particularly, we will show that $\|y\| \ge \delta/\sqrt{n}$.

Let $M := [m_1, \dots, m_n] := (U^{\mathrm{T}})^{-1}$. Due to $\Delta_z = My$, we obtain $1 = \|\Delta_z\| \le \|M\| \cdot \|y\|$, which implies $\|y\| \ge 1/\|M\|$. In accordance with Lemma 3.2, Claim 1, we obtain

$$\max_{k \in [n]} \|m_k\| = \frac{1}{\delta(u_1, \dots, u_n)} \le \frac{1}{\delta} \,.$$

Consequently,

$$\|Mx\| \le \sum_{k=1}^{n} \|m_k\| \cdot |x_k| \le \sum_{k=1}^{n} \frac{1}{\delta} \cdot |x_k| = \frac{\|x\|_1}{\delta} \le \frac{\sqrt{n} \cdot \|x\|}{\delta}$$

for any vector $x \ne 0$, i.e., $\|M\| = \sup_{x \ne 0} \|Mx\|/\|x\| \le \sqrt{n}/\delta$. Summarizing the previous observations, we obtain $\|y\| \ge 1/\|M\| \ge \delta/\sqrt{n}$.

For the last part of the proof we observe that there exists an index $i \in [n]$ such that $|y_i| \ge \delta/n$. We apply the principle of deferred decisions an assume that all coefficients λ_j for $j \ne i$ are fixed arbitrarily. By the chain of equivalences

$$\begin{aligned} &\lambda^{\mathrm{T}} y \in [-2n\varepsilon^{1/3}, 2n\varepsilon^{1/3}] \\ \iff &\sum_{k=1}^{n} \frac{\lambda_k \cdot y_k}{y_i} \in \left[-\frac{2n\varepsilon^{1/3}}{|y_i|}, \frac{2n\varepsilon^{1/3}}{|y_i|}\right] \\ \iff &\lambda_i \in \left[-\frac{2n\varepsilon^{1/3}}{|y_i|}, \frac{2n\varepsilon^{1/3}}{|y_i|}\right] - \sum_{k \ne i} \frac{\lambda_k \cdot y_k}{y_i} \end{aligned}$$

we see that the event $\lambda^{\mathrm{T}} y \in [-2n\varepsilon^{1/3}, 2n\varepsilon^{1/3}]$ occurs if and only if the coefficient λ_i, which we did not fix, falls into a certain fixed interval of length $4n\varepsilon^{1/3}/|y_i|$. The probability for this to happen is at most $4n\varepsilon^{1/3}/|y_i| \leq 4n^2\varepsilon^{1/3}/\delta$. The claim follows by applying a union bound over all pairs $(z_1, z_2) \in V_2(P)$, which gives us the additional factor of m^n. □

The next observation characterizes the situation when the projections of two linearly independent vectors in $\mathbb{R}^n$ are projected onto two linearly dependent vectors in $\mathbb{R}^2$ by the function $x \mapsto (\hat{c}^{\mathrm{T}} x, \hat{w}^{\mathrm{T}} x)$.

Observation 3.23. *Let $(z_1, z_2, z_3) \in V_3(P)$, let $\Delta_1 := z_2 - z_1$ and $\Delta_2 := z_3 - z_2$, and let $\hat{c}, \hat{w} \in \mathbb{R}^n$ be vectors for which $\hat{w}^{\mathrm{T}}\Delta_1 \neq 0$, $\hat{w}^{\mathrm{T}}\Delta_2 \neq 0$, and*

$$\frac{\hat{w}^{\mathrm{T}}\Delta_1}{\hat{c}^{\mathrm{T}}\Delta_1} = \frac{\hat{w}^{\mathrm{T}}\Delta_2}{\hat{c}^{\mathrm{T}}\Delta_2}\,.$$

Then $\hat{c}^{\mathrm{T}} x = 0$ for $x := \Delta_1 - \mu \cdot \Delta_2$, where $\mu = \hat{w}^{\mathrm{T}}\Delta_1/\hat{w}^{\mathrm{T}}\Delta_2$.

Note that, by the definition of x, the equation $\hat{w}^{\mathrm{T}} x = 0$ trivially holds. For the equation $\hat{c}^{\mathrm{T}} x = 0$ we require that the projections of Δ_1 and Δ_2 are linearly dependent as it is assumed in Observation 3.23. Furthermore, let us remark that in the formulation above we allow $\hat{c}^{\mathrm{T}}\Delta_1 = 0$ or $\hat{c}^{\mathrm{T}}\Delta_2 = 0$ using the convention $x/0 = +\infty$ for $x > 0$ and $x/0 = -\infty$ for $x < 0$.

Proof. The claim follows from

$$\begin{aligned} \hat{c}^{\mathrm{T}} x = \hat{c}^{\mathrm{T}}\Delta_1 - \mu \cdot \hat{c}^{\mathrm{T}}\Delta_2 &= \hat{c}^{\mathrm{T}}\Delta_2 \cdot \frac{\hat{w}^{\mathrm{T}}\Delta_1}{\hat{w}^{\mathrm{T}}\Delta_2} - \mu \cdot \hat{c}^{\mathrm{T}}\Delta_2 \\ &= \hat{c}^{\mathrm{T}}\Delta_2 \cdot \frac{\mu \cdot \hat{w}^{\mathrm{T}}\Delta_2}{\hat{w}^{\mathrm{T}}\Delta_2} - \mu \cdot \hat{c}^{\mathrm{T}}\Delta_2 = 0\,. \end{aligned}$$

□

We are now able to prove an analog of Lemma 3.9.

Lemma 3.24. *The probability that there exist a triple $(z_1, z_2, z_3) \in V_3(P)$ and vectors $\hat{\lambda} \in U_\varepsilon(\lambda)$ and $\hat{c} \in U_\varepsilon(c)$ for which*

$$\frac{\hat{w}^{\mathrm{T}}\Delta_1}{\hat{c}^{\mathrm{T}}\Delta_1} = \frac{\hat{w}^{\mathrm{T}}\Delta_2}{\hat{c}^{\mathrm{T}}\Delta_2}\,,$$

where $\Delta_1 := z_2 - z_1$, $\Delta_2 := z_3 - z_2$, and $\hat{w} = -[u_1, \ldots, u_n] \cdot \hat{\lambda}$, is bounded from above by $12m^n n^2 \varepsilon^{1/3}\phi/\delta$.

Proof. Let us introduce the following events:

- With event A we refer to the event stated in Lemma 3.24.
- Event B occurs if there exist a pair $(z_1, z_2) \in V_2(P)$ and a vector $\hat{\lambda} \in U_\varepsilon(\lambda)$ such that $|\hat{w}^{\mathrm{T}} \cdot (z_2 - z_1)| \leq n\varepsilon^{1/3} \cdot \|z_2 - z_1\|$ (cf. Lemma 3.22).
- Event C occurs if there is a triple $(z_1, z_2, z_3) \in V_3(P)$ such that $|c^{\mathrm{T}} x| \leq (4\sqrt{n}\varepsilon^{1/3}/\delta) \cdot \|x\|$, where $x = x(w, z_1, z_2, z_3) := \Delta_1 - \mu \cdot \Delta_2$ for $\Delta_1 := z_2 - z_1$, $\Delta_2 := z_3 - z_2$, and $\mu = w^{\mathrm{T}}\Delta_1/w^{\mathrm{T}}\Delta_2$ if $w^{\mathrm{T}}\Delta_2 \neq 0$ and $\mu = 0$ otherwise (cf. Observation 3.23).

In the first part of the proof we will show that $A \subseteq B \cup C$. For this, it suffices to show that $A \setminus B \subseteq C$. Let us consider realizations $w \in (0,1]^n$ and $c \in [-1,1]^n$ for which event A occurs, but not event B. Let $(z_1, z_2, z_3) \in V_3(P)$, $\hat{\lambda} \in U_\varepsilon(\lambda)$, and $\hat{c} \in U_e(c)$ be the vectors mentioned in the definition of event A. Our goal is to show that $|c^{\mathrm{T}} x| \leq (4\sqrt{n}\varepsilon^{1/3}/\delta) \cdot \|x\|$ for $x = x(w, z_1, z_2, z_3)$. As event B does not occur, we know that

$$\begin{aligned} |w^{\mathrm{T}}\Delta_1| &\geq n\varepsilon^{1/3} \cdot \|\Delta_1\|\,, & |\hat{w}^{\mathrm{T}}\Delta_1| &\geq n\varepsilon^{1/3} \cdot \|\Delta_2\|\,, \\ |w^{\mathrm{T}}\Delta_2| &\geq n\varepsilon^{1/3} \cdot \|\Delta_2\|\,, \quad \text{and} & |\hat{w}^{\mathrm{T}}\Delta_2| &\geq n\varepsilon^{1/3} \cdot \|\Delta_2\|\,. \end{aligned}$$

Furthermore, note that

$$|\hat{w}^{\mathrm{T}}\Delta_1 - w^{\mathrm{T}}\Delta_1| \leq \|\Delta_w\| \cdot \|\Delta_1\| \leq n\varepsilon \cdot \|\Delta_1\|$$

and, similarly,

$$|\hat{w}^{\mathrm{T}}\Delta_2 - w^{\mathrm{T}}\Delta_2| \leq n\varepsilon \cdot \|\Delta_2\|\,.$$

Therefore,

$$\begin{aligned} |\hat{w}^{\mathrm{T}}\Delta_1 - w^{\mathrm{T}}\Delta_1| &\leq n\varepsilon \cdot \|\Delta_1\| \leq \varepsilon^{2/3} \cdot |w^{\mathrm{T}}\Delta_1| \quad \text{and} \\ |\hat{w}^{\mathrm{T}}\Delta_2 - w^{\mathrm{T}}\Delta_2| &\leq n\varepsilon \cdot \|\Delta_2\| \leq \varepsilon^{2/3} \cdot |\hat{w}^{\mathrm{T}}\Delta_2|\,, \end{aligned}$$

and, consequently

$$\begin{aligned} \frac{|\hat{w}^{\mathrm{T}}\Delta_1|}{|\hat{w}^{\mathrm{T}}\Delta_2|} &\leq \frac{(1+\varepsilon^{2/3}) \cdot |w^{\mathrm{T}}\Delta_1|}{\frac{1}{1+\varepsilon^{2/3}} \cdot |w^{\mathrm{T}}\Delta_2|} = (1+\varepsilon^{2/3})^2 \cdot \frac{|w^{\mathrm{T}}\Delta_1|}{|w^{\mathrm{T}}\Delta_2|} \leq (1+3\varepsilon^{2/3}) \cdot \frac{|w^{\mathrm{T}}\Delta_1|}{|w^{\mathrm{T}}\Delta_2|} \quad \text{and} \\ \frac{|\hat{w}^{\mathrm{T}}\Delta_1|}{|\hat{w}^{\mathrm{T}}\Delta_2|} &\geq \frac{(1-\varepsilon^{2/3}) \cdot |w^{\mathrm{T}}\Delta_1|}{\frac{1}{1-\varepsilon^{2/3}} \cdot |w^{\mathrm{T}}\Delta_2|} = (1-\varepsilon^{2/3})^2 \cdot \frac{|w^{\mathrm{T}}\Delta_1|}{|w^{\mathrm{T}}\Delta_2|} \geq (1-3\varepsilon^{2/3}) \cdot \frac{|w^{\mathrm{T}}\Delta_1|}{|w^{\mathrm{T}}\Delta_2|}\,. \end{aligned}$$

Here we again used $\varepsilon \leq 1$. Observe that both, $\hat{w}^{\mathrm{T}}\Delta_1$ and $w^{\mathrm{T}}\Delta_1$, as well as $\hat{w}^{\mathrm{T}}\Delta_2$ and $w^{\mathrm{T}}\Delta_2$, have the same sign, since their absolute values are larger than $n\varepsilon^{1/3} \cdot \|\Delta_1\|$ and $n\varepsilon^{1/3} \cdot \|\Delta_2\|$, but their difference is at most $n\varepsilon \cdot \|\Delta_1\|$ and $n\varepsilon\|\Delta_2\|$, respectively. Hence,

$$\left| \frac{\hat{w}^{\mathrm{T}}\Delta_1}{\hat{w}^{\mathrm{T}}\Delta_2} - \frac{w^{\mathrm{T}}\Delta_1}{w^{\mathrm{T}}\Delta_2} \right| = \left| \left| \frac{\hat{w}^{\mathrm{T}}\Delta_1}{\hat{w}^{\mathrm{T}}\Delta_2} \right| - \left| \frac{w^{\mathrm{T}}\Delta_1}{w^{\mathrm{T}}\Delta_2} \right| \right| \leq 3\varepsilon^{2/3} \cdot \frac{|w^{\mathrm{T}}\Delta_1|}{|w^{\mathrm{T}}\Delta_2|}\,.$$

As event A occurs, but not event B, Observation 3.23 yields $\hat{c}^{\mathrm{T}} x(\hat{w}, z_1, z_2, z_3) = 0$. With the previous inequality we obtain

$$\begin{aligned}
|\hat{c}^{\mathrm{T}}x(w,z_1,z_2,z_3)| &= \left|\hat{c}^{\mathrm{T}}\cdot\left(x(w,z_1,z_2,z_3)-x(\hat{w},z_1,z_2,z_3)\right)\right| \\
&\le \|\hat{c}\|\cdot\|x(w,z_1,z_2,z_3)-x(\hat{w},z_1,z_2,z_3)\| \\
&= \|\hat{c}\|\cdot\left|\frac{w^{\mathrm{T}}\Delta_1}{w^{\mathrm{T}}\Delta_2}-\frac{\hat{w}^{\mathrm{T}}\Delta_1}{\hat{w}^{\mathrm{T}}\Delta_2}\right|\cdot\|\Delta_2\| \\
&\le \sqrt{n}\cdot 3\varepsilon^{2/3}\cdot\frac{|w^{\mathrm{T}}\Delta_1|}{|w^{\mathrm{T}}\Delta_2|}\cdot\|\Delta_2\| \\
&\le \sqrt{n}\cdot 3\varepsilon^{2/3}\cdot\frac{\|w\|\cdot\|\Delta_1\|}{n\varepsilon^{1/3}\cdot\|\Delta_2\|}\cdot\|\Delta_2\| \\
&\le \sqrt{n}\cdot 3\varepsilon^{2/3}\cdot\frac{n\cdot\|\Delta_1\|}{n\varepsilon^{1/3}\cdot\|\Delta_2\|}\cdot\|\Delta_2\| \\
&= 3\sqrt{n}\varepsilon^{1/3}\cdot\|\Delta_1\|\,.
\end{aligned}$$

In the remainder of this proof, with x we refer to the vector $x(w,z_1,z_2,z_3)$ (and not to, e.g., $x(\hat{w},z_1,z_2,z_3)$). Now we show that $\|x\|\ge\delta\cdot\|\Delta_1\|$. For this, let ${a_i}^{\mathrm{T}}$ be a row of matrix A for which ${a_i}^{\mathrm{T}}z_1<b_i$, but ${a_i}^{\mathrm{T}}z_2={a_i}^{\mathrm{T}}z_3=b_i$, i.e., the i^{th} constraint is tight for z_2 and z_3, but not for z_1. Such a constraint exists as z_1 and z_3 are distinct neighbors of z_2. Consequently, ${a_i}^{\mathrm{T}}\Delta_1>0$ and ${a_i}^{\mathrm{T}}\Delta_2=0$. Hence,

$$|{a_i}^{\mathrm{T}}x| = |{a_i}^{\mathrm{T}}\cdot(\Delta_1-\mu\cdot\Delta_2)| = |{a_i}^{\mathrm{T}}\cdot\Delta_1|\ge\delta\cdot\|\Delta_1\|\,,$$

where the last inequality is due to Lemma 3.2, Claim 3. As $\|a_i\|=1$, we obtain

$$\|x\|\ge\frac{|{a_i}^{\mathrm{T}}x|}{\|a_i\|}=|{a_i}^{\mathrm{T}}x|\ge\delta\cdot\|\Delta_1\|\,.$$

Summarizing the previous observations yields

$$|\hat{c}^{\mathrm{T}}x|\le 3\sqrt{n}\varepsilon^{1/3}\cdot\|\Delta_1\|\le\frac{3\sqrt{n}\varepsilon^{1/3}}{\delta}\cdot\|x\|\,.$$

Now that we have bounded $|\hat{c}^{\mathrm{T}}x|$ from above, we easily get an upper bound for $|c^{\mathrm{T}}x|$. Since

$$|c^{\mathrm{T}}x-\hat{c}^{\mathrm{T}}x|\le\|\Delta_c\|\cdot\|x\|\le\sqrt{n}\varepsilon\cdot\|x\|\,,$$

we obtain

$$|c^{\mathrm{T}}x|\le|\hat{c}^{\mathrm{T}}x|+|c^{\mathrm{T}}x-\hat{c}^{\mathrm{T}}x|\le\frac{3\sqrt{n}\varepsilon^{1/3}}{\delta}\cdot\|x\|+\sqrt{n}\varepsilon\cdot\|x\|\le\frac{4\sqrt{n}\varepsilon^{1/3}}{\delta}\cdot\|x\|\,,$$

i.e., event C occurs.

In the second part of the proof we show that $\mathbf{Pr}[C]\le 8m^n n^2\varepsilon^{1/3}\phi/\delta$. Due to $A\subseteq B\cup C$, $\phi\ge 1$, and Lemma 3.22, it then follows that

$$\mathbf{Pr}[A]\le 4m^n n^2\varepsilon^{1/3}/\delta+8m^n n^2\varepsilon^{1/3}\phi/\delta\le 12m^n n^2\varepsilon^{1/3}\phi/\delta\,.$$

Let $(z_1, z_2, z_3) \in V_3(P)$ be a triple of vertices of P. We apply the principle of deferred decisions twice: First, we assume that λ has already been fixed arbitrarily. Hence, the vector $x = x(w, z_1, z_2, z_3) \neq 0$ is also fixed. Let $z = (1/\|x\|) \cdot x$ be the normalization of x. As $|c^{\mathrm{T}}x| \leq (4\sqrt{n}\varepsilon^{1/3}/\delta) \cdot \|x\|$ holds if and only if $|c^{\mathrm{T}}z| \leq 4\sqrt{n}\varepsilon^{1/3}/\delta$, we will analyze the probability of the latter event.

There exists an index i such that $|z_i| \geq 1/\sqrt{n}$. Now we again apply the principle of deferred decisions an assume that all coefficients c_j for $j \neq i$ are fixed arbitrarily. Then

$$|c^{\mathrm{T}}z| \leq 4\sqrt{n}\varepsilon^{1/3}/\delta \iff \sum_{j=1}^{n} c_j \cdot \frac{z_j}{z_i} \in \left[-\frac{4\sqrt{n}\varepsilon^{1/3}}{\delta \cdot |z_i|}, \frac{4\sqrt{n}\varepsilon^{1/3}}{\delta \cdot |z_i|}\right]$$
$$\iff c_i \in \left[-\frac{4\sqrt{n}\varepsilon^{1/3}}{\delta \cdot |z_i|}, \frac{4\sqrt{n}\varepsilon^{1/3}}{\delta \cdot |z_i|}\right] - \sum_{j \neq i} c_j \cdot \frac{z_j}{z_i}.$$

Hence, the random coefficient c_i must fall into a fixed interval of length $8\sqrt{n}\varepsilon^{1/3}/(\delta \cdot |z_i|)$. The probability for this to happen is at most

$$\frac{8\sqrt{n}\varepsilon^{1/3}}{\delta \cdot |z_i|} \cdot \phi \leq \frac{8\sqrt{n}\varepsilon^{1/3}}{\delta \cdot \frac{1}{\sqrt{n}}} \cdot \phi = \frac{8n\varepsilon^{1/3}\phi}{\delta}.$$

A union bound over all triples $(z_1, z_2, z_3) \in V_3(P)$ gives the additional factor of $V_3(P) \leq m^n n$. □

Lemma 3.25. *Let us consider the shadow vertex algorithm given as Algorithm 2 for $\phi \geq \sqrt{n}$. If we replace the draw of each continuous random variable by the draw of at least*

$$B(m, n, \phi, \delta) := \lceil 6n \log_2 m + 6 \log_2 n + 3 \log_2 \phi + 3 \log_2(1/\delta) + 12 \rceil$$

random bits as described earlier in this section, then the expected number of pivots is $O(\frac{mn^2}{\delta^2} + \frac{m\sqrt{n}\phi}{\delta})$.

Proof. As discussed in the beginning of this section, instead of drawing k random bits to simulate a uniform random draw from an interval $[a, b)$, we can draw a uniform random variable X from $[0, 1)$ and apply the function $g(X) = h(\lfloor X \cdot 2^k \rfloor / 2^k)$ for $h(x) = a + (b - a) \cdot x$ to obtain a discrete random variable with the same distribution. Observe, that $|X - g(X)| \leq (b - a)/2^k$. In the shadow vertex algorithm all intervals are of length 1 or of length $1/\phi \leq 1$. Hence, $|X - g(X)| \leq 2^{-k}$. As we use $k \geq B(m, n, \phi, \delta)$ bits for each draw, we obtain $g(X) \in U_\varepsilon(X)$ for

$$\varepsilon = 2^{-B(m,n,\phi,\delta)} \leq \frac{\delta^3}{2^{12} m^{6n} n^6 \phi^3} = \left(\frac{\delta}{16 m^{2n} n^2 \phi}\right)^3.$$

Now let c and λ denote the continuous random vectors and let $\bar{c} \in U_\varepsilon(c)$ and $\bar{\lambda} \in U_\varepsilon(\lambda)$ denote the discrete random vectors obtained from c and λ as described above. Furthermore, let $w = -[u_1, \ldots, u_n] \cdot \lambda$ and $\bar{w} = -[u_1, \ldots, u_n] \cdot \bar{\lambda}$. We introduce the event D which occurs if one of the following holds:

1. There exists a pair $(z_1, z_2) \in V_2(P)$ such that $c^{\mathrm{T}}z_1$ and $c^{\mathrm{T}}z_2$ are not in the same relation as $\bar{c}^{\mathrm{T}}z_1$ and $\bar{c}^{\mathrm{T}}z_2$ or $c^{\mathrm{T}}z_1 = c^{\mathrm{T}}z_2$ or $\bar{c}^{\mathrm{T}}z_1 = \bar{c}^{\mathrm{T}}z_2$.

2. There exists a triple $(z_1, z_2, z_3) \in V_3(P)$ such that $\frac{w^{\mathrm{T}} \cdot (z_2 - z_1)}{c^{\mathrm{T}} \cdot (z_2 - z_1)}$ and $\frac{w^{\mathrm{T}} \cdot (z_3 - z_2)}{c^{\mathrm{T}} \cdot (z_3 - z_2)}$ are not in the same relation as $\frac{\bar{w}^{\mathrm{T}} \cdot (z_2 - z_1)}{\bar{c}^{\mathrm{T}} \cdot (z_2 - z_1)}$ and $\frac{\bar{w}^{\mathrm{T}} \cdot (z_3 - z_2)}{\bar{c}^{\mathrm{T}} \cdot (z_3 - z_2)}$.

Here, a and b being in the same relation as $\bar{a}$ and $\bar{b}$ means that $\operatorname{sgn}(a - b) = \operatorname{sgn}(\bar{a} - \bar{b})$, where $\operatorname{sgn}(x) = -1$ for $x < 0$, $\operatorname{sgn}(x) = 0$ for $x = 0$, and $\operatorname{sgn}(x) = +1$ for $x > 0$.

Let X and $\bar{X}$ denote the number of pivots of the shadow vertex algorithm with continuous random vectors c and λ and with discrete random vectors $\bar{c}$ and $\bar{\lambda}$, respectively. We will first argue that $X = \bar{X}$ if event D does not occur. In both cases, we start in the same vertex x_0. In each vertex x, the algorithm chooses among the neighbors of x with a larger c-value (or $\bar{c}$-value, respectively) the neighbor z with the smallest slope $\frac{w^{\mathrm{T}} \cdot (z - x)}{c^{\mathrm{T}} \cdot (z - x)}$ (or $\frac{\bar{w}^{\mathrm{T}} \cdot (z - x)}{\bar{c}^{\mathrm{T}} \cdot (z - x)}$, respectively). If event D does not occur, then in both cases the same neighbors of x are considered and, additionally, the order of their slopes is the same. Hence, in both cases the same sequence of vertices is considered.

Now let Y be the random variable that takes the value m^n if event D occurs and the value 0 otherwise. Clearly, $\bar{X} \leq X + Y$ and, thus,

$$\mathbf{E}\left[\bar{X}\right] \leq \mathbf{E}\left[X\right] + \mathbf{E}\left[Y\right] \leq O\left(\frac{mn^2}{\delta^2} + \frac{m\sqrt{n}\phi}{\delta}\right) + m^n \cdot \mathbf{Pr}\left[D\right] ,$$

where the last inequality stems from Theorem 3.7. In the remainder of this proof we show that the probability $\mathbf{Pr}\left[D\right]$ of event D is bounded from above by $1/m^n$. For this, let us assume that the first part of the definition of event D is fulfilled for a pair $(z_1, z_2) \in V_2(P)$. If $c^{\mathrm{T}} z_1$ and $c^{\mathrm{T}} z_2$ are not in the same relation as $\bar{c}^{\mathrm{T}} z_1$ and $\bar{c}^{\mathrm{T}} z_2$, then there exists a $\mu \in [0, 1]$ such that

$$\mu \cdot (c^{\mathrm{T}} z_1 - c^{\mathrm{T}} z_2) + (1 - \mu) \cdot (\bar{c}^{\mathrm{T}} z_1 - \bar{c}^{\mathrm{T}} z_2) = 0 .$$

If we consider the vector $\hat{c} := \mu \cdot c + (1 - \mu) \cdot \bar{c} \in U_\varepsilon(c)$, then we obtain

$$\hat{c}^{\mathrm{T}} \cdot (z_2 - z_1) = \mu \cdot c^{\mathrm{T}} \cdot (z_2 - z_1) + (1 - \mu) \cdot \bar{c}^{\mathrm{T}} \cdot (z_2 - z_1) = 0 .$$

Hence, the event described in Lemma 3.21 occurs. This event also occurs if $c^{\mathrm{T}} z_1 = c^{\mathrm{T}} z_2$ or $\bar{c}^{\mathrm{T}} z_1 = \bar{c}^{\mathrm{T}} z_2$.

Let us now assume that the second part of the definition of event D is fulfilled for a triple $(z_1, z_2, z_3) \in V_3(P)$, but not the first one, and let us consider the function $f \colon [0, 1] \to \mathbb{R}$, defined by

$$f(\mu) = \frac{(\mu \cdot w + (1 - \mu) \cdot \bar{w})^{\mathrm{T}} \cdot (z_2 - z_1)}{(\mu \cdot c + (1 - \mu) \cdot \bar{c})^{\mathrm{T}} \cdot (z_2 - z_1)} - \frac{(\mu \cdot w + (1 - \mu) \cdot \bar{w})^{\mathrm{T}} \cdot (z_3 - z_2)}{(\mu \cdot c + (1 - \mu) \cdot \bar{c})^{\mathrm{T}} \cdot (z_3 - z_2)} .$$

The denominators of both fractions are linear in μ and, since the first part of the definition of event D does not hold, the signs for $\mu = 0$ and $\mu = 1$ are the same and different from 0. Hence, both denominators are different from 0 for all $\mu \in [0, 1]$. Consequently, function f is continuous (on $[0, 1]$). As we have

$$f(0) = \frac{\bar{w}^{\mathrm{T}} \cdot (z_2 - z_1)}{\bar{c}^{\mathrm{T}} \cdot (z_2 - z_1)} - \frac{\bar{w}^{\mathrm{T}} \cdot (z_3 - z_2)}{\bar{c}^{\mathrm{T}} \cdot (z_3 - z_2)}$$

and

$$f(1) = \frac{w^{\mathrm{T}} \cdot (z_2 - z_1)}{c^{\mathrm{T}} \cdot (z_2 - z_1)} - \frac{w^{\mathrm{T}} \cdot (z_3 - z_2)}{c^{\mathrm{T}} \cdot (z_3 - z_2)}$$

and these differences have different signs as the second part of the definition of event D is fulfilled, there must be a value $\mu \in [0,1]$ for which $f(\mu) = 0$. This implies

$$\frac{\hat{w}^{\mathrm{T}} \cdot (z_2 - z_1)}{\hat{c}^{\mathrm{T}} \cdot (z_2 - z_1)} = \frac{\hat{w}^{\mathrm{T}} \cdot (z_3 - z_2)}{\hat{c}^{\mathrm{T}} \cdot (z_3 - z_2)}$$

for $\hat{c} := \mu \cdot c + (1-\mu) \cdot \bar{c} \in U_\varepsilon(c)$, $\hat{\lambda} := \mu \cdot \lambda + (1-\mu) \cdot \bar{\lambda} \in U_\varepsilon(\lambda)$, and $\hat{w} := -[u_1, \ldots, u_n] \cdot \hat{\lambda} = \mu \cdot w + (1-\mu) \cdot \bar{w}$. Thus, the event described in Lemma 3.24 occurs.

By applying Lemma 3.21 and Lemma 3.24 we obtain

$$\begin{aligned}\mathbf{Pr}[D] &\leq 2m^n n^{3/2} \varepsilon \phi + \frac{12 m^n n^2 \varepsilon^{1/3} \phi}{\delta} \leq \frac{4 m^n n^2 \varepsilon^{1/3} \phi}{\delta} + \frac{12 m^n n^2 \varepsilon^{1/3} \phi}{\delta} \\ &= \frac{16 m^n n^2 \phi}{\delta} \cdot \varepsilon^{1/3} \leq \frac{1}{m^n}.\end{aligned}$$

This completes the proof. □

Lemma 3.25 states that if we draw $2n \cdot B(m,n,\phi,\delta)$ random bits for the $2n$ components of c and λ, then the expected number of pivots does not increase significantly. We consider now the case that the parameter δ is not known (and also no good lower bound). We will use the fraction $\hat{\delta} = \hat{\delta}(n,\phi) := 2n^{3/2}/\phi$ as an estimate for δ. For the case $\phi > 2n^{3/2}/\delta$, in which the repeated shadow vertex algorithm is guaranteed to yield the optimal solution, this is a valid lower bound for δ. For the case $\phi < 2n^{3/2}/\delta$ this estimate is too large and we would draw too few random bits, leading to a (for our analysis) unpredictable running time behavior of the shadow vertex method. To solve this problem, we stop the shadow vertex method after at most $8n \cdot p(m,n,\phi,\hat{\delta}(n,\phi))$ pivots, where $p(m,n,\phi,\delta) = O\big(\frac{mn^2}{\delta^2} + \frac{m\sqrt{n}\phi}{\delta}\big)$ is the upper bound for the expected number of pivots stated in Lemma 3.25. When the shadow vertex method stops, we assume that the current choice of ϕ is too small (although this does not have to be the case) and restart the repeated shadow vertex algorithm with 2ϕ. Recall that this is the same doubling strategey that is applied when the repeated shadow vertex algorithm yields a non-optimal solution for the original linear program. We call this algorithm the shadow vertex algorithm with random bits.

Theorem 3.26. *The shadow vertex algorithm with random bits solves linear programs with n variables and m constraints satisfying the δ-distance property using $O\big(\frac{mn^4}{\delta^2} \cdot \log\big(\frac{1}{\delta}\big)\big)$ pivots in expectation if a feasible solution is given.*

Note that, in analogy, all other results stated in Theorem 1.8 and Theorem 1.9 also hold for the shadow vertex algorithm with random bits with an additional $O(n)$-factor (or $O(m)$-factor when no feasible solution is given).

Proof. Let us assume that the shadow vertex algorithm with random bits does not find the optimal solution before the first iteration $i^\star$ for which $\phi_{i^\star} > 2n^{3/2}/\delta$. For iterations $i \geq i^\star$ we know that the shadow vertex algorithm will return the optimal solution (or detect, that the linear program is unbounded) if it is not stopped because the number of pivots exceeds $8n \cdot p(m,n,\phi_i,\hat{\delta}(n,\phi_i))$. Due to Markov's inequality, the probability of the latter event is bounded from above by $1/8n$ (for each facet of the optimal solution) because $p(m,n,\phi_i,\hat{\delta}(n,\phi_i)) \geq p(m,n,\phi_i,\delta)$ due to $\hat{\delta}(n,\phi_i) \leq \delta$ and $p(m,n,\phi_i,\delta)$ is an upper bound for the expected number of pivots. As n facets have to be identified in iteration i, the probability that the shadow vertex method stops because of too many pivots is bounded from above by $n \cdot 1/8n = 1/8$. Hence, the expected number of pivots of all iterations $i \geq i^\star$, provided that iteration $i^\star$ is reached, is at most

$$\begin{aligned}
&\sum_{i=i^\star}^{\infty} \left(\frac{1}{8}\right)^{i-i^\star} \cdot \frac{7}{8} \cdot n \cdot 8n \cdot p(m,n,\phi_i,\hat{\delta}(n,\phi_i)) \\
=&7n^2 \cdot \sum_{i=i^\star}^{\infty} \frac{1}{8^{i-i^\star}} \cdot p\left(m,n,\phi_i,\frac{2n^{3/2}}{\phi_i}\right) \\
=&O\left(8^{i^\star} n^2 \cdot \sum_{i=i^\star}^{\infty} \frac{1}{8^i} \cdot \frac{m\sqrt{n}\phi_i}{\frac{2n^{3/2}}{\phi_i}}\right) = O\left(8^{i^\star} n \cdot \sum_{i=i^\star}^{\infty} \frac{1}{8^i} \cdot m\phi_i^2\right) \\
=&O\left(8^{i^\star} n \cdot \sum_{i=i^\star}^{\infty} \frac{1}{8^i} \cdot m \cdot (2^i n^{3/2})^2\right) = O\left(8^{i^\star} n \cdot \sum_{i=i^\star}^{\infty} \frac{1}{2^i} \cdot mn^3\right) \\
=&O(4^{i^\star} mn^4) = O\left(\frac{mn^4}{\delta^2}\right) .
\end{aligned}$$

Some equations require further explanation. The factor $n \cdot 8n \cdot p(m,n,\phi_i,\hat{\delta}(n,\phi_i))$ stems from the fact that we have to identify n facets, and for each we stop after at most $8n \cdot p(m,n,\phi_i,\hat{\delta}(n,\phi_i))$ pivots. The second equation is in accordance with Lemma 3.25, which states that $p(m,n,\phi,\delta) = O(\frac{mn^2}{\delta^2} + \frac{m\sqrt{n}\phi}{\delta})$. As the term mn^2/δ^2 is dominated by the term $m\sqrt{n}\phi/\delta$ when $\phi \geq n^{3/2}/\delta$, it can be omitted in the O-notation for such values. Above we only consider iterations $i \geq i^\star$, i.e., $\phi_i \geq \phi_{i^\star} > 2n^{3/2}/\delta$. The last equation is due to the fact that

$$2^{i^\star-1} n^{3/2} = \phi_{i^\star-1} \leq \frac{2n^{3/2}}{\delta} ,$$

i.e., $2^{i^\star} \leq 4/\delta$ and, hence, $4^{i^\star} = O(1/\delta^2)$.

To finish the proof, we observe that the iterations $i = 1, \ldots, i^\star$ require at most

$$\begin{aligned}
&\sum_{i=1}^{i^\star-1} n \cdot 8n \cdot p(m,n,\phi_i,\hat{\delta}(n,\phi)) = \sum_{i=1}^{i^\star-1} n \cdot 8n \cdot p\left(m,n,\phi_i,\frac{2n^{3/2}}{\phi_i}\right) \\
=&O\left(\sum_{i=1}^{i^\star-1} n^2 \cdot \frac{mn^2}{\delta^2}\right) = O\left(i^\star \cdot \frac{mn^4}{\delta^2}\right) = O\left(\log\left(\frac{1}{\delta}\right) \cdot \frac{mn^4}{\delta^2}\right)
\end{aligned}$$

pivots in expectation. The second equation stems from Lemma 3.25, which states that $p(m,n,\phi,\delta) = O(\frac{mn^2}{\delta^2} + \frac{m\sqrt{n}\phi}{\delta})$. The second term in the sum can be omitted if $\phi = O(n^{3/2}/\delta)$, which is the case for $\phi_1,\ldots,\phi_{i^\star-1}$. Finally, $i^\star$ is the smallest integer i for which $2^i n^{3/2} > 2n^{3/2}/\delta$. Hence, $i^\star = O(\log(1/\delta))$. $\square$

3.6 Finding a Basic Feasible Solution

In this section we discuss how Phase 1 can be realized. In general there are, of course, several known textbook methods how Phase 1 can be implemented. However, for our purposes it is crucial that the parameter δ (or Δ) is not too small (or too large) for the linear program that needs to be solved in Phase 1. Ideally we would like it to be identical with the parameter δ (or Δ) of the matrix A of the original linear program. Eisenbrand and Vempala have addressed this problem and have presented a method to implement Phase 1. Their method is, however, very different from usual textbook methods and needs to solve m different linear programs to find an initial feasible solution for the given linear program.

In this section we will argue that also one of the usual textbook methods can be applied. We argue that $1/\delta$ increases by a factor of at most $\sqrt{m}$ and that Δ does not change at all in case one considers integer matrices (in particular, for totally unimodular matrices).

Let m and n be arbitrary positive integers, let $A \in \mathbb{R}^{m\times n}$ be an arbitrary matrix without zero-rows, and let $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ be arbitrary vectors. For finding a basic feasible solution of the linear program

$$\text{(LP)} \begin{cases} \max\ c^{\mathrm{T}}x \\ \ \text{s.t.}\ Ax \leq b \end{cases}$$

if one exists, or detecting that none exists, otherwise, we can solve the following linear program:

$$\text{(LP')} \begin{cases} \min \displaystyle\sum_{i=1}^{m} y_i \\ \text{s.t.}\ Ax - y \leq b \\ \qquad y \geq 0 \end{cases}$$

In the remainder of this section let us assume that matrix A has full column rank, that is, $\operatorname{rank}(A) = n$. Otherwise, we can transform the linear program (LP) as stated in Section 3.7.1 before considering (LP'). Furthermore, let us assume that the matrix $\bar{A}$, formed by the first n rows of matrix A, is invertible. This entails no loss of generality as this can always be achieved by permuting the rows of matrix A.

Let $\bar{b}$ denote the vector given by the first n entries of vector b and let $\bar{x}$ denote the vector for which $\bar{A}\bar{x} = \bar{b}$. The vector $(x', y') = (\bar{x}, \max\{A\bar{x} - b, 0\})$ is a feasible solution of (LP'), where the maximum is meant component-wise and 0 denotes the m-dimensional null vector. This is true because $Ax' - y' \leq A\bar{x} - (A\bar{x} - b) = b$ and $y' \geq 0$. Moreover, (x', y') is a basic solution: By the choice of $\bar{x}$ the first n inequalities of $Ax - y \leq b$ are tight as well as the first n non-negativity constraints. For each $k > m$ the k^{th} inequality of $Ax - y \leq b$ or the k^{th} non-negativity constraint is tight. Hence, the number of tight constraints is at least $2n + (m - n) = m + n$, which equals the number of variables of (LP').

Finally, we observe that a vector $(x, 0)$ is a basic feasible solution of (LP') if and only if x is a basic feasible solution of (LP). Consequently, by solving the linear program (LP') we obtain a basic feasible solution of the linear program (LP) (if the optimal value is 0) or we detect that (LP) is infeasible (if the optimal value is larger than 0). The linear program (LP') can be solved as described in Section 3.3.3. However, the running time is now expressed in the parameters $m' = 2m$, $n' = m + n$ and $\delta(B)$ (or $\Delta(B)$) of the matrix

$$B = \begin{bmatrix} A & -\mathbb{I}_m \\ \mathbb{O}_{m \times n} & -\mathbb{I}_m \end{bmatrix} \in \mathbb{R}^{(m+m)\times(n+m)} .$$

Before analyzing the parameters $\delta(B)$ and $\Delta(B)$, let us show that matrix B has full column rank.

Lemma 3.27. *The rank of matrix B is $m + n$.*

Proof. Recall that we assumed that the matrix $\bar{A}$ given by the first n rows of matrix A is invertible. Now consider the first n rows and the last m rows of matrix B. These rows form a submatrix $\bar{B}$ of B of the form

$$\bar{B} = \begin{bmatrix} \bar{A} & C \\ \mathbb{O}_{m \times n} & -\mathbb{I}_m \end{bmatrix}$$

for $C = [-\mathbb{I}_{n\times n}, \mathbb{O}_{n\times(m-n)}]$. As $\bar{B}$ is a 2×2-block-triangular matrix, we obtain $\det(\bar{B}) = \det(\bar{A}) \cdot \det(-\mathbb{I}_n) \neq 0$, that is, the first n rows and the last m rows of matrix B are linearly independent. Hence, $\operatorname{rank}(B) = m + n$. □

The remainder of this section is devoted to the analysis of $\delta(B)$ and $\Delta(B)$, respectively.

3.6.1 A Lower Bound for $\delta(B)$

Before we derive a bound for the value $\delta(B)$, let us give a characterization of $\delta(M)$ for a matrix M with full column rank.

Lemma 3.28. *Let $M \in \mathbb{R}^{m\times n}$ be a matrix with rank n. Then*

$$\frac{1}{\delta(M)} = \max_{k\in[n]} \max \left\{ \|z\| \;\middle|\; \begin{matrix} r_1{}^{\mathrm{T}}, \dots, r_n{}^{\mathrm{T}} \textit{ linear independent rows} \\ \textit{of } M \textit{ and } [\mathcal{N}(r_1), \dots, \mathcal{N}(r_n)]^{\mathrm{T}} \cdot z = e_k \end{matrix} \right\} ,$$

where e_k denotes the k^{th} unit vector.

Proof. The correctness of the above statement follows from

$$\begin{aligned} \frac{1}{\delta(M)} &= \max \left\{ \frac{1}{\delta(r_1, \dots, r_n)} \,\middle|\, r_1{}^{\mathrm{T}}, \dots, r_n{}^{\mathrm{T}} \text{ lin. indep. rows of } M \right\} \\ &= \max \left\{ \frac{1}{\delta(\mathcal{N}(r_1), \dots, \mathcal{N}(r_n))} \,\middle|\, r_1{}^{\mathrm{T}}, \dots, r_n{}^{\mathrm{T}} \text{ lin. indep. rows of } M \right\} \\ &= \max \left\{ \max_{k\in[n]} \|v_k\| \;\middle|\; \begin{matrix} r_1{}^{\mathrm{T}}, \dots, r_n{}^{\mathrm{T}} \text{ lin. indep. rows of } M \text{ and} \\ [v_1, \dots, v_n]^{-1} = [\mathcal{N}(r_1), \dots, \mathcal{N}(r_n)]^{\mathrm{T}} \end{matrix} \right\} . \end{aligned}$$

The first equation is due to the definition of δ, the second equation holds as δ is invariant under scaling of rows, and the third equation is due to Claim 1 of Lemma 3.2. The vector v_k from the last line is exactly the vector z for which $[\mathcal{N}(r_1), \dots, \mathcal{N}(r_n)]^{\mathrm{T}} \cdot z = e_k$. This finishes the proof. □

For the following lemma let us without loss of generality assume that the rows of matrix A are normalized. This does neither change the rank of A nor the value $\delta(A)$.

Lemma 3.29. *Let A and B be matrices of the form described above. Then*

$$\frac{1}{\delta(B)} \leq \frac{2\sqrt{m-n+1}}{\delta(A)} .$$

Proof. In accordance with Lemma 3.28, it suffices to show that for any $m+n$ linearly independent rows ${r_1}^{\mathrm{T}}, \dots, {r_{m+n}}^{\mathrm{T}}$ of B and any $k = 1, \dots, m+n$ the inequality

$$\|z\| \leq \frac{2\sqrt{m-n+1}}{\delta(A)}$$

holds, where z is the vector for which $[\mathcal{N}(r_1), \dots, \mathcal{N}(r_{m+n})]^{\mathrm{T}} \cdot z = e_k$.

Let ${r_1}^{\mathrm{T}}, \dots, {r_{m+n}}^{\mathrm{T}}$ be arbitrary $m+n$ linearly independent rows of B and let $k \in [m+n]$ be an arbitrary integer. We consider the equation $\hat{B} \cdot z = e_k$, where $\hat{B} = [\mathcal{N}(r_1), \dots, \mathcal{N}(r_{m+n})]^{\mathrm{T}}$. Each row r_ℓ is of either one of the two following types: Type 1 rows correspond to a row from A and for these we have $\|r_\ell\| = 2$ as the rows of A are normalized. Type 2 rows correspond to a non-negativity constraint of a variable y_i. For these we have $\|r_\ell\| = 1$. Observe that each row has exactly one "-1"-entry within the last m columns.

We categorize type 1 and type 2 rows further depending on the other selected rows: Type 1a rows are type 1 rows for which a type 2 row exists among the rows $r_1, \dots, r_{m+n}$ which has its "-1"-entry in the same column. This type 2 row is then classified as a type 2a row. The remaining type 1 and type 2 rows are classified as type 1b and type 2b rows, respectively. Observe that we can permute the rows of matrix $\hat{B}$ arbitrarily as we show the claim for all unit vectors e_k. Furthermore, we can permute the columns of $\hat{B}$ arbitrarily because this only permutes the rows of the solution vector z. This does not influence its norm. Hence, without loss of generality, matrix $\hat{B}$ contains normalizations of type 1a, of type 2a, of type 1b, and of type 2b rows in this order and the normalizations of the type 2a rows are ordered the same way as the normalizations of their corresponding type 1a rows.

Let m_1, m_2, and m_3 denote the number of type 1a, type 1b, and type 2b rows, respectively. Observe that the number of type 2a rows is also m_1. As matrix $\hat{B}$ is invertible, each column contains at least one non-zero entry. Hence, we can permute the columns of $\hat{B}$ such that $\hat{B}$ is of the form

$$\hat{B} = \begin{bmatrix} \frac{1}{2}A_1 & -\frac{1}{2}\mathbb{I}_{m_1} & \mathbb{O} & \mathbb{O} \\ \mathbb{O} & -\mathbb{I}_{m_1} & \mathbb{O} & \mathbb{O} \\ \frac{1}{2}A_2 & \mathbb{O} & -\frac{1}{2}\mathbb{I}_{m_2} & \mathbb{O} \\ \mathbb{O} & \mathbb{O} & \mathbb{O} & -\mathbb{I}_{m_3} \end{bmatrix} \in \mathbb{R}^{(m+n)\times(m+n)} ,$$

where A_1 and A_2 are $m_1 \times n$- and $m_2 \times n$-submatrices of A, respectively. The number of rows of $\hat{B}$ is $2m_1 + m_2 + m_3 = m+n$, whereas the number of columns of $\hat{B}$ is $n + m_1 + m_2 + m_3 = m+n$. This implies $m_1 = n$ and $m_2 \leq m-n$. Particularly, A_1 is a square matrix. As matrix $\hat{B}$ is a 2×2-block-triangular matrix and the top left and the bottom right block are 2×2-block-triangular matrices as well, we obtain

$$\det(\hat{B}) = \det\left(\frac{1}{2}A_1\right) \cdot (-1)^{m_1} \cdot \left(-\frac{1}{2}\right)^{m_2} \cdot (-1)^{m_3} = \pm \det(A_1) \cdot \frac{1}{2^{n+m_2}} \,.$$

Due to the linear independence of the rows ${r_1}^{\mathrm{T}}, \ldots, {r_{m+n}}^{\mathrm{T}}$ we have $\det(\hat{B}) \neq 0$. Consequently, $\det(A_1) \neq 0$, that is, matrix A_1 is invertible.

We partition vector z and vector e_k into four components $z_1, \ldots, z_4$ and $e_k^{(1)}, \ldots, e_k^{(4)}$, respectively, and rewrite the system $\hat{B} \cdot z = e_k$ of linear equations as follows:

$$\begin{aligned} \frac{1}{2}A_1 z_1 - \frac{1}{2}z_2 &= e_k^{(1)} \\ -z_2 &= e_k^{(2)} \\ \frac{1}{2}A_2 z_1 - \frac{1}{2}z_3 &= e_k^{(3)} \\ -z_4 &= e_k^{(4)} \end{aligned}$$

Now we distinguish between four pairwise distinct cases $e_k^{(i)} \neq 0$ for $i = 1, \ldots, 4$. In any case recall that the rows of A_1 and A_2 are rows of A, which are normalized. Furthermore, recall that the rows of A_1 are linearly independent.

- **Case 1:** $e_k^{(1)} \neq 0$. In this case we obtain $z_2 = 0$ and $z_4 = 0$. This implies $z_1 = 2\hat{z}$, where $\hat{z}$ is the solution of the equation $A_1\hat{z} = e_k^{(1)} + \frac{1}{2} \cdot 0 = e_k^{(1)}$. As the rows of matrix A_1 are normalized, Lemma 3.28 yields $\|\hat{z}\| \leq 1/\delta(A)$ and, hence, $\|z_1\| \leq 2/\delta(A)$. Next, we obtain $z_3 = A_2 z_1 - 2 \cdot e_k^{(3)} = A_2 z_1 - 0 = A_2 z_1$. Each entry of z_3 is a dot product of a (normalized) row from A and z_1. Hence, the absolute value of each entry is bounded by $\|z_1\| \leq 2/\delta(A)$. This yields the inequality

$$\begin{aligned} \|z\| = \sqrt{\|z_1\|^2 + \|z_2\|^2 + \|z_3\|^2 + \|z_4\|^2} &\leq \sqrt{(1 + m_2) \cdot (2/\delta(A))^2} \\ &\leq \frac{2\sqrt{m - n + 1}}{\delta(A)} \,. \end{aligned}$$

  For the last inequality we used the fact that $m_2 \leq m - n$.

- **Case 2:** $e_k^{(2)} \neq 0$. Here we obtain $z_2 = -e_k^{(2)}$, $z_4 = 0$, and $A_1 z_1 = 2 \cdot e_k^{(1)} + z_2 = 2 \cdot 0 - e_k^{(2)} = -e_k^{(2)}$, that is, $z_1 = -\hat{z}$, where $\hat{z}$ is the solution of the equation $A_1\hat{z} = e_k^{(2)}$. Analogously as in Case 1, we obtain $\|\hat{z}\| \leq 1/\delta(A)$ and, hence, $\|z_1\| \leq 1/\delta(A)$. Moreover, we obtain $z_3 = A_2 z_1 - 2 \cdot e_k^{(3)} = A_2 z_1 - 0 = A_2 z_1$, that is, the absolute value of each entry of z_3 is bounded by $\|z_1\| \leq 1/\delta(A)$. Consequently,

$$\|z\| \leq \sqrt{1 + (1 + m_2) \cdot (1/\delta(A))^2} \leq \frac{\sqrt{m - n + 2}}{\delta(A)} \leq \frac{2\sqrt{m - n + 1}}{\delta(A)} \,.$$

  For the second inequality we used $m_2 \leq m - n$ and $\delta(A) \leq 1$ by definition of $\delta(A)$. In the last inequality we used the fact that $m - n + 1 \geq 1$ and $\sqrt{x + 1} \leq 2\sqrt{x}$ for all $x \geq 1/3$.

- **Case 3:** $e_k^{(3)} \neq 0$. In this case we obtain $z_2 = 0$, $z_4 = 0$, and hence, $z_1 = 0$. This yields $z_3 = -2 \cdot e_k^{(3)}$ and
$$\|z\| = \|z_3\| = 2 \leq \frac{2\sqrt{m-n+1}}{\delta(A)} \,,$$
where we again used $\delta(A) \leq 1$.

- **Case 4:** $e_k^{(4)} \neq 0$. Here we obtain $z_2 = 0$, $z_4 = -e_k^{(4)}$, and hence, $z_1 = 0$ and $z_3 = 0$. Consequently, we get
$$\|z\| = \|z_4\| = 1 \leq \frac{2\sqrt{m-n+1}}{\delta(A)} \,,$$
which completes this case distinction.

As we have seen, in any case the inequality $\|z\| \leq 2\sqrt{m-n+1}/\delta(A)$ holds, which finishes the proof. □

3.6.2 An Upper Bound for $\Delta(B)$

Although parameter $\Delta(B)$ can be defined for arbitrary real-valued matrices, its meaning is limited to integer matrices when considering our analysis of the expected running time of the shadow vertex method. Hence, in this section we only deal with the case that matrix A is integral. Unlike in Section 3.6.1, we do not normalize the rows of matrix A before considering the linear program (LP'). As a consequence, matrix B is also integral.

The following lemma establishes a connection between $\Delta(A)$ and $\Delta(B)$.

Lemma 3.30. *Let A and B be of the form described above. Then $\Delta(B) = \Delta(A)$.*

Proof. It is clear that $\Delta(B) \geq \Delta(A)$ as matrix B contains matrix A as a submatrix. Thus, we can concentrate on proving that $\Delta(B) \leq \Delta(A)$. For this, consider an arbitrary $k \times k$-submatrix $\hat{B}$ of B. Matrix $\hat{B}$ is of the form
$$\hat{B} = \begin{bmatrix} A' & -I_1 \\ \mathbb{O}_{k_1 \times (k-k_2)} & -I_2 \end{bmatrix} \,,$$
where A' is a $(k-k_1) \times (k-k_2)$-submatrix of A and I_1 and I_2 are $(k-k_1) \times k_2$- and $k_1 \times k_2$-submatrices of $\mathbb{I}_m$, respectively. Our goal is to show that $|\det(\hat{B})| \leq \Delta(A)$. By analogy with the proof of Lemma 3.29 we partition the rows of $\hat{B}$ into classes. A row of $\hat{B}$ is of type 1 if it contains a row from A'. Otherwise, it is of type 2. Consequently, there are $k-k_1$ type 1 and k_1 type 2 rows.

These type 1 and type 2 rows are further categorized into three subtypes depending on the "-1"-entry (if exists) within the last k_2 columns. Type 1 and type 2 rows that only have zeros in the last k_2 entries are classified as type 1c and type 2c rows, respectively. The remaining type 1 and type 2 rows have exactly one "-1"-entry within the last k_2 columns. These are partitioned into subclasses as follows: If there are a type 1 row and a type 2 row that have their "-1"-entry in the same column, then these rows are classified as type 1a and type 2a, respectively. The type 1 and type 2 rows that are neither type 1a nor type 1c nor type 2a nor type 2c are referred to as type 1b and type 2b rows, respectively.

Note that type 2c rows only contain zeros. If matrix $\hat{B}$ contains such a row, then $|\det(\hat{B})| = 0 \leq \Delta(A)$. Hence, in the remainder we only consider the case that matrix $\hat{B}$ does not contain type 2c rows. With the same argument we can assume, without loss of generality, that matrix $\hat{B}$ does not contain a column with only zeros. As permuting the rows and columns of matrix $\hat{B}$ does not change the absolute value of its determinant, we can assume that $\hat{B}$ contains type 1a, type 1c, type 2a, type 1b, and type 2b rows in this order and that the type 2a rows are ordered the same ways as their corresponding type 1a rows. Furthermore, we can permute the columns of $\hat{B}$ such that it has the following form:

$$\hat{B} = \begin{bmatrix} A_1 & -\mathbb{I} & \mathbb{O} & \mathbb{O} \\ A_2 & \mathbb{O} & \mathbb{O} & \mathbb{O} \\ \mathbb{O} & -\mathbb{I} & \mathbb{O} & \mathbb{O} \\ A_3 & \mathbb{O} & -\mathbb{I} & \mathbb{O} \\ \mathbb{O} & \mathbb{O} & \mathbb{O} & -\mathbb{I} \end{bmatrix},$$

where A_1, A_2, and A_3 are submatrices of A' and, hence, of A. Iteratively decomposing matrix $\hat{B}$ into blocks and exploiting the block-triangular form of the matrices obtained in each step yields

$$\begin{aligned} |\det(\hat{B})| &= \left|\det\left(\begin{bmatrix} A_1 & -\mathbb{I} \\ A_2 & \mathbb{O} \\ \mathbb{O} & -\mathbb{I} \end{bmatrix}\right)\right| \cdot \left|\det\left(\begin{bmatrix} -\mathbb{I} & \mathbb{O} \\ \mathbb{O} & -\mathbb{I} \end{bmatrix}\right)\right| = \left|\det\left(\begin{bmatrix} A_1 & -\mathbb{I} \\ A_2 & \mathbb{O} \\ \mathbb{O} & -\mathbb{I} \end{bmatrix}\right)\right| \\ &= \left|\det\left(\begin{bmatrix} A_1 \\ A_2 \end{bmatrix}\right)\right| \cdot |\det(-\mathbb{I})| = \left|\det\left(\begin{bmatrix} A_1 \\ A_2 \end{bmatrix}\right)\right|. \end{aligned}$$

The absolute value of the latter determinant is bounded from above by $\Delta(A)$. This completes the proof. □

3.7 Justification of Assumptions

We assumed the matrix $A \in \mathbb{R}^{m \times n}$ to have full column rank and we assumed the polyhedron $\{x \in \mathbb{R}^n \mid Ax \leq b\}$ to be bounded. In this section we show that this entails no loss of generality by giving transformations of arbitrary linear programs into linear programs with full column rank whose polyhedra of feasible solutions are bounded.

3.7.1 Raising the Rank of Matrix A

For the algorithm we have assumed that the matrix A determining the polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ has full column rank. In this section we provide a solution if this condition is not met. For this, we describe the transformation of A into a matrix A' with full column rank by adding new linearly independent rows (we will ensure that the δ-distance property respectively the value of Δ is not violated by the transformation of A into A').

Transformation with respect to δ

Assume that we have an arbitrary matrix $A = [a_1, \dots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$ with rank $r = \operatorname{rank}(A) < n$. This implies that the polyhedron $P = \{x \,|\, Ax \le b\}$ has no vertices. Let $c \in \mathbb{R}^n$ be an arbitrary vector. Then the linear program $\max\{c^{\mathrm{T}}x \,|\, Ax \le b\}$ has either no solution (this is true if P is empty or $c^{\mathrm{T}}x$ is unbounded) or infinitely many solutions. We distinguish two different cases.

Case 1: $c \in \operatorname{span}\{a_1, \dots, a_m\}$
Let $\operatorname{span}\{a_1, \dots, a_m\}^{\perp}$ denote the orthogonal complement of $\operatorname{span}\{a_1, \dots, a_m\}$. Furthermore let $o_1, \dots, o_{n-r}$ be an orthonormal basis of $\operatorname{span}\{a_1, \dots, a_m\}^{\perp}$. Then the set of solutions $\mathbb{L} = \{\arg\max c^{\mathrm{T}}x \,|\, Ax \le b\}$ equals the set

$$\tilde{\mathbb{L}} = \{v + \arg\max c^{\mathrm{T}}x \,|\, Ax \le b, \tilde{A}x = \mathbb{O}, v \in \operatorname{span}\{a_1, \dots, a_m\}^{\perp}\},$$

where $\tilde{A} = [o_1, \dots, o_{n-r}]$. Thus we can add rows $[o_1, -o_1, \dots, o_{n-r}, -o_{n-r}]$ and extend the vector b by zero entries and calculate the set of solutions (note that the δ-distance-property does not change under this extension of A by Lemma 3.31). This equals the case where $n - r$ basis variables are known and we can proceed as in Section 3.3.1 by reducing the polyhedron to dimension r.

Case 2: $c \notin \operatorname{span}\{a_1, \dots, a_m\}$
We maintain the notation from above. Then we have a linear combination

$$c = \sum_{i=1}^{r} \ell_i \cdot a_i + \sum_{i=1}^{n-r} \tilde{\ell}_i \cdot o_i$$

where $\tilde{\ell}_k \neq 0$ for at least one $k \in [n-r]$. Without loss of generality we may assume that $\tilde{\ell}_k > 0$. But x is not bounded for direction o_k by A and thus o_k's coefficient in the linear combination of x may be chosen arbitrarily large. Thus $\max\{c^{\mathrm{T}}x \,|\, Ax \le b\}$ is unbounded.

Finally we prove that adding rows from the orthogonal complement of $\langle a_1, \dots, a_m\rangle$ to A does not change the δ-distance property.

Lemma 3.31. *Let $A = [a_1, \dots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$ be an arbitrary matrix of rank $r \le n-1$ and $\|a_i\| = 1$ for $i \in [m]$. Let $v \in \mathbb{R}^n$ be a vector such that $\|v\| = 1$ and $\langle v, a_i\rangle = 0$ for $i \in [m]$. Then $\operatorname{rank}(A') = r + 1$ and furthermore $\delta(A') = \delta(A)$ where $A' = [a_1, \dots, a_m, v]^{\mathrm{T}}$ is defined by adding the new row v^{T} to matrix A.*

Proof. First choose r linearly independent rows of A. Without loss of generality we may assume $a_1, \dots, a_r$. To calculate $\delta(\{a_1 \dots, a_{r-1}\}, a_r)$ we choose a vertex $x \in \operatorname{span}\{a_1, \dots, a_r\}$ with $x \cdot a_i = 0$ for $i \in [r-1]$ and $x \cdot a_r = 1$. Let α be the angle between a_r and x. Then $\delta(\{a_1 \dots, a_{r-1}\}, a_r) = \sin(\pi/2 - \alpha) = \cos(\alpha) = \frac{\|a_r\|}{\|x\|} = \frac{1}{\|x\|}$.

Moreover let $v, o_2, \dots, o_{n-r}$ be an orthonormal basis of the orthogonal complement $\operatorname{span}\{a_1, \dots, a_r\}^{\perp}$. Then $x \cdot v = 0$ and $x \cdot o_{i+1} = 0$ for $i \in [n-r-1]$ because of $x \in \operatorname{span}\{a_1, \dots, a_r\}$. Thus, x is the unique solution of the system of linear equations

$$[a_1, \dots, a_r, v, o_2, \dots, o_{n-r}]^{\mathrm{T}} x = e_r,$$

where $e_r \in \mathbb{R}^n$ denotes the rth canonical unit vector.

In accordance with Definition 3.1 and Lemma 3.2 1, we obtain $\delta(A)$ by choosing a solution with minimum norm over all such systems of linear equations and all vectors $e_i \in \mathbb{R}^n$ with $i \in [r]$. Consider now the matrix A' which is obtained by adding the row v to A. To calculate $\delta(A')$ we have to calculate the minimal norm $\frac{1}{\|x\|}$ of the set of solutions of the systems of linear equations of the form

$$[a'_1, \dots, a'_r, v, o_2, \dots, o_{n-r}]^{\mathrm{T}} x = e_i,$$

with $i \in [r+1]$. In the case where $i \leq r$ the set of systems of linear equations equals the set of systems of linear equation from the case where we calculate $\delta(A)$. Thus the minimum norm does not change and we obtain $\delta(A') = \delta(A)$.

In the case where $i = r+1$ the solution of the systems of linear equations is given by $x = v$ and we obtain $1/\|x\| = 1$. But this is the maximum norm, which can be reached by a solution x and thus the minimum norm does not change at all which completes the proof. □

Transformation with respect to Δ

If we want to ensure that the value $\Delta(A)$ does not change under the tranformation (which means $\Delta(A') = \Delta(A)$) we have to consider a slight modification of the above transformation. Especially, we will add vectors e_i for $i \in [n]$ which are part of the canonical basis of $\mathbb{R}^n$ such that $e_i \notin \operatorname{span}\{a_1, \dots, a_m\}$.
Again, we know that in Case 1 the polyhedron is either empty or has infinitely many solutions. Thus, if we find a solution

$$x' \in \{\arg\max c^{\mathrm{T}} x \mid Ax \leq b, \tilde{A}x = \mathbb{O}\},$$

where $\tilde{A} = [e_{i_1}, \dots, e_{i_{n-r}}]$ we already know that x' also maximizes c with respect to P. Furthermore if we are in Case 2 which means $c \notin \operatorname{span}\{a_1, \dots, a_m\}$ then the function $c^{\mathrm{T}}x$ is unbounded for elements $x \in P$. It remains to show that Δ does not change by adding rows e_i to A.

Lemma 3.32. *Let $A = [a_1, \dots, a_m]^{\mathrm{T}} \in \mathbb{Z}^{m \times n}$ be an arbitrary matrix of rank $r \leq n-1$. Let $e_i \in \mathbb{R}^n$ be a vector part of the canonical basis of $\mathbb{R}^n$ such that $e_i \notin \operatorname{span}\{a_1, \dots, a_m\}$. Then* $\operatorname{rank}(A') = r + 1$ *and furthermore $\Delta(A') = \Delta(A)$ where $A' = [a_1, \dots, a_m, e_i]^{\mathrm{T}}$ is defined by adding the new row ${e_i}^{\mathrm{T}}$ to matrix A.*

Proof. Let B be a submatrix of A'. Then either B contains no entries from row ${e_i}^{\mathrm{T}}$ (which means $\det(B) \leq \Delta(A)$) or one row of B is a subvector e' of ${e_i}^{\mathrm{T}}$. We distinguish two different cases:

Case 1: $e' = 0$. Then B has a zero row and thus $\det(B) = 0 \leq \Delta(A)$.

Case 2: $e' \neq 0$. In this case B has a row e'^{T}, which is element of the canonical basis. Then B has the form

$$\begin{pmatrix} \tilde{A} \\ e'^{\mathrm{T}} \end{pmatrix}.$$

Using the Laplace expansion, the absolute value of the determinant of B is at most the absolute value of a determinant of a submatrix of $\tilde{A}$ which is a submatrix of A. We obtain $\det(B) \leq \Delta(A)$ which concludes the proof. □

3.7.2 Translation into a Bounded Polyhedron

For the algorithm we have assumed that the polyhedron $P = \{x \in \mathbb{R} \mid Ax \leq b\}$ is bounded. This may be done because in the case where P is unbounded we tranform P into a polytope P' and run the algorithm for P'. If the optimum solution is unique and not a vertex of P, then we assert that the linear program $\max\{c^{\mathrm{T}}x \mid Ax \leq b\}$ is unbounded. To transform P we use the construction applied in [25]. First we choose n linearly independent rows of A. Without loss of generality we may assume the rows are given by $a_1, \dots, a_n$. If we find a ball $B(0)$ with radius r which contains all vertices of P, then we define a parallelpiped

$$Z = \{x \in \mathbb{R}^n \mid -r \leq a_i \cdot x \leq r, i \in [n]\},$$

which contains all vertices of P and does not violate the δ-distance property since it is defined by rows of A. Finally, set $P' = P \cap Z$ and start the algorithm on polytope P'. Note that P' has δ-distance since the set of rows of A did not change during the transformation.

To construct a ball with the desired properties we have to assume $A \in \mathbb{Q}^{m \times n}$ such as $b \in \mathbb{Q}^m$, which means no loss of generality for the implementation of the algorithm. By a slight generalization of Lemma 3.1.33 of [29] all vertices of the polyhedron P are contained in a ball $B(0)$ with radius $r = \sqrt{n} \cdot 2^{\mathrm{enc}(A,b)-n^2} \cdot \mathrm{lcm}(A)^n$ if $A \in \mathbb{Q}$, where the function enc returns the encoding length and the function $\mathrm{lcm}(M)$ for a rational matrix $M \in \mathbb{Q}^{m \times n}$ returns the least common multiple of the denominators of the entries of M. As a convention, the denominator of 0 is defined as 1.

Lemma 3.33. *If $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ for $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$, then all vertices of P are contained in a ball B around* 0 *with radius $r = \sqrt{n} \cdot 2^{\mathrm{enc}(A,b)-n^2} \cdot \mathrm{lcm}(A)^n$.*

Proof. We have to calculate an upper bound for the length of all vertices. Thus, for each submatrix B of A of rank n and the corresponding subvector b' of b we have to bound the length of the solution x of $Bx = b'$. Applying Cramer's rule, the components x_i of x are given by

$$x_i = \frac{\det(B_i)}{\det(B)},$$

where B_i equals B after replacing the i^{th} column of B by b'. We obtain $\mathrm{lcm}(B)^n \cdot \det(B) = \det(\mathrm{lcm}(B) \cdot B) \geq 1$ since $\mathrm{lcm}(B) \cdot B$ is integral and non-singular. All together we obtain

$$\begin{aligned} x_i = \frac{\det(B_i)}{\det(B)} &\leq \det(B_i) \cdot \mathrm{lcm}(B)^n \\ &\leq 2^{\mathrm{enc}(B_i)-n^2} \cdot \mathrm{lcm}(B)^n \\ &\leq 2^{\mathrm{enc}(B,b')-n^2} \cdot \mathrm{lcm}(B)^n. \end{aligned}$$

Thus, choosing $r = \sqrt{n} \cdot 2^{\mathrm{enc}(A,b)-n^2} \cdot \mathrm{lcm}(A)^n$ the ball $B(0)$ with radius r contains all vertices of P. □

Chapter 4

Conclusion

Hierarchical Clustering In this dissertation we gave the first known lower bound for the existence of hierarchical clusterings for the k-center problem. This leaves open a gap between the lower bound of 2 and the upper bound of 4 also shown in this dissertation. Moreover it leaves room for further lower bounds with respect other objective functions like k-means. We have initiated the theoretical study of the approximation guarantee of Ward's method. In particular, we have shown that Ward computes a 2-approximation on well-separated instances, which can be seen as the first theoretical explanation for its popularity in applications. We have also seen that its worst-case approximation guarantee increases exponentially with the dimension of the input and that it computes an $\mathcal{O}(1)$-approximation on one-dimensional instances. These results leave room for further research. It would be particularly interesting to better understand the worst-case behavior of Ward's method. It is not clear, for example, if it computes a constant-factor approximation if the dimension is constant. Our analysis of the one-dimensional case is very complex and the factor hidden in the $\mathcal{O}$-notation is large. It would be interesting to simplify our analysis and to improve the approximation factor. We conjecture that the instance shown in Figure 2.14 is the worst one-dimensional instance for Ward's method with an approximation factor of $2 + \sqrt{2} \approx 3.41$.

We improved the known approximation guarantees for the popular complete-linkage method which yields an $O(1)$-approximate hierarchical clusterings for the diameter k-clustering problem and the (discrete) k-center problem, assuming that d is a constant. For this it was sufficient to improve the second phase of the analysis by Ackermann et al. [2] (i.e., the last k merge operations). We used their results about the first phase to obtain our results. It is a very interesting question if the dependence on the dimension can be improved in the first phase. If we express the known lower bound of Ackermann et al. [2] in terms of d then it becomes $\Omega(\sqrt[p]{\log d})$. Hence, in terms of d, there is still a huge gap between the known upper and lower bounds. Another interesting question is whether the upper bound of $O(\log k)$ holds also for metrics that are not induced by norms.

Shadow Vertex Algorithm We also have shown that the shadow vertex algorithm can be used to solve linear programs possessing the δ-distance property in strongly polynomial time with respect to n, m, and $1/\delta$. The bound we obtained in Theorem 1.8 depends quadratically on $1/\delta$. Roughly speaking, one term $1/\delta$ is due to the fact that the smaller δ the less random is the objective function $w^{\mathrm{T}}x$. This term could in fact be replaced by $1/\delta(B)$ where B is the matrix that contains only the rows that are tight for x. The other term $1/\delta$ is due to our application of the principle of deferred decisions in the proof of Lemma 3.14. The smaller δ the less random is $w(Z)$.

For packing linear programs, in which all coefficients of A and b are non-negative and one has $x \geq 0$ as additional constraint, it is, for example, clear that $x = 0^n$ is a basic feasible solution. That is, one does not need to run Phase 1. Furthermore as in this solution without loss of generality exactly the constraints $x \geq 0$ are tight, $\delta(B) = 1$ and one occurrence of $1/\delta$ in Theorem 1.8 can be removed.

Bibliography

[1] Margareta Ackerman and Shai Ben-David. Clusterability: A theoretical study. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1–8, 2009.

[2] Marcel R. Ackermann, Johannes Blömer, Daniel Kuntze, and Christian Sohler. Analysis of agglomerative clustering. *Algorithmica*, 69(1):184–215, 2014.

[3] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–72, 2017.

[4] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[5] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.

[6] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1-2):49–54, 2012.

[7] Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of euclidean k-means. In *Proceedings of the 31st International Symposium on Computational Geometry (SoCG)*, pages 754–767, 2015.

[8] Pranjal Awasthi and Or Sheffet. Improved spectral-norm bounds for clustering. In *Proceedings of the 15th APPROX and 16th RANDOM*, pages 37–49, 2012.

[9] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 671–680, 2008.

[10] Maria-Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. *SIAM Journal on Computing*, 45(1):102–155, 2016.

[11] Maria-Florina Balcan, Yingyu Liang, and Pramod Gupta. Robust hierarchical clustering. *Journal of Machine Learning Research*, 15(1):3831–3871, 2014. Appendix C, page 4048.

[12] Shai Ben-David. Computational feasibility of clustering under clusterability assumptions. *CoRR*, abs/1501.00437, 2015.

[13] Shai Ben-David and Nika Haghtalab. Clustering in the presence of background noise. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 280–288, 2014.

[14] Nicolas Bonifas, Marco Di Summa, Friedrich Eisenbrand, Nicolai Hähnle, and Martin Niemeier. On sub-determinants and the diameter of polyhedra. In *Proceedings of the 28th ACM Symposium on Computational Geometry (SoCG)*, pages 357–362, 2012.

[15] Karl Heinz Borgwardt. *A probabilistic analysis of the simplex method.* Springer-Verlag New York, Inc., New York, NY, USA, 1986.

[16] Tobias Brunsch. *Smoothed Analysis of Selected Optimization Problems and Algorithms.* PhD thesis, University of Bonn, 2014. http://nbn-resolving.de/urn:nbn:de:hbz:5n-35439.

[17] Tobias Brunsch and Heiko Röglin. Finding short paths on polytopes by the shadow vertex algorithm. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 279–290, 2013.

[18] James R. Cole, Qiong Wang, Jordan A. Fish, Benli Chai, Donna M. McGarrell, Yanni Sun, C. Titus Brown, Andrea Porras-Alfaro, Cheryl R. Kuske, and James M. Tiedje. Ribosomal database project: data and tools for high throughput rrna analysis. *Nucleic Acids Research*, 2013.

[19] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proc. of the 3rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 1971.

[20] Amit Daniely, Nati Linial, and Michael E. Saks. Clustering is difficult only when it does not matter. *CoRR*, abs/1205.4891, 2012.

[21] G.B. Dantzig. Programming in a linear structure. *Comptroller, United States Air Force, Washington DC*, 1948.

[22] Aparna Das and Claire Kenyon-Mathieu. On hierarchical diameter-clustering and the supplier problem. *Theory of Computing Systems*, 45(3):497–511, 2009.

[23] Sanjoy Dasgupta and Philip M. Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555–569, 2005.

[24] Martin E. Dyer and Alan M. Frieze. Random walks, totally unimodular matrices, and a randomised dual simplex algorithm. *Mathematical Programming*, 64:1–16, 1994.

[25] Friedrich Eisenbrand and Santosh Vempala. Geometric random edge. *CoRR*, abs/1404.1568, 2014.

[26] Tomás Feder and Daniel H. Greene. Optimal algorithms for approximate clustering. In *Proc. of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 434–444, 1988.

[27] Houman Ghaemmaghami, David Dean, Robbie Vogt, and Sridha Sridharan. Speaker attribution of multiple telephone conversations using a complete-linkage clustering approach. In *Proc. of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4185–4188, 2012.

[28] Teofilo F. Gonzalés. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[29] Martin Grötschel, Laszlo Lovasz, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag New York, Inc., New York, NY, USA, 1980.

[30] I. Heller. On linear systems with integral valued solutions. *Pacific Journal of Mathematics*, 7(3):1351–1364, 1957.

[31] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Computational Geometry*, 28(2-3):89–112, 2004.

[32] Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[33] Leonid Khachiyan. A polynomial algorithm in linear programming. *Dokl Akad Nauk SSSR*, 244:1093–1096, 1979.

[34] Joshua Marc Koehnen. Ward-Verfahren mit k-median Zielfunktion, 2018. Bachelor's Thesis, in German.

[35] Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *Proceedings of the 51th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 299–308, 2010.

[36] Shrinu Kushagra, Samira Samadi, and Shai Ben-David. Finding meaningful cluster structure amidst background noise. In *27th International Conference on Algorithmic Learning Theory (ALT)*, pages 339–354, 2016.

[37] Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43, 2017.

[38] Guolong Lin, Chandrashekhar Nagarajan, Rajmohan Rajaraman, and David P. Williamson. A general approach for incremental approximation and hierarchical clustering. *SIAM Journal on Computing*, 39(8):3633–3669, 2010.

[39] Stuart P. Lloyd. Least squares quantization in PCM. *Bell Laboratories Technical Memorandum*, 1957. later published as [40].

[40] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[41] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi R. Varadarajan. The Planar k-means Problem is NP-Hard. In *Proceedings of the 3rd Workshop on Algorithms and Computation (WALCOM)*, pages 274–285, 2009.

[42] Konstantin Makarychev and Yury Makarychev. Metric perturbation resilience. *CoRR*, abs/1607.06442, 2016.

[43] Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. *SIAM Journal on Computing*, 32(3):816–832, 2003.

[44] Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The effectiveness of Lloyd-type methods for the k-means problem. *Journal of the ACM*, 59(6):28:1–28:22, 2012.

[45] C. Greg Plaxton. Approximation algorithms for hierarchical location problems. *Journal of Computer and System Sciences*, 72(3):425–443, 2006.

[46] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19(4):639–668, 2011.

[47] Melanie Schmidt. *Coresets and streaming algorithms for the k-means problem and related clustering objectives.* PhD thesis, Universität Dortmund, 2014.

[48] Armin Schrenk. Konstruktion unterer Schranken für die maximale Approximationsgüte von Ward's method, 2017. Bachelor's Thesis, in German.

[49] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

[50] Éva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

[51] Joe H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.

Appendix A

Proofs from Section 3.4

In this chapter we give the omitted proofs from Section 3.4. These are merely contained for the sake of completeness because they are very similar to the corresponding proofs in [17].

Lemma 3.9. *The probability of event $\mathcal{F}_\varepsilon$ tends to* 0 *for* $\varepsilon \to 0$.

Lemma A.1. *The probability that there are two neighboring vertices z_1, z_2 of P such that $|c^{\mathrm{T}} \cdot (z_2 - z_1)| \le \varepsilon \cdot \|z_2 - z_1\|$ is bounded from above by $2m^n n\varepsilon\phi$.*

Proof. Let z_1 and z_2 be arbitrary points in $\mathbb{R}^n$, let $u = z_2 - z_1$, and let A_ε denote the event that $|c^{\mathrm{T}} \cdot u| \le \varepsilon \cdot \|u\|$. As this inequality is invariant under scaling, we can assume that $\|u\| = 1$. Hence, there exists an index i for which $|u_i| \ge 1/\sqrt{n} \ge 1/n$. We apply the principle of deferred decisions and assume that the coefficients c_j for $j \ne i$ are already fixed arbitrarily. Then event A_ε occurs if and only if $c_i \cdot u_i \in [-\varepsilon, \varepsilon] - \sum_{j \ne i} c_j u_j$. Hence, for event A_ε to occur the random coefficient c_i must fall into an interval of length $2\varepsilon/|u_i| \le 2n\varepsilon$. The probability for this is bounded from above by $2n\varepsilon\phi$.

As we have to consider at most $\binom{m}{n-1} \le m^n$ pairs of neighbors (z_1, z_2), a union bound yields the additional factor of m^n. □

Proof of Lemma 3.9. Let z_1, z_2, z_3 be pairwise distinct vertices of P such that z_1 and z_3 are neighbors of z_2 and let $\Delta_z := z_2 - z_1$ and $\Delta'_z := z_3 - z_2$. We assume that $\|\Delta_z\| = \|\Delta'_z\| = 1$. This entails no loss of generality as the fractions in Definition 3.8 are invariant under scaling. Let $i_1, \dots, i_{n-1} \in [m]$ be the $n-1$ indices for which ${a_{i_k}}^{\mathrm{T}} z_1 = b_{i_k} = {a_{i_k}}^{\mathrm{T}} z_2$. For the ease of notation let us assume that $i_k = k$. The rows $a_1, \dots, a_{n-1}$ are linearly independent because P is non-degenerate. Since z_1, z_2, z_3 are distinct vertices of P and since z_1 and z_3 are neighbors of z_2, there is exactly one index ℓ for which ${a_\ell}^{\mathrm{T}} z_3 < b_\ell$, i.e., ${a_\ell}^{\mathrm{T}} \Delta'_z \ne 0$. Otherwise, z_1, z_2, z_3 would be collinear which would contradict the fact that they are pairwise distinct vertices of P. Without loss of generality assume that $\ell = n-1$. Since ${a_k}^{\mathrm{T}} \Delta_z = 0$ for each $k \in [n-1]$, the vectors $a_1, \dots, a_{n-1}, \Delta_z$ are linearly independent.

We apply the principle of deferred decisions and assume that c is already fixed. Thus, $c^{\mathrm{T}} \Delta_z$ and $c^{\mathrm{T}} \Delta'_z$ are fixed as well. Moreover, we assume that $c^{\mathrm{T}} \Delta_z \ne 0$ and $c^{\mathrm{T}} \Delta'_z \ne 0$ since this happens almost surely due to Lemma A.1. Now consider the matrix $M = [a_1, \dots, a_{n-2}, \Delta_z, a_{n-1}]$ and the random vector $(Y_1, \dots, Y_{n-1}, Z)^{\mathrm{T}} = M^{-1} \cdot w = -M^{-1} \cdot$

$[u_1, \dots, u_n] \cdot \lambda$. For fixed values $y_1, \dots, y_{n-1}$ let us consider all realizations of λ for which $(Y_1, \dots, Y_{n-1}) = (y_1, \dots, y_{n-1})$. Then

$$\begin{aligned} w^{\mathrm{T}} \Delta_z &= \big(M \cdot (y_1, \dots, y_{n-1}, Z)^{\mathrm{T}}\big)^{\mathrm{T}} \Delta_z \\ &= \sum_{k=1}^{n-2} y_k \cdot {a_k}^{\mathrm{T}} \Delta_z + y_{n-1} \cdot {\Delta_z}^{\mathrm{T}} \Delta_z + Z \cdot {a_{n-1}}^{\mathrm{T}} \Delta_z \\ &= y_{n-1}\,, \end{aligned}$$

i.e., the value of $w^{\mathrm{T}} \Delta_z$ does not depend on the outcome of Z since Δ_z is orthogonal to all a_k. For Δ'_z we obtain

$$\begin{aligned} w^{\mathrm{T}} \Delta'_z &= \big(M \cdot (y_1, \dots, y_{n-1}, Z)^{\mathrm{T}}\big)^{\mathrm{T}} \Delta'_z \\ &= \sum_{k=1}^{n-2} y_k \cdot {a_k}^{\mathrm{T}} \Delta'_z + y_{n-1} \cdot {\Delta_z}^{\mathrm{T}} \Delta'_z + Z \cdot {a_{n-1}}^{\mathrm{T}} \Delta'_z \\ &= y_{n-1} \cdot {\Delta_z}^{\mathrm{T}} \Delta'_z + Z \cdot {a_{n-1}}^{\mathrm{T}} \Delta'_z \end{aligned}$$

as Δ'_z is orthogonal to all a_k except for $k = \ell = n-1$. The chain of equivalences

$$\begin{aligned} & \left| \frac{w^{\mathrm{T}} \Delta_z}{c^{\mathrm{T}} \Delta_z} - \frac{w^{\mathrm{T}} \Delta'_z}{c^{\mathrm{T}} \Delta'_z} \right| \leq \varepsilon \\ \iff\; & \frac{w^{\mathrm{T}} \Delta'_z}{c^{\mathrm{T}} \Delta'_z} \in [-\varepsilon, \varepsilon] + \frac{w^{\mathrm{T}} \Delta_z}{c^{\mathrm{T}} \Delta_z} \\ \iff\; & w^{\mathrm{T}} \Delta'_z \in \Big[-\varepsilon \cdot |c^{\mathrm{T}} \Delta'_z|, \varepsilon \cdot |c^{\mathrm{T}} \Delta'_z| \Big] + \frac{w^{\mathrm{T}} \Delta_z}{c^{\mathrm{T}} \Delta_z} \cdot c^{\mathrm{T}} \Delta'_z \\ \iff\; & Z \cdot {a_{n-1}}^{\mathrm{T}} \Delta'_z \in \Big[-\varepsilon \cdot |c^{\mathrm{T}} \Delta'_z|, \varepsilon \cdot |c^{\mathrm{T}} \Delta'_z| \Big] + \frac{w^{\mathrm{T}} \Delta_z}{c^{\mathrm{T}} \Delta_z} \cdot c^{\mathrm{T}} \Delta'_z - y_{n-1} \cdot {\Delta_z}^{\mathrm{T}} \Delta'_z \end{aligned}$$

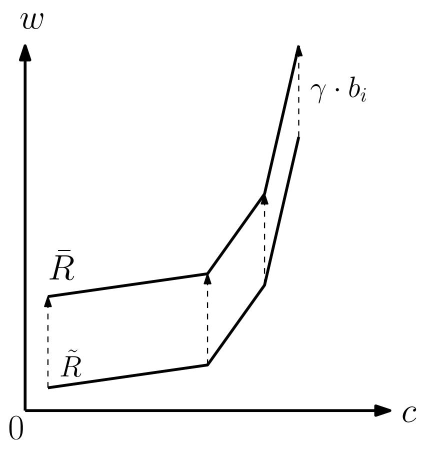
implies, that for event $\mathcal{F}_\varepsilon$ to occur Z must fall into an interval $I = I(y_1, \dots, y_{n-1})$ of length $2\varepsilon \cdot |c^{\mathrm{T}} \Delta'_z| / |{a_{n-1}}^{\mathrm{T}} \Delta'_z|$. The probability for this to happen is bounded from above by

$$\frac{2n \cdot 2\varepsilon \cdot \frac{|c^{\mathrm{T}} \Delta'_z|}{|{a_{n-1}}^{\mathrm{T}} \Delta'_z|}}{\delta(r_1, \dots, r_n) \cdot \min_{k \in [n]} \|r_k\|} = \underbrace{\frac{4n \cdot |c^{\mathrm{T}} \Delta'_z|}{\delta(r_1, \dots, r_n) \cdot \min_{k \in [n]} \|r_k\| \cdot |{a_{n-1}}^{\mathrm{T}} \Delta'_z|}}_{=:\gamma} \cdot \varepsilon\,,$$

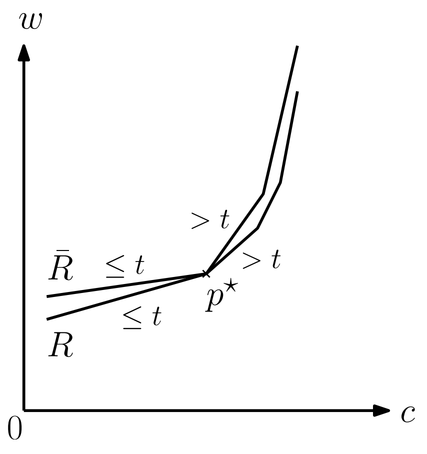
where $[r_1, \dots, r_n] = -M^{-1} \cdot [u_1, \dots, u_n]$. This is due to $(Y_1, \dots, Y_{n-1}, Z)^{\mathrm{T}} = [r_1, \dots, r_n] \cdot \lambda$ and Corollary 3.4 (applied with $\phi = 1$). Since the vectors $r_1, \dots, r_n$ are linearly independent, $\delta(r_1, \dots, r_n)$ is a well-defined positive value and $\min_{k \in [n]} \|r_k\| > 0$. Furthermore, $|{a_{n-1}}^{\mathrm{T}} \Delta'_z| > 0$ since i_{n-1} is the constraint which is not tight for z_3, but for z_2. Hence, $\gamma < \infty$, and thus $\mathbf{Pr}\Big[\Big| \frac{w^{\mathrm{T}} \Delta_z}{c^{\mathrm{T}} \Delta_z} - \frac{w^{\mathrm{T}} \Delta'_z}{c^{\mathrm{T}} \Delta'_z} \Big| \leq \varepsilon \Big] \to 0$ for $\varepsilon \to 0$.

As there are at most m^{3n} triples (z_1, z_2, z_3) we have to consider, the claim follows by applying a union bound. □

Lemma 3.10. *Let $\tilde{\pi} = \pi_{c,\tilde{w}}$ and let $\tilde{R} = R_{c,\tilde{w}}$ be the path from $\tilde{\pi}(x_0)$ to the rightmost vertex $\tilde{p}_r$ of the projection $\tilde{\pi}(P)$ of polytope P. Furthermore, let $\tilde{p}^\star$ be the rightmost vertex of $\tilde{R}$ whose slope does not exceed t. Then $\tilde{p}^\star = \tilde{\pi}(x^\star)$.*



(a) Relation between $\bar{R}$ and $\tilde{R}$



(b) Relation between $\bar{R}$ an R

Figure A.1: Relations between R, $\tilde{R}$, and $\bar{R}$

Proof of Lemma 3.10. We consider a linear auxiliary function $\bar{w} \colon \mathbb{R}^n \to \mathbb{R}$, given by $\bar{w}(x) := \tilde{w}^{\mathrm{T}} x + \gamma \cdot b_i$. The paths $\bar{R} = R_{c,\bar{w}}$ and $\tilde{R}$ are identical except for a shift by $\gamma \cdot b_i$ in the second coordinate because for $\bar{\pi} = \pi_{c,\bar{w}}$ we obtain

$$\bar{\pi}(x) = (c^{\mathrm{T}} x, \tilde{w}^{\mathrm{T}} x + \gamma \cdot b_i) = (c^{\mathrm{T}} x, \tilde{w}^{\mathrm{T}} x) + (0, \gamma \cdot b_i) = \tilde{\pi}(x) + (0, \gamma \cdot b_i)$$

for all $x \in \mathbb{R}^n$. Consequently, the slopes of $\bar{R}$ and $\tilde{R}$ are exactly the same (see Figure A.1a).

Let $x \in P$ be an arbitrary point from the polytope P. Then, $\tilde{w}^{\mathrm{T}} x = w^{\mathrm{T}} x - \gamma \cdot {a_i}^{\mathrm{T}} x \geq w^{\mathrm{T}} x - \gamma \cdot b_i$. The inequality is due to $\gamma \geq 0$ and ${a_i}^{\mathrm{T}} x \leq b_i$ for all $x \in P$. Equality holds, among others, for $x = x^\star$ due to the choice of a_i. Hence, for all points $x \in P$ the two-dimensional points $\pi(x)$ and $\bar{\pi}(x)$ agree in the first coordinate while the second coordinate of $\pi(x)$ is at most the second coordinate of $\bar{\pi}(x)$ as $\bar{w}(x) = \tilde{w}^{\mathrm{T}} x + \gamma \cdot b_i \geq w^{\mathrm{T}} x$. Additionally, we have $\pi(x^\star) = \bar{\pi}(x^\star)$. Thus, path $\bar{R}$ is above path R but they have point $p^\star = \pi(x^\star)$ in common. Hence, the slope of $\bar{R}$ to the left (right) of $p^\star$ is at most (at least) the slope of R to the left (right) of $p^\star$ which is at most (greater than) t (see Figure A.1b). Consequently, $p^\star$ is the rightmost vertex of $\bar{R}$ whose slope does not exceed t. Since $\bar{R}$ and $\tilde{R}$ are identical up to a shift of $(0, \gamma \cdot b_i)$, $\tilde{\pi}(x^\star)$ is the rightmost vertex of $\tilde{R}$ whose slope does not exceed t, i.e., $\tilde{\pi}(x^\star) = \tilde{p}^\star$. □

Lemma 3.14. *For any $\phi \geq \sqrt{n}$, any $t \geq 0$, and any $\varepsilon > 0$ the probability of event $A_{t,\varepsilon}$ is bounded by*

$$\mathbf{Pr}\left[A_{t,\varepsilon}\right] \leq \frac{2mn^2\varepsilon}{\max\left\{\frac{n}{2}, t\right\} \cdot \delta^2} \leq \frac{4mn\varepsilon}{\delta^2}\,.$$

Proof of Lemma 3.14. Due to Lemma 3.13 it suffices to show that

$$\mathbf{Pr}\left[E_{i,t,\varepsilon}\right] \leq \frac{1}{m} \cdot \frac{2mn^2\varepsilon}{\max\left\{\frac{n}{2}, t\right\} \cdot \delta^2} = \frac{2n^2\varepsilon}{\max\left\{\frac{n}{2}, t\right\} \cdot \delta^2}$$

for any index $i \in [m]$.

We apply the principle of deferred decisions and assume that vector c is already fixed. Now we extend the normalized vector a_i to an orthonormal basis $\{q_1, \ldots, q_{n-1}, a_i\}$ of $\mathbb{R}^n$ and consider the random vector $(Y_1, \ldots, Y_{n-1}, Z)^{\mathrm{T}} = Q^{\mathrm{T}} w$ given by the matrix vector product of the transpose of the orthogonal matrix $Q = [q_1, \ldots, q_{n-1}, a_i]$ and the vector $w = -[u_1, \ldots, u_n] \cdot \lambda$. For fixed values $y_1, \ldots, y_{n-1}$ let us consider all realizations of λ such that $(Y_1, \ldots, Y_{n-1}) = (y_1, \ldots, y_{n-1})$. Then, w is fixed up to the ray

$$w(Z) = Q \cdot (y_1, \ldots, y_{n-1}, Z)^{\mathrm{T}} = \sum_{j=1}^{n-1} y_j \cdot q_j + Z \cdot a_i = v + Z \cdot a_i$$

for $v = \sum_{j=1}^{n-1} y_j \cdot q_j$. All realizations of $w(Z)$ that are under consideration are mapped to the same value $\tilde{w}$ by the function $w \mapsto \tilde{w}(w, i)$, i.e., $\tilde{w}(w(Z), i) = \tilde{w}$ for any possible realization of Z. In other words, if $w = w(Z)$ is specified up to this ray, then the path $R_{c,\tilde{w}(w,i)}$ and, hence, the vectors $y^\star$ and $\hat{y}$ from the definition of event $E_{i,t,\varepsilon}$, are already determined.

Let us only consider the case that the first condition of event $E_{i,t,\varepsilon}$ is fulfilled. Otherwise, event $E_{i,t,\varepsilon}$ cannot occur. Thus, event $E_{i,t,\varepsilon}$ occurs iff

$$(t, t+\varepsilon] \ni \frac{w^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{c^{\mathrm{T}} \cdot (\hat{y} - y^\star)} = \underbrace{\frac{v^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{c^{\mathrm{T}} \cdot (\hat{y} - y^\star)}}_{=:\alpha} + Z \cdot \underbrace{\frac{a_i{}^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{c^{\mathrm{T}} \cdot (\hat{y} - y^\star)}}_{=:\beta} .$$

The next step in this proof will be to show that the inequality $|\beta| \geq \max\left\{\frac{n}{2}, t\right\} \cdot \frac{\delta}{n}$ is necessary for event $E_{i,t,\varepsilon}$ to happen. For the sake of simplicity let us assume that $\|\hat{y} - y^\star\| = 1$ since β is invariant under scaling. If event $E_{i,t,\varepsilon}$ occurs, then $a_i{}^{\mathrm{T}} y^\star = b_i$, $\hat{y}$ is a neighbor of $y^\star$, and $a_i{}^{\mathrm{T}} \hat{y} \neq b_i$. That is, by Lemma 3.2, Claim 3 we obtain $|a_i{}^{\mathrm{T}} \cdot (\hat{y} - y^\star)| \geq \delta \cdot \|\hat{y} - y^\star\| = \delta$ and, hence,

$$|\beta| = \left| \frac{a_i{}^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{c^{\mathrm{T}} \cdot (\hat{y} - y^\star)} \right| \geq \frac{\delta}{|c^{\mathrm{T}} \cdot (\hat{y} - y^\star)|} .$$

On the one hand we have $|c^{\mathrm{T}} \cdot (\hat{y} - y^\star)| \leq \|c\| \cdot \|\hat{y} - y^\star\| \leq \left(1 + \frac{\sqrt{n}}{\phi}\right) \cdot 1 \leq 2$, where the second inequality is due to the choice of c as perturbation of the unit vector c_0 and the third inequality is due to the assumption $\phi \geq \sqrt{n}$. On the other hand, due to $\frac{w^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{c^{\mathrm{T}} \cdot (\hat{y} - y^\star)} \geq t$ we have

$$|c^{\mathrm{T}} \cdot (\hat{y} - y^\star)| \leq \frac{|w^{\mathrm{T}} \cdot (\hat{y} - y^\star)|}{t} \leq \frac{\|w\| \cdot \|\hat{y} - y^\star\|}{t} \leq \frac{n}{t} .$$

Consequently,

$$|\beta| \geq \frac{\delta}{\min\left\{2, \frac{n}{t}\right\}} = \max\left\{\frac{n}{2}, t\right\} \cdot \frac{\delta}{n} .$$

Summarizing the previous observations we can state that if event $E_{i,t,\varepsilon}$ occurs, then $|\beta| \geq \max\left\{\frac{n}{2}, t\right\} \cdot \frac{\delta}{n}$ and $\alpha + Z \cdot \beta \in (t, t+\varepsilon]$. Hence,

$$Z \cdot \beta \in (t, t+\varepsilon] - \alpha ,$$

i.e., Z falls into an interval $I(y_1, \dots, y_{n-1})$ of length at most $\varepsilon/(\max\{\frac{n}{2}, t\} \cdot \delta/n) = n\varepsilon/(\max\{\frac{n}{2}, t\} \cdot \delta)$ that only depends on the realizations $y_1, \dots, y_{n-1}$ of $Y_1, \dots, Y_{n-1}$. Let $B_{i,t,\varepsilon}$ denote the event that Z falls into the interval $I(Y_1, \dots, Y_{n-1})$. We showed that $E_{i,t,\varepsilon} \subseteq B_{i,t,\varepsilon}$. Consequently,

$$\mathbf{Pr}[E_{i,t,\varepsilon}] \leq \mathbf{Pr}[B_{i,t,\varepsilon}] \leq \frac{2n \cdot \frac{n\varepsilon}{\max\{\frac{n}{2}, t\} \cdot \delta}}{\delta(Q^{\mathrm{T}} u_1, \dots, Q^{\mathrm{T}} u_n)} \leq \frac{2n^2 \varepsilon}{\max\{\frac{n}{2}, t\} \cdot \delta^2},$$

where the second inequality is due to Corollary 3.4 (applied with $\phi = 1$): By definition, we have

$$(Y_1, \dots, Y_{n-1}, Z)^{\mathrm{T}} = Q^{\mathrm{T}} w = Q^{\mathrm{T}} \cdot -[u_1, \dots, u_n] \cdot \lambda = [-Q^{\mathrm{T}} u_1, \dots, -Q^{\mathrm{T}} u_n] \cdot \lambda\,.$$

The third inequality stems from the fact that $\delta(-Q^{\mathrm{T}} u_1, \dots, -Q^{\mathrm{T}} u_n) = \delta(u_1, \dots, u_n) \geq \delta$, where the equality is due to the orthogonality of $-Q$ (Claim 2 of Lemma 3.2). $\square$