Smoothed Analysis of Selected Optimization Problems and Algorithms

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Tobias Brunsch

aus

Karl-Marx-Stadt, jetzt Chemnitz

Bonn, 2014

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen
Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Heiko Röglin

2. Gutachter: Prof. Dr. Marek Karpinski

Tag der Promotion: 28. Februar 2014

Erscheinungsjahr: 2014

Abstract

Optimization problems arise in almost every field of economics, engineering, and science. Many of these problems are well-understood in theory and sophisticated algorithms exist to solve them efficiently in practice. Unfortunately, in many cases the theoretically most efficient algorithms perform poorly in practice. On the other hand, some algorithms are much faster than theory predicts. This discrepancy is a consequence of the pessimism inherent in the framework of worst-case analysis, the predominant analysis concept in theoretical computer science.

We study selected optimization problems and algorithms in the framework of smoothed analysis in order to narrow the gap between theory and practice. In smoothed analysis, an adversary specifies the input, which is subsequently slightly perturbed at random. As one example we consider the successive shortest path algorithm for the minimum-cost flow problem. While in the worst case the successive shortest path algorithm takes exponentially many steps to compute a minimum-cost flow, we show that its running time is polynomial in the smoothed setting.

Another problem studied in this thesis is makespan minimization for scheduling with related machines. It seems to be unlikely that there exist fast algorithms to solve this problem exactly. This is why we consider three approximation algorithms: the jump algorithm, the lex-jump algorithm, and the list scheduling algorithm. In the worst case, the approximation guarantees of these algorithms depend on the number of machines. We show that there is no such dependence in smoothed analysis.

We also apply smoothed analysis to multicriteria optimization problems. In particular, we consider integer optimization problems with several linear objectives that have to be simultaneously minimized. We derive a polynomial upper bound for the size of the set of Pareto-optimal solutions contrasting the exponential worst-case lower bound.

As the icing on the cake we find that the insights gained from our smoothed analysis of the running time of the successive shortest path algorithm lead to the design of a randomized algorithm for finding short paths between two given vertices of a polyhedron. We see this result as an indication that, in future, smoothed analysis might also result in the development of fast algorithms.

Zusammenfassung

Optimierungsprobleme treten in allen wirtschaftlichen, naturwissenschaftlichen und technischen Gebieten auf. Viele dieser Probleme sind ausführlich untersucht und aus praktischer Sicht effizient lösbar. Leider erweisen sich in vielen Fällen die theoretisch effizientesten Algorithmen in der Praxis als ungeeignet. Auf der anderen Seite sind einige Algorithmen viel schneller als die Theorie vorhersagt. Dieser scheinbare Widerspruch resultiert aus dem Pessimismus, der dem in der theoretischen Informatik vorherrschenden Analysekonzept, der Worst-Case-Analyse, innewohnt.

Um die Lücke zwischen Theorie und Praxis zu verkleinern, untersuchen wir ausgewählte Optimierungsprobleme und Algorithmen auf gegnerisch vorgegebenen Instanzen, die durch ein leichtes Zufallsrauschen gestört werden. Solche perturbierten Instanzen bezeichnen wir als semi-zufällige Eingaben. Als Beispiel betrachten wir den Successive-Shortest-Path-Algorithmus für das Minimum-Cost-Flow-Problem. Während dieser Algorithmus im Worst Case exponentiell viele Schritte benötigt, um einen Minimum-Cost-Flow zu berechnen, zeigen wir, dass seine Laufzeit auf semi-zufälligen Eingaben polynomiell ist.

Ein weiteres Problem, das wir in dieser Arbeit untersuchen, ist die Minimierung des Makespans für Scheduling auf unterschiedlich schnellen Maschinen. Es scheint, dass dieses Problem nicht effizient gelöst werden kann. Daher betrachten wir drei Approximationsalgorithmen: den Jump-, den Lex-Jump- und den List-Scheduling-Algorithmus. Im Worst Case hängt die Approximationsgüte dieser Algorithmen von der Anzahl der Maschinen ab. Wir zeigen, dass das auf semi-zufälligen Eingaben nicht der Fall ist.

Des Weiteren betrachten wir ganzzahlige Optimierungsprobleme mit mehreren linearen Zielfunktionen, die simultan minimiert werden sollen. Wir leiten eine polynomielle obere Schranke für die Größe der Pareto-Menge auf semi-zufälligen Eingaben her, die im Gegensatz zu der exponentiellen unteren Worst-Case-Schranke steht.

Mit den Erkenntnissen aus der Laufzeitanalyse des Successive-Shortest-Path-Algorithmus entwerfen wir einen randomisierten Algorithmus zur Bestimmung eines kurzen Pfades zwischen zwei gegebenen Ecken eines Polyeders. Wir betrachten dieses Ergebnis als ein Indiz dafür, dass in Zukunft Analysen auf semi-zufälligen Eingaben auch zu der Entwicklung schneller Algorithmen führen könnten.

Acknowledgments

The last four years have been an exciting experience and I enjoyed every single day despite the uncountable moments I got stuck with research. In these times it was you, Heiko, who motivated me with his unlimited optimism. Every now and then I have to smile when I remember you saying that ‘this makes the problem only more interesting’. It took me a while to believe you, but now I am convinced that you were right. Thanks to your guidance and your brilliant ideas, which enlightened me in gloomy situations when no approach seemed to work, I have reached far more than I ever expected or hoped for. Furthermore, I thank Prof. Dr. Marek Karpinski and Prof. Dr. Tobias Friedrich for their efforts as co-referees of this thesis.

I am also grateful having such fine colleagues here in Bonn. You often found it amusing when I came along for lunch and then only had a salad. When you think twice you will come to realize that it was not because of the food that I joined. Most notably you, Alex, who did not just support me with his mathematical expertise in many fruitful discussions, but also taught me correct German, made me laugh a lot. On top of that I really liked your world history in a nutshell lectures for which I came home late several times.

Let me additionally thank my new colleagues Lena and Michael, who started reinforcing our group last year. From the first moment we got on well with each other and formed a nice team. Thank you for the pleasant conversations and for proofreading my thesis. Your comments significantly improved the quality of this work. I am looking forward to watching your progress and listening to your discussions about max-cut and k-means. You sure will make our group shine bright.

Although I left Chemnitz for becoming a PhD student, I was always welcomed home warmly by the group of Christoph Helmberg and Frank Göring. You provided me with delicious cake and tea and listened to several of my talks. Despite the overwhelming amount of information I put into my presentations you stayed interested and even asked for more details. Your toughness is admirable. I enjoyed being temporarily part of your group. Thank you for your kindness.

I also did not forget the research meetings with my coauthors Bodo Manthey, Cyriel Rutten, Kamiel Cornelissen, and Tjark Vredeveld. Sometimes it was really frustrating and only the cookies made me persevere. But somehow everything worked out in the end.

Maybe it is as Bodo once said: ‘every meeting with Heiko results in one paper’. It was a pleasure working with you all.

Most of all I am indebted to my family. You have always been there for me such that there were not many things I had to worry about. In particular you, mom and dad, advanced my interest in mathematics culminating in this thesis. I can hardly express the extend of my gratitude, but I hope that you know how I feel.

The last decade has been full of changes, not all of them being positive. You, Sina, have been accompanying me all these years. You bring light into my life and make me smile every day. You are wonderful. I love you!

Contents

Chapter 1

Introduction

Optimization problems play a central role in many areas of economics, engineering, and science. Especially since the invention of the computer the interest in fast algorithms for solving these problems has been growing rapidly among both practitioners and theoreticians. Today, many of these problems are well-understood. The knowledge we have is, however, very inconsistent. In practice, the theoretically most efficient algorithms are often outperformed by algorithms whose theoretical running time is much worse. This unsatisfying discrepancy between theory and practice results from the pessimism inherent in the framework of worst-case analysis, the predominant analysis concept in theoretical computer science. In worst-case analysis, instances for a problem are considered on which the algorithm at hand performs as badly as possible. There are many algorithms whose worst cases are usually not encountered in practice. For these algorithms worst-case predictions about their running time behavior completely deviate from what can be observed on real-life instances.

In order to narrow the gap between theory and practice, Spielman and Teng introduced the framework of *smoothed analysis*. In smoothed analysis, the power of the adversary, which creates artificially bad instances, is reduced by adding some amount of random noise to these instances. Thus, smoothed analysis can be seen as a hybrid between worst-case analysis and average-case analysis having the advantages of both: As in average-case analysis, pathological instances that spoil the analysis are ruled out due to some amount of randomness. As in worst-case analysis, good bounds usually indicate good behavior on real-life instances.

In this thesis, we study selected optimization problems and algorithms in the framework of smoothed analysis to explain why they usually behave significantly better than theory predicts. In the first part we deal with the *minimum-cost flow problem*, which is solvable in strongly-polynomial time. Orlin's algorithm, which is the theoretically fastest algorithm for this problem known today, is practically not competitive. The same holds for some other strongly-polynomial time algorithms. These algorithms are outperformed by the *successive shortest path algorithm*, which is one of the simplest algorithms for the minimum-cost flow problem and has exponential worst-case running time. We show that the successive shortest path algorithm has smoothed polynomial running time.

The ideas developed in part one turn out to be useful in a similar context that is studied in the second part of this thesis. Here, we are interested in finding a short path between two given vertices of a *polyhedron*. We analyze a randomized variant of the *simplex method* with the *shadow vertex pivot rule* and show that it has expected polynomial running time for some classes of polyhedra like flow polytopes.

In the third part we consider the *scheduling problem*. Here the task is to assign jobs with individual sizes to machines with individual speeds in order to minimize the time it takes to process all jobs. Even if all machines have the same speed this problem is strongly $\mathcal{NP}$-hard if the number of machines is part of the input. This can be seen as an indication that no efficient algorithm for the scheduling problem exists and led to the development of several approximation algorithms and heuristics. We study three of them in this dissertation: the *jump algorithm*, the *lex-jump algorithm*, and the *list scheduling algorithm*. The last one can also be used in an online setting. If the machines have different speeds, then the worst-case approximation guarantees of these algorithms depend on the number of machines. All known instances on which these algorithms show such a dependency seem to be very artificial and are not likely to be encountered in practice. We apply a smoothed analysis to derive upper bounds for the approximation guarantees of the aforementioned algorithms. These bounds depend only on the amount of random noise the instances are subject to. This gives evidence that no dependence on the number of machines should be observed on real-life instances.

What all the previous problems have in common is that they consist of a single objective which has to be optimized. Apart from these *single-criterion optimization problems* there exists a variety of *multicriteria optimization problems* with several conflicting objectives. For instance, in situations in which the economic point of view is of concern one usually has to balance some quality aspects against the cost aspect. Without any further information about how to weight these objectives, which would allow for a transformation of this problem into a single-criterion optimization problem, the predominant notion is the one of *Pareto optimality*. Solutions which cannot be improved in any of the objectives without loss in any of the other objectives are called *Pareto-optimal* solutions or *Pareto optima*. Pareto-optimal solutions are the reasonable trade-offs for a multiobjective optimization problem. Computing the set of Pareto optima, or *Pareto set* for short, supports the decision maker in choosing the "optimal" solution (with respect to his possibly complex, imprecise, and/or only subconsciously available weighting of the conflicting objectives) and can sometimes be used as an algorithmic tool for solving a single-criterion optimization problem. For both applications the Pareto set should be small to be useful.

We study integer optimization problems with multiple linear objective functions in the fourth part of this thesis and analyze the number of Pareto-optimal solutions. Already for two objectives, instances can be generated whose Pareto set is exponentially large in the number of variables. In contrast, typical real-life instances tend to have only a small number of Pareto-optimal solutions. We support this observation by deriving a polynomial smoothed upper bound.

In the remainder of this introduction we present the framework of smoothed analysis and the problems and algorithms that we analyze in this dissertation more detailed.

1.1 Smoothed Analysis

Smoothed analysis was introduced by Spielman and Teng [ST04] to explain why the simplex method is efficient in practice despite its exponential worst-case running time. In the original model, an adversary chooses an arbitrary instance which is subsequently slightly perturbed at random by adding some Gaussian noise. Thus, pathological instances no longer dominate the analysis. Good smoothed bounds usually indicate good practical behavior because in practice inputs are often subject to a small amount of random noise. For instance, this random noise can stem from measurement errors, numerical imprecision, or rounding errors. It can also model influences that cannot be quantified exactly but for which there is no reason to believe that they are adversarial. Since its invention, smoothed analysis has been successfully applied in a variety of contexts. Two recent surveys [MR11, ST09] summarize some of these results.

We follow a more general model of smoothed analysis due to Beier and Vöcking [BV04]. In this model, the adversary is even allowed to specify the probability distribution of the random noise. The influence he can exert is described by the so-called *smoothing parameter* ϕ denoting the maximum density of the noise.

Let us describe these models more formally. As an example we assume that our problem is parameterized by n variables $x_1, \dots, x_n$, which can take arbitrary real values, and we do not care about how these values are encoded. The input size of an instance consisting of the values $x_1, \dots, x_n$ is defined as n. We consider an algorithm for this problem whose running time T_n for n parameters depends on $x_1, \dots, x_n$. The worst-case running time of this algorithm is defined as $T_{\mathrm{WC}}(n) = \sup_{x_1,\dots,x_n \in \mathbb{R}} T_n(x_1, \dots, x_n)$, which is a function in n. This model assumes that there is an omnipotent adversary who generates, for a given integer n, the worst instance with n parameters for the algorithm at hand.

Smoothed analysis also assumes the existence of an adversary. However, in contrast to worst-case analysis the power of the adversary is limited: In the model of Spielman and Teng, the instance of the adversary is perturbed by adding independent Gaussian random variables $N_1, \dots, N_n \sim \mathcal{N}(0, \sigma^2)$. The *smoothed running time* of the algorithm under consideration is then defined as

$$T_{\mathrm{Sm}}(n, \sigma) = \sup_{x_1,\dots,x_n \in \mathbb{R}} \mathbf{E}_{N_1,\dots,N_n \sim \mathcal{N}(0,\sigma^2)} \left[T_n(x_1 + N_1, \dots, x_n + N_n)\right]$$

and depends on n and, additionally, on σ. This model allows the adversary to specify the mean of the random variables $X_i = x_i + N_{x_i}$ for which the running time $T_n(X_1, \dots, X_n)$ is considered, but not the type of the random noise. The standard deviation σ determines the magnitude of the Gaussian noise: If σ is close to 0, then the instance of the adversary is only changed slightly and we are close to worst-case analysis. Large values of σ make the perturbed instance behave almost random. We say that the algorithm has *smoothed polynomial running time* (in the model of Spielman and Teng) if the smoothed running time $T_{\mathrm{Sm}}(n, \sigma)$ is polynomially bounded in n and $1/\sigma$.

Problems and algorithms are often invariant under scaling. For our simple example this means that $T_n(x_1, \dots, x_n) = T_n(\lambda x_1, \dots, \lambda x_n)$ for any $n \in \mathbb{N}$, any $x_1, \dots, x_n \in \mathbb{R}$, and any $\lambda > 0$. For these problems and algorithms, only values from $[-1, 1]$ have to be considered for $x_1, \dots, x_n$. If this is the case, then our model, which follows the model of Beier and Vöcking, can be applied. Here, the adversary specifies probability density functions $f_1, \dots, f_n \colon [-1, 1] \to [0, \phi]$ according to which random variables X_i are drawn independently of each other. The smoothed running time is then defined as

$$T_{\mathrm{Sm}}(n, \phi) = \sup_{f_1, \dots, f_n \colon [-1,1] \to [0, \phi]} \mathbf{E}_{X_1 \sim f_1, \dots, X_n \sim f_n} \left[T_n(X_1, \dots, X_n) \right]$$

and depends on n and ϕ: If $\phi = 1/2$, then the adversary has to choose $f_i = f$ for any $i = 1, \dots, n$, where $f(z) = 1/2$ for $z \in [-1, 1]$ and $f(z) = 0$ otherwise, because the integral $\int_{-1}^{1} f_i(x)\,\mathrm{d}x$ must be 1. That is, the smoothed running time $T_{\mathrm{Sm}}(n, 1/2)$ equals the average-case running time when all parameters are drawn uniformly at random from the interval $[-1, 1]$. On the other hand, the adversary can, for instance, specify intervals I_i of length $1/\phi$ that contain worst-case choices for the values x_i (with respect to T_n) from which the random variables X_i are drawn uniformly at random. That is, $f_i(z) = \phi$ if $z \in I_i$ and $f_i(z) = 0$ otherwise. Hence, if ϕ becomes large, then the smoothed running time $T_{\mathrm{Sm}}(n, \phi)$ approaches the worst-case running time $T_{\mathrm{WC}}(n)$. We say that the algorithm has smoothed polynomial running time (in our model) if the smoothed running time $T_{\mathrm{Sm}}(n, \phi)$ is polynomially bounded in n and ϕ.

Note that in our model the adversary can specify the type of random noise, contrasting the model of Spielman and Teng. This makes this model more general. Furthermore, let us remark that the previous considerations only highlight the essential differences between the worst-case model, the smoothed model of Spielman and Teng, and our model. Usually not all of the parameters are perturbed in order to obtain stronger theoretical results or because the model would be unrealistic otherwise. This is why we do not introduce a general smoothed input model but we present concrete models tailored to the problems dealt within this dissertation.

1.2 The Minimum-Cost Flow Problem

Flow problems have gained a lot of attention in the second half of the twentieth century to model, for example, transportation and communication networks [AMO93, FF62]. In the *minimum-cost flow problem* we are given a simple directed graph $G = (V, E)$, where each edge e is assigned a *capacity* $u_e \geq 0$ which limits the amount of flow on it and a *cost* $c_e \geq 0$ which states how much it costs to send one unit of flow over it. We call this graph a *flow network*. For convenience, we assume that there are no directed cycles of length two. Additionally, each node v is assigned a *balance value* $b_v \in \mathbb{R}$ which states how much flow the node supplies ($b_v > 0$) or how much it demands ($b_v < 0$). A *feasible b-flow* is a function $f\colon E \to \mathbb{R}_{\geq 0}$ that obeys the capacity constraint $0 \leq f(e) \leq u_e$ for each edge e and *Kirchhoff's law* adapted to the balance values. That is, the effective outflow of each node v equals its balance value, or formally, $b_v = \sum_{e'=(v,w)\in E} f(e') - \sum_{e=(u,v)\in E} f(e)$. The *cost* $c(f) = \sum_{e\in E} f(e) \cdot c_e$ of a b-flow f is the product of the amount of flow sent over an edge e times the cost c_e of this edge, aggregated over all edges e. The minimum-cost flow problem asks to compute a cheapest feasible b-flow, a so-called *minimum-cost b-flow*, if one exists.

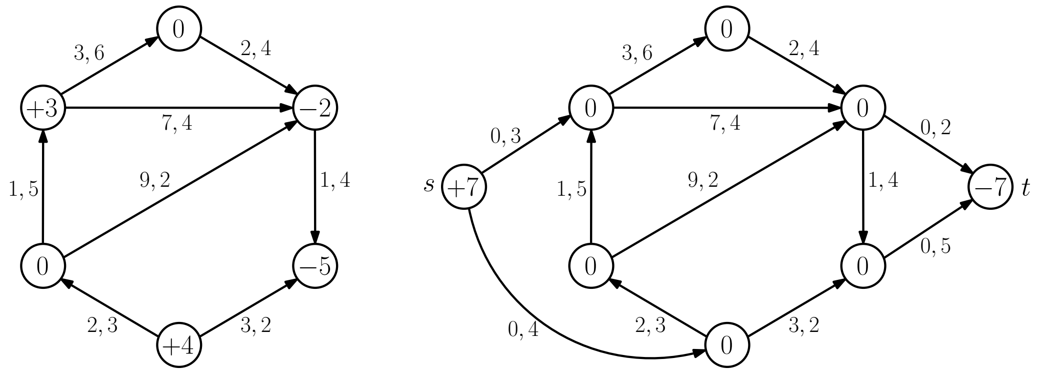
1.2.1 The Successive Shortest Path Algorithm

One algorithm for solving the minimum-cost flow problem is the *successive shortest path (SSP) algorithm*. We start with introducing some essential notation in order to describe this algorithm. For a pair $e = (u, v)$, we denote by e^{-1} the pair (v, u). Let G be a flow network with edge capacities u_e and edge costs c_e, and let f be a b-flow (for an appropriate choice of balance values b_v). The *residual network* G_f is the directed graph with vertex set V, with edge set $E' = E_\mathrm{f} \cup E_\mathrm{b}$, where $E_\mathrm{f} = \{e : e \in E \text{ and } f(e) < u_e\}$ is the set of so-called *forward edges* and $E_\mathrm{b} = \{e^{-1} : e \in E \text{ and } f(e) > 0\}$ is the set of so-called *backward edges*, and with *residual edge capacities* and *residual edge costs*

$$u'_e = \begin{cases} u_e - f(e) & \text{if } e \in E\,, \\ f(e^{-1}) & \text{if } e^{-1} \in E \end{cases} \quad \text{and} \quad c'_e = \begin{cases} c_e & \text{if } e \in E\,, \\ -c_{e^{-1}} & \text{if } e^{-1} \in E\,. \end{cases}$$

In practice, the simplest way to implement the SSP algorithm is to transform the instance to an equivalent instance with only one *supply node* (a node with positive balance value) and one *demand node* (a node with negative balance value). For this, we add two nodes s and t to the network which we call *master source* and *master sink*, edges (s, v) for any supply node v, and edges (w, t) for any demand node w. The capacities of these *auxiliary edges* (s, v) and (w, t) are set to $b_v > 0$ and $-b_w > 0$, respectively. The costs of the auxiliary edges are set to 0. Now we set $b'_v = 0$ for any node $v \notin \{s, t\}$ and $b'_s = -b'_t = \max\{U_s, U_t\}$, where U_s is the sum of the capacities of the auxiliary edges incident to s and U_t is the sum of the capacities of the auxiliary edges incident to t. Note that $U_s \neq U_t$ implies that there exists neither a feasible b'-flow for the transformed flow network nor a feasible b-flow for the original flow network. A minimum-cost b'-flow f' for the transformed flow network can easily be transformed into a minimum-cost b-flow f for the original flow network by setting $f(e) = f'(e)$ for any edge e of the original flow network.

One example for the transformation of a multiple-source-multiple-sink minimum-cost flow network into a single-source-single-sink minimum-cost flow network is depicted in Figure 1.1. The balance values are given in the center of the nodes, the edge costs and edge capacities are given for each edge e by the label c_e, u_e.



(a) Original minimum-cost flow network

(b) Transformed minimum-cost flow network with master source s and master sink t

Figure 1.1: Transformation of a multiple-source-multiple-sink minimum-cost flow network into a single-source-single-sink minimum-cost flow network

In the remainder we use the term *flow* to refer to a feasible b'-flow for arbitrary balance values b'_v with the property that $b'_s = -b'_t$ and $b'_v = 0$ for any $v \notin \{s, t\}$. We will denote by $|f| = \sum_{e=(s,v)\in E} f(e)$ the *value* of flow f, that is, the amount of flow shipped from s to t in flow f.

The SSP algorithm for a minimum-cost flow network with a single source s, a single sink t, and with $b_s = -b_t = z > 0$ is given as Algorithm 1.

Algorithm 1 SSP algorithm for single-source-single-sink minimum-cost flow networks with $b_s = -b_t = z$.

```
1: start with the empty flow $f_0 \equiv 0$
2: for $i = 1, 2, \ldots$ do
3:     if $|f_{i-1}| = z$ then output $f_i$
4:     if $G_{f_{i-1}}$ does not contain a (directed) $s$-$t$ path then output an error message
5:     find a shortest $s$-$t$ path $P_i$ in $G_{f_{i-1}}$ with respect to the edge costs
6:     augment the flow as much as possible* along path $P_i$ to obtain a new flow $f_i$
7: end for
```

*The value $|f_i|$ of flow f_i must not exceed z and f_i must obey all capacity constraints.

Theorem 1.2.1. *In any iteration i, the flow f_i that is computed by Algorithm 1 is a minimum-cost $b^{(i)}$-flow for the balance values $b_s^{(i)} = -b_t^{(i)} = |f_i|$ and $b_v^{(i)} = 0$ for any $v \notin \{s, t\}$.*

Theorem 1.2.1 is due to Jewell [Jew62], Iri [Iri60], and Busacker and Gowen [BG60]. We refer to Korte and Vygen [KV07] for a proof. As a consequence, no residual network G_{f_i} contains a directed cycle with negative total costs. Otherwise, we could augment along such a cycle to obtain a $b^{(i)}$-flow f' with smaller costs than f_i. In particular, this implies that the shortest paths in G_{f_i} from s to nodes $v \in V$ form a shortest path tree rooted at s. Since the choice of the value z only influences the last augmentation of the algorithm, the algorithm performs the same augmentations when run for two different values $z_1 < z_2$ until the flow value $|f_i|$ exceeds z_1. We will exploit this observation in our analysis.

1.2.2 Related Work

Plenty of algorithms have been developed for the minimum-cost flow problem over the last fifty years. The first pseudo-polynomial algorithm was the out-of-kilter algorithm independently proposed by Minty [Min60] and by Fulkerson [Ful61]. The simplest pseudo-polynomial algorithms are the primal cycle canceling algorithm by Klein [Kle67] and the dual *successive shortest path (SSP) algorithm* by Jewell [Jew62], Iri [Iri60], and Busacker and Gowen [BG60]. By introducing a scaling technique, Edmonds and Karp [EK72] modified the SSP algorithm to obtain the *capacity scaling algorithm*, which was the first polynomial-time algorithm for the minimum-cost flow problem.

The first strongly polynomial algorithms were given by Tardos [Tar85] and by Orlin [Orl84]. Later, Goldberg and Tarjan [GT89] proposed a pivot rule for the cycle canceling algorithm to obtain the strongly polynomial *minimum-mean cycle canceling (MMCC) algorithm*. The fastest known strongly polynomial algorithm up to now is the enhanced capacity scaling algorithm due to Orlin [Orl93] and has a running time of $O(m \log(n)(m+n \log n))$, where $m = |E|$ is the number of edges and $n = |V|$ is the number of nodes of the flow network. For an extensive overview of minimum-cost flow algorithms we suggest the paper of Goldberg and Tarjan [GT90], the paper of Vygen [Vyg02], and the book of Ahuja et al. [AMO93].

Zadeh [Zad73] showed that the SSP algorithm has an exponential worst-case running time. Contrary to this, the worst-case running times of the capacity scaling algorithm and the MMCC algorithm are $O(m(\log U)(m+n \log n))$ [EK72] and $O(m^2 n^2 \min\{\log(nC), m\})$ [RG94], respectively. Here, U denotes the maximum edge capacity and C denotes the maximum edge cost. In particular, the former is polynomial whereas the latter is even strongly polynomial. However, the notions of pseudo-polynomial, polynomial, and strongly polynomial algorithms always refer to worst-case running times, which do not always resemble the algorithms' behavior on real-life instances. Algorithms with large worst-case running times do not inevitably perform poorly in practice. An experimental study of Király and

Kovács [KK12] indeed observes running time behaviors significantly deviating from what the worst-case running times indicate. The MMCC algorithm is completely outperformed by the SSP algorithm. The capacity scaling algorithm is the fastest of these three algorithms, but its running time seems to be in the same order of magnitude as the running time of the SSP algorithm.

1.2.3 Our Results

We explain why the SSP algorithm comes off so well by applying the framework of smoothed analysis. Without loss of generality we can assume that all edge costs c_e are from the interval $[0, 1]$. This can be obtained by scaling down all edge costs equally. In our input model the adversary does not fix the edge costs $c_e \in [0, 1]$ for each edge e, but he specifies probability density functions $f_e \colon [0, 1] \to [0, \phi]$ according to which the costs c_e are randomly drawn independently of each other. As in worst-case analysis, the flow network G, the edge capacities u_e, and the balance values b_v of the nodes are chosen adversarially. We define the smoothed running time of an algorithm as the worst expected running time the adversary can achieve and we prove the following theorem.

Theorem 1.2.2. *The SSP algorithm requires $O(mn\phi)$ iterations in expectation and has a smoothed running time of $O(mn\phi(m + n \log n))$.*

If ϕ is a constant – which seems to be a reasonable assumption if it models, for example, measurement errors – then the smoothed bound simplifies to $O(mn(m+n \log n))$. Hence, it is unlikely to encounter instances on which the SSP algorithm requires an exponential amount of time. Note, that the bound on the smoothed running time immediately follows from the bound on the expected number of iterations as each iteration can be implemented to run in time $O(m+n \log n)$ (see, e.g., the book of Korte and Vygen [KV07] for details).

Now let us compare the running times of the SSP algorithm and the strongly polynomial MMCC algorithm. Radzik and Goldberg [RG94] presented a family of worst-case instances on which the MMCC algorithm requires $\Omega(m^2 n)$ minimum-mean cost cycle cancellations, which for $m = \Omega(n \log n)$ is already in the same order as the smoothed running time of the SSP algorithm. Together with the observation that the best known algorithm for computing minimum-mean cost cycles has a running time of $\Theta(mn)$ [DG97], this can be viewed as the first theoretical indication for why the SSP algorithm clearly outperforms the strongly polynomial MMCC algorithm in experiments. However, admittedly this comparison is not entirely fair because it compares the smoothed running time of the SSP algorithm with the worst-case running time of the MMCC algorithm. For a complete explanation why the SSP algorithm outperforms the MMCC algorithm in practice a smoothed lower bound of the running time of the latter algorithm must be derived that exceeds the upper bound stated in Theorem 1.2.2.

Although the MMCC algorithm was mainly considered for theoretical reasons and not necessarily intended for practical purposes, this comparison serves as an example for a phenomenon which can be explained by smoothed analysis but not by worst-case analysis.

In his master's thesis, Rösner [Rös14] constructs an instance on which the SSP algorithm requires $\Omega(m\phi \cdot \min\{n, \phi\})$ iterations showing that the bound stated in Theorem 1.2.2 is tight for $\phi = \Omega(n)$.

Applications to Linear Programming A *linear program (LP)* is an optimization problem where the objective function is linear and the set of feasible solutions is a *polyhedron P* which is defined by *linear constraints.* It is a fundamental property of LPs that, except for a few exceptions which are not relevant for this thesis, the optimal value is attained in a vertex of the polyhedron.

A classic algorithm for solving linear programs is the *simplex method.* This algorithm starts in some vertex x_0 of the polyhedron P and walks along the edges of P such that the objective value does not deteriorate. Usually, a vertex has several neighbors whose objective values are better than its own value. A so-called *pivot rule* specifies which of these neighbors should be visited next. The *shadow vertex pivot rule* has gained a lot of attention among theoreticians due to the average-case analysis of Borgwardt [Bor86] and due to Spielman and Teng [ST04] who showed that the simplex method with the shadow vertex pivot rule, or *shadow vertex method* for short, has a polynomial running time if the coefficients of the linear constraints are subject to Gaussian noise.

Let w denote the vector that describes the linear objective $x \mapsto w^{\mathrm{T}}x$ of the linear program. The shadow vertex method computes a vector c such that the objective function $x \mapsto c^{\mathrm{T}}x$ is optimized by x_0 and no other vertex of P. Then it projects the polyhedron P onto the 2-dimensional plane that is spanned by the vectors w and c. If we assume for the sake of simplicity that P is bounded, then the resulting projection is a polygon Q. The crucial properties of the polygon Q are as follows: both the projection of x_0 and the projection of the optimal solution $x^\star$ are vertices of Q, and every edge of Q corresponds to an edge of P. The shadow vertex method follows the edges of Q from the projection of x_0 to the projection of $x^\star$. The aforementioned properties guarantee that this corresponds to a feasible walk on the polyhedron P. For a complete and more formal description, we refer the reader to [Bor86] or [ST04].

The result of Spielman and Teng, which has later been significantly improved by Vershynin [Ver09], was a breakthrough in algorithm analysis and made smoothed analysis a generally accepted alternative to worst-case analysis in theory. However, their smoothed input model cannot be applied to all linear programs. There are many examples for which perturbations of the constraints' coefficients completely destroy the structure of the problem the LP describes. One example is the linear program for the *maximum flow problem.* In the maximum flow problem we are given a directed graph $G = (V, E)$ with two special nodes $s, t \in V$, which we refer to as the *source* and the *sink*, and with an *edge capacity* $u_e \geq 0$ for each edge e. This graph is called a *flow network.* A *feasible flow* is a function $f \colon E \to \mathbb{R}_{\geq 0}$ that obeys all capacity constraints $f(e) \leq u_e$ and *Kirchhoff's* *law*, that is, $\sum_{e'=(v,w)\in E} f(e') = \sum_{e=(u,v)\in E} f(e)$ for any $v \in V \setminus \{s,t\}$. In other words, the in-flow and the out-flow of each node except the source and the sink are equal. The objective of the maximum flow problem is to compute a flow f with maximum *value* $|f| = \sum_{e'=(s,w)\in E} f_{e'} - \sum_{e=(u,s)\in E} f_e$. The following *maximum flow linear program* is an equivalent formulation of the maximum flow problem.

$$\begin{aligned}
\max\ & \sum_{e'=(s,w)\in E} f_{e'} - \sum_{e=(u,s)\in E} f_e \\
\text{s.t. } & \forall e \in E\colon f_e \geq 0 && \text{(non-negativity constraints)} \\
& \forall e \in E\colon f_e \leq u_e && \text{(capacity constraints)} \\
& \forall v \in V \setminus \{s,t\}: \sum_{e'=(v,w)\in E} f_{e'} - \sum_{e=(u,v)\in E} f_e = 0 && \text{(Kirchhoff's law)}
\end{aligned}$$

As can be seen, all coefficients of the constraints are from $\{-1,0,1\}$ and perturbing some of them would completely destroy the structure of the problem. Hence, the results of Spielman and Teng and of Vershynin do not have algorithmic consequences here. In the remainder of this section we show an interesting connection between the SSP algorithm and the shadow vertex method for the maximum flow linear program which allows us to apply Theorem 1.2.2.

The empty flow $f_0 \equiv 0$ is a vertex of the polytope defined by the linear constraints of the maximum flow linear program. In particular, it is a feasible solution with minimum costs for any cost vector c with positive entries. Hence, any such cost vector c is a valid choice in the shadow vertex method. For this choice every feasible flow f is projected to the pair $\pi(f) := (|f|, c(f))$, where $c(f)$ denotes the cost of flow f. Theorem 1.2.1 guarantees that the projections $\pi(f_i)$ of the flows f_i encountered by the SSP algorithm are the vertices that define the lower envelope of the polygon that results from projecting the polytope of feasible flows. There are two possibilities for the shadow vertex method for the first step: it can choose to follow either the upper or the lower envelope of this polygon. If it decides for the lower envelope, then it will encounter exactly the same sequence of flows as the SSP algorithm.

This means that Theorem 1.2.2 can also be interpreted as a statement about the shadow vertex method applied to the maximum-flow linear program:

Corollary 1.2.3. *The shadow vertex method for the maximum flow linear program that starts with the empty flow $f_0 \equiv 0$ and draws a vector $c \in (0,1)^{|E|}$ uniformly at random as the second projection direction has expected polynomial running time.*

Corollary 1.2.3 is not interesting with regard to algorithmic consequences for solving the maximum flow problem. It is interesting itself since the described algorithm is a simple variant of the simplex method with polynomial running time for the class of maximum flow linear programs. We do not assume any further properties like smoothness – the flow network including the edge capacities can be arbitrary.

1.3 Finding Short Paths on Polyhedra

Motivated by the connection between the SSP algorithm for the minimum-cost flow problem and the *shadow vertex method* for the maximum flow linear program (see Section 1.2.3) we consider the following problem: Given a matrix $A = [a_1, \dots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$, a vector $b \in \mathbb{R}^m$, and two vertices x_1 and x_2 of the *polyhedron* $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, find a short path from x_1 to x_2 along the edges of P efficiently, where the length of a path is defined as the number of its edges. In this context *efficient* means that the running time of the algorithm is polynomially bounded in m, n, and the length of the path it computes. Note that the polyhedron P does not have to be bounded. This problem is the algorithmic version of the problem that asks for a small upper bound for the *diameter* $d(P)$ of the polyhedron P. The diameter of a polyhedron P is the smallest integer that bounds the length of the shortest path between any two vertices of P from above.

The *polynomial Hirsch conjecture* states that $d(P)$ is polynomially bounded in m and n for any matrix A and any vector b. As long as this conjecture remains unresolved, it is unclear whether there always exists a path of polynomial length between the given vertices x_1 and x_2. Moreover, even if such a path exists, it is open whether there is a polynomial time algorithm to find it.

1.3.1 Our Algorithm

Our algorithm is inspired by the *shadow vertex pivot rule* for the *simplex method* and the insights we gain from the analysis of the successive shortest path algorithm (Chapter 4): First 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 such that x_2 uniquely maximizes $w_2^{\mathrm{T}} x$ subject to $x \in P$. Then project the polyhedron onto the plane spanned by w_1 and w_2 in order to obtain a (possibly unbounded) polygon P'. Let us call the projection π. By the same arguments as for the shadow vertex pivot rule, 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 of same length. 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. We choose w_1 and w_2 randomly from these cones. A more precise description of this algorithm is given as Algorithm 2. Here, by $N(x)$ we refer to the normalization $N(x) = (1/\|x\|_2) \cdot x$ of a vector $x \neq 0$.

Let us give some remarks on the algorithm above. The vectors $u_1, \dots, u_n$ in Line 1 and the vectors $v_1, \dots, v_n$ in Line 2 must exist because x_1 and x_2 are vertices of P. The only point where our algorithm makes use of randomness is in Line 3. By the choice of w_1 and w_2 in Line 4, x_1 is the unique optimum of the linear program $\min w_1^{\mathrm{T}} x$ s.t. $x \in P$ and x_2 is the unique optimum of the linear program $\max w_2^{\mathrm{T}} x$ s.t. $x \in P$. The former

Algorithm 2 Shadow Vertex Algorithm

```
1: Determine $n$ linearly independent rows $u_k^{\mathrm{T}}$ of $A$ for which $u_k^{\mathrm{T}} x_1 = b_k$.
2: Determine $n$ linearly independent rows $v_k^{\mathrm{T}}$ of $A$ for which $v_k^{\mathrm{T}} x_2 = b_k$.
3: Draw vectors $\lambda, \mu \in (0,1]^n$ independently and uniformly at random.
4: Set $w_1 = -[N(u_1), \dots, N(u_n)] \cdot \lambda$ and $w_2 = [N(v_1), \dots, N(v_n)] \cdot \mu$.
5: Use the function $\pi : x \mapsto \big(w_1^{\mathrm{T}} x, w_2^{\mathrm{T}} x\big)$ to project $P$ onto the Euclidean plane and
   obtain the shadow vertex polygon $P' = \pi(P)$.
6: Walk from $\pi(x_1)$ along the edges of $P'$ in increasing direction of the second coordinate
   until $\pi(x_2)$ is found.
7: Output the corresponding path of $P$.
```

follows because for any $y \in P$ with $y \neq x_1$ there must be an index $k \in [n]$ with $u_k^{\mathrm{T}} x_1 < b_k$. The latter follows analogously. Note that $\|w_1\| \leq \sum_{k=1}^n \lambda_k \cdot \|N(u_k)\| = \sum_{k=1}^n \lambda_k \leq n$ and, similarly, $\|w_2\| \leq n$. The shadow vertex polygon P' in Line 5 has several important properties: The projections of x_1 and x_2 are vertices of P' and all edges of P' correspond to projected edges of P. Hence, any path on the edges of P' is the projection of a path on the edges of P. Though we call P' a polygon, it does not have to be bounded. This is the case if P is unbounded in the directions w_1 or $-w_2$. Nevertheless, there is always a path from x_1 to x_2 which will be found in Line 6. For more details about the shadow vertex pivot rule and formal proofs of these properties, we refer to the book of Borgwardt [Bor86].

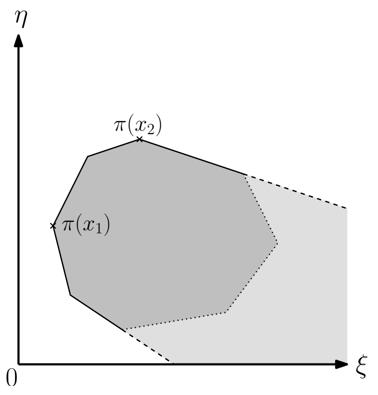


Figure 1.2: Shadow polygon P'

To provide some intuition why these statements hold true, consider the projection depicted in Figure 1.2. We denote the first coordinate of the Euclidean plane by ξ and the second coordinate by η. Since w_1 and w_2 are chosen such that x_1 and x_2 are, among the points of P, optimal for the functions $x \mapsto w_1^{\mathrm{T}} x$ and $x \mapsto w_2^{\mathrm{T}} x$, respectively, the projections $\pi(x_1)$ and $\pi(x_2)$ of x_1 and x_2 must be the leftmost vertex and the topmost vertex of $P' = \pi(P)$, respectively. As P' is a (not necessarily bounded) polygon, this implies that if we start in vertex $\pi(x_1)$ and follow the edges of P' in the direction of increasing values of η, then we will end up in $\pi(x_2)$ after a finite number of steps. This is not only true if P' is bounded (as depicted by the dotted line and the dark gray area) but also if P is unbounded (as depicted by the dashed lines and the dark gray plus the light gray area). Moreover, note that the slopes of the edges of the path from $\pi(x_1)$ to $\pi(x_2)$ are positive and monotonically decreasing.

1.3.2 Related Work

The diameter of polyhedra has been studied extensively in the last decades. In 1957, Hirsch conjectured that the diameter $d(P)$ of P is bounded by $m - n$ for any matrix A and any vector b (see Dantzig's seminal book about linear programming [Dan63]). This conjecture has been disproven by Klee and Walkup [KW67] who gave an unbounded counterexample. However, it remained open for quite a long time whether the conjecture holds for bounded polyhedra, so-called *polytopes*. More than fourty years later Santos [San10] gave the first counterexample to this refined conjecture showing that there are polytopes P for which $d(P) \geq (1 + \varepsilon) \cdot m$ for some $\varepsilon > 0$. This is the best known lower bound today. On the other hand, the best known upper bound of $O(m^{1+\log n})$ due to Kalai and Kleitman [KK92] is only quasi-polynomial. It is still an open question whether $d(P)$ is always polynomially bounded in m and n. This has only been shown for special classes of polyhedra like 0/1 polytopes, flow polytopes, and the transportation polytope. For these classes of polyhedra bounds of $m - n$ (Naddef [Nad89]), $O(mn \log n)$ (Orlin [Orl97]), and $O(m)$ (Brightwell et al. [BvdHS06]) have been shown, respectively. On the other hand, there are bounds on the diameter of far more general classes of polyhedra that depend polynomially on m, n, and on additional parameters. Recently, Bonifas et al. [BDE$^+$12] showed that the diameter of polyhedra P defined by integer matrices A is bounded by a polynomial in n and a parameter that depends on the matrix A. They showed that $d(P) = O(\Delta^2 n^4 \log(n\Delta))$, where Δ is the largest absolute value of any sub-determinant of A. Although the parameter Δ can be very large in general, this approach allows to obtain bounds for classes of polyhedra for which Δ is known to be small. For example, if the matrix A is *totally unimodular*, that is, if all sub-determinants of A are from $\{-1, 0, 1\}$, then their bound simplifies to $O(n^4 \log n)$, improving the previously best known bound of $O(m^3 n^{16} \log^3(mn))$ by Dyer and Frieze [DF94].

We are not only interested in the existence of a short path between two vertices of a polyhedron but we want to compute such a path efficiently. It is clear that lower bounds for the diameter of polyhedra have direct (negative) consequences for this algorithmic problem. However, upper bounds for the diameter do not necessarily resolve the algorithmic version of the problem as they might be non-constructive. The aforementioned bounds of Orlin, Brightwell et al., and Dyer and Frieze are constructive, whereas the bound of Bonifas et al. is not.

1.3.3 Our Results

We give a constructive upper bound for the diameter of the (not necessarily bounded) polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ for arbitrary matrices $A \in \mathbb{R}^{m \times n}$ and arbitrary vectors $b \in \mathbb{R}^m$. This bound is polynomial in m, n, and a parameter $1/\delta$, which depends only on the matrix A and is a measure for the angle between edges of the polyhedron P and their neighboring facets. We say that a facet F of the polyhedron P is neighboring an edge e if exactly one of the endpoints of e belongs to F. The parameter δ denotes the smallest sine of any angle between an edge and a neighboring facet in P. If, for example, every edge is orthogonal to its neighboring facets, then $\delta = 1$. On the other hand, if there exists an edge that is almost parallel to a neighboring facet, then $\delta \approx 0$. The formal definition of δ is deferred to Section 5.4. We obtain the constructive bound by analyzing Algorithm 2, which we will call *shadow vertex method* in the following.

Theorem 1.3.1. *Given vertices x_1 and x_2 of P, the shadow vertex method efficiently computes a path from x_1 to x_2 on the polyhedron P with expected length $O(mn^2/\delta^2)$.*

Let us emphasize that the algorithm is very simple and its running time depends only polynomially on m, n and the length of the path it computes.

Theorem 1.3.1 does not resolve the polynomial Hirsch conjecture as the value δ can be exponentially small. Furthermore, it does not imply a good running time of the shadow vertex method for optimizing linear programs because for the variant considered in this thesis both vertices have to be known. Contrary to this, in the optimization problem one has to determine the optimal vertex for a given linear objective. To compare our results with the result by Bonifas et al. [BDE$^+$12], we show that if A is an integer matrix, then $1/\delta \leq n \cdot \Delta^2$, which yields the following corollary.

Corollary 1.3.2. *Let $A \in \mathbb{Z}^{m \times n}$ be an integer matrix and let $b \in \mathbb{R}^m$ be a real-valued vector. Given vertices x_1 and x_2 of P, the shadow vertex method efficiently computes a path from x_1 to x_2 on the polyhedron P with expected length $O(\Delta^4 mn^4)$.*

This bound is worse than the bound of Bonifas et al., but it is constructive. Furthermore, if A is a totally unimodular matrix, then $\Delta = 1$. Hence, we obtain the following corollary.

Corollary 1.3.3. *Let $A \in \mathbb{Z}^{m \times n}$ be a totally unimodular matrix and let $b \in \mathbb{R}^m$ be a vector. Given vertices x_1 and x_2 of P, the shadow vertex method efficiently computes a path from x_1 to x_2 on the polyhedron P with expected length $O(mn^4)$.*

The bound stated in Corollary 1.3.3 significantly improves upon the previously best known constructive bound of $O(m^3 n^{16} \log^3(mn))$ due to Dyer and Frieze.

1.4 Scheduling

Parallelization is an important concept used to increase utilization of resources when processing a bunch of tasks. It is present in almost all economical areas. One example is the parallel manufacturing of parts which are later assembled to obtain the final product. *Scheduling theory*, which dates back to the early fifties of the 20th century (see, for instance, the work of Jackson [Jac55], Johnson [Joh54], and Smith [Smi56] and the survey of Graham et al. [GLLK79]), has also influenced the design of modern operating systems. Nowadays most of the computers have a multi-core processor that allows several processes to be executed simultaneously. Universities and large companies even operate clusters of hundreds of computers to enhance parallelization.

In many scenarios where parallelization is applied, minimizing the time it takes to process a set of given tasks is one main objective. The more independent the tasks are of each other the better they can be parallelized: If a task cannot be executed before another task has been processed, then serialization of these two tasks is necessary preventing them to be executed in parallel. In the scheduling variants we consider in this thesis, the tasks, which we refer to as *jobs*, are completely independent of each other. This property holds, for instance, for a typical set of user application processes which do not interact with each other and, hence, can be assigned to different cores of a multi-core processor.

Let $J = \{1, \dots, n\}$ be the set of jobs and let $M = \{1, \dots, m\}$ be the set of available cores, which we refer to as *machines*. The time p_{ij} it takes to fully process a job j on a machine i depends on the size $p_j > 0$ of job j, usually called its *processing requirement*, the *speed* $s_i > 0$ of machine i, and whether job j is allowed to be processed on machine i or not. The set of machines on which job j may be processed is denoted by $\mathcal{M}_j \subseteq M$. With this notation we can define p_{ij} as follows:

$$p_{ij} = \begin{cases} p_j/s_i & \text{if } i \in \mathcal{M}_j\,, \\ \infty & \text{otherwise}\,. \end{cases}$$

If all machines i have the same speed, without loss of generality $s_i = 1$, then we call this setting *scheduling with identical machines*. Otherwise, we call it *scheduling with related machines*. Independently of these settings we distinguish between the scenario where each job can be processed on each machine and the scenario where there are jobs that are not allowed to be processed on all machines. We call these settings *scheduling with unrestricted machines* and *scheduling with restricted machines*, respectively. In the former setting we do not explicitly mention the sets $\mathcal{M}_j$ as they all equal set M.

A scheduling policy has to decide which jobs to process on which machine. We only consider scenarios in which jobs have no arrival times or deadlines at which they have to be processed. All jobs are available for processing right from the beginning. In the scheduling variants we analyze in this thesis we do not allow jobs to migrate from one machine to another during their time of execution. Consequently, any job is assigned to exactly one machine where it is completely processed. The time it takes for a machine i to process all jobs j assigned to it equals the sum of the times p_{ij} of these jobs. When the assignment from jobs to machines, which we call a *schedule*, is known, then this time is already determined and neither depends on the order in which the jobs are executed nor on whether the jobs are allowed to be preempted (when execution is later resumed on the same machine). The goal of the scheduling problems we consider here is to minimize the time when the last machine finishes processing all of its jobs (assuming that all machines start processing jobs at time 0), which is called the *makespan* of the given schedule.

For ease of explanation we assume that the jobs that are scheduled on a machine i share this processor in such a way that they all finish at the same time. This does not change the problem itself since we are only interested in minimizing the time when the last job is processed completely. The advantage of this viewpoint is that the *finishing time* of a job is well-defined without specifying an order in which the jobs are processed. With this definition, the algorithms we want to study can be described much better.

An instance I of a scheduling problem consists of machine speeds $s_1, \ldots, s_m$, processing requirements $p_1, \ldots, p_n$, and, in the restricted setting, sets $\mathcal{M}_1, \ldots, \mathcal{M}_n \subseteq M$ of allowed machines. Even in the case that all speeds are equal, the problems under consideration are known to be strongly $\mathcal{NP}$-hard if the number m of machines is part of the input (see, for example, Garey and Johnson [GJ79]). This has motivated a lot of research in the previous decades on approximation algorithms for scheduling problems. Since some of the theoretically best approximation algorithms are rather involved, a lot of research has focused on simple heuristics like *greedy algorithms* and *local search algorithms* which are easy to implement. While greedy algorithms make reasonable ad hoc decisions to obtain a schedule, local search algorithms start with some schedule and iteratively improve the current schedule by performing some kind of local improvements until no such is possible anymore. In this thesis, we consider the following algorithms that can be applied to all scheduling variants that we have described above:

- The *list scheduling algorithm* is a greedy algorithm that starts from an empty schedule and a list of jobs. Then it repeatedly selects the next unscheduled job from the list and assigns it to the machine on which it will be completed the earliest with respect to the current partial schedule. We call any schedule that can be generated by list scheduling a *list schedule*.

- The *jump algorithm* and the *lex-jump algorithm* are local search algorithms that start with an arbitrary schedule and iteratively perform a *local improvement step*. In each improvement step, one job is reassigned (it jumps) from a machine i to a different machine i' where it finishes earlier. In the jump algorithm, only jobs on *critical machines*, that is, machines with maximum finishing time, are allowed to be reassigned. In the lex-jump algorithm, each job can be reassigned at any time. A schedule for which there is no jump improvement step or no lex-jump improvement step is called *jump optimal* or *lex-jump optimal*, respectively.

For each of these three algorithms we are interested in their performance guarantees, that is, the worst-case bound on the ratio of the makespan of a schedule returned by the algorithm over the makespan of an optimal schedule. The final schedule returned by a local search algorithm is called a *local optimum.* Usually, there are multiple local optima for a given scheduling instance both for the jump and the lex-jump algorithm with varying quality. As we do not know which local optimum is found by the local search, we will always bound the quality of the worst local optimum. Since local optima for lex-jump and pure Nash equilibria are the same (see for example [Vöc07]), this corresponds to bounding the *price of anarchy* in the scheduling game that is obtained if jobs are selfish agents trying to minimize their own completion time and if the makespan is considered as the *social welfare function.* Similarly, the list scheduling algorithm can produce different schedules depending on how the jobs in the list are ordered. Hence, also for the list scheduling algorithm we will bound the quality of the worst schedule that can be obtained.

1.4.1 Related Work

The approximability of scheduling with makespan minimization is well understood. Cho and Sahni [CS80] showed that the list scheduling algorithm has a performance guarantee of at most $1 + \sqrt{2m-2}/2$ for $m \geq 3$ and that it is at least $\Omega(\log m)$. Aspnes et al. [AAF$^+$97] improved the upper bound to $O(\log m)$ matching the lower bound asymptotically. Hochbaum and Shmoys [HS88] designed a polynomial time approximation scheme for this problem. Polynomial time approximation algorithms and polynomial time approximation schemes for special cases of the problem on restricted related machines are given in, among others, [Li06, GK07, OLL08]. More work on restricted related machines is discussed in the survey of Leung and Li [LL08].

In the last decade, there has been a strong interest in understanding the worst-case behavior of local optima. We refer to the survey [Ang06] and the book [MAK07] for a comprehensive overview of the worst-case analysis and other theoretical aspects of local search. It follows from the work of Cho and Sahni [CS80] that for the problem on unrestricted related machines the performance guarantee of the jump algorithm is $\big(1+\sqrt{4m-3}\big)/2$. This bound is tight [SV07]. Czumaj and Vöcking [CV07] showed that the performance guarantee of the lex-jump algorithm is $\Theta\big(\min\{(\log m)/(\log\log m), \log(s_{\max}/s_{\min})\}\big)$, where $s_{\max}$ and $s_{\min}$ denote the speed of the fastest and the slowest machine, respectively. For the problem on restricted related machines, Rutten et al. [RRSV12] showed that the performance guarantee of the jump algorithm is $\big(1+\sqrt{1+4(m-1)s_{\max}/s_{\min}}\big)/2$ and that this bound is tight up to a constant factor. Moreover, they showed that the performance guarantee of the lex-jump algorithm is $\Theta\big((\log S)/(\log\log S)\big)$, where $S = \big(\sum_{i=1}^{m} s_i\big)/s_{\min}$. When all machines have the same speed, Awerbuch et al. [AART06] showed that the performance guarantee of the lex-jump algorithm is $\Theta\big((\log m)/(\log\log m)\big)$.

Up to now, smoothed analysis has been mainly applied to running time analyses (see, for example, [ST09] for a survey). The first exception is the paper by Becchetti et al. [BLMS$^+$06] who introduced the concept of smoothed competitive analysis, which is equivalent to smoothed performance guarantees for online algorithms. Schäfer and Sivadasan [SS05] performed a smoothed competitive analysis for metrical task systems. Englert et al. [ERV07] considered the 2-Opt algorithm for the traveling salesman problem and determined, among others, the smoothed performance guarantee of local optima of the 2-Opt algorithm. Hoefer and Souza [HS10] presented one of the first average case analyses for the price of anarchy.

1.4.2 Our Results

The performance guarantee of local search and greedy algorithms for scheduling problems is well studied and understood. For most algorithms, matching upper and lower bounds on their approximation ratio are known (see Table 1.1). The worst-case approximation guarantees of the jump algorithm and the lex-jump algorithm are known for all scheduling variants and it is constant only for the simplest case with unrestricted identical machines. In all other cases it increases with the number m of machines. The lower bounds are often somewhat contrived, however, and it is questionable whether they resemble typical instances in practical applications. This is why we study these algorithms in the framework of smoothed analysis, in which instances are subject to some degree of random noise. By doing so, we find out for which heuristics and scheduling variants the lower bounds are robust and for which they are fragile and not very likely to occur in practical applications.

As usual, due to a scaling argument we can assume without loss of generality that all processing requirements are from the interval $[0,1]$ and that the slowest machine has speed $s_{\min} = 1$. In our model the adversary specifies probability densities $f_j \colon [0,1] \to [0,\phi]$ according to which the processing requirements p_j are drawn at random independently of each other. All others parameters, that is, the number m of machines, the machine speeds $s_{\max} := s_1 \geq \ldots \geq s_m =: s_{\min} = 1$ (in the case of non-identical machines), the number n of jobs, and the sets $\mathcal{M}_1, \ldots, \mathcal{M}_n \subseteq M$ of allowed machines (in the case of restricted machines), can be chosen arbitrarily by the adversary. We call such an instance $\mathcal{I}$ for scheduling a *ϕ-smooth* instance. Formally, a ϕ-smooth instance is not a single instance but a distribution over instances. We write $I \sim \mathcal{I}$ to denote that the instance I is drawn from the ϕ-smooth instance $\mathcal{I}$.

In this thesis, we analyze the *smoothed performance guarantee* of the jump algorithm, the lex-jump algorithm, and the list scheduling algorithm. As mentioned earlier, to define the approximation guarantee of these algorithms on a given instance, we consider the worst local optimum (for the jump algorithm and the lex-jump algorithm) or the worst order of the jobs in the list (for the list scheduling algorithm). Now, the smoothed performance is defined to be the worst expected approximation guarantee of any ϕ-smooth instance.

Our results for the jump and lex-jump algorithm are summarized in Table 1.1. The smoothed performance guarantees for all variants of restricted machines turned out to be robust against random noise. We show that even for large perturbations, that is, when ϕ

	jump algorithm		lex-jump algorithm	
	worst-case	ϕ-smooth	worst-case	ϕ-smooth
unrestricted identical	$\Theta(1)$ [FH79, SV07]	$\Theta(1)$	$\Theta(1)$ [FH79, SV07]	$\Theta(1)$
unrestricted related	$\Theta\left(\sqrt{m}\right)$ [CS80, SV07]	$\Theta(\phi)$ [6.2.1]	$\Theta\left(\frac{\log m}{\log\log m}\right)$ [CV07]	$\Theta(\log\phi)$ [6.2.2, 6.2.3]
restricted identical	$\Theta\left(\sqrt{m}\right)$ [RRSV12]	$\Theta\left(\sqrt{m}\right)$ [6.3.1]	$\Theta\left(\frac{\log m}{\log\log m}\right)$ [AART06]	$\Theta\left(\frac{\log m}{\log\log m}\right)$ [6.3.2]
restricted related	$\Theta\left(\sqrt{m \cdot s_{\max}}\right)$ [RRSV12]	$\Theta\left(\sqrt{m \cdot s_{\max}}\right)$ [6.3.1]	$\Theta\left(\frac{\log S}{\log\log S}\right)$ [RRSV12]	$\Omega\left(\frac{\log m}{\log\log m}\right)$ [6.3.2]

Table 1.1: Worst-case and smoothed performance guarantees for the jump algorithm and the lex-jump algorithm. Here, $S = \sum_{i=1}^{m} s_i$, and we assume without loss of generality that $s_{\min} = 1$. With [6.X.Y] we refer to the section in this thesis where the bound is shown.

is a constant, the worst-case lower bounds carry over. This can be seen as an indication that, in general, neither the jump algorithm nor the lex-jump algorithm yield a good approximation ratio for scheduling with restricted machines in practice.

The situation is much more promising for the unrestricted variants. Here, the worst-case bounds are fragile and do not carry over to the smoothed case. The interesting case is the one of unrestricted and related machines. Even though both for the jump algorithm and for the lex-jump algorithm the worst-case lower bound is not robust, there is a significant difference between these two: while the smoothed approximation ratio of the jump algorithm grows linearly with the perturbation parameter ϕ, the smoothed approximation ratio of the lex-jump algorithm grows only logarithmically in ϕ. This proves that also in the presence of random noise lex-jump optimal schedules are significantly better than jump optimal schedules. As mentioned earlier, this also implies that the smoothed price of anarchy is $\Theta(\log\phi)$. Additionally, we show that the smoothed approximation ratio of the list scheduling algorithm is $\Theta(\log\phi)$ in the model with unrestricted related machines, even when the order of the list may be specified after the realizations of the processing times are known. This indicates that both the lex-jump algorithm and the list scheduling algorithm should yield good approximations on practical instances.

Our results particularly imply that the average-case performance guarantees for the jump algorithm, the lex-jump algorithm, and the list-scheduling algorithm are constant.

In his bachelor's thesis, Etscheid [Ets13] considers a smoothed input model where the machine speeds rather then the processing requirements are perturbed. For the environment with unrestricted machines he obtains the same smoothed bounds for the jump, the lex-jump, and the list scheduling algorithm as we did. For the environment with restricted related machines, the worst-case bounds for the jump and the lex-jump algorithm are rather robust against perturbations of processing requirements. Due to their dependence on the machine speeds these bounds should, however, be more fragile in the smoothed input model of Etscheid. Indeed, Etscheid derives bounds of $\Theta(m\sqrt{\phi})$ and $\Theta(\min\{m, (\log(m\phi))/(\log\log(m\phi))\})$ for the jump and the lex-jump algorithm.

1.5 Pareto-Optimal Solutions

In most real-life decision-making problems there is more than one objective to be optimized. For example, when booking a train ticket, one wishes to minimize the travel time, the fare, and the number of train changes. As different objectives are often conflicting, usually no solution is simultaneously optimal in all criteria and one has to make a trade-off between different objectives. The most common way to filter out unreasonable trade-offs and to reduce the number of solutions the decision maker has to choose from is to determine the set of Pareto-optimal solutions, where a solution is called *Pareto-optimal* or a *Pareto optimum* if no other solution is simultaneously better in all criteria.

Multiobjective optimization problems have been studied extensively in operations research and theoretical computer science (see, for example, [Ehr05] for a comprehensive survey). In particular, many algorithms for generating the set of Pareto-optimal solutions for various optimization problems such as the (bounded) knapsack problem [NU69, KW00], the multiobjective shortest path problem [CM85, Han80, SA00], and the multiobjective network flow problem [Ehr99, MG98] have been proposed. Enumerating the set of Pareto-optimal solutions is not only used as a preprocessing step to eliminate unreasonable trade-offs, but often it is also used as an intermediate step in algorithms for solving optimization problems. For example, the Nemhauser–Ullmann algorithm [NU69] treats the single-criterion knapsack problem as a bicriteria optimization problem in which a solution with small weight and large profit is sought, and it generates the set of Pareto-optimal solutions, ignoring the given capacity of the knapsack. After this set has been generated, the algorithm returns the solution with the highest profit among all Pareto-optimal solutions with weight not exceeding the knapsack capacity. This solution is optimal for the given instance of the knapsack problem.

Generating the set of Pareto-optimal solutions, or *Pareto set* for short, only makes sense if few solutions are Pareto-optimal. Otherwise, it is too costly and it does not provide enough guidance to the decision maker. While, in many applications, it has been observed that the Pareto set is indeed usually small (see, for example, [MHW01] for an experimental study of the multiobjective shortest path problem), one can, for almost every problem with more than one objective function, easily find instances with an exponential number of Pareto-optimal solutions (see, for example, [Ehr05]).

1.5.1 Model and Previous Work

We consider a very general model of multiobjective optimization problems. An instance of such a problem consists of $d+1$ objective functions $V^1, \dots, V^{d+1}$ that are to be optimized over a set $\mathcal{S} \subseteq \{0, \dots, \mathcal{K}\}^n$ of feasible solutions for some integer $\mathcal{K}$. While the set $\mathcal{S}$ and the last objective function $V^{d+1} \colon \mathcal{S} \to \mathbb{R}$ can be arbitrary, the first d objective functions have to be linear, that is, they are of the form $V^t(x) = V^t_1 x_1 + \dots + V^t_n x_n$ for $x = (x_1, \dots, x_n) \in \mathcal{S}$ and $t \in \{1, \dots, d\}$. We assume without loss of generality that all objectives are to be minimized and that all coefficients V_i^t are from the interval $[-1, 1]$. A solution $x \in \mathcal{S}$ is called *Pareto-optimal* if there is no solution $y \in \mathcal{S}$ which is at least as good as x in all of the objectives and even better than x in at least one. We will introduce this notion formally in Section 7.2. The set of Pareto-optimal solutions is called the *Pareto set.* We are interested in the size of this set. As a convention, we count distinct Pareto-optimal solutions that coincide in all objective values only once. Since we compare solutions based on their objective values, there is no need to consider more than one solution with exactly the same values.

If one is allowed to choose the set $\mathcal{S}$, the objective function V^{d+1}, and the coefficients of the linear objective functions $V^1, \dots, V^d$ arbitrarily, then, even for $d = 1$, one can easily construct instances with an exponential number of Pareto-optimal solutions. For this reason, Beier and Vöcking [BV04] introduced the model of *ϕ-smooth instances*, in which an adversary can choose the set $\mathcal{S}$ and the objective function V^{d+1} arbitrarily while he can only specify a probability density function $f_i^t \colon [-1, 1] \to [0, \phi]$ for each coefficient V_i^t according to which it is chosen independently of the other coefficients.

The *smoothed number of Pareto-optimal solutions* depends on the number n of integer variables, the maximum integer $\mathcal{K}$, and the perturbation parameter ϕ. It is defined to be the largest expected number of Pareto-optimal solutions the adversary can achieve by any choice of $\mathcal{S} \subseteq \{0, \dots, \mathcal{K}\}^n$, $V^{d+1} \colon \mathcal{S} \to \mathbb{R}$, and the densities $f_i^t \colon [-1, 1] \to [0, \phi]$. In the following we assume that the adversary has made arbitrary fixed choices for these entities. Then we can associate with every matrix $V \in \mathbb{R}^{d \times n}$ the number $\mathrm{PO}(V)$ of Pareto-optimal solutions in $\mathcal{S}$ when the coefficients V_i^t of the d linear objective functions take the values given in V. Assuming that the adversary has made worst-case choices for $\mathcal{S}$, V^{d+1}, and the densities f_i^t, the *smoothed number of Pareto-optimal solutions* is the expected value $\mathbf{E}_V[\mathrm{PO}(V)]$, where the coefficients in V are chosen according to the densities f_i^t. For $c \geq 1$ we call $\mathbf{E}_V[\mathrm{PO}^c(V)]$ the *c-th moment of the smoothed number of Pareto-optimal solutions.*

Beier and Vöcking [BV04] showed that for the binary bicriteria case (that is, $\mathcal{K} = d = 1$) the smoothed number of Pareto-optimal solutions is $O(n^4\phi)$ and $\Omega(n^2)$. The upper bound was later simplified and improved by Beier et al. [BRV07] to $O(n^2\phi)$. In his PhD thesis [Bei04], Beier conjectured that the smoothed number of Pareto-optimal solutions is polynomially bounded in n and ϕ for $\mathcal{K} = 1$ and every constant d. This conjecture was proven by Röglin and Teng [RT09], who showed that for binary solutions and for any fixed $d \geq 1$, the smoothed number of Pareto-optimal solutions is $O((n^2\phi)^{f(d)})$, where the function f is roughly $f(d) = 2^d d!$. They also proved that for any constant c the c-th moment of the smoothed number of Pareto-optimal solutions is bounded by $O\big(\big((n^2\phi)^{f(d)}\big)^c\big)$. Recently, Moitra and O'Donnell [MO11] improved the bound for the smoothed number of Pareto-optimal solutions significantly to $O(n^{2d}\phi^{d(d+1)/2})$. However, it remained unclear how to improve the bound for the moments by their methods.

1.5.2 Our Results

Besides general ϕ-smooth instances, we additionally consider the special case of *quasiconcave density functions.* This means that we assume that every coefficient V_i^t is chosen independently according to its own density function $f_i^t \colon [-1,1] \to [0,\phi]$ with the additional requirement that for every density f_i^t there is a value $x_i^t \in [-1,1]$ such that f_i^t is monotonically increasing in the interval $[-1,x_i^t]$ and monotonically decreasing in the interval $[x_i^t,1]$. We do not think that this is a severe restriction because all natural perturbation models, like Gaussian or uniform perturbations, use quasiconcave density functions. Furthermore, quasiconcave densities capture the essence of a perturbation: each coefficient V_i^t has an unperturbed value x_i^t and the probability that the perturbed coefficient takes a value z becomes smaller with increasing distance $|z - x_i^t|$. We will call these instances *quasiconcave ϕ-smooth instances* in the following.

We present significantly improved bounds for the smoothed number of Pareto optima and the moments, answering two questions posed by Moitra and O'Donnell [MO11].

Theorem 1.5.1. *For any $d \geq 1$, the smoothed number of Pareto-optimal solutions is $(\mathcal{K}+1)^{2(d+1)^2} \cdot O(n^{2d}\phi^{d(d+1)})$ for general ϕ-smooth instances and $(\mathcal{K}+1)^{2(d+1)^2} \cdot O(n^{2d}\phi^{d})$ for quasiconcave ϕ-smooth instances.*

The bound of Theorem 1.5.1 for quasiconcave ϕ-smooth instances improves the previously best known bound of $O(n^{2d}\phi^{d(d+1)/2})$ in the binary case (which is, however, valid also for non-quasiconcave densities) and it answers a question posed by Moitra and O'Donnell whether it is possible to improve the factor of $\phi^{d(d+1)/2}$ in their bound [MO11].

Theorem 1.5.2. *For any $d \geq 1$, there is a quasiconcave ϕ-smooth instance for $\mathcal{K} = 1$ such that the smoothed number of Pareto-optimal solutions is $\Omega(\min\{n^{f(d)}\phi^d, 2^{\Theta(n)}\})$, where $f(1) = 2$ and $f(d) = d - 1.5$ for $d \geq 2$.*

Theorem 1.5.1 and Theorem 1.5.2 together show that the exponents of both n and ϕ are linear in d. The minimum in Theorem 1.5.2 is no severe restriction because the number of Pareto-optimal solutions never exceeds 2^n in the binary case. Furthermore, for $d = 1$ this bound matches the upper bound of $O(n^2\phi)$ of Beier et al. [BRV07].

The following result bound higher moments of the smoothed number of Pareto-optimal solutions.

Theorem 1.5.3. *For any $d \geq 1$ and any constant $c \in \mathbb{N}$, the c-th moment of the smoothed number of Pareto-optimal solutions is $(\mathcal{K}+1)^{(c+1)^2(d+1)^2} \cdot O((n^{2d}\phi^{d(d+1)})^c)$ for general ϕ-smooth instances and $(\mathcal{K}+1)^{(c+1)^2(d+1)^2} \cdot O((n^{2d}\phi^{d})^c)$ for quasiconcave ϕ-smooth instances.*

This answers a question in [MO11] whether it is possible to improve the bounds for the moments in [RT09] and it yields better concentration bounds for the smoothed number of Pareto-optimal solutions. Our results also have immediate consequences for the expected running times of various algorithms because most heuristics for generating the Pareto set of some problem (including the ones mentioned at the beginning of the introduction) have a running time that depends linearly or quadratically on the size of the Pareto set. The improved bounds on the smoothed number of Pareto-optimal solutions and the second moment of this number yield improved bounds on the smoothed running times of these algorithms.

Beier and Vöcking originally only considered ϕ-smooth instances for binary bicriteria optimization problems (that is, for the case $\mathcal{K} = d = 1$). The above described canonical generalization of this model to binary multiobjective optimization problems, on which Röglin and Teng's [RT09] and Moitra and O'Donnell's results [MO11] are based, appears to be very general and flexible on the first glance. However, one aspect limits its applicability severely and makes it impossible to formulate certain multiobjective linear optimization problems in this model. The weak point of the model is that it assumes that every binary variable x_i appears in every linear objective function as it is not possible to set some coefficients V_i^t deterministically to zero.

Already Spielman and Teng [ST04] and Beier and Vöcking [BV06] observed that the zeros often encode an essential part of the combinatorial structure of a problem and they suggested to analyze *zero-preserving perturbations* in which it is possible to either choose a density f_i^t according to which the coefficient V_i^t is chosen or to set it deterministically to zero. Zero-preserving perturbations have been studied in [SST06] and [BV06] for analyzing smoothed condition numbers of matrices and the smoothed complexity of binary optimization problems. For the smoothed number of Pareto-optimal solutions no upper bounds are known that are valid for zero-preserving perturbations (except trivial worst-case bounds), and in particular the bounds proven in [RT09] and [MO11] do not seem to generalize easily to zero-preserving perturbations. In this thesis, we present new techniques for analyzing the smoothed number of Pareto-optimal solutions that can also be used for analyzing zero-preserving perturbations.

Theorem 1.5.4. *For any $d \geq 1$, the smoothed number of Pareto-optimal solutions is $(\mathcal{K}+1)^{(d+1)^5} \cdot O((n\phi)^{d^3+d^2+d})$ for general ϕ-smooth instances with zero-preserving perturbations and $(\mathcal{K}+1)^{(d+1)^5} \cdot O(n^{d^3+d^2+d}\phi^d)$ for quasiconcave ϕ-smooth instances with zero-preserving perturbations.*

Note, that for constant $\mathcal{K}$ like in the binary case the factor $(\mathcal{K}+1)^{\mathrm{poly}(d)}$ is a constant for fixed d. We will see that they allow us not only to extend the smoothed analysis to linear multiobjective optimization problems that are not captured by the previous model without zero-preserving perturbations, but that they also enable us to bound the smoothed number of Pareto-optimal solutions in problems with *non-linear objective functions*. In particular, the number of Pareto-optimal solutions for polynomial objective functions can be bounded by Theorem 1.5.4.

Moreover note that our analysis also covers the general case when the set $\mathcal{S}$ is an arbitrary subset of $\{-\mathcal{K}, \dots, \mathcal{K}\}^n$. In this case, consider the shifted set $\mathcal{S}' = \{x + u : x \in \mathcal{S}\} \subseteq$

$\{0,\dots,2\mathcal{K}\}$ for $u = (\mathcal{K},\dots,\mathcal{K})$ and the functions $W^1,\dots,W^{d+1}\colon \mathcal{S}' \to \mathbb{R}$, defined as $W^t x = V^t x$ for $t = 1,\dots,d$ and $W^{d+1}x = V^{d+1}(x-u)$. The Pareto set with respect to $\mathcal{S}$ and $\{V^1,\dots,V^{d+1}\}$ and the Pareto set with respect to $\mathcal{S}'$ and $\{W^1,\dots,W^{d+1}\}$ are identical except for a shift of $(V^1u,\dots,V^du,0)$ in the image space. Hence, the sizes of both sets are equal. All aforementioned results can be applied for $\mathcal{S}'$ and $\{W^1,\dots,W^{d+1}\}$, so they also hold for $\mathcal{S}$ and $\{V^1,\dots,V^{d+1}\}$ if one replaces $\mathcal{K}$ by $2\mathcal{K}$.

Let us remark that for the bicriteria case $d = 1$, which was studied by Beier and Vöcking, zero-preserving perturbations are not more powerful than other perturbations (see Section 7.1). Zero-preserving perturbations become interesting when we consider more than two objectives.

Applications to Path Trading Berger et al. [BRvdZ11] study a model for routing in networks. In their model there is a graph $G = (V,E)$ whose vertex set V is partitioned into mutually disjoint sets $V_1,\dots,V_k$. We can think of G as the Internet graph whose vertices are owned and controlled by k different *autonomous systems* (ASes). We denote by $E_i \subseteq E$ the set of edges inside V_i. The graph G is undirected, and each edge $e \in E$ has a length $\ell_e \in \mathbb{R}_{\geq 0}$. The traffic is modeled by a set of requests, where each request is characterized by its source node $s \in V$ and its target node $t \in V$. The *Border Gateway Protocol (BGP)* determines for each request (s,t) the order in which it has to be routed through the ASes. We say that a path P from s to t is *valid* if it connects s to t and visits the ASes in the order specified by the BGP protocol. This means that the first AS has to choose a path P_1 inside V_1 from s to some node in V_1 that is connected to some node $v_2 \in V_2$. Then the second AS has to choose a path P_2 inside V_2 from v_2 to some node in V_2 that is connected to some node $v_3 \in V_3$ and so on. For simplicity, the costs of routing a packet between two ASes are assumed to be zero, whereas AS i incurs costs of $\sum_{e\in P_i} \ell_e$ for routing the packet inside V_i along path P_i. In the common *hot-potato routing*, every AS is only interested in minimizing its own costs for each request. To model this, there are k objective functions that map each valid path P to a cost vector $(C_1(P),\dots,C_k(P))$, where

$$C_i(P) = \sum_{e\in P\cap E_i} \ell_e \quad \text{for } i \in \{1,\dots,k\}\,.$$

In [BRvdZ11] the problem of *path trading* is considered. If there is only one request, then no AS has an incentive to deviate from the hot-potato strategy. The problem becomes more interesting if there are multiple requests that have to be satisfied. Consider, for example, the three ASes depicted in Figure 1.3 and assume that there are three requests (s_1,t_1), (s_2,t_2), and (s_3,t_3). Moreover, assume that the BGP specifies that any request from $s \in V_i$ to $t \in V_j$ shall be routed directly from AS i to AS j. If all ASes follow the hot-potato strategy, then they decide for the routes (s_1,u_1,w_2,s_1), (s_2,u_2,w_3,t_2), and (s_3,u_3,w_1,t_3). Each AS i incurs costs of 1 for the request (s_i,t_i) and costs of 9 for the request (s_j,t_j) for which $t_j \in V_i$.

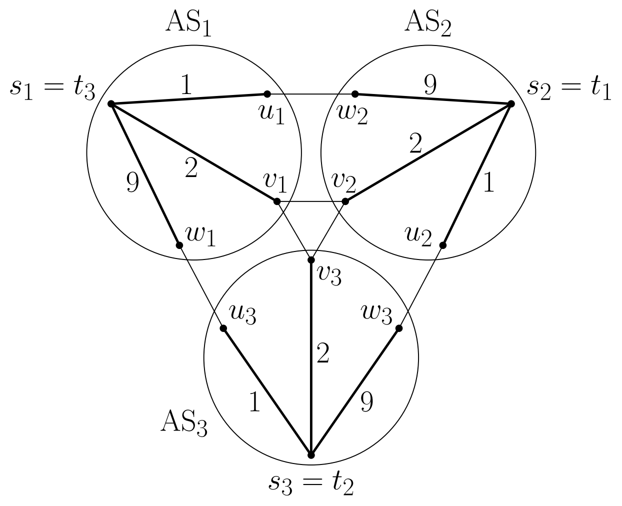


Figure 1.3: A network graph with three autonomous systems

Now assume that AS i routes request (s_i, t_i) from s_i to v_i. Then it incurs costs of 2 (instead of 1) for this route, which is worse than if it had chosen the hot-potato route. However, if all ASes agree on this new strategy, then each AS i only incurs costs of 2 (instead of 9) for the request (s_j, t_j) for which $t_j \in V_i$. Hence, the total costs of each AS for satisfying the three requests (s_i, t_i) is 4 instead of 10.

The path trading problem asks whether there exist routes for given requests (s_i, t_i) such that the total costs of each involved AS is less or equal the total costs it would incur if all would follow the hot-potato strategy. Such routes are called *feasible path trades.*

Consider $\mathcal{K}$ requests $(s_1, t_1), \dots, (s_\mathcal{K}, t_\mathcal{K})$ and s_i-t_i-paths $P_1, \dots, P_\mathcal{K}$ that comply with the BGP. For an edge $e \in E$ let $x_e \in \{0, \dots, \mathcal{K}\}$ be the number of paths $P_1, \dots, P_\mathcal{K}$ that contain edge e. We can encode the routes $P_1, \dots, P_\mathcal{K}$ by an integer vector $x \in \{0, \dots, \mathcal{K}\}^{|E|}$ consisting of the values x_e. Let $\mathcal{S}$ denote the set of encodings of all valid routes $P_1, \dots, P_\mathcal{K}$. The question whether there is a feasible path trade for the requests (s_i, t_i) reduces to the question whether the vector $x^\star$ that encodes the hot-potato routes $P_1^\star, \dots, P_\mathcal{K}^\star$ is not Pareto-optimal with respect to $\mathcal{S}$ and $\{C_1, \dots, C_k\}$, where the objectives $C_i \colon \mathcal{S} \to \mathbb{R}$,

$$C_i(x) = \sum_{e \in E_i} \ell_e x_e \,,$$

describe the total costs of AS i for the routes encoded by x. As the Pareto set can be exponentially large in the worst case, Berger et al. [BRvdZ11] proposed to study ϕ-smooth instances in which an adversary chooses the graph G and a density $f_e \colon [0,1] \to [0, \phi]$ for every edge length ℓ_e according to which it is chosen. It seems as if we could easily apply the results in [RT09] and [MO11] to bound the smoothed number of Pareto-optimal paths because all objective functions C_i are linear in the binary variables x_e, $e \in E$.

Note, however, that different objective functions contain different variables x_e because the coefficients of all x_e with $e \notin E_i$ are set to zero in C_i. This is an important combinatorial property of the path trading problem that has to be obeyed. In the model in [RT09] and [MO11] it is not possible to set coefficients deterministically to zero. The best we can do is to replace each zero by a uniform density on the interval $[0, 1/\phi]$. Then, however, an AS would incur positive costs for any edge and not only for its own edges that are used, which does not resemble the structure of the problem. Theorem 1.5.4, which allows zero-preserving perturbations, yields immediately the following result.

Corollary 1.5.5. *The smoothed number of Pareto-optimal valid paths is polynomially bounded in $|E|$, ϕ, and $\mathcal{K}$ for any constant number k of ASes.*

Applications to Non-linear Objective Functions Even though we assumed above that the objective functions $V^1, \dots, V^d$ are linear, we can also extend the smoothed analysis to non-linear objective functions. We consider first the bicriteria case $d = 1$. As above, we assume that the adversary has chosen an arbitrary set $\mathcal{S}$ of feasible solutions and an arbitrary injective objective function $V^2 \colon \mathcal{S} \to \mathbb{R}$. In addition to that the adversary can choose m_1 arbitrary functions $I_i^1 \colon \mathcal{S} \to \{0, \dots, \mathcal{K}\}$, $i \in \{1, \dots, m_1\}$. The objective function $V^1 \colon \mathcal{S} \to \mathbb{R}$ is defined to be a weighted sum of the functions I_i^1:

$$V^1(x) = \sum_{i=1}^{m_1} w_i^1 I_i^1(x)\,,$$

where each weight w_i^1 is randomly chosen according to a density $f_i^1 \colon [-1, 1] \to [0, \phi]$ given by the adversary. There is a wide variety of functions $V^1(x)$ that can be expressed in this way. We can, for example, express every polynomial if we let $I_1^1, \dots, I_{m_1}^1$ be its monomials. Note, that the value $\mathcal{K}$ then depends on the set $\mathcal{S}$ and the maximum degree of the monomials.

We can linearize the problem by introducing a binary variable for every function I_i^1. Using the function $\pi \colon \mathcal{S} \to \{0, \dots, \mathcal{K}\}^{m_1}$, defined by $\pi(x) = (I_1^1(x), \dots, I_{m_1}^1(x))$, the set of feasible solutions becomes $\mathcal{S}' = \{\pi(x) : x \in \mathcal{S}\} \subseteq \{0, \dots, \mathcal{K}\}^{m_1}$. For this set of feasible solutions we define $W^1 \colon \mathcal{S}' \to \mathbb{R}$ and $W^2 \colon \mathcal{S}' \to \mathbb{R}$ as follows:

$$W^1(y) = \sum_{i=1}^{m_1} w_i^1 y_i \quad \text{and} \quad W^2(y) = \min\left\{V^2(x) : x \in \mathcal{S} \text{ and } \pi(x) = y\right\}\,.$$

One can easily verify that the problem defined by $\mathcal{S}$, V^1, and V^2 and the problem defined by $\mathcal{S}'$, W^1, and W^2 are equivalent and have the same number of Pareto-optimal solutions. The latter problem is linear and hence we can apply the result by Beier et al. [BRV07], which yields that the smoothed number of Pareto-optimal solutions is bounded by $\text{poly}(\mathcal{K}+1) \cdot O(m_1^2 \phi)$. This shows in particular that the smoothed number of

Pareto-optimal solutions is polynomially bounded in the number of monomials, the maximum integer in the monomials' ranges, and the density parameter for every polynomial objective function V^1.

We can easily extend these considerations to multiobjective problems with $d \geq 2$. For these problems the adversary chooses an arbitrary set $\mathcal{S}$, numbers $m_1, \dots, m_d \in \mathbb{N}$, and an arbitrary injective objective function $V^{d+1} \colon \mathcal{S} \to \mathbb{R}$. In addition to that he chooses arbitrary functions $I_i^t \colon \mathcal{S} \to \{0, \dots, \mathcal{K}\}$ for $t \in \{1, \dots, d\}$ and $i \in \{1, \dots, m_t\}$. Every objective function $V^t \colon \mathcal{S} \to \mathbb{R}$ is a weighted sum of the functions I_i^t:

$$V^t(x) = \sum_{i=1}^{m_t} w_i^t I_i^t(x) \,,$$

where each weight w_i^t is randomly chosen according to a density $f_i^t \colon [-1,1] \to [0,\phi]$ chosen by the adversary. Similar as the bicriteria case, also this problem can be linearized. However, the previously known results about the smoothed number of Pareto-optimal solutions can only be applied if every objective function V^t is composed of exactly the same functions I_i^t. Theorem 1.5.4 yields the following result.

Corollary 1.5.6. *The smoothed number of Pareto-optimal solutions for multiobjective optimization problems with non-linear integral objectives is polynomially bounded in $\sum_{i=1}^t m_i$, $\mathcal{K}$, and ϕ.*

As a special case of non-linear objective functions we consider polynomials. Now, our set $\mathcal{S}$ is again a subset of $\{0, \dots, \mathcal{K}\}^n$ and the objective functions V^t, $t \in \{1, \dots, d\}$, are polynomials. That is, the objectives V^t are weighted sums of monomials. The adversary can specify a ϕ-bounded density on $[-1,1]$ for every weight according to which it is chosen. We call such an instance a *ϕ-smooth instance with polynomial objective functions.* Denote the total number of monomials by m and let Δ denote the maximum degree of the monomials. Then the following corollary holds.

Corollary 1.5.7. *For any $d \geq 1$, the smoothed number of Pareto-optimal solutions is $(\mathcal{K}^\Delta + 1)^{(d+1)^5} \cdot O((m\phi)^{d^3+d^2+d})$ for general ϕ-smooth instances with polynomial objective functions and $(\mathcal{K}^\Delta + 1)^{(d+1)^5} \cdot O(m^{d^3+d^2+d}\phi^d)$ for quasiconcave ϕ-smooth instances with polynomial objective functions.*

1.6 Outline and Bibliographical Notes

In Chapter 2 we introduce the essential techniques that are used throughout this thesis. Fundamental theorems concerning random variables are stated in Chapter 3. Subsequently, we prove the results about the successive shortest path algorithm (Chapter 4), about finding short paths on polyhedra (Chapter 5), about the jump, the lex-jump, and the list scheduling algorithm (Chapter 6), and about Pareto-optimal solutions (Chapter 7). Chapter 8 is devoted to conclusions and directions of future research.

The results of Chapter 4 have been presented in preliminary form at the following conference:

⋆ [BCMR13] Tobias Brunsch, Kamiel Cornelissen, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the successive shortest path algorithm. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1180–1189, 2013.

The results of Chapter 5 have been presented in preliminary form at the following conference:

⋆ [BR13] 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), Part I*, pages 279–290, 2013.

The results of Chapter 6 have been presented in part in preliminary form at the following conference:

⋆ [BRRV11] Tobias Brunsch, Heiko Röglin, Cyriel Rutten, and Tjark Vredeveld. Smoothed performance guarantees for local search. In *Proceedings of the 19th Annual European Symposium on Algorithms (ESA)*, pages 772–783, 2011.

This publication will also appear in *Mathematical Programming, Series A*. These results have also appeared in the PhD thesis of Cyriel Rutten [Rut13].

The results of Chapter 7 have been presented in part in preliminary form at the following conferences:

⋆ [BR11] Tobias Brunsch and Heiko Röglin. Lower bounds for the smoothed number of pareto optimal solutions. In *Proceedings of the 8th Annual Conference on Theory and Applications of Models of Computation (TAMC)*, pages 416–427, 2011.

⋆ [BR12] Tobias Brunsch and Heiko Röglin. Improved smoothed analysis of multiobjective optimization. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*, pages 407–426, 2012.

Chapter 2

Probabilistic Tools for Smoothed Analyses

In this chapter we present several important tools and techniques that are often used in smoothed or other probabilistic analyses, particularly in this thesis. For a better understanding we will also give motivations for these tools and show exemplary applications illustrating their usefulness.

2.1 Union Bound

Given n events $A_1, \dots, A_n$, the probability of the union $A = \bigcup_{i=1}^n A_i$ is bounded by $\mathbf{Pr}[A] \le \sum_{i=1}^n \mathbf{Pr}[A_i]$. If the events A_i are pairwise disjoint, then we even have equality. On the other hand, the more events A_i can occur simultaneously, the worse the quality of the union bound. If, for instance, $A_1 = \dots = A_n = A$, then the union bound yields $\mathbf{Pr}[A] \le \sum_{i=1}^n \mathbf{Pr}[A] = n \cdot \mathbf{Pr}[A]$. This means that we lose a factor of n.

We often have a non-trivial bound $\mathbf{Pr}[A_i] \le p$ for each event A_i and are interested in the probability of an event $B \subseteq A$. In other words, event B is "covered" by the events A_i whose probabilities we can bound from above. Applying a union bound, we obtain $\mathbf{Pr}[B] \le np$. Note that this bound is non-trivial only for $p < 1/n$.

The union bound is a very simple tool. Nevertheless, it can make an analysis much shorter if one is only interested in an upper bound. As an example, we consider a deck of 32 cards containing 4 aces and 28 other cards. Let us assume that there are four players, each of which draws two cards uniformly at random from the deck and keeps them as hand cards. What is the probability of event A that at least one of the players has two aces? If we are interested in the exact value, then one way to compute this value is to analyze the probability of the complementary event $\bar{A}$. We can express event $\bar{A}$ as the disjoint union of the events $B_0, \dots, B_4$, where B_i denotes the event that exactly i players draw exactly one ace whereas the other players do not draw any ace. Due to a symmetry argument and further considerations we obtain that the probability of event B_i equals $\mathbf{Pr}\left[B_i\right] = \binom{4}{i} \cdot 2^i \cdot \left(\binom{4}{i} \cdot \binom{28}{8-i}\right) / \left(\binom{32}{8} \cdot \binom{8}{i}\right)$. Consequently, the probability of event A equals

$$\mathbf{Pr}\left[A\right] = 1 - \sum_{i=0}^{4} \binom{4}{i} \cdot 2^i \cdot \frac{\binom{4}{i} \cdot \binom{28}{8-i}}{\binom{32}{8} \cdot \binom{8}{i}} \approx 0.0482\,.$$

Even for such a simple random experiment the analysis is not trivial. Let us now denote by A_i the event that player i draws two aces. With this notion, event A is the union of the events $A_1, \dots, A_4$. This union is not disjoint as there might be two players that both draw two aces. However, this is very unlikely. Hence, a union bound should yield a good approximation. The probability of each event A_i equals $\mathbf{Pr}\left[A_i\right] = \binom{4}{2} / \binom{32}{2}$. Applying a union bound yields

$$\mathbf{Pr}\left[A\right] \le 4 \cdot \frac{\binom{4}{2}}{\binom{32}{2}} \approx 0.0484\,.$$

This bound is very close to the exact value of $\mathbf{Pr}\left[A\right]$. The example above demonstrates that a union bound is sometimes a good alternative to an exact analysis if one is only interested in an upper bound.

2.2 Principle of Deferred Decisions

We consider an event $A = A(X, Y)$ which depends on the independent random vectors X and Y. Let $\mathbf{Pr}\left[A \mid Y = y\right]$ denote the probability of event A to occur under the condition that $Y = y$, that is, whether A occurs now only depends on X. The principle of deferred decisions states that if $\mathbf{Pr}\left[A \mid Y = y\right] \le p$ for any possible realization y of Y, then $\mathbf{Pr}\left[A\right] \le p$. The same holds true if we exchange both upper bounds by lower bounds or by equalities.

In many analyses handling the whole amount of information about a random experiment is very difficult. Sometimes, a part of this information is not essential for bounding or computing the probability of a given event A. If this is the case, then we can reduce the randomness of the experiment to the intrinsic portion X by letting an adversary specify the irrelevant part Y arbitrarily.

For an application of the principle of deferred decisions we consider the following example (compare [MU05], Exercise 1.15). Given n independent random variables $X_1, \dots, X_n$ that are uniformly drawn from $[0, 1)$, what is the probability that the fractional part $R(S) := S - \lfloor S \rfloor$ of the sum $S = \sum_{i=1}^{n} X_i$ falls into a given interval $[a, b] \subseteq [0, 1)$?

Without the principle of deferred decisions this is an overwhelming task. We could, for instance, compute the density of S and integrate it over the set $A = \bigcup_{k=0}^{n-1} [a + k, b + k]$. Both subproblems are very challenging. With the principle of deferred decisions the problem becomes much easier: We observe that for any fixed real s the fractional part $R(s + X_1)$ of $s + X_1$ is uniformly distributed on $[0, 1)$. In consequence, the probability that $R(s + X_1)$ falls into the interval $[a, b]$ equals the length $b - a$ of the interval $[a, b]$, independently of the choice of s. Now we fix the random vector $(X_2, \dots, X_n)$ arbitrarily, say $(X_2, \dots, X_n) = (x_2, \dots, x_n)$, and set $s = \sum_{i=2}^n x_i$. Then $S = s + X_1$. As X_1 and $X_2, \dots, X_n$ are independent of each other, we obtain

$$\begin{aligned}
&\mathbf{Pr}\left[R(S) \in [a,b] \mid (X_2, \dots, X_n) = (x_2, \dots, x_n)\right] \\
&\quad = \mathbf{Pr}\left[R(s + X_1) \in [a,b] \mid (X_2, \dots, X_n) = (x_2, \dots, x_n)\right] \\
&\quad = b - a
\end{aligned}$$

due to previous considerations. As this equation holds for any possible realization of $(X_2, \dots, X_n)$, the principle of deferred decisions yields $\mathbf{Pr}\left[R(S) \in [a,b]\right] = b - a$.

2.3 Interval Probability Bound

The interval probability bound is probably the simplest observation tailored to the input model of smoothed analysis. In its pure form it states that the probability that a random variable which is drawn according to a probability density $f \colon \mathbb{R} \to [0, \phi]$ falls into a given interval I of length ℓ is bounded from above by $\ell\phi$. This statement can easily be generalized to an arbitrary number n of random variables and to arbitrary measurable sets in $\mathbb{R}^n$. In this thesis we will only consider Cartesian products of intervals of same length, that is, hypercubes.

The interval probability bound becomes more interesting when we combine it with the principle of deferred decisions. Let C be a function mapping points $(y_1, \dots, y_n)$ from $\mathbb{R}^n$ to an arbitrary hypercube $C(y_1, \dots, y_n) \subseteq \mathbb{R}^k$ with side length ℓ and let $Y_1, \dots, Y_n$ and $Z_1, \dots, Z_k$ be independent random variables that are drawn according to probability densities $f_i \colon \mathbb{R} \to [0, \phi]$ and $g_j \colon \mathbb{R} \to [0, \phi]$, respectively. Then the probability $\mathbf{Pr}\left[(Z_1, \dots, Z_k) \in C(Y_1, \dots, Y_n)\right]$ that $(Z_1, \dots, Z_k)$ falls into the random hypercube $C(Y_1, \dots, Y_n)$ is bounded from above by $(\ell\phi)^k = \ell^k\phi^k$.

Again, this statement can be further generalized: Let $X_1, \dots, X_m$ be independent random variables drawn according to probability densities $f_i \colon \mathbb{R} \to [0, \phi]$. Furthermore, let $A \in \mathbb{R}^{(n+k) \times m}$ be a matrix with linearly independent rows. We consider the random variables $Y_1, \dots, Y_n, Z_1, \dots, Z_k$ that are defined by $(Y_1, \dots, Y_n, Z_1, \dots, Z_k)^{\mathrm{T}} = A \cdot (X_1, \dots, X_m)^{\mathrm{T}}$, that is, they are linearly independent linear combinations of the random variables $X_1, \dots, X_m$. As before, the function C maps a tuple $(y_1, \dots, y_n)$ to a hypercube with side length ℓ. Röglin and Teng [RT09] already observed that the probability $\mathbf{Pr}\left[(Z_1, \dots, Z_k) \in C(Y_1, \dots, Y_n)\right]$ that $(Z_1, \dots, Z_k)$ falls into the random hypercube $C(Y_1, \dots, Y_n)$ is bounded from above by $\ell^k\phi^{n+k} \cdot g(A)$ for some appropriate function g. If we compare this bound with the bound $\ell^k\phi^k$ for the simple setting discussed before, then we observe the same dependence on ℓ. This result implies that linearly independent linear combinations of random variables can, in some sense, be treated like independent random variables by sacrificing a factor of $\phi^n \cdot g(A)$.

Sometimes it would be desirable to dispose of the additional factor ϕ^n. Without further restrictions to the densities this might be very difficult or even impossible. In this thesis we give a very natural restriction for the densities for which the dependence on ϕ decreases from ϕ^{n+k} to ϕ^k: If all densities f_i are *quasiconcave*, that is, each of them is monotonically increasing up to some point starting from which it is monotonically decreasing, then the probability $\mathbf{Pr}\left[(Z_1, \dots, Z_k) \in C(Y_1, \dots, Y_n)\right]$ that $(Z_1, \dots, Z_k)$ falls into the random hypercube $C(Y_1, \dots, Y_n)$ is bounded from above by $\ell^k \phi^k \cdot h(A)$ for some appropriate function h.

For the details and a proof we refer to Theorem 3.3. In Theorem 5.6.1 and Theorem 7.9.1 we consider specific classes of matrices A and derive adequate upper bounds for $g(A)$ and $h(A)$.

2.4 Extraction of Fragile Worst-Case Properties

Smoothed analysis is used to explain why some algorithms behave much better in practice than in the worst case. For this to work, worst-case instances need to have an inherent property that is not robust against random noise. The strategy of the extraction of fragile worst-case properties makes this fact explicit and splits the analysis into two parts: In the first part, we identify a property of worst-case instances which dominates worst-case analyses but is unlikely to be encountered in practice. For this part we consider arbitrary inputs and make no assumptions like smoothness. In the second part we show that the extracted property is fragile in the smoothed input model.

This separation into two parts increases the reusability of insights gained from the analysis. All results from the first part do not depend on the smoothed input model and are also valid for worst-case analyses or smoothed analyses with a different input model. The bachelor's thesis of Etscheid [Ets13] about scheduling algorithms serves as an example. Etscheid analyzes the same scheduling heuristics as we consider in Chapter 6 but for a different input model. In this model the machine speeds rather than the jobs' processing requirements are perturbed. Several statements used to bound the performance guarantees of the lex-jump algorithm and the list scheduling algorithm from above in the setting with unrestricted related machines also proved valuable in the analysis of Etscheid.

Note that the strategy as it is described above only works if there is a strict separation into good and bad instances for the algorithm at hand. Usually, the measure which describes how good the algorithm performs on which instance is not that binary. This is why, in general, the property identified in the first part of the analysis is parameterized. Let us illustrate this with the following example. We consider a heuristic for an optimization problem on a set $\mathcal{I}$ of instances. For an instance $I \in \mathcal{I}_n$ of input length n the heuristic computes a solution which is worse than the optimal solution by a factor of $\alpha(I) \geq 1$. Thus, the worst-case approximation factor of the heuristic, which we allow to be a function in n, is $\alpha^\star(n) = \max_{I \in \mathcal{I}_n} \alpha(I)$ which can be arbitrarily large. Now let us turn to a smoothed analysis. In the first part of the analysis we introduce a property P_t for which we show that for any $t \geq 1$ each instance I with $\alpha(I) \geq t$ has propery P_t. This is the part of the analysis that exposes the nature of worst-case instances for the heuristic at hand: Instances that cause the approximation factor to be larger than some value t must have property P_t.

In the second part of the analysis we show that instances which have property P_t are unlikely to be encountered in the smoothed input model if t is large. This explains why worst-case instances can be considered degenerate in some sense. For this, we consider arbitrary densities that are bounded by the smoothing parameter ϕ and fix them once and for all. These densities define a probability distribution on the set $\mathcal{I}_n$ of instances I with input length n which we express by $I \sim \mathcal{I}_n$. Let us denote by $\mathcal{E}_t = \mathcal{E}_t(I)$ the event that an instance I has property P_t. The next step is to derive an upper bound $g(n, \phi, t)$ for the probability of event $\mathcal{E}_t$ where $g(n, \phi, t) \to 0$ for fixed n and ϕ and for $t \to \infty$. This upper bound must be valid independly of the probability densities. Hence, for any feasible choice of probability densities one can conclude that

$$\mathbf{Pr}_{I \sim \mathcal{I}_n}\left[\alpha(I) \geq t\right] \leq \mathbf{Pr}_{I \sim \mathcal{I}_n}\left[I \text{ has property } P_t\right] = \mathbf{Pr}_{I \sim \mathcal{I}_n}\left[\mathcal{E}_t\right] \leq g(n, \phi, t)\,, \tag{2.1}$$

where the first inequality stems from part one and the second inequality stems from part two of the analysis. This bound can be used to obtain a bound that holds with high probability. Let $\hat{t}(n, \phi)$ be a function for which $g(n, \phi, \hat{t}(n, \phi)) \leq 1/n$ for any n and any ϕ. Then, with high probability, the heuristic has an approximation ratio of at most $\hat{t}(n, \phi)$.

Inequality (2.1) can also be used to bound the expectation of $\alpha(I)$. Using the well-known formula $\mathbf{E}\left[X\right] = \int_0^\infty \mathbf{Pr}\left[X \geq x\right] \mathrm{d}x$ for non-negative random variables X we obtain

$$\mathbf{E}_{I \sim \mathcal{I}_n}\left[\alpha(I)\right] = \int_0^\infty \mathbf{Pr}_{I \sim \mathcal{I}_n}\left[\alpha(I) \geq t\right] \mathrm{d}t \leq 1 + \int_1^\infty g(n, \phi, t)\, \mathrm{d}t\,.$$

As a concrete example for the extraction of fragile worst-case properties we consider the lex-jump algorithm and the list scheduling algorithm for scheduling with unrestriced related machines in the model described in Section 1.4.2. We show that any instance I on which these algorithms generate a schedule whose makespan is larger than the optimal makespan by a factor $\alpha(I) \geq t$ has the following property P_t: at least half of the jobs of I have a processing requirement of at most $2^{-t/6+3}$. As this bound decreases exponentially with t, this implies that the expected approximation ratios of the *lex-jump algorithm* and the *list scheduling algorithm* are $O(\log \phi)$ on unrestricted related machines.

2.5 Discretization

A strong technique for analyzing counting problems is discretization. We are interested in a discrete random variable N that takes positive integers and we want to bound the expected value of N. We are not given direct information about N, but we have N random variables $X_1, \ldots, X_N$ with domain $[0, \ell]$ and the following properties:

(1) $\mathbf{Pr}\left[\exists i\colon X_i \in [t, t+\varepsilon)\right] \le k_1\varepsilon$ for arbitrary real numbers t and $\varepsilon > 0$ and an appropriate constant k_1,

(2) $\mathbf{Pr}\left[\exists i,j : i \neq j \text{ and } |X_i - X_j| < \varepsilon\right] \to 0$ for $\varepsilon \to 0$, and

(3) $N \le k_2$ for an appropriate constant k_2.

Then $\mathbf{E}\left[N\right] \le k_1\ell$. To get a better feeling for this concept let us give a very abstract view of how to bound the expectation of the number N of augmentation steps of the *successive shortest path (SSP) algorithm* for the *minimum-cost flow problem* in the model described in Section 1.2.3. We identify the i^{th} iteration of the SSP algorithm with the length X_i of the path the SSP algorithm augments along in this iteration. We can show that the lengths X_i are monotonically increasing. As the cost of each edge is at most 1 and as each path contains at most n edges, this implies that $X_i \in [0, n]$ for any iteration i. Now we can translate the three properties above as follows.

(1) For arbitrary real numbers t and $\varepsilon > 0$ the probability that the SSP algorithm augments along a path whose length lies in the interval $[t, t+\varepsilon)$ is (at most) proportional to ε. Essentially, we show that $\mathbf{Pr}\left[\exists i\colon X_i \in [t, t+\varepsilon)\right] \le 2m\phi\varepsilon$.

(2) It is unlikely that the lengths of the augmenting paths of two consecutive iterations differ only by a tiny amount. In particular, $\mathbf{Pr}\left[\exists i \neq j : |X_i - X_j| < \varepsilon\right] \le 2n^{2n}\phi\varepsilon$, which tends to 0 as ε tends to 0.

(3) The worst-case number of iterations of the SSP algorithm is finite.

These three properties imply that the expected number of augmentation steps of the successive shortest path algorithm is bounded by $\mathbf{E}\left[N\right] \le 2m\phi n = O(mn\phi)$.

We do not prove the correctness of discretization formally here, but describe the idea how the claimed bound can be derived. If we choose a large integer n and partition the interval $[0, \ell]$ into n subintervals $I_1, \ldots, I_n$ of length ℓ/n, then each interval contains at most one variable X_i with high probability (Property (2)). Hence, the expected number of X_i's in an interval I_k equals the probability that I_k contains a variable X_i. The probability for this event is bounded by $k_1 \cdot (\ell/n)$ for each interval (Property (1)). Since we have n intervals, we obtain the claimed bound by linearity of expectation. Property (3) is used to cover the failure event that there is an interval which contains more than one variable X_i. In this case, we bound N by its worst-case bound yielding an additional term of $f(n) \cdot k_2$, where $f(n) \to 0$ for $n \to \infty$. As we can choose n arbitrarily large for the analysis, this additional term can be neglected.

Discretization can also be generalized to higher dimensions and to partitioning the interval $[0, \ell]$ into variable sized subintervals. Often, the most effort for applying discretization is spent on proving Property (1).

Chapter 3

Some Probability Theory

In this chapter we state three theorems about random variables that are applied in this dissertation. The first two of them are well-known and we only list them for the sake of completeness. Theorem 3.3 considers linear combinations of independent random variables. A special case of this theorem has already been stated by Röglin and Teng ([RT09], Lemma 3.3). The theorem turns out to be useful in Chapter 5 (see Theorem 5.6.1) and in Chapter 7 (see Theorem 7.9.1).

Theorem 3.1 (*Markov's inequality*)**.** *Let X be a nonnegative random variable. For any real $t > 0$ the probability that X exceeds t can be bounded from above by*

$$\mathbf{Pr}[X \geq t] \leq \frac{\mathbf{E}[X]}{t} .$$

Theorem 3.2 (*Hoeffding's inequality*)**.** *Let $X_1, \dots, X_n$ be independent random variables, let $X = \sum_{j=1}^{n} X_j$, and let $\mu = \mathbf{E}[X]$. If each random variable X_j takes values in $[a_j, b_j]$ for some reals a_j and b_j, then for any $t > 0$ the probability that X deviates from its expectation can be bounded from above by*

$$\mathbf{Pr}[X \leq \mathbf{E}[X] - t] \leq \exp\left(\frac{-2t}{\sum_{j=1}^{n}(b_j - a_j)^2}\right) \quad \textit{and}$$

$$\mathbf{Pr}[X \geq \mathbf{E}[X] + t] \leq \exp\left(\frac{-2t}{\sum_{j=1}^{n}(b_j - a_j)^2}\right) .$$

The following theorem bounds the probability that linear combinations of independent random variables fall into a hypercube of side length ε that depends on other linear combinations of these variables. For more explanation see the discussion about the *interval probability bound* in Chapter 2.

Theorem 3.3. *Let $m \leq n$ be integers and let $X_1, \dots, X_n$ be independent random variables, each with a probability density function $f_i \colon [-1,1] \to [0,\phi]$, let $A \in \mathbb{R}^{m \times n}$ be a matrix of rank m, let $k \in [m-1]$ be an integer, let $(Y_1, \dots, Y_{m-k}, Z_1, \dots, Z_k)^{\mathrm{T}} = A \cdot (X_1, \dots, X_n)^{\mathrm{T}}$ be the linear combinations of $X_1, \dots, X_n$ given by A, and let C be a function mapping a tuple $(y_1, \dots, y_{m-k}) \in \mathbb{R}^{m-k}$ to a hypercube $C(y_1, \dots, y_{m-k}) \subseteq \mathbb{R}^k$ with side length ε. Then*

$$\mathbf{Pr}\left[(Z_1, \dots, Z_k) \in C(Y_1, \dots, Y_{m-k})\right] \leq \frac{(2m\alpha)^{m-k}}{|\det(\hat{A})|} \cdot \phi^m \varepsilon^k ,$$

where $\hat{A}$ is an arbitrary full-rank $m \times m$-submatrix of A and α denotes the maximum absolute value of the entries of $\hat{A}$. If all densities f_i are quasiconcave, then even the stronger bound

$$\mathbf{Pr}\left[(Z_1, \dots, Z_k) \in C(Y_1, \dots, Y_{m-k})\right] \leq 2^k \cdot \sum_I \frac{|\det(\hat{A}_I)|}{|\det(\hat{A})|} \cdot \phi^k \varepsilon^k$$

holds. The sum runs over all tuples $I = (i_1, \dots, i_k)$ for which $1 \leq i_1 < \ldots < i_k \leq m$. For such a tuple I, matrix $\hat{A}_I$ is the $(m-k) \times (m-k)$-submatrix of $\hat{A}$ that is obtained from $\hat{A}$ by removing the last k rows and the columns with the numbers $i \in I$.

Let us remark that Röglin and Teng [RT09] gave the proof for the first claim of Theorem 3.3 for binary matrices. This proof can be generalized to arbitrary matrices and we only state it for the sake of completeness. Most part of the work has to be done for proving the second claim of Theorem 3.3. Note, that a function $f \colon [a,b] \to \mathbb{R}$ is called *quasiconcave* if there exists a real $x \in [a,b]$ such that f is monotonically increasing on $[a,x]$ and monotonically decreasing on $[x,b]$.

It is not clear that the bound stated in the second claim is stronger than the one from the first claim. However, in this thesis our focus is on the exponent of ϕ.

Proof. First of all we show that we can assume w.l.o.g. that $n = m$. Otherwise, we can choose m indices $i_1 < \ldots < i_m \in [n]$ for which the matrix $A' = [a_{i_1}, \dots, a_{i_m}]$ is a full-rank square submatrix of A. For the sake of simplicity let us assume that $i_k = k$ for $k = 1, \dots, m$. We apply the principle of deferred decisions and assume that $X_{m+1}, \dots, X_n$ are fixed arbitrarily to some values $x_{m+1}, \dots, x_n$.

Let $A'' = [a_{m+1}, \dots, a_n]$, $A''_1 = A''|_{1,\dots,m-k}$, $A''_2 = A''|_{m-k+1,\dots,m}$, and $x = (x_{m+1}, \dots, x_n)$. Let us further introduce the random vector $(Y'_1, \dots, Y'_{m-k}, Z'_1, \dots, Z'_k) = A' \cdot (X_1, \dots, X_m)$ and the function $C'(Y'_1, \dots, Y'_{m-k}) = C((Y'_1, \dots, Y'_{m-k}) + A''_1 \cdot x) - A''_2 \cdot x$. Observing that $(Y'_1, \dots, Y'_{m-k}) + A''_1 \cdot x = (Y_1, \dots, Y_{m-k})$ and $(Z'_1, \dots, Z'_k) + A''_2 \cdot x = (Z_1, \dots, Z_k)$, we obtain

$$\begin{aligned}(Z_1, \dots, Z_k) \in C(Y_1, \dots, Y_{m-k}) &\iff (Z'_1, \dots, Z'_k) \in C(Y_1, \dots, Y_{m-k}) - A''_2 \cdot x \\ &\iff (Z'_1, \dots, Z'_k) \in C'(Y'_1, \dots, Y'_{m-k}) .\end{aligned}$$

The probability of the last event can be bounded by applying Theorem 3.3 for the $m \times m$-matrix A'.

In the remainder of this proof we assume that $n = m$. As matrix A is a full-rank square matrix, its inverse A^{-1} exists and we can write

$$\begin{aligned}
\mathbf{Pr}\left[(Z_1, \dots, Z_k) \in C(Y_1, \dots, Y_{n-k})\right] &= \int_{y \in \mathbb{R}^{n-k}} \int_{z \in C(y)} f_{Y,Z}(y, z) \, \mathrm{d}z \, \mathrm{d}y \\
&= \int_{y \in \mathbb{R}^{n-k}} \int_{z \in C(y)} |\det(A^{-1})| \cdot f_X\big(A^{-1} \cdot (y, z)\big) \mathrm{d}z \, \mathrm{d}y \\
&= |\det(A^{-1})| \cdot \int_{y \in \mathbb{R}^{n-k}} \int_{z \in C(y)} f_X\big(A^{-1} \cdot (y, z)\big) \mathrm{d}z \, \mathrm{d}y \\
&\leq |\det(A^{-1})| \cdot \varepsilon^k \cdot \int_{y \in \mathbb{R}^{n-k}} \max_{z \in \mathbb{R}^k} f_X\big(A^{-1} \cdot (y, z)\big) \mathrm{d}y \,,
\end{aligned}$$

where $f_{Y,Z}$ denotes the common density of the variables $Y_1, \dots, Y_{n-k}, Z_1, \dots, Z_k$ and $f_X = \prod_{i=1}^n f_i$ denotes the common density of the variables $X_1, \dots, X_n$. The second equality is due to a change of variables. In general, we can bound the integral in the formula above by

$$\begin{aligned}
\int_{y \in \mathbb{R}^{n-k}} \max_{z \in \mathbb{R}^k} f_X\big(A^{-1} \cdot (y, z)\big) \mathrm{d}y &\leq \int_{y \in [-n\alpha, n\alpha]^{n-k}} \max_{z \in \mathbb{R}^k} f_X\big(A^{-1} \cdot (y, z)\big) \mathrm{d}y \\
&\leq \int_{y \in [-n\alpha, n\alpha]^{n-k}} \phi^n \mathrm{d}y = (2n\alpha)^{n-k} \phi^n = (2m\alpha)^{m-k} \phi^m \,,
\end{aligned}$$

where the first inequality is due to the fact that all variables Y_i can only take values in the interval $[-n\alpha, n\alpha]$ as all entries of matrix A are from $[-\alpha, \alpha]$ and as all variables X_j can only take values in the interval $[-1, 1]$.

To prove the statement about quasiconcave functions we first consider arbitrary rectangular functions, i.e., functions that are constant on a given interval, and zero otherwise. This will be the main part of our analysis. Afterwards, we analyze sums of rectangular functions and, finally, we show that quasiconcave functions can be approximated by such sums.

Lemma 3.4. *For $i \in [n]$ let $\phi_i \geq 0$, let $I_i \subseteq \mathbb{R}$ be an interval of length ℓ_i, and let $f_i \colon \mathbb{R} \to \mathbb{R}$ be the function*

$$f_i(x) = \begin{cases} \phi_i & \text{if } x \in I_i, \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, let $f \colon \mathbb{R}^n \to \mathbb{R}$ be the function $f(x_1, \dots, x_n) = \prod_{i=1}^n f_i(x_i)$ and let $A \in \mathbb{R}^{n \times n}$ be an invertible matrix. Then

$$\int_{y \in \mathbb{R}^{n-k}} \max_{z \in \mathbb{R}^k} f\big(A^{-1} \cdot (y, z)\big) \mathrm{d}y \leq 2^k \chi \cdot \sum_I \left(|\det(A_I)| \cdot \prod_{i \notin I} \ell_i \right)$$

where $\chi = \prod_{i=1}^n \phi_i$ and where the sum runs over all tuples $I = (i_1, \dots, i_k)$ for which $1 \le i_1 < \dots < i_k \le n$. For such a tuple I, matrix A_I is the $(n-k) \times (n-k)$-submatrix of A that is obtained from A by removing the last k rows and the columns with the numbers $i \in I$.

Proof. Function f takes the value χ on the n-dimensional box $Q = \prod_{i=1}^n I_i$ and is zero otherwise. Hence,

$$\int_{y \in \mathbb{R}^{n-k}} \max_{z \in \mathbb{R}^k} f\big(A^{-1} \cdot (y,z)\big) \mathrm{d}y = \chi \cdot \mathrm{vol}(Q')$$

for

$$\begin{aligned} Q' &= \big\{y \in \mathbb{R}^{n-k} : \exists z \in \mathbb{R}^k \text{ such that } A^{-1} \cdot (y,z) \in Q\big\} \\ &= \big\{y \in \mathbb{R}^{n-k} : \exists z \in \mathbb{R}^k \exists x \in Q \text{ such that } (y,z) = A \cdot x\big\} \\ &= (P \cdot A)(Q)\,, \end{aligned}$$

where $P := \big[\mathbb{I}_{n-k}, \mathbb{O}_{(n-k)\times k}\big]$ is the projection matrix that removes the last k entries from a vector of length n. In the remainder of this proof we bound the volume of $M(Q)$ where $M := P \cdot A \in \mathbb{R}^{(n-k)\times n}$. Let $a_i =: c_i^0$ and $b_i =: c_i^1$ be the left and the right bound of interval I_i, respectively. For an index tuple $I = (i_1, \dots, i_k)$, $1 \le i_1 < \dots < i_k \le n$, and a bit tuple $J = (j_1, \dots, j_k) \in \{0,1\}^k$, let

$$F_I^J = \prod_{i=1}^n \begin{cases} \{c_{i_t}^{j_t}\} & \text{if } i = i_t \in I\,, \\ I_i & \text{if } i \notin I\,, \end{cases}$$

be one of the $2^k \cdot \binom{n}{k}$ $(n-k)$-dimensional faces of Q. We show that $M(Q) \subseteq \bigcup_I \bigcup_J M\big(F_I^J\big)$. Let $y \in M(Q)$, i.e., there is a vector $x \in Q$ for which $y = M \cdot x$. Now, consider the polytope

$$R = \{(x', s') \in \mathbb{R}^n \times \mathbb{R}^n : M \cdot x' = y' \text{ and } x' + s' = b' \text{ and } x', s' \ge 0\}\,,$$

where $y' = y - M \cdot a$ and $b' = b - a$ for $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$. This polytope is bounded and non-empty because $(x - a, b - x) \in R$. Consequently, there exists a basic feasible solution $(x^\star, s^\star)$. As there are $2n$ variables and $2n - k$ constraints, this solution has at least k zero-entries, i.e., there are indices $1 \le i_1 < \dots < i_k \le n$ for which either $x_{i_t}^\star = 0$ (in that case we set $j_t := 0$) or $x_{i_t}^\star = b'_{i_t}$ (in that case we set $j_t := 1$) for any $t \in [k]$. Now, consider the vector $\hat{x} = x^\star + a \in [0, b'] + a = Q$. We obtain $M \cdot \hat{x} = y$ and $\hat{x}_{i_t} = c_{i_t}^{j_t}$ for all $t \in [k]$. Hence, $x \in F_I^J$ for $I = (i_1, \dots, i_k)$ and $J = (j_1, \dots, j_k)$, and thus $y \in M\big(F_I^J\big)$.

Due to this observation we can bound the volume of $M(Q)$ by $\sum_I \sum_J \mathrm{vol}\big(M\big(F_I^J\big)\big)$. It remains to bound the volume $\mathrm{vol}\big(M\big(F_I^J\big)\big)$. For this, we enumerate the indices of $[n] \setminus I$ by $1 \le \hat{i}_1 < \dots < \hat{i}_{n-k} \le n$. Now, we define vectors $t_i \in \mathbb{R}^{n-k}$ and a vector $v \in \mathbb{R}^n$ as follows.

$$t_i = \begin{cases} 0 & \text{if } i \in I\,, \\ e_t & \text{if } i = \hat{i}_t\,, \end{cases} \quad \text{and} \quad v_i = \begin{cases} 0 & \text{if } i \notin I\,, \\ c_{i_t}^{j_t} & \text{if } i = i_t\,. \end{cases}$$

Let $T = [t_1, \dots, t_n]^{\mathrm{T}} = [e_{\hat{i}_1}, \dots, e_{\hat{i}_{n-k}}] \in \mathbb{R}^{n \times (n-k)}$, $F' = \prod_{t=1}^{n-k} I_{\hat{i}_t} \subseteq \mathbb{R}^{n-k}$, and consider the linear function $\phi \colon F' \to F_I^J$, defined as $\phi(x) = Tx + v$. Function ϕ is the canonical bijection from F' to the face F_I^J: If $y = \phi(x)$, then $y_{\hat{i}_t} = e_t^{\mathrm{T}} x + 0 = x_t$ for $t \in [n-k]$ and $y_{i_t} = 0^{\mathrm{T}} x + c_{i_t}^{j_t} = c_{i_t}^{j_t}$ for $t \in [k]$. Now consider the function $\psi = M \circ \phi \colon F' \to M(F_I^J)$. Note, that both F' and $M(F_I^J)$ are subsets of $\mathbb{R}^{n-k}$ and that $\psi(x) = MTx + Mv = A_I x + Mv$ for the $(n-k) \times (n-k)$-submatrix $A_I := MT = PAT$ of A containing the rows of A with the numbers $1, \dots, n-k$ and the columns of A with the numbers $\hat{i}_1, \dots, \hat{i}_{n-k}$. If ψ is not injective, then $\psi(F') = M(F_I^J)$ is not full-dimensional, i.e., $\mathrm{vol}\big(M\big(F_I^J\big)\big) = 0$. Otherwise, we obtain

$$\begin{aligned}
\mathrm{vol}\big(M\big(F_I^J\big)\big) &= \int_{\psi(F')} 1 \,\mathrm{d}x = \int_{F'} |\det \mathrm{D}\psi(x)| \,\mathrm{d}x = \int_{F'} |\det(A_I)| \,\mathrm{d}x \\
&= |\det(A_I)| \cdot \mathrm{vol}(F') = |\det(A_I)| \cdot \prod_{t=1}^{n-k} |I_{\hat{i}_t}| = |\det(A_I)| \cdot \prod_{t=1}^{n-k} \ell_{\hat{i}_t} \,.
\end{aligned}$$

This yields

$$\begin{aligned}
\int_{y \in \mathbb{R}^{n-k}} \max_{z \in \mathbb{R}^k} f\big(A^{-1} \cdot (y,z)\big) \mathrm{d}y = \chi \cdot \mathrm{vol}(M(Q)) &\le \chi \cdot \sum_I \sum_J \mathrm{vol}\big(M\big(F_I^J\big)\big) \\
&\le \chi \cdot \sum_I \sum_J \left(|\det(A_I)| \cdot \prod_{t=1}^{n-k} \ell_{\hat{i}_t} \right) \\
&= 2^k \chi \cdot \sum_I \left(|\det(A_I)| \cdot \prod_{i \notin I} \ell_i \right) \,. \qquad \square
\end{aligned}$$

In the next step we generalize the statement of Lemma 3.4 to sums of rectangular functions.

Corollary 3.5. *Let $N_1, \dots, N_n$ be positive integers, let $\phi_{i,k} \ge 0$ be a non-negative real, let $I_{i,k} \subseteq \mathbb{R}$ be an interval of length $\ell_{i,k}$, and let $f_{i,k} \colon \mathbb{R} \to \mathbb{R}$ be the function*

$$f_{i,k}(x) = \begin{cases} \phi_{i,k} & \text{if } x \in I_{i,k}\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

$i \in [n]$, $k \in [N_i]$. Furthermore, let $f_i \colon \mathbb{R} \to \mathbb{R}$ be the step function $f_i = \sum_{k=1}^{N_i} f_{i,k}$, let $f \colon \mathbb{R}^n \to \mathbb{R}$ be the function $f(x_1, \dots, x_n) = \prod_{i=1}^n f_i(x_i)$, and let $A \in \mathbb{R}^{n \times n}$ be an invertible matrix. Then

$$\int_{y \in \mathbb{R}^{n-k}} \max_{z \in \mathbb{R}^k} f\big(A^{-1} \cdot (y,z)\big) \mathrm{d}y \le 2^k \cdot \sum_I \left(|\det(A_I)| \cdot \left(\prod_{i \notin I} \sigma_i \right) \cdot \left(\prod_{i \in I} \chi_i \right) \right) ,$$

where $\sigma_i = \sum_{k=1}^{N_i} \phi_{i,k} \cdot \ell_{i,k}$ *and* $\chi_i = \sum_{k=1}^{N_i} \phi_{i,k}$ *and where the sum runs over all tuples* $I = (i_1, \ldots, i_k)$ *for which* $1 \le i_1 < \ldots < i_k \le n$*. For such a tuple* I*, matrix* A_I *is the* $(n-k) \times (n-k)$*-submatrix of* A *that is obtained from* A *by removing the last* k *rows and the columns with the numbers* $i \in I$*.*

Proof. For indices $k_i \in [N_i]$ let $f_{k_1,\ldots,k_n}(x_1, \ldots, x_n) = \prod_{i=1}^n f_{i,k_i}(x_i)$. This function is of the form assumed in Lemma 3.4 and takes only values zero and $\chi_{k_1,\ldots,k_n} = \prod_{i=1}^n \phi_{i,k_i}$. We can write function f as

$$\begin{aligned} f(x_1, \ldots, x_n) &= \prod_{i=1}^n f_i(x_i) = \prod_{i=1}^n \sum_{k_i=1}^{N_i} f_{i,k_i}(x_i) = \sum_{k_1=1}^{N_1} \cdots \sum_{k_n=1}^{N_n} \prod_{i=1}^n f_{i,k_i}(x_i) \\ &= \sum_{k_1=1}^{N_1} \cdots \sum_{k_n=1}^{N_n} f_{k_1,\ldots,k_n}(x_1, \ldots, x_n) , \end{aligned}$$

i.e., as a sum of rectangular functions. For the sake of simplicity we write $\sum_{k_i}$ instead of $\sum_{k_i=1}^{N_i}$ and $\sum_{k_i \colon i \in (i_1,\ldots,i_\ell)}$ instead of $\sum_{k_{i_1}} \ldots \sum_{k_{i_\ell}}$. We can bound the integral as follows:

$$\begin{aligned} \int_{y \in \mathbb{R}^{n-k}} \max_{z \in \mathbb{R}^k} f\big(A^{-1} \cdot (y,z)\big) \mathrm{d}y &= \int_{y \in \mathbb{R}^{n-k}} \max_{z \in \mathbb{R}^k} \sum_{k_i \colon i \in [n]} f_{k_1,\ldots,k_n}\big(A^{-1} \cdot (y,z)\big) \mathrm{d}y \\ &\le \int_{y \in \mathbb{R}^{n-k}} \sum_{k_i \colon i \in [n]} \max_{z \in \mathbb{R}^k} f_{k_1,\ldots,k_n}\big(A^{-1} \cdot (y,z)\big) \mathrm{d}y \\ &= \sum_{k_i \colon i \in [n]} \int_{y \in \mathbb{R}^{n-k}} \max_{z \in \mathbb{R}^k} f_{k_1,\ldots,k_n}\big(A^{-1} \cdot (y,z)\big) \mathrm{d}y \\ &\le \sum_{k_i \colon i \in [n]} \left(2^k \chi_{k_1,\ldots,k_n} \cdot \sum_I \left(|\det(A_I)| \cdot \prod_{i \notin I} \ell_{i,k_i} \right) \right) \\ &= 2^k \cdot \sum_{k_i \colon i \in [n]} \left(\prod_{i \in [n]} \phi_{i,k_i} \cdot \sum_I \left(|\det(A_I)| \cdot \prod_{i \notin I} \ell_{i,k_i} \right) \right) \\ &= 2^k \cdot \sum_I \left(|\det(A_I)| \cdot \sum_{k_i \colon i \in [n]} \left(\prod_{i \in [n]} \phi_{i,k_i} \cdot \prod_{i \notin I} \ell_{i,k_i} \right) \right) , \end{aligned}$$

where the second inequality is due to Lemma 3.4. Now

$$\sum_{k_i \colon i \in [n]} \left(\prod_{i \in [n]} \phi_{i,k_i} \cdot \prod_{i \notin I} \ell_{i,k_i} \right) = \sum_{k_i \colon i \in [n]} \left(\prod_{i \in I} \phi_{i,k_i} \cdot \prod_{i \notin I} (\phi_{i,k_i} \cdot \ell_{i,k_i}) \right)$$

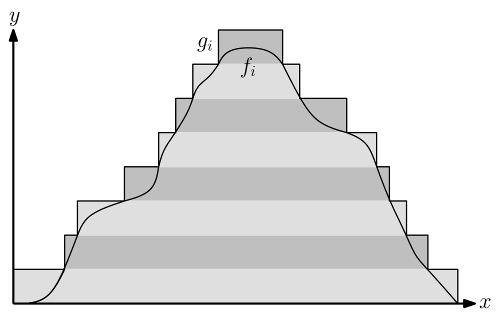


Figure 3.1: Area of a quasi-concave function covered by a "stack" of rectangles with approximately the same area

$$= \left(\sum_{k_i\,:\,i\in I}\prod_{i\in I}\phi_{i,k_i}\right)\cdot\left(\sum_{k_i\,:\,i\notin I}\prod_{i\notin I}(\phi_{i,k_i}\cdot\ell_{i,k_i})\right)$$
$$= \left(\prod_{i\in I}\sum_{k_i}\phi_{i,k_i}\right)\cdot\left(\prod_{i\notin I}\sum_{k_i}(\phi_{i,k_i}\cdot\ell_{i,k_i})\right)$$
$$= \left(\prod_{i\in I}\chi_i\right)\cdot\left(\prod_{i\notin I}\sigma_i\right),$$

which completes the proof of Corollary 3.5. □

To finish the proof of Theorem 3.3 we round the probability densities f_i as follows: For an arbitrarily small positive real δ let $g_i := \lceil f_i/\delta\rceil\cdot\delta$, i.e., we round f_i up to the next integral multiple of δ. As the densities f_i are quasiconcave, there is a decomposition of g_i such that $g_i = \sum_{k=1}^{N_i} f_{i,k}$ where

$$f_{i,k} = \begin{cases}\phi_{i,k} & : \ x\in I_{i,k}\,,\\ 0 & : \ \text{otherwise}\,,\end{cases}\qquad\text{and}\qquad \chi_i := \sum_{k=1}^{N_i}\phi_{i,k} = \max_{x\in[-1,1]} g_i(x)\,,$$

where $I_{i,k}$ are intervals of length $\ell_{i,k}$ and $\phi_{i,k}$ are positive reals. The second property is the interesting one and stems from the quasiconcaveness of f_i. Informally speaking the two-dimensional shape bounded by the horizontal axis and the graph of g_i is a stack of rectangles aligned with axes (see Figure 3.1). Therefore, the sum χ_i of the rectangles' heights which appears in the formula of Corollary 3.5 is approximately ϕ. Without the quasiconcaveness χ_i might be unbounded.

Applying Corollary 3.5, we obtain

$$\int_{y\in\mathbb{R}^{n-k}}\max_{z\in\mathbb{R}^k} f_X\big(A^{-1}\cdot(y,z)\big)\mathrm{d}y = \int_{y\in\mathbb{R}^{n-k}}\max_{z\in\mathbb{R}^k}\prod_{i=1}^{n} f_i\Big(\big(A^{-1}\cdot(y,z)\big)_i\Big)\mathrm{d}y$$

$$\begin{aligned}
&\leq \int_{y\in\mathbb{R}^{n-k}} \max_{z\in\mathbb{R}^k} \prod_{i=1}^{n} g_i\bigl(\bigl(A^{-1}\cdot(y,z)\bigr)_i\bigr)\,\mathrm{d}y \\
&\leq 2^k \cdot \sum_I |\det(A_I)| \cdot \left(\prod_{i\notin I}\sum_{k_i=1}^{N_i}(\phi_{i,k_i}\cdot \ell_{i,k_i})\right)\cdot\left(\prod_{i\in I}\chi_i\right) \\
&= 2^k \cdot \sum_I |\det(A_I)| \cdot \left(\prod_{i\notin I}\int_{[-1,1]} g_i\,\mathrm{d}x\right)\cdot\left(\prod_{i\in I}\chi_i\right) .
\end{aligned}$$

Since $0 \leq \int_{[-1,1]} g_i\,\mathrm{d}x \leq \int_{[-1,1]}(f_i+\delta)\,\mathrm{d}x = 1+2\delta$ and $0 \leq \chi_i \leq \sup_{x\in[-1,1]} f_i(x)+\delta \leq \phi+\delta$, this implies

$$\begin{aligned}
\int_{y\in\mathbb{R}^{n-k}} \max_{z\in\mathbb{R}^k} f_X\bigl(A^{-1}\cdot(y,z)\bigr)\mathrm{d}y &\leq 2^k\cdot\sum_I|\det(A_I)|\cdot\left(\prod_{i\notin I}(1+2\delta)\right)\cdot\left(\prod_{i\in I}(\phi+\delta)\right) \\
&= 2^k\cdot\sum_I|\det(A_I)|\cdot(1+2\delta)^{n-k}\cdot(\phi+\delta)^k .
\end{aligned}$$

As this bound is true for arbitrarily small reals $\delta > 0$, it also holds for $\delta = 0$ and we obtain

$$\begin{aligned}
\mathbf{Pr}\left[(Z_1,\dots,Z_k)\in C(Y_1,\dots,Y_{n-k})\right] &\leq |\det(A^{-1})|\cdot\varepsilon^k\cdot\int_{y\in\mathbb{R}^{n-k}}\max_{z\in\mathbb{R}^k} f_X\bigl(A^{-1}\cdot(y,z)\bigr)\mathrm{d}y \\
&\leq 2^k\cdot\sum_I\frac{|\det(A_I)|}{|\det(A)|}\cdot\phi^k\varepsilon^k . \qquad\square
\end{aligned}$$

Chapter 4

The Successive Shortest Path Algorithm

This chapter is devoted to the study of the *successive shortest path (SSP) algorithm* in the framework of smoothed analysis. After introducing some notation that we will use in the remainder of this chapter we give an outline of our analysis of the SSP algorithm. There we will also discuss a connection to the analysis of the worst-case bound in the integer case. The main technical section of this chapter is devoted to the proof of Theorem 1.2.2 (Section 4.3).

4.1 Terminology and Notation

Consider the run of the SSP algorithm on the *flow network* G. We denote the set $\{f_0, f_1, \ldots\}$ of all *flows* encountered by the SSP algorithm by $\mathcal{F}_0(G)$. Furthermore, we set $\mathcal{F}(G) = \mathcal{F}_0(G) \setminus \{f_0\}$. (We omit the parameter G if it is clear from the context.)

By f_0 and $f_{\max}$, we denote the *empty flow* and the *maximum flow*, i.e., the flow that assigns 0 to all edges e and the flow of maximum value encountered by the SSP algorithm, respectively.

Let f_{i-1} and f_i be two consecutive flows encountered by the SSP algorithm and let P_i be the shortest path in the *residual network* $G_{f_{i-1}}$, i.e., the SSP algorithm augments along P_i to increase flow f_{i-1} to obtain flow f_i. We call P_i the *next path* of f_{i-1} and the *previous path* of f_i. To distinguish between the original network G and some residual network G_f in the remainder of this chapter, we refer to the edges in the residual network as *arcs*, whereas we refer to the edges in the original network as *edges*.

For a given arc e in a residual network G_f, we denote by e_0 the corresponding edge in the original network G, i.e., $e_0 = e$ if $e \in E$ (i.e., e is a *forward arc*) and $e_0 = e^{-1}$ if $e \notin E$ (i.e., e is a *backward arc*). An arc e is called *empty* (with respect to some residual network G_f) if e belongs to G_f, but e^{-1} does not. Empty arcs e are either forward arcs that do not carry flow or backward arcs whose corresponding edge e_0 carries as much flow as possible. We say that an arc *becomes saturated* (during an augmentation) when it is contained in the current augmenting path, but it does not belong to the residual network that we obtain after this augmentation.

In the remainder, a *path* is always a simple directed path. Let P be a path, and let u and v be contained in P in this order. With $u \overset{P}{\rightsquigarrow} v$, we refer to the subpath of P starting from node u going to node v. We call any flow network G' a *possible residual network* (of G) if there is a flow f for G such that $G' = G_f$. Paths in possible residual networks are called *possible paths*.

4.2 Outline of Our Approach

We can concentrate on counting the number of augmenting steps of the SSP algorithm since each step can be implemented to run in time $O(m + n \log n)$ using Dijkstra's algorithm (see, e.g., the book of Korte and Vygen [KV07] for details). To establish a connection between our smoothed analysis and the worst-case analysis of the SSP algorithm on instances with integral edge costs, let us first consider the case that all edge costs are from $\{1, \dots, C\}$. In this case the length of any possible path is bounded by nC. We will see that the lengths of the augmenting paths are monotonically increasing. If there is no unique shortest path to augment flow along and ties are broken by choosing one with the fewest number of arcs, then the number of successive augmenting paths with the same length is bounded by $O(mn)$. Hence, the SSP algorithm terminates within $O(mn^2C)$ steps.

Now let us perturb the edge costs of such an integral instance independently by, for example, uniform additive noise from the interval $[-1, 1]$. This scenario is not covered by bounds for the integral case. Indeed, instances can be generated for which the number of augmentation steps is exponential in m and n. Nevertheless, an immediate consequence of Theorem 1.2.2 is that, in expectation, the SSP algorithm terminates within $O(mnC)$ steps on instances of this form. To see this, scale all edge costs by a factor of $1/(C+1)$. Each of these random variables is now drawn uniformly at random from an interval of length $2/(C+1)$ which is a subset of $[0, 1]$. Consequently, their probability densities are of the form assumed by Theorem 1.2.2 and are bounded by $(C+1)/2 = O(C)$.

Our analysis of the SSP algorithm is based on the following idea: We identify a flow $f_i \in \mathcal{F}_0$ with a real number by mapping f_i to the length ℓ_i of the previous path P_i of f_i. The flow f_0 is identified with $\ell_0 = 0$. In this way, we obtain a sequence $L = (\ell_0, \ell_1, \dots)$ of real numbers. We show that this sequence is strictly monotonically increasing with a probability of 1. Since all costs are drawn from the interval $[0, 1]$, each element of L is from the interval $[0, n]$. To count the number of elements of L, we use *discretization*: We partition the interval $[0, n]$ into small subintervals of length ε and sum up the number of elements of L in these intervals. By linearity of expectation, this approach carries over to the expected number of elements of L. If ε is very small, then – with sufficiently high probability – each interval contains at most one element. Thus, it suffices to bound the probability that an element of L falls into some interval $(d, d+\varepsilon]$.

For this, assume that there is an integer i such that $\ell_i \in (d, d+\varepsilon]$. By the previous assumption that for any interval of length ε there is at most one path whose length is within this interval, we obtain that $\ell_{i-1} \leq d$. We show that the augmenting path P_i uses an empty arc e. Moreover, we will see that we can reconstruct flow f_{i-1} without knowing the cost of edge e_0 that corresponds to arc e in the original network. Hence, we do not have to reveal c_{e_0} for this. However, the length of P_i, which equals ℓ_i, depends linearly on c_{e_0}, and the coefficient is $+1$ or -1. Consequently, the probability that ℓ_i falls into the interval $(d, d+\varepsilon]$ is bounded by $\varepsilon\phi$, as the probability density of c_{e_0} is bounded by ϕ. Since the arc e is not always the same, we have to apply a *union bound* over all $2m$ possible arcs. Summing up over all n/ε intervals the expected number of flows encountered by the SSP algorithm can be bounded by roughly $(n/\varepsilon) \cdot 2m \cdot \varepsilon\phi = 2mn\phi$.

There are some parallels to the analysis of the smoothed number of Pareto-optimal solutions in bicriteria linear optimization problems by Beier and Vöcking [BV06], although we have only one objective function. In this context, we would call f_i the loser, f_{i-1} the winner, and the difference $\ell_i - d$ the loser gap. Beier and Vöcking's analysis is also based on the observation that the winner (which in their analysis is a Pareto-optimal solution and not a flow) can be reconstructed when all except for one random coefficients are revealed. While this reconstruction is simple in the setting of bicriteria optimization problems, the reconstruction of the flow f_{i-1} in our setting is significantly more challenging and a main difficulty in our analysis.

4.3 Analysis of the SSP Algorithm

Before we start with the analysis, note that due to our transformation of the general minimum-cost flow problem to a single-source-single-sink minimum-cost flow problem the cost perturbations only affect the original edges. The costs of the auxiliary edges are not perturbed but set to 0. Thus, we will slightly deviate from what we described in the outline by treating empty arcs corresponding to auxiliary edges separately.

Lemma 4.3.1. *For any real $\varepsilon > 0$ the probability that there are two nodes u and v and two distinct possible u-v-paths whose lengths differ by at most ε is bounded from above by $2n^{2n}\varepsilon\phi$.*

Proof. Fix two nodes u and v and two distinct possible u-v-paths P_1 and P_2. If we consider the paths P_1 and P_2 as undirected paths by interpreting a directed edge (x, y) as an undirected edge $\{x, y\}$, then the symmetric difference $P_1 \,\Delta\, P_2 = (V(P_1) \cup V(P_2), E(P_1) \,\Delta\, E(P_2))$ of both paths does not contain a node of degree 1. On the other hand, not all nodes can be isolated nodes because $P_1 \neq P_2$. Consequently, the graph $P_1 \,\Delta\, P_2$ contains a cycle. This cycle must contain an edge that does not correspond to an auxiliary edge of the original graph G. Moreover, it corresponds to an edge e such that one of the paths – without loss of generality path P_1 – contains arc e or e^{-1}, but the other one contains neither of them. If we fix all edge costs except the cost of edge e, then the length of P_2 is already determined whereas the length of P_1 depends on the cost c_e. Hence, c_e must fall into a fixed interval of length 2ε in order for the path lengths of P_1 and P_2 to differ by at most ε. The probability for this is bounded by $2\varepsilon\phi$ because c_e is chosen according to a density function that is bounded from above by ϕ. A *union bound* over all pairs (u, v) and all possible u-v-paths concludes the proof. □

According to Lemma 4.3.1 we can assume that there is no s-t-path of length 0 and that the following property holds since it holds with a probability of 1.

Property 1. For any nodes u and v the lengths of all possible u-v-paths are pairwise distinct.

Lemma 4.3.2. *Let $d_i(v)$ denote the distance from s to node v in the residual network G_{f_i}. Then the sequence $\big(d_i(v)\big)_{i\geq 0}$ is monotonically increasing.*

Proof. Let $i \geq 0$ be an arbitrary integer. We show $d_i(v) \leq d_{i+1}(v)$ by induction on the depth of node v in the shortest path tree T_{i+1} of the residual network $G_{f_{i+1}}$ rooted at s. For the root s, the claim holds since $d_i(s) = d_{i+1}(s) = 0$. Now assume that the claim holds for all nodes up to a certain depth k, consider a node v with depth $k+1$, and let u denote its parent. Consequently, $d_{i+1}(v) = d_{i+1}(u) + c_e$ for $e = (u, v)$. If arc e has been available in G_{f_i}, then $d_i(v) \leq d_i(u) + c_e$. If not, then the SSP algorithm must have augmented along e^{-1} in step $i+1$ to obtain flow f_{i+1} and, hence, $d_i(u) = d_i(v) + c_{e^{-1}} = d_i(v) - c_e$. In both cases the inequality $d_i(v) \leq d_i(u) + c_e$ holds. Applying the induction hypothesis for node u, we obtain $d_i(v) \leq d_i(u) + c_e \leq d_{i+1}(u) + c_e = d_{i+1}(v)$. □

Definition 4.3.3. For a flow $f_i \in \mathcal{F}_0$, we denote by $\ell_-^G(f_i)$ and $\ell_+^G(f_i)$ the length of the previous path P_i and the next path P_{i+1} of f_i, respectively. By convention, we set $\ell_-^G(f_0) = 0$ and $\ell_+^G(f_{\max}) = \infty$. If the network G is clear from the context, then we simply write $\ell_-(f_i)$ and $\ell_+(f_i)$. By $\mathscr{C}$ we denote the cost function that maps reals x from the interval $\big[0, |f_{\max}|\big]$ to the cost of the cheapest flow f with value x, that is, $\mathscr{C}(x) = \min\{c(f) : |f| = x\}$.

The lengths $\ell_-(f_i)$ correspond to the lengths ℓ_i mentioned in the outline. The apparent notational overhead is necessary for formal correctness. In Lemma 4.3.5, we will reveal a connection between the values $\ell_-(f_i)$ and the function $\mathscr{C}$. Based on this, we can focus on analyzing function $\mathscr{C}$.

Corollary 4.3.4. *Let $f_i, f_j \in \mathcal{F}_0$ be two flows with $i < j$. Then $\ell_-(f_i) \leq \ell_-(f_j)$.*

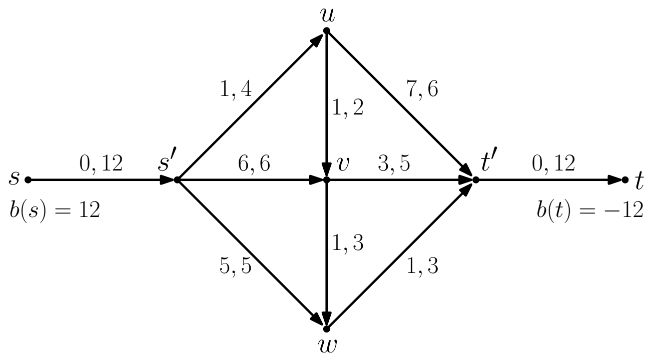
Proof. The claim follows from Lemma 4.3.2, the fact that the SSP algorithm does not augment along the same path in two consecutive steps, and the fact that different paths have different lengths (see Property 1). □

Lemma 4.3.5. *The function $\mathscr{C}$ is continuous, monotonically increasing, and piecewise linear. The break points of $\mathscr{C}$ are the values of the flows $f \in \mathcal{F}_0$ with $\ell_-(f) < \ell_+(f)$. For each flow $f \in \mathcal{F}_0$, the slopes of $\mathscr{C}$ to the left and to the right of $|f|$ equal $\ell_-(f)$ and $\ell_+(f)$, respectively.*

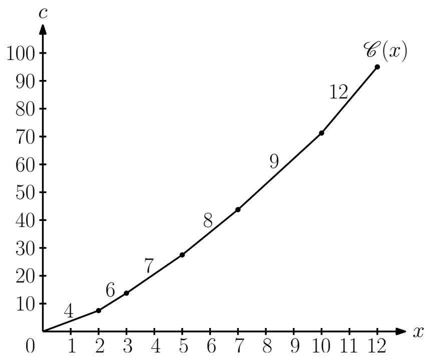
Proof. The proof follows from Theorem 1.2.1 and the observation that the cost of the flow is linearly increasing when gradually increasing the flow along the shortest path in the residual network until at least one arc becomes saturated. The slope of the cost function is given by the length ℓ of that path: As long as no arc becomes saturated, the cost of the flow increases by $\Delta_f \cdot \ell$ if the flow increases by Δ_f. □

Example 4.3.6. Consider the flow network depicted in Figure 4.1a. The cost c_e and the capacity u_e of an edge e are given by the notation c_e, u_e. Observe, that the costs c_e are not scaled down to the interval $[0, 1]$ to make the example easier to read.

For each step of the SSP algorithm, Table 4.1 lists the relevant part of the augmenting path (excluding s, s', t', and t), its length, the amount of flow that is sent along that path, the value and the cost of the current flow, and the arcs that become saturated. As can be seen in the table, the values $|f|$ of the encountered flows $f \in \mathcal{F}_0$ are 0, 2, 3, 5, 7, 10, and 12. These are the breakpoints of the cost function $\mathscr{C}$, and the lengths of the augmenting paths equal the slopes of $\mathscr{C}$ (see Figure 4.1b).



(a) Flow network G



(b) Cost function $\mathscr{C}$

Figure 4.1: A minimum-cost flow network and the corresponding cost function $\mathscr{C}$

With the following definition, we lay the foundation for distinguishing between original edges with perturbed costs and auxiliary edges whose costs are set to 0.

step	1	2	3	4	5	6
path	u, v, w	w	w, v	u	v	v, u
path length	4	6	7	8	9	12
amount of flow	2	1	2	2	3	2
current flow value	2	3	5	7	10	12
current flow cost	8	14	28	44	71	95
saturated arcs	(u, v)	(w, t')	(w, v)	(s', u)	(v, t')	(v, u)

Table 4.1: The augmenting paths for network G

Definition 4.3.7. Let $f \in \mathcal{F}_0$ be an arbitrary flow. An empty arc e in the residual network G_f that does not correspond to an auxiliary edge is called a *good arc.* We call f a *good flow* if $f \neq f_0$ and if the previous path of f contains a good arc in the previous residual network. Otherwise, f is called a *bad flow.*

Now we derive a property of good arcs that are contained in the previous path of good flows. This property allows us to bound the probability that one of the lengths $\ell_-(f_i)$ falls into a given interval of length ε.

Lemma 4.3.8. *Let $f \in \mathcal{F}_0$ be a predecessor of a good flow for which $\ell_-^G(f) < \ell_+^G(f)$ holds, and let $d \in \big[\ell_-^G(f), \ell_+^G(f)\big)$ be an arbitrary real number. Additionally, let e be a good arc in the next path of f, and let e_0 be the edge in G that corresponds to e. Now change the cost of e_0 to $c'_{e_0} = 1$ ($c'_{e_0} = 0$) if $e_0 = e$ ($e_0 = e^{-1}$), i.e., when e is a forward (backward) arc. In any case, the cost of arc e increases. We denote the resulting flow network by G'. Then $f \in \mathcal{F}_0(G')$. Moreover, the inequalities $\ell_-^{G'}(f) \leq \ell_-^G(f) \leq d < \ell_+^G(f) \leq \ell_+^{G'}(f)$ hold.*

Proof. Let $\mathscr{C}$ and $\mathscr{C}'$ be the cost functions of the original network G and the modified network G', respectively. Both functions are of the form described in Lemma 4.3.5. In particular, they are continuous and the breakpoints correspond to the values of the flows $\tilde{f} \in \mathcal{F}_0(G)$ and $\hat{f} \in \mathcal{F}_0(G')$ with $\ell_-^G(\tilde{f}) < \ell_+^G(\tilde{f})$ and $\ell_-^{G'}(\hat{f}) < \ell_+^{G'}(\hat{f})$, respectively.

We start with analyzing the case $e_0 = e$. In this case, we set $\mathscr{C}'' = \mathscr{C}'$ and observe that by increasing the cost of edge e_0 to 1 the cost of no flow can decrease. Hence, $\mathscr{C}'' \geq \mathscr{C}$. Since flow f does not use arc e, its costs remain unchanged, i.e., $\mathscr{C}''(|f|) = \mathscr{C}(|f|)$.

If $e_0 = e^{-1}$, then we set $\mathscr{C}'' = \mathscr{C}' + \Delta_{e_0}$ for $\Delta_{e_0} = u_{e_0} \cdot c_{e_0}$. This function is also piecewise linear and has the same breakpoints and slopes as $\mathscr{C}'$. Since the flow on edge e_0 cannot exceed the capacity u_{e_0} of edge e_0 and since the cost on that edge has been reduced by c_{e_0} in G', the cost of each flow is reduced by at most Δ_{e_0} in G'. Furthermore, this gain is only achieved for flows that entirely use edge e_0 like f does. Hence, $\mathscr{C}'' \geq \mathscr{C}$ and $\mathscr{C}''(|f|) = \mathscr{C}(|f|)$.

Due to $\mathscr{C}'' \geq \mathscr{C}$, $\mathscr{C}''(|f|) = \mathscr{C}(|f|)$, and the form of both functions, the left-hand derivative of $\mathscr{C}''$ at $|f|$ is at most the left-hand derivative of $\mathscr{C}$ at $|f|$ (see Figure 4.2).

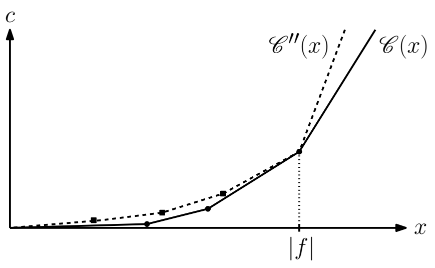


Figure 4.2: Cost function $\mathscr{C}$ and function $\mathscr{C}''$.

Since $|f|$ is a breakpoint of $\mathscr{C}$, this implies that $|f|$ is also a breakpoint of $\mathscr{C}''$ and that the slope to the left of $\mathscr{C}''$ at $|f|$ is at most the slope to the left of $\mathscr{C}$ at $|f|$. For the same reasons, the right-hand derivative of $\mathscr{C}''$ at $|f|$ is at least the right-hand derivative of $\mathscr{C}$ at $|f|$ and the slope to the right of $\mathscr{C}''$ at $|f|$ is at least the slope to the right of $\mathscr{C}$ at $|f|$. These properties carry over to $\mathscr{C}'$. Hence, $f \in \mathcal{F}_0(G')$. Recalling $d \in \big[\ell_-^G(f), \ell_+^G(f)\big)$ and the fact that the slopes correspond to shortest s-t-path lengths, the stated chain of inequalities follows. □

Lemma 4.3.8 suggests Algorithm 3 (Reconstruct) for reconstructing a flow f based on a good arc e that belongs to the shortest path in the residual network G_f and on a threshold $d \in \big[\ell_-(f), \ell_+(f)\big)$. The crucial fact that we will later exploit is that for this reconstruction the cost c_{e_0} of edge e_0 does not have to be known. (Note that we only need Reconstruct for the analysis in order to show that the flow f can be reconstructed.)

Algorithm 3 Reconstruct(e, d)

```
1: let $e_0$ be the edge that corresponds to arc $e$ in the original network $G$
2: change the cost of edge $e_0$ to $c'_{e_0} = 1$ if $e$ is a forward arc or to $c'_{e_0} = 0$ if $e$ is a
   backward arc
3: start running the SSP algorithm on the modified network $G'$
4: stop when the length of the shortest $s$-$t$-path in the residual network of the current
   flow $f'$ exceeds $d$
5: output $f'$
```

Corollary 4.3.9. *Let $f \in \mathcal{F}_0$ be a predecessor of a good flow, let e be a good arc in the next path of f, and let $d \in \big[\ell_-(f), \ell_+(f)\big)$ be a real. Then* Reconstruct(e, d) *outputs flow f.*

Proof. By applying Lemma 4.3.8, we obtain $f \in \mathcal{F}_0(G')$ and $\ell_-^{G'}(f) \leq d < \ell_+^{G'}(f)$. Together with Corollary 4.3.4, this implies that Reconstruct(e, d) does not stop before encountering flow f and stops once it encounters f. Hence, Reconstruct(e, d) outputs flow f. □

Corollary 4.3.9 is an essential component of the proof of Theorem 1.2.2 but it only describes how to reconstruct predecessor flows f of good flows with $\ell_-(f) < \ell_+(f)$. In the next part of this section we show that most of the flows are good flows.

Lemma 4.3.10. *In any step of the SSP algorithm, any s-t-path in the residual network contains at least one empty arc.*

Proof. The claim is true for the empty flow f_0. Now consider a flow $f_i \in \mathcal{F}$, its predecessor flow f_{i-1}, the path P_i which is a shortest path in the residual network $G_{f_{i-1}}$, and an arbitrary s-t-path P in the current residual network G_{f_i}. We show that at least one arc in P is empty.

For this, fix one arc $e = (x, y)$ from P_i that is not contained in the current residual network G_{f_i} since it became saturated through the augmentation along P_i. Let v be the first node of P that occurs in the subpath $y \overset{P_i}{\rightsquigarrow} t$ of P_i, and let u be the last node in the subpath $s \overset{P}{\rightsquigarrow} v$ of P that belongs to the subpath $s \overset{P_i}{\rightsquigarrow} x$ of P_i (see Figure 4.3). By the choice of u and v, all nodes on the subpath $P' = u \overset{P}{\rightsquigarrow} v$ of P except u and v do not belong to P_i. Hence, the arcs of P' are also available in the residual network $G_{f_{i-1}}$ and have the same capacity in both residual networks $G_{f_{i-1}}$ and G_{f_i}.

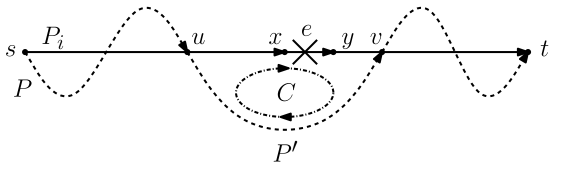


Figure 4.3: Paths P and P_i in the residual network G_{f_i}.

In the remainder of this proof, we show that at least one arc of P' is empty. Assume to the contrary that none of the arcs is empty in G_{f_i} and, hence, in $G_{f_{i-1}}$. This implies that, for each arc $e \in P'$, the residual network $G_{f_{i-1}}$ also contains the arc e^{-1}. Since P_i is the shortest s-t-path in $G_{f_{i-1}}$ and since the lengths of all possible s-t-paths are pairwise distinct, the path $s \overset{P_i}{\rightsquigarrow} u \overset{P}{\rightsquigarrow} v \overset{P_i}{\rightsquigarrow} t$ is longer than P_i. Consequently, the path $P' = u \overset{P}{\rightsquigarrow} v$ is longer than the path $u \overset{P_i}{\rightsquigarrow} v$. This contradicts the fact that flow f_{i-1} is optimal since the arcs of path $u \overset{P_i}{\rightsquigarrow} v$ and the reverse arcs e^{-1} of the arcs e of path P' form a directed cycle C in $G_{f_{i-1}}$ of negative costs. □

We use the technique of *discretization* and partition the interval $[0, n]$ into small subintervals of length ε. That way we can treat the number of lengths $\ell_-(f_i)$ that fall into a given subinterval almost as a binary random variable. This may be wrong if there are two possible s-t-paths whose lengths differ by at most ε. In this case, whose probability tends to 0 (see Lemma 4.3.1), we will simply bound the number of augmentation steps of the SSP algorithm by a worst-case bound according to the following lemma.

Lemma 4.3.11. *The number $|\mathcal{F}_0|$ of flows encountered by the SSP algorithm is bounded by $|\mathcal{F}_0| \le 3^{m+n}$.*

Proof. We call two possible residual networks equivalent if they contain the same arcs. Equivalent possible residual networks have the same shortest s-t-path in common. The length of this path is also the same. Hence, for two distinct flows $f_i, f_j \in \mathcal{F}_0$, the residual networks G_{f_i} and G_{f_j} are not equivalent due to Corollary 4.3.4 and Property 1. The number of equivalence classes is bounded by 3^{m+n} since there are m original edges and at most n auxiliary edges. This completes the proof. □

Lemma 4.3.12. *There are at most n bad flows $f \in \mathcal{F}$.*

Proof. According to Lemma 4.3.10, the augmenting path contains an empty arc e in each step. If e is an arc that corresponds to an auxiliary edge (this is the only case when e is not a good arc), then e is not empty after the augmentation. Since the SSP algorithm does not augment along arcs e^{-1} if e is an arc that corresponds to an auxiliary edge, non-empty arcs that correspond to auxiliary edges cannot be empty a second time. Thus, there can be at most n steps where the augmenting path does not contain a good arc. This implies that there are at most n bad flows $f \in \mathcal{F}$. □

We can now bound the probability that there is a flow $f_i \in \mathcal{F}$ whose previous path's length $\ell_-(f_i)$ falls into a given subinterval of length ε. Though we count bad flows separately, they also play a role in bounding the probability that there is a *good* flow $f_i \in \mathcal{F}$ such that $\ell_-(f_i)$ falls into a given subinterval of length ε.

Lemma 4.3.13. *For a fixed real $d \ge 0$, let $\mathcal{E}_{d,\varepsilon}$ be the event that there is a flow $f \in \mathcal{F}$ for which $\ell_-(f) \in (d, d+\varepsilon]$, and let $B_{d,\varepsilon}$ be the event that there is a bad flow $f' \in \mathcal{F}$ for which $\ell_-(f') \in (d, d+\varepsilon]$. Then the probability of $\mathcal{E}_{d,\varepsilon}$ can be bounded by*

$$\mathbf{Pr}\left[\mathcal{E}_{d,\varepsilon}\right] \le 2m\varepsilon\phi + 2 \cdot \mathbf{Pr}\left[B_{d,\varepsilon}\right] .$$

Proof. Let $A_{d,\varepsilon}$ be the event that there is a good flow $f \in \mathcal{F}$ for which $\ell_-(f) \in (d, d+\varepsilon]$. Since $\mathcal{E}_{d,\varepsilon} = A_{d,\varepsilon} \cup B_{d,\varepsilon}$, it suffices to show that $\mathbf{Pr}[A_{d,\varepsilon}] \le 2m\varepsilon\phi + \mathbf{Pr}[B_{d,\varepsilon}]$. Consider the event that there is a good flow whose previous path's length lies in the interval $(d, d+\varepsilon]$. Among all these good flows, let $\hat{f}$ be the one with the smallest value $\ell_-(\hat{f})$, i.e., $\hat{f}$ is the first good flow f encountered by the SSP algorithm for which $\ell_-(f) \in (d, d+\varepsilon]$, and let $f^\star$ be its previous flow. Flow $f^\star$ always exists since $\hat{f}$ cannot be the empty flow f_0. Corollary 4.3.4 and Property 1 yield $\ell_-(f^\star) < \ell_-(\hat{f})$. Thus, there can only be two cases: If $\ell_-(f^\star) \in (d, d+\varepsilon]$, then $f^\star$ is a bad flow by the choice of $\hat{f}$ and, hence, event $B_{d,\varepsilon}$ occurs. The interesting case which we consider now is when $\ell_-(f^\star) \le d$ holds. If this is true, then $d \in [\ell_-(f^\star), \ell_+(f^\star))$ due to $\ell_+(f^\star) = \ell_-(\hat{f})$.

As $\hat{f}$ is a good flow, the shortest path in the residual network $G_{f^\star}$ contains a good arc $e = (u, v)$. Applying Corollary 4.3.9 we obtain that we can reconstruct flow $f^\star$ by calling

$\mathsf{Reconstruct}(e,d)$. The shortest s-t-path P in the residual network $G_{f^\star}$ is the previous path of $\hat{f}$ and its length equals $\ell\_(\hat{f})$. Furthermore, P is of the form $s \overset{P}{\rightsquigarrow} u \rightarrow v \overset{P}{\rightsquigarrow} t$, where $s \overset{P}{\rightsquigarrow} u$ and $v \overset{P}{\rightsquigarrow} t$ are shortest paths in $G_{f^\star}$ from s to u and from v to t, respectively. These observations yield

$$A_{d,\varepsilon} \subseteq \bigcup_{e \in E} R_{e,d,\varepsilon} \cup \bigcup_{e \in E} R_{e^{-1},d,\varepsilon} \cup B_{d,\varepsilon}\,,$$

where $R_{e,d,\varepsilon}$ for an arc $e=(u,v)$ denotes the following event: By calling $\mathsf{Reconstruct}(e,d)$, we obtain a certain flow f. Let ℓ be the length of the shortest s-t-path in G_f that uses arc e. Then event $R_{e,d,\varepsilon}$ occurs if $\ell \in (d, d+\varepsilon]$. Therefore, the probability of event $A_{d,\varepsilon}$ is bounded by

$$\sum_{e \in E} \mathbf{Pr}\left[R_{e,d,\varepsilon}\right] + \sum_{e \in E} \mathbf{Pr}\left[R_{e^{-1},d,\varepsilon}\right] + \mathbf{Pr}\left[B_{d,\varepsilon}\right]\,.$$

We conclude the proof by showing $\mathbf{Pr}[R_{e,d,\varepsilon}] \le \varepsilon\phi$. For this, let e_0 be the edge corresponding to arc $e=(u,v)$ in the original network. If we fix all edge costs except cost c_{e_0} of edge e_0, then the output f of $\mathsf{Reconstruct}(e,d)$ is already determined. The same holds for the shortest s-t-path in G_f that uses arc e since it is of the form $s \rightsquigarrow u \rightarrow v \rightsquigarrow t$ where $P_1 = s \rightsquigarrow u$ is a shortest s-u-path in G_f that does not use v and where $P_2 = v \rightsquigarrow t$ is a shortest v-t-path in G_f that does not use u. The length ℓ of this path, however, depends linearly on the cost c_{e_0}. To be more precise, $\ell = \ell' + c_e = \ell' + \mathsf{sgn}(e) \cdot c_{e_0}$, where ℓ' is the length of P_1 plus the length of P_2 and where

$$\mathsf{sgn}(e) = \begin{cases} +1 & \text{if } e_0 = e\,, \\ -1 & \text{if } e_0 = e^{-1}\,. \end{cases}$$

Hence, ℓ falls into the interval $(d, d+\varepsilon]$ if and only if c_{e_0} falls into some fixed interval of length ε. The probability for this is bounded by $\varepsilon\phi$ as c_{e_0} is drawn according to a distribution whose density is bounded by ϕ. □

Corollary 4.3.14. *The expected number of augmentation steps the SSP algorithm performs is bounded by* $2mn\phi + 2n$.

Proof. Let $X = |\mathcal{F}|$ be the number of augmentation steps of the SSP algorithm. For reals $d, \varepsilon > 0$, let $\mathcal{E}_{d,\varepsilon}$ and $B_{d,\varepsilon}$ be the events defined in Lemma 4.3.13, let $X_{d,\varepsilon}$ be the number of flows $f \in \mathcal{F}$ for which $\ell\_(f) \in (d, d+\varepsilon]$, and let $Z_{d,\varepsilon} = \min\{X_{d,\varepsilon}, 1\}$ be the indicator variable of event $\mathcal{E}_{d,\varepsilon}$.

Since all costs are drawn from the interval $[0,1]$, the length of any possible s-t-path is bounded by n. Furthermore, according to Corollary 4.3.4, all lengths are non-negative (and positive with a probability of 1). Let F_ε denote the event that there are two possible

s-t-paths whose lengths differ by at most ε. Then for any positive integer k, we obtain

$$X = \sum_{i=0}^{k-1} X_{i\cdot\frac{n}{k},\frac{n}{k}} \begin{cases} = \sum_{i=0}^{k-1} Z_{i\cdot\frac{n}{k},\frac{n}{k}} & \text{if } F_{\frac{n}{k}} \text{ does not occur}\,, \\ \leq 3^{m+n} & \text{if } F_{\frac{n}{k}} \text{ occurs}\,. \end{cases}$$

With the concept of *discretization* we obtain

$$\begin{aligned} \mathbf{E}\left[X\right] &\leq \sum_{i=0}^{k-1} \mathbf{E}\left[Z_{i\cdot\frac{n}{k},\frac{n}{k}}\right] + 3^{m+n} \cdot \mathbf{Pr}\left[F_{\frac{n}{k}}\right] \\ &= \sum_{i=0}^{k-1} \mathbf{Pr}\left[\mathcal{E}_{i\cdot\frac{n}{k},\frac{n}{k}}\right] + 3^{m+n} \cdot \mathbf{Pr}\left[F_{\frac{n}{k}}\right] \\ &\leq 2mn\phi + 2 \cdot \sum_{i=0}^{k-1} \mathbf{Pr}\left[B_{i\cdot\frac{n}{k},\frac{n}{k}}\right] + 3^{m+n} \cdot \mathbf{Pr}\left[F_{\frac{n}{k}}\right] \\ &\leq 2mn\phi + 2n + 3^{m+n} \cdot \mathbf{Pr}\left[F_{\frac{n}{k}}\right]\,. \end{aligned}$$

The second inequality is due to Lemma 4.3.13 whereas the third inequality stems from Lemma 4.3.12. The claim follows since $\mathbf{Pr}\left[F_{n/k}\right] \to 0$ for $k \to \infty$ in accordance with Lemma 4.3.1. □

Proof of Theorem 1.2.2. Since each step of the SSP algorithm runs in time $O(m+n\log n)$ using Dijkstra's algorithm (see, e.g., the book of Korte and Vygen [KV07] for details), applying Corollary 4.3.14 yields the desired result. □

Chapter 5

Finding Short Paths on Polyhedra

In this chapter we study the problem of finding a short path between two given vertices x_1 and x_2 of a *polyhedron* $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ for a matrix $A = [a_1, \ldots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$. In particular, we will give an upper bound on the expected length of the path computed by the *shadow vertex method* (given as Algorithm 2) that depends polynomially on m, n, and a parameter δ which describes the flatness of the vertices of P. Section 5.2 explains why we can concentrate on analyzing non-degenerate polyhedra. In Section 5.3 we give an outline of our analysis, which is inspired by our analysis of the *successive shortest path algorithm* (Chapter 4), and present the main ideas. After that, in Section 5.4, we introduce the parameter δ formally and discuss some of its properties. Section 5.5 is devoted to the proof of Theorem 1.3.1. The probabilistic foundations of our analysis are provided in Section 5.6.

5.1 Notation

Let us first introduce some notation. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. Let $A \in \mathbb{R}^{m \times n}$ be an $m \times n$-matrix and let $i \in [m]$ and $j \in [n]$ be indices. With $A_{i,j}$ we refer to the $(m-1) \times (n-1)$-submatrix obtained from A by removing the i^{th} row and the j^{th} column. We call the determinant of any $k \times k$-submatrix of A a *sub-determinant* of A of size k. By $\mathbb{I}_n$ we denote the $n \times n$-identity matrix $\operatorname{diag}(1, \ldots, 1)$ and by $\mathbb{O}_{m \times n}$ the $m \times n$-zero matrix. If $n \in \mathbb{N}$ is clear from the context, then we define vector e_i to be the i^{th} column of $\mathbb{I}_n$. For a vector $x \in \mathbb{R}^n$ we denote by $\|x\| = \|x\|_2$ the Euclidean norm of x and by $N(x) = (1/\|x\|) \cdot x$ for $x \neq 0$ the normalization of vector x.

5.2 Degeneracy

A polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is called *degenerate* if there exists a vertex x of P for which more than n of the linear inequalities are tight. That is, there exist $n+1$

distinct rows a_i^{T} of A for which $a_i^{\mathrm{T}}x = b_i$. Any degenerate polyhedron P can be made non-degenerate by perturbing the vector b by a tiny amount of random noise. Thus, another polyhedron $\tilde{P}$ is obtained that is non-degenerate with probability one. Any degenerate vertex of P at which $\ell > n$ constraints are tight generates at most $\binom{\ell}{n}$ vertices of $\tilde{P}$ that are all very close to each other if the perturbation of b is small. We say that two vertices of $\tilde{P}$ that correspond to the same vertex of P are in the same equivalence class.

If the perturbation of the vector b is small enough, then any edge between two vertices of $\tilde{P}$ in different equivalence classes corresponds to an edge in P between the vertices that generated these equivalence classes. We apply the shadow vertex method to the polyhedron $\tilde{P}$ to find a path R between two arbitrary vertices from the equivalence classes generated by x_1 and x_2, respectively. Then we translate this path into a walk from x_1 to x_2 on the polyhedron P by mapping each vertex on the path R to the vertex that generated its equivalence class. This way, we obtain a walk from x_1 to x_2 on the polyhedron P that may visit vertices multiple times and may also stay in the same vertex for some steps. In the latter type of steps only the algebraic representation of the current vertex is changed. As this walk on P has the same length as the path that the shadow vertex algorithm computes on $\tilde{P}$, the upper bound we derive for the length of R also applies to the degenerate polyhedron P.

Of course the perturbation of the vector b might change the shape of the polyhedron P. The bounds we derive for the (expected) length of the path computed by the shadow vertex method only depend on the matrix A, which is the same for both polyhedra P and $\tilde{P}$. Consequently, the bounds we derive for the polyhedron $\tilde{P}$ and which also hold for the polyhedron P are already expressed in the parameters of P.

5.3 Outline of the Analysis

In the remainder of this thesis we assume that the polyhedron P is *non-degenerate*, i.e., for each vertex x of P there are exactly n indices i for which $a_i^{\mathrm{T}}x = b_i$. This implies that for any edge between two vertices x and y of P there are exactly $n-1$ indices i for which $a_i^{\mathrm{T}}x = a_i^{\mathrm{T}}y = b_i$. According to Section 5.2 this assumption is justified.

From the description of the shadow vertex method it is clear that the main step in proving Theorem 1.3.1 is to bound the expected number of edges on the path from $\pi(x_1)$ to $\pi(x_2)$ on the polygon P'. In order to do this, we look at the slopes of the edges on this path. As we discussed in Section 1.3.1, the sequence of slopes is monotonically decreasing. We will show that due to the randomness in the objective functions w_1 and w_2, it is even strictly decreasing with probability one. Furthermore all slopes on this path are bounded from below by 0.

Instead of counting the edges on the path from $\pi(x_1)$ to $\pi(x_2)$ directly, we will count the number of different slopes in the interval $[0, 1]$ and we observe that the expected number of slopes from the interval $[0, \infty)$ is twice the expected number of slopes from the interval $[0,1]$. In order to count the number of slopes in $[0,1]$, we partition the interval $[0,1]$ into several small subintervals and we bound for each of these subintervals I the expected number of slopes in I. Then we use linearity of expectation to obtain an upper bound on the expected number of different slopes in $[0,1]$, which directly translates into an upper bound on the expected number of edges on the path from $\pi(x_1)$ to $\pi(x_2)$.

We apply the principle of *discretization* and choose the subintervals so small that, with high probability, none of them contains more than one slope. Then the expected number of slopes in a subinterval $I = (t, t+\varepsilon]$ is approximately equal to the probability that there is a slope in the interval I. In order to bound this probability, we use an advanced version of the *interval probability bound.* The main idea is to split the random draw of the vectors w_1 and w_2 in the shadow vertex method into two steps. The first step reveals enough information about the realizations of these vectors to determine the last edge $e = (\hat{p}, p^\star)$ on the path from $\pi(x_1)$ to $\pi(x_2)$ whose slope is bigger than t (see Figure 5.1). Even though e is determined in the first step, its slope is not. We argue that there is still enough randomness left in the second step to bound the probability that the slope of e lies in the interval $(t, t+\varepsilon]$ from above, yielding Theorem 1.3.1.

We will now give some more details on how the random draw of the vectors w_1 and w_2 is partitioned. Let $\hat{x}$ and $x^\star$ be the vertices of the polyhedron P that are projected onto $\hat{p}$ and $p^\star$, respectively. Due to the non-degeneracy of the polyhedron P, there are exactly $n-1$ constraints that are tight for both $\hat{x}$ and $x^\star$ and there is a unique constraint $a_i^{\mathrm{T}} x \le b_i$ that is tight for $x^\star$ but not for $\hat{x}$. In the first step the vector w_1 is completely revealed while instead of w_2 only an element $\tilde{w}_2$ from the ray $\{w_2 + \gamma \cdot a_i :\ \gamma \ge 0\}$ is revealed. We then argue that knowing w_1 and $\tilde{w}_2$ suffices to identify the edge e. The only randomness left in the second step is the exact position of the vector w_2 on the ray $\{\tilde{w}_2 - \gamma \cdot a_i :\ \gamma \ge 0\}$, which suffices to bound the probability that the slope of e lies in the interval $(t, t+\varepsilon]$.

5.4 The Parameter δ

In this section we define the parameter δ that describes the flatness of the vertices of the polyhedron P and state some relevant properties.

Definition 5.4.1.

1. Let $z_1, \dots, z_n \in \mathbb{R}^n$ be linearly independent vectors and let $\varphi \in (0, \pi/2]$ be the angle between z_n and the hyperplane $\operatorname{span}\{z_1, \dots, z_{n-1}\}$. By $\hat{\delta}(\{z_1, \dots, z_{n-1}\}, z_n) = \sin\varphi$ we denote the sine of angle φ. Moreover, we set
$$\delta(z_1, \dots, z_n) = \min_{k \in [n]} \hat{\delta}(\{z_i :\ i \in [n] \setminus \{k\}\}, z_k)\,.$$
2. Given a matrix $A = [a_1, \dots, a_m]^{\mathrm{T}} \in \mathbb{R}^{m \times n}$, we set
$$\delta(A) = \min\left\{\delta(a_{i_1}, \dots, a_{i_n}) :\ a_{i_1}, \dots, a_{i_n} \text{ linearly independent}\right\}\,.$$

The value $\hat{\delta}(\{z_1,\dots,z_{n-1}\},z_n)$ describes how orthogonal z_n is to the hyperplane spanned by the vectors $z_1,\dots,z_{n-1}$. If $\varphi \approx 0$, that is, if z_n is close to the span of $z_1,\dots,z_{n-1}$, then $\hat{\delta}(\{z_1,\dots,z_{n-1}\},z_n) \approx 0$. On the other hand, if z_n is orthogonal to the vectors $z_1,\dots,z_{n-1}$, then $\varphi = \pi/2$ and, hence, $\hat{\delta}(\{z_1,\dots,z_{n-1}\},z_n) = 1$. The value $\hat{\delta}(\{z_1,\dots,z_{n-1}\},z_n)$ equals the distance between both faces of the parallelotope Q, given by $Q = \{\sum_{i=1}^n \alpha_i \cdot N(z_i) : \alpha_i \in [0,1]\}$, that are parallel to $\operatorname{span}\{z_1,\dots,z_{n-1}\}$ and is scale invariant.

The value $\delta(z_1,\dots,z_n)$ equals twice the inner radius r_n of the parallelotope Q and, thus, is a measure of the flatness of Q: A value $\delta(z_1,\dots,z_n) \approx 0$ implies that Q is nearly $(n-1)$-dimensional. On the other hand, if $\delta(z_1,\dots,z_n) = 1$, then the vectors $z_1,\dots,z_n$ are pairwise orthogonal, that is, Q is an n-dimensional unit cube.

The next lemma lists some useful statements concerning the parameter $\delta := \delta(A)$ including a connection to the parameters Δ_1, Δ_{n-1}, and Δ introduced in the paper of Bonifas et al. [BDE+12].

Lemma 5.4.2. *Let $z_1,\dots,z_n \in \mathbb{R}^n$ be linearly independent vectors, let $A \in \mathbb{R}^{m\times n}$ be a matrix, let $b \in \mathbb{R}^m$ be a vector, and let $\delta = \delta(A)$. Then the following claims hold true:*

1. *If M is the inverse of $[N(z_1),\dots,N(z_n)]^{\mathrm{T}}$, then*
$$\delta(z_1,\dots,z_n) = \frac{1}{\max_{k\in[n]} \|m_k\|} \le \frac{\sqrt{n}}{\max_{k\in[n]} \|M_k\|}\,,$$
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 $P = \{x \in \mathbb{R}^n : Ax \le b\}$ and let a_i^{T} be a row of A. If $a_i^{\mathrm{T}} \cdot (y_2 - y_1) \neq 0$, then $|a_i^{\mathrm{T}} \cdot (y_2 - y_1)| \ge \delta \cdot \|y_2 - y_1\|$.*
4. *If A is an integral matrix, then $1/\delta \le n\Delta_1\Delta_{n-1} \le n\Delta^2$, 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.*

Proof. First of all we derive a simple formula for $\hat{\delta}(\{z_1,\dots,z_{n-1}\},z_n)$. For this, assume that the vectors $z_1,\dots,z_n$ are normalized. Now consider a normal vector $x \neq 0$ of $\operatorname{span}\{z_1,\dots,z_{n-1}\}$ that lies in the same halfspace as z_n. Let $\varphi \in (0,\pi/2]$ be the angle between z_n and $\operatorname{span}\{z_1,\dots,z_{n-1}\}$ and let $\psi \in [0,\pi/2)$ be the angle between z_n and x. Clearly, $\varphi + \psi = \pi/2$. Consequently,
$$\hat{\delta}(\{z_1,\dots,z_{n-1}\},z_n) = \sin\varphi = \sin\left(\frac{\pi}{2} - \psi\right) = \cos\psi = \frac{z_n^{\mathrm{T}}x}{\|x\|}\,.$$
The last fraction is invariant under scaling of x. Since x and z_n lie in the same halfspace, we can assume w.l.o.g. that $z_n^{\mathrm{T}}x = 1$. Hence, $\hat{\delta}(\{z_1,\dots,z_{n-1}\},z_n) = 1/\|x\|$, where x is the

unique solution of the equation $[z_1, \dots, z_{n-1}, z_n]^{\mathrm{T}} \cdot x = (0, \dots, 0, 1)^{\mathrm{T}} = e_n$. If the vectors $z_1, \dots, z_n$ are not normalized, then we obtain

$$\hat{\delta}(\{z_1, \dots, z_{n-1}\}, z_n) = \hat{\delta}(\{N(z_1), \dots, N(z_{n-1})\}, N(z_n)) = \frac{1}{\|x\|},$$

where $x = [N(z_1), \dots, N(z_{n-1}), N(z_n)]^{-\mathrm{T}} \cdot e_n$. Since for the previous line of reasoning we can relabel the vectors $z_1, \dots, z_n$ arbitrarily, this implies

$$\begin{aligned}\delta(z_1, \dots, z_n) &= \min_{k \in [n]} \frac{1}{\|[N(z_1), \dots, N(z_n)]^{-\mathrm{T}} \cdot e_k\|} \\ &= \frac{1}{\max\left\{\|x\| : x \text{ is column of } [N(z_1), \dots, N(z_n)]^{-\mathrm{T}}\right\}}.\end{aligned}$$

This yields the equation in Claim 1. Due to

$$\left(\max_{k \in [n]} \|M_k\|\right)^2 \le \sum_{k \in [n]} \|M_k\|^2 = \sum_{k \in [n]} \|m_k\|^2 \le n \cdot \left(\max_{k \in [n]} \|m_k\|\right)^2$$

we obtain the inequality

$$\frac{1}{\max_{k \in [n]} \|m_k\|} \le \frac{\sqrt{n}}{\max_{k \in [n]} \|M_k\|}$$

stated in Claim 1. For Claim 2 observe that

$$\begin{aligned}[N(Qz_1), \dots, N(Qz_n)]^{-\mathrm{T}} &= [QN(z_1), \dots, QN(z_n)]^{-\mathrm{T}} \\ &= (Q \cdot [N(z_1), \dots, N(z_n)])^{-\mathrm{T}} \\ &= ([N(z_1), \dots, N(z_n)]^{-1} \cdot Q^{\mathrm{T}})^{\mathrm{T}} \\ &= Q \cdot [N(z_1), \dots, N(z_n)]^{-\mathrm{T}}\end{aligned}$$

for any orthogonal matrix Q. Therefore, we get

$$\begin{aligned}\frac{1}{\delta(Qz_1, \dots, Qz_n)} &= \max\left\{\|x\| : x \text{ is column of } [N(Qz_1), \dots, N(Qz_n)]^{-\mathrm{T}}\right\} \\ &= \max\left\{\|Qy\| : y \text{ is column of } [N(z_1), \dots, N(z_n)]^{-\mathrm{T}}\right\} \\ &= \max\left\{\|y\| : y \text{ is column of } [N(z_1), \dots, N(z_n)]^{-\mathrm{T}}\right\} \\ &= \frac{1}{\delta(z_1, \dots, z_n)}.\end{aligned}$$

For Claim 3 let y_1 and y_2 be two neighboring vertices of P. Then there are exactly $n-1$ indices j for which $a_j^{\mathrm{T}} \cdot (y_2 - y_1) = 0$. We denote them by $j_1, \dots, j_{n-1}$. If there is an index i

for which $a_i^{\mathrm{T}} \cdot (y_2 - y_1) \neq 0$, then $a_{j_1}, \ldots, a_{j_{n-1}}, a_i$ are linearly independent. Consequently, $\delta(a_{j_1}, \ldots, a_{j_{n-1}}, a_i) \geq \delta$. Let us assume that $a_i^{\mathrm{T}} \cdot (y_2 - y_1) \geq 0$. (Otherwise, consider $a_i^{\mathrm{T}} \cdot (y_1 - y_2)$ instead.) Since $y_2 - y_1$ is a normal vector of $\operatorname{span}\{a_{j_1}, \ldots, a_{j_{n-1}}\}$ that lies in the same halfspace as a_i, we obtain

$$\frac{a_i^{\mathrm{T}} \cdot (y_2 - y_1)}{\|y_2 - y_1\|} = \hat{\delta}(\{a_{j_1}, \ldots, a_{j_{n-1}}\}, a_i) \geq \delta(a_{j_1}, \ldots, a_{j_{n-1}}, a_i) \geq \delta$$

and, thus, $a_i^{\mathrm{T}} \cdot (y_2 - y_1) \geq \delta \cdot \|y_2 - y_1\|$. For proving Claim 4 we can focus on showing the first inequality. The second one follows from $\Delta \geq \max\{\Delta_1, \Delta_{n-1}\}$. For this, it suffices to show that for n arbitrary linearly independent rows $a_{i_1}^{\mathrm{T}}, \ldots, a_{i_n}^{\mathrm{T}}$ of A the inequality

$$\frac{1}{\hat{\delta}(\{a_{i_1}, \ldots, a_{i_{n-1}}\}, a_{i_n})} \leq n\Delta_1\Delta_{n-1}$$

holds. By previous observations we know that

$$\frac{1}{\hat{\delta}(\{a_{i_1}, \ldots, a_{i_{n-1}}\}, a_{i_n})} = \|x\|$$

where x is the unique solution of $\hat{A}x = e_n$ for $\hat{A} = [N(a_{i_1}), \ldots, N(a_{i_n})]^{\mathrm{T}}$. Let $\tilde{A} = [a_{i_1}, \ldots, a_{i_n}]^{\mathrm{T}}$. Then

$$\begin{aligned}\|x\|^2 &= \sum_{k=1}^{n} x_k^2 = \sum_{k=1}^{n} \left(\frac{\det(\hat{A}_{n,k})}{\det(\hat{A})}\right)^2 = \sum_{k=1}^{n} \left(\frac{\det(\tilde{A}_{n,k}) \cdot \prod_{j=1}^{n-1} \frac{1}{\|a_{i_j}\|}}{\det(\tilde{A}) \cdot \prod_{j=1}^{n} \frac{1}{\|a_{i_j}\|}}\right)^2 \\ &= \sum_{k=1}^{n} \left(\frac{\det(\tilde{A}_{n,k}) \cdot \|a_{i_n}\|}{\det(\tilde{A})}\right)^2 \leq \sum_{k=1}^{n} \left(\frac{\Delta_{n-1} \cdot \sqrt{n}\Delta_1}{1}\right)^2 = n^2\Delta_1^2\Delta_{n-1}^2 .\end{aligned}$$

Some of the equations need further explanation: Due to Cramer's rule, we have $x_k = \det(\bar{A}) / \det(\hat{A})$, where $\bar{A}$ is obtained from $\hat{A}$ by replacing the k^{th} column by the right-hand side e_n of the equation $\hat{A}x = e_n$. Laplace's formula yields $|\det(\bar{A})| = |\det(\hat{A}_{n,k})|$. Hence, the second equation is true. For the third equation note that the k^{th} row of matrix $\hat{A}$ is the same as the k^{th} row of matrix $\tilde{A}$ up to a factor of $1/\|a_{i_k}\|$. The inequality follows from $|\det(\tilde{A}_{n,k})| \leq \Delta_{n-1}$ since this is a sub-determinant of A of size $n-1$, from $\|a_{i_n}\| \leq \sqrt{n} \cdot \|a_{i_n}\|_\infty \leq \sqrt{n}\Delta_1$, since $\|a_{i_n}\|_\infty$ is a sub-determinant of A of size 1, and from $|\det(\tilde{A})| \geq 1$ since $\tilde{A}$ is invertible and integral by assumption. Hence,

$$\frac{1}{\hat{\delta}(\{a_{i_1}, \ldots, a_{i_{n-1}}\}, a_{i_n})} = \|x\| \leq n\Delta_1\Delta_{n-1} . \qquad \square$$

5.5 Analysis

For the proof of Theorem 1.3.1 we assume that $\|a_i\| = 1$ for all $i \in [m]$. This entails no loss of generality since normalizing the rows of matrix A (and scaling the right-hand side b appropriately) does neither change the behavior of Algorithm 2 nor does it change the parameter $\delta = \delta(A)$.

For two functions $f_1 \colon \mathbb{R}^n \to \mathbb{R}$ and $f_2 \colon \mathbb{R}^n \to \mathbb{R}$ we denote by $\pi = \pi_{f_1,f_2}$ the function $\pi \colon \mathbb{R}^n \to \mathbb{R}^2$, defined by $\pi(x) = (f_1(x), f_2(x))$. For our analysis only linear functions are of interest. We will treat an n-dimensional vector v as the linear function $x \mapsto v^{\mathrm{T}} x$. By $P' = P'_{f_1,f_2}$ we denote the projection $\pi(P)$ of the polyhedron P onto the Euclidean plane. If f_1 and f_2 are linear, then P' is a polygon and we denote by $R = R_{f_1,f_2}$ the path from $\pi(x_1)$ to $\pi(x_2)$ along the edges of P'.

Our goal is to bound the expected number of edges of the path $R = R_{w_1,w_2}$ which is random since w_1 and w_2 depend on the realizations of the random vectors λ and μ (see Line 4 of Algorithm 2). Each edge of R corresponds to a slope in $(0, \infty)$. These slopes are pairwise distinct with probability one (see Lemma 5.5.3). Hence, the number of edges of R equals the number of distinct slopes of R. In order to bound the expected number of distinct slopes we first restrict our attention to slopes in the interval $(0, 1]$.

Definition 5.5.1. 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

$$\left| \frac{w_2^{\mathrm{T}} \cdot (z_2 - z_1)}{w_1^{\mathrm{T}} \cdot (z_2 - z_1)} - \frac{w_2^{\mathrm{T}} \cdot (z_3 - z_2)}{w_1^{\mathrm{T}} \cdot (z_3 - z_2)} \right| \leq \varepsilon \,.$$

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.

Lemma 5.5.2. *The probability that there are two neighboring vertices z_1, z_2 of P such that $|w_1^{\mathrm{T}} \cdot (z_2 - z_1)| \leq \varepsilon \cdot \|z_2 - z_1\|$ is bounded from above by $2m^n\varepsilon/\delta$.*

Proof. Let z_1 and z_2 be two neighbors of P. Let $\Delta_z = z_2 - z_1$. Because the claim we want to show is invariant under scaling, we can assume without loss of generality that $\|\Delta_z\| = 1$. There are $n - 1$ indices $i_1, \ldots, i_{n-1} \in [m]$ such that $a_{i_k}^{\mathrm{T}} z_1 = b_{i_k} = a_{i_k}^{\mathrm{T}} z_2$. Recall that $w_1 = -[u_1, \ldots, u_n] \cdot \lambda$, where $\lambda = (\lambda_1, \ldots, \lambda_n)$ is drawn uniformly at random from $(0, 1]^n$. There must be an index i such that $a_{i_1}, \ldots, a_{i_{n-1}}, u_i$ are linearly independent. Hence, $\kappa := u_i^{\mathrm{T}} \Delta_z \neq 0$ and, thus, $|\kappa| \geq \delta$ due to Lemma 5.4.2, Claim 3. We apply the *principle of deferred decisions* and assume that all λ_j for $j \neq i$ are already drawn. Then

$$w_1^{\mathrm{T}} \Delta_z = -\sum_{j=1}^{n} \lambda_j \cdot u_j^{\mathrm{T}} \Delta_z = \underbrace{-\sum_{j \neq i} \lambda_j \cdot u_j^{\mathrm{T}} \Delta_z}_{=:\gamma} - \lambda_i \cdot \kappa \,.$$

Thus,

$$|w_1^{\mathrm{T}}\Delta_z| \le \varepsilon \iff \lambda_i \cdot \kappa \in [\gamma - \varepsilon, \gamma + \varepsilon] \iff \lambda_i \in \left[\frac{\gamma}{\kappa} - \frac{\varepsilon}{|\kappa|}, \frac{\gamma}{\kappa} + \frac{\varepsilon}{|\kappa|}\right].$$

The probability for the latter event is bounded by the length of the interval, that is, by $2\varepsilon/|\kappa| \le 2\varepsilon/\delta$. Since we have to consider at most $\binom{m}{n-1} \le m^n$ pairs of neighbors (z_1, z_2), applying a *union bound* yields the additional factor of m^n. $\square$

Lemma 5.5.3. *The probability of event $\mathcal{F}_\varepsilon$ tends to 0 for $\varepsilon \to 0$.*

Proof. 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 5.5.1 are invariant under scaling. Let $i_1, \dots, i_{n-1} \in [m]$ be the $n-1$ indices for which $a_{i_k}^{\mathrm{T}} z_1 = b_{i_k} = a_{i_k}^{\mathrm{T}} z_2$. The rows $a_{i_1}, \dots, a_{i_{n-1}}$ 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 i_ℓ for which $a_{i_\ell}^{\mathrm{T}} z_3 < b_{i_\ell}$, i.e., $a_{i_\ell}^{\mathrm{T}} \Delta'_z \ne 0$. Otherwise, z_1, z_2, z_3 would be collinear which would contradict the fact that they are distinct vertices of P. Without loss of generality assume that $\ell = n-1$. Since $a_{i_k}^{\mathrm{T}} \Delta_z = 0$ for each $k \in [n-1]$, the vectors $a_{i_1}, \dots, a_{i_{n-1}}, \Delta_z$ are linearly independent.

We apply the *principle of deferred decisions* and assume that w_1 is already fixed. Thus, $w_1^{\mathrm{T}}\Delta_z$ and $w_1^{\mathrm{T}}\Delta'_z$ are fixed as well. Moreover, we assume that $w_1^{\mathrm{T}}\Delta_z \ne 0$ and $w_1^{\mathrm{T}}\Delta'_z \ne 0$ since this happens almost surely due to Lemma 5.5.2. Now consider the matrix $M = [a_{i_1}, \dots, a_{i_{n-2}}, \Delta_z, a_{i_{n-1}}]$ and the random vector $(Y_1, \dots, Y_{n-1}, Z)^{\mathrm{T}} = M^{-1} \cdot w_2 = M^{-1} \cdot [v_1, \dots, v_n] \cdot \mu$. For fixed values $y_1, \dots, y_{n-1}$ let us consider all realizations of μ for which $(Y_1, \dots, Y_{n-1}) = (y_1, \dots, y_{n-1})$. Then

$$\begin{aligned} w_2^{\mathrm{T}}\Delta_z &= \left(M \cdot (y_1, \dots, y_{n-1}, Z)^{\mathrm{T}}\right)^{\mathrm{T}} \Delta_z \\ &= \sum_{k=1}^{n-2} y_k \cdot a_{i_k}^{\mathrm{T}} \Delta_z + y_{n-1} \cdot \Delta_z^{\mathrm{T}} \Delta_z + Z \cdot a_{i_{n-1}}^{\mathrm{T}} \Delta_z \\ &= y_{n-1}\,, \end{aligned}$$

i.e., the value of $w_2^{\mathrm{T}}\Delta_z$ does not depend on the outcome of Z since Δ_z is orthogonal to all a_{i_k}. For Δ'_z we obtain

$$\begin{aligned} w_2^{\mathrm{T}}\Delta'_z &= \left(M \cdot (y_1, \dots, y_{n-1}, Z)^{\mathrm{T}}\right)^{\mathrm{T}} \Delta'_z \\ &= \sum_{k=1}^{n-2} y_k \cdot a_{i_k}^{\mathrm{T}} \Delta'_z + y_{n-1} \cdot \Delta_z^{\mathrm{T}} \Delta'_z + Z \cdot a_{i_{n-1}}^{\mathrm{T}} \Delta'_z \\ &= \underbrace{y_{n-1} \cdot \Delta_z^{\mathrm{T}} \Delta'_z}_{=:\kappa} + Z \cdot a_{i_{n-1}}^{\mathrm{T}} \Delta'_z \end{aligned}$$

as Δ'_z is orthogonal to all a_{i_k} except for $k = \ell = n-1$. The chain of equivalences

$$\begin{aligned}
&\left|\frac{w_2^{\mathrm{T}}\Delta_z}{w_1^{\mathrm{T}}\Delta_z} - \frac{w_2^{\mathrm{T}}\Delta'_z}{w_1^{\mathrm{T}}\Delta'_z}\right| \le \varepsilon \\
\iff &\frac{w_2^{\mathrm{T}}\Delta'_z}{w_1^{\mathrm{T}}\Delta'_z} \in \left[\frac{w_2^{\mathrm{T}}\Delta_z}{w_1^{\mathrm{T}}\Delta_z} - \varepsilon, \frac{w_2^{\mathrm{T}}\Delta_z}{w_1^{\mathrm{T}}\Delta_z} + \varepsilon\right] \\
\iff &w_2^{\mathrm{T}}\Delta'_z \in \left[\frac{w_2^{\mathrm{T}}\Delta_z}{w_1^{\mathrm{T}}\Delta_z} \cdot w_1^{\mathrm{T}}\Delta'_z - \varepsilon \cdot |w_1^{\mathrm{T}}\Delta'_z|, \frac{w_2^{\mathrm{T}}\Delta_z}{w_1^{\mathrm{T}}\Delta_z} \cdot w_1^{\mathrm{T}}\Delta'_z + \varepsilon \cdot |w_1^{\mathrm{T}}\Delta'_z|\right] \\
\iff &Z \cdot a_{i_{n-1}}^{\mathrm{T}}\Delta'_z \in \left[\frac{w_2^{\mathrm{T}}\Delta_z}{w_1^{\mathrm{T}}\Delta_z} \cdot w_1^{\mathrm{T}}\Delta'_z - \kappa - \varepsilon \cdot |w_1^{\mathrm{T}}\Delta'_z|, \frac{w_2^{\mathrm{T}}\Delta_z}{w_1^{\mathrm{T}}\Delta_z} \cdot w_1^{\mathrm{T}}\Delta'_z - \kappa + \varepsilon \cdot |w_1^{\mathrm{T}}\Delta'_z|\right]
\end{aligned}$$

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 $2\varepsilon \cdot |w_1^{\mathrm{T}}\Delta'_z|/|a_{i_{n-1}}^{\mathrm{T}}\Delta'_z|$. The probability of this is bounded from above by

$$\frac{2n \cdot 2\varepsilon \cdot \frac{|w_1^{\mathrm{T}}\Delta'_z|}{|a_{i_{n-1}}^{\mathrm{T}}\Delta'_z|}}{\delta(r_1, \dots, r_n) \cdot \min_{k\in[n]} \|r_k\|} = \underbrace{\frac{4n \cdot |w_1^{\mathrm{T}}\Delta'_z|}{\delta(r_1, \dots, r_n) \cdot \min_{k\in[n]} \|r_k\| \cdot |a_{i_{n-1}}^{\mathrm{T}}\Delta'_z|}}_{=:\gamma} \cdot \varepsilon \,,$$

where $[r_1, \dots, r_n] = M^{-1} \cdot [v_1, \dots, v_n]$. This is due to $(Y_1, \dots, Y_{n-1}, Z)^{\mathrm{T}} = [r_1, \dots, r_n] \cdot \mu$ and Theorem 5.6.1. Since the vectors $r_1, \dots, r_n$ are linearly independent, we have $\delta(r_1, \dots, r_n) > 0$ and $\min_{k\in[n]} \|r_k\| > 0$. Furthermore, $|a_{i_{n-1}}^{\mathrm{T}}\Delta'_z| > 0$ since i_{n-1} is the constraint which is not tight for z_3, but for z_2. Hence, $\gamma < \infty$, and thus

$$\mathbf{Pr}\left[\left|\frac{w_2^{\mathrm{T}}\Delta_z}{w_1^{\mathrm{T}}\Delta_z} - \frac{w_2^{\mathrm{T}}\Delta'_z}{w_1^{\mathrm{T}}\Delta'_z}\right| \le \varepsilon\right] \to 0$$

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*. □

Let $p \neq \pi(x_2)$ be a vertex of R. We call the slope s of the edge incident to p to the right of p *the slope of* p. As a convention, we set the slope of $\pi(x_2)$ to 0 which is smaller than the slope of any other vertex p of R.

Let $t \ge 0$ be an arbitrary real, let $\hat{p}$ be the right-most vertex of R whose slope is larger than t, and let $p^\star$ be the right neighbor of $\hat{p}$ (see Figure 5.1). Furthermore, let $\hat{x}$ and $x^\star$ be the neighboring vertices of P with $\pi(\hat{x}) = \hat{p}$ and $\pi(x^\star) = p^\star$, respectively, and 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 polyhedron P. For an arbitrary real $\gamma \ge 0$ we consider the vector $\tilde{w}_2 = w_2 + \gamma \cdot a_i$.

Lemma 5.5.4. *Let $\tilde{\pi} = \pi_{w_1,\tilde{w}_2}$ and let $\tilde{R} = R_{w_1,\tilde{w}_2}$ be the path from $\tilde{\pi}(x_1)$ to $\tilde{\pi}(x_2)$ in the projection $\tilde{P}' = P'_{w_1,\tilde{w}_2}$ of polyhedron P. Furthermore, let $\tilde{p}^\star$ be the left-most vertex of $\tilde{R}$ whose slope does not exceed t. Then $\tilde{p}^\star = \tilde{\pi}(x^\star)$.*

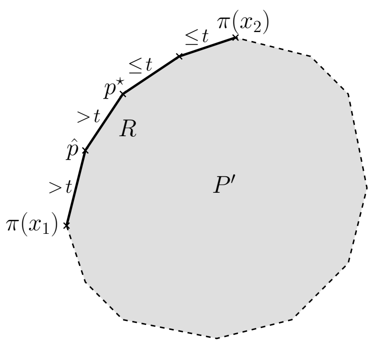


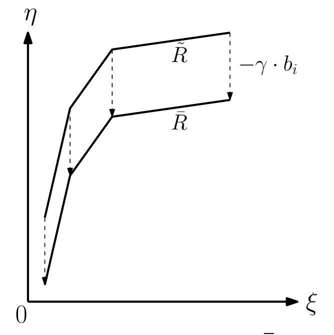
Figure 5.1: Slopes of the vertices of R

Let us reformulate the statement of Lemma 5.5.4 as follows: The vertex $\tilde{p}^\star$ is defined for the path $\tilde{R}$ of polygon $\tilde{P}'$ with the same rules as used to define the vertex $p^\star$ of the original path R of polygon 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 polyhedron P, that is, $p^\star = \pi(x^\star)$ and $\tilde{p}^\star = \tilde{\pi}(x^\star)$. Let us remark that Lemma 5.5.4 is a significant generalization of Lemma 4.3.8.

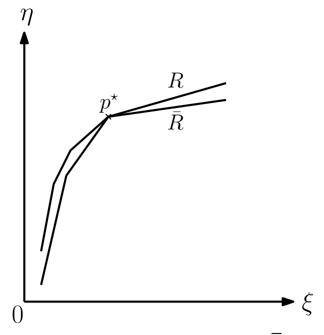
Proof. We consider a linear auxiliary function $\bar{w}_2 \colon \mathbb{R}^n \to \mathbb{R}$, given by $\bar{w}_2(x) = \tilde{w}_2^{\mathrm{T}} x - \gamma \cdot b_i$. The paths $\bar{R} = R_{w_1,\bar{w}_2}$ and $\tilde{R}$ are identical except for a shift by $-\gamma \cdot b_i$ in the second coordinate because for $\bar{\pi} = \pi_{w_1,\bar{w}_2}$ we obtain

$$\bar{\pi}(x) = (w_1^{\mathrm{T}} x, \tilde{w}_2^{\mathrm{T}} x - \gamma \cdot b_i) = (w_1^{\mathrm{T}} x, \tilde{w}_2^{\mathrm{T}} x) - (0, \gamma \cdot b_i) = \tilde{\pi}(x) - (0, \gamma \cdot b_i)$$

for all $x \in \mathbb{R}^n$. Consequently, the slopes of $\bar{R}$ and $\tilde{R}$ are exactly the same (see Figure 5.2a).



(a) Relation between $\bar{R}$ and $\tilde{R}$



(b) Relation between $\bar{R}$ an R

Figure 5.2: Relations between R, $\tilde{R}$, and $\bar{R}$

Let $x \in P$ be an arbitrary point from the polyhedron P. Then

$$\tilde{w}_2^{\mathrm{T}} x = w_2^{\mathrm{T}} x + \gamma \cdot a_i^{\mathrm{T}} x \leq w_2^{\mathrm{T}} x + \gamma \cdot b_i \,.$$

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 least the second coordinate of $\bar{\pi}(x)$ as $\bar{w}_2(x) = \tilde{w}_2^{\mathrm{T}} x - \gamma \cdot b_i \leq w_2^{\mathrm{T}} x$. Additionally, we have $\pi(x^\star) = \bar{\pi}(x^\star)$. Thus, path $\bar{R}$ is below path R but they meet at point $p^\star = \pi(x^\star)$. Hence, the slope of $\bar{R}$ to the left (right) of $p^\star$ is at least (at most) the slope of R to the left (right) of $p^\star$ which is greater than (at most) t (see Figure 5.2b). Consequently, $p^\star$ is the left-most 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 left-most vertex of $\tilde{R}$ whose slope does not exceed t, i.e., $\tilde{\pi}(x^\star) = \tilde{p}^\star$. □

Lemma 5.5.4 holds for any vector $\tilde{w}_2$ on the ray $\vec{r} = \{w_2 + \gamma \cdot a_i : \gamma \geq 0\}$. As $\|w_2\| \leq n$ (see Section 1.3.1), we have $w_2 \in [-n, n]^n$. Hence, ray $\vec{r}$ intersects the boundary of $[-n, n]^n$ in a unique point z. We choose $\tilde{w}_2 = \tilde{w}_2(w_2, i) := z$ and obtain the following result.

Corollary 5.5.5. *Let $\tilde{\pi} = \pi_{w_1, \tilde{w}_2(w_2, i)}$ and let $\tilde{p}^\star$ be the left-most vertex of path $\tilde{R} = R_{w_1, \tilde{w}_2(w_2, i)}$ whose slope does not exceed t. Then $\tilde{p}^\star = \tilde{\pi}(x^\star)$.*

Note, that Corollary 5.5.5 only holds for the right choice of index $i = i(x^\star, \hat{x})$. The vector $\tilde{w}_2(w_2, i)$ is defined for any vector $w_2 \in [-n, n]^n$ 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 w_1 and w_2.

Definition 5.5.6. For an index $i \in [m]$ and a real $t \geq 0$ let $\tilde{p}^\star$ be the left-most vertex of $\tilde{R} = R_{w_1, \tilde{w}_2(w_2, i)}$ whose slope does not exceed t and let $y^\star$ be the corresponding vertex of P. For a real $\varepsilon > 0$ we denote by $\mathcal{E}_{i,t,\varepsilon}$ the event that the conditions

- $a_i^{\mathrm{T}} y^\star = b_i$ and
- $w_2^{\mathrm{T}}(\hat{y} - y^\star)/w_1^{\mathrm{T}}(\hat{y} - y^\star) \in (t, t+\varepsilon]$, 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 polyhedron 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_2 from the line $L := \{w_2 + x \cdot a_i : x \in \mathbb{R}\}$ are mapped to the same vector $\tilde{w}_2(w_2, i)$. Consequently, if w_1 is fixed and if we only consider realizations of μ for which $w_2 \in L$, then vertex $\tilde{p}^\star$ and, hence, vertex $y^\star$ from Definition 5.5.6 are already determined. However, since w_2 is not completely specified, we have some randomness left for event $\mathcal{E}_{i,t,\varepsilon}$ to occur. This allows us to bound the probability of event $\mathcal{E}_{i,t,\varepsilon}$ from above (see proof of Lemma 5.5.8). The next lemma shows why this probability matters.

Lemma 5.5.7. *For reals $t \geq 0$ and $\varepsilon > 0$ let $A_{t,\varepsilon}$ denote the event that the path $R = R_{w_1,w_2}$ has a slope in $(t, t+\varepsilon]$. Then $A_{t,\varepsilon} \subseteq \bigcup_{i=1}^{m} \mathcal{E}_{i,t,\varepsilon}$.*

Proof. Assume that event $A_{t,\varepsilon}$ occurs. Let $\hat{p}$ be the right-most vertex of R whose slope exceeds t, let $p^\star$ be the right neighbor of $\hat{p}$, and let $\hat{x}$ and $x^\star$ be the neighboring vertices of P for which $\pi(\hat{x}) = \hat{p}$ and $\pi(x^\star) = p^\star$, where $\pi = \pi_{w_1,w_2}$. Moreover, let $i = i(x^\star, \hat{x})$ be the index for which $a_i^{\mathrm{T}} x^\star = b_i$ but $a_i^{\mathrm{T}} \hat{x} < b_i$. We show that event $\mathcal{E}_{i,t,\varepsilon}$ occurs.

Consider the left-most vertex $\tilde{p}^\star$ of $\tilde{R} = R_{w_1,\tilde{w}_2(w_2,i)}$ whose slope does not exceed t and let $y^\star$ be the corresponding vertex of P. In accordance with Corollary 5.5.5 we obtain $y^\star = x^\star$. Hence, $a_i^{\mathrm{T}} y^\star = b_i$, i.e., the first condition of event $\mathcal{E}_{i,t,\varepsilon}$ holds. Now let $\hat{y}$ be the unique neighbor y of $y^\star$ for which $a_i^{\mathrm{T}} y < b_i$. Since $y^\star = x^\star$, we obtain $\hat{y} = \hat{x}$. Consequently,

$$\frac{w_2^{\mathrm{T}}(\hat{y} - y^\star)}{w_1^{\mathrm{T}}(\hat{y} - y^\star)} = \frac{w_2^{\mathrm{T}}(\hat{x} - x^\star)}{w_1^{\mathrm{T}}(\hat{x} - x^\star)} \in (t, t+\varepsilon]\,,$$

since this is the smallest slope of R that exceeds t and since there is a slope in $(t, t+\varepsilon]$ by assumption. Hence, event $\mathcal{E}_{i,t,\varepsilon}$ occurs since the second condition for event $\mathcal{E}_{i,t,\varepsilon}$ to happen holds as well. □

With Lemma 5.5.7 we can now bound the probability of event $A_{t,\varepsilon}$.

Lemma 5.5.8. *For reals $t \geq 0$ and $\varepsilon > 0$ the probability of event $A_{t,\varepsilon}$ is bounded by $\mathbf{Pr}[A_{t,\varepsilon}] \leq 4mn^2\varepsilon/\delta^2$.*

Proof. Due to Lemma 5.5.7 it suffices to show that $\mathbf{Pr}[\mathcal{E}_{i,t,\varepsilon}] \leq (1/m) \cdot 4mn^2\varepsilon/\delta^2 = 4n^2\varepsilon/\delta^2$ for any index $i \in [m]$. We apply the *principle of deferred decisions* and assume that vector $\lambda \in (0,1]^n$ is not random anymore, but arbitrarily fixed. Thus, vector w_1 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 $(Y_1, \ldots, Y_{n-1}, Z)^{\mathrm{T}} = Q^{\mathrm{T}} w_2$ 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 $w_2 = [v_1, \ldots, v_n] \cdot \mu$. For fixed values $y_1, \ldots, y_{n-1}$ let us consider all realizations of μ such that $(Y_1, \ldots, Y_{n-1}) = (y_1, \ldots, y_{n-1})$. Then w_2 is fixed up to the ray

$$w_2(Z) = Q \cdot (y_1, \ldots, y_{n-1}, Z)^{\mathrm{T}} = \sum_{j=1}^{n-1} y_j \cdot q_j + Z \cdot a_i = w + Z \cdot a_i$$

for $w = \sum_{j=1}^{n-1} y_j \cdot q_j$. All realizations of $w_2(Z)$ that are under consideration are mapped to the same value $\tilde{w}_2$ by the function $w_2 \mapsto \tilde{w}_2(w_2, i)$, i.e., $\tilde{w}_2(w_2(Z), i) = \tilde{w}_2$ for any possible realization of Z. In other words, if $w_2 = w_2(Z)$ is specified up to this ray, then the path $R_{w_1,\tilde{w}_2(w_2,i)}$ and, hence, the vectors $y^\star$ and $\hat{y}$ used for the definition of event $\mathcal{E}_{i,t,\varepsilon}$, are already determined.

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

$$(t, t+\varepsilon] \ni \frac{w_2^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{w_1^{\mathrm{T}} \cdot (\hat{y} - y^\star)} = \underbrace{\frac{w^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{w_1^{\mathrm{T}} \cdot (\hat{y} - y^\star)}}_{=:\alpha} + Z \cdot \underbrace{\frac{a_i^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{w_1^{\mathrm{T}} \cdot (\hat{y} - y^\star)}}_{=:\beta} .$$

The next step in this proof will be to show that the inequality $|\beta| \geq \delta/n$ is necessary for event $\mathcal{E}_{i,t,\varepsilon}$ to happen. For the sake of simplicity let us assume that $\|\hat{y} - y^\star\| = 1$ since β is invariant under scaling. If event $\mathcal{E}_{i,t,\varepsilon}$ 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 5.4.2, Claim 3 we obtain $|a_i^{\mathrm{T}} \cdot (\hat{y} - y^\star)| \geq \delta \cdot \|\hat{y} - y^\star\| = \delta$ and, hence,

$$|\beta| = \left| \frac{a_i^{\mathrm{T}} \cdot (\hat{y} - y^\star)}{w_1^{\mathrm{T}} \cdot (\hat{y} - y^\star)} \right| \geq \frac{\delta}{|w_1^{\mathrm{T}} \cdot (\hat{y} - y^\star)|} \geq \frac{\delta}{\|w_1\| \cdot \|\hat{y} - y^\star)\|} \geq \frac{\delta}{n \cdot 1} .$$

Summarizing the previous observations we can state that if event $\mathcal{E}_{i,t,\varepsilon}$ occurs, then $|\beta| \geq \delta/n$ and $\alpha + Z \cdot \beta \in (t, t+\varepsilon] \subseteq [t-\varepsilon, t+\varepsilon]$. Hence,

$$Z \in \left[\frac{t-\alpha}{\beta} - \frac{\varepsilon}{|\beta|}, \frac{t-\alpha}{\beta} + \frac{\varepsilon}{|\beta|} \right] \subseteq \left[\frac{t-\alpha}{\beta} - \frac{\varepsilon}{\frac{\delta}{n}}, \frac{t-\alpha}{\beta} + \frac{\varepsilon}{\frac{\delta}{n}} \right] =: I(y_1, \dots, y_{n-1}) .$$

Let $B_{i,t,\varepsilon}$ denote the event that Z falls into the interval $I(Y_1, \dots, Y_{n-1})$ of length $2n\varepsilon/\delta$. We showed that $\mathcal{E}_{i,t,\varepsilon} \subseteq B_{i,t,\varepsilon}$. Consequently,

$$\mathbf{Pr}\left[\mathcal{E}_{i,t,\varepsilon}\right] \leq \mathbf{Pr}\left[B_{i,t,\varepsilon}\right] \leq \frac{2n \cdot \frac{2n\varepsilon}{\delta}}{\delta(Q^{\mathrm{T}} v_1, \dots, Q^{\mathrm{T}} v_n)} \leq \frac{4n^2 \varepsilon}{\delta^2} ,$$

where the second inequality is due to Theorem 5.6.1: By definition, we have

$$(Y_1, \dots, Y_{n-1}, Z)^{\mathrm{T}} = Q^{\mathrm{T}} w_2 = Q^{\mathrm{T}} \cdot [v_1, \dots, v_n] \cdot \mu = [Q^{\mathrm{T}} v_1, \dots, Q^{\mathrm{T}} v_n] \cdot \mu .$$

The third inequality stems from the fact that $\delta(Q^{\mathrm{T}} v_1, \dots, Q^{\mathrm{T}} v_n) = \delta(v_1, \dots, v_n) \geq \delta$, where the equality is due to the orthogonality of Q (Claim 2 of Lemma 5.4.2). □

Lemma 5.5.9. *Let Y be the number of slopes of $R = R_{w_1,w_2}$ that lie in the interval* $(0, 1]$. *Then* $\mathbf{E}[Y] \leq 4mn^2/\delta^2$.

Proof. For a real $\varepsilon > 0$ let $\mathcal{F}_\varepsilon$ denote the event from Definition 5.5.1. Recall that all slopes of R differ by more than ε if $\mathcal{F}_\varepsilon$ does not occur. 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 there is such a slope and $Z_{t,\varepsilon} = 0$ otherwise. Then for any integer $k \geq 1$

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

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

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

where the second inequality stems from Lemma 5.5.8. The claim follows since the bound on $\mathbf{E}[Y]$ holds for any integer $k \ge 1$ and since $\mathbf{Pr}[\mathcal{F}_\varepsilon] \to 0$ for $\varepsilon \to 0$ in accordance with Lemma 5.5.3. □

Proof of Theorem 1.3.1. Lemma 5.5.9 bounds only the expected number of edges on the path R that have a slope in the interval $(0,1]$. However, the lemma can also be used to bound the expected number of edges whose slope is larger than 1. For this, one only needs to exchange the order of the objective functions $x \mapsto w_1^{\mathrm{T}} x$ and $x \mapsto w_2^{\mathrm{T}} x$ in the projection π. Then any edge with a slope of $s > 0$ becomes an edge with slope $1/s$. Due to the symmetry in the choice of w_1 and w_2, Lemma 5.5.9 can also be applied to bound the expected number of edges whose slope lies in $(0,1]$ for this modified projection, which are exactly the edges whose original slope lies in $[1,\infty)$.

Formally we can argue as follows. Consider the vertices $x_1' = x_2$ and $x_2' = x_1$, the directions $w_1' = -w_2$ and $w_2' = -w_1$, and the projection $\pi' = \pi_{w_1',w_2'}$, yielding a path R' from $\pi'(x_1')$ to $\pi'(x_2')$. Let X be the number of slopes of R and let Y and Y' be the number of slopes of R and of R', respectively, that lie in the interval $(0,1]$. The paths R and R' are identical except for the linear transformation $(x,y) \mapsto (-y,-x)$. Consequently, s is a slope of R if and only if $1/s$ is a slope of R' and, hence, $X \le Y + Y'$. One might expect equality here but in the unlikely case that R contains an edge with slope equal to 1 we have $X = Y + Y' - 1$. The expectation of Y is given by Lemma 5.5.9. Since this result holds for any two vertices x_1 and x_2 it also holds for x_1' and x_2'. Note, that w_1' and w_2' have exactly the same distribution as the directions the shadow vertex algorithm computes for x_1' and x_2'. Therefore, Lemma 5.5.9 can also be applied to bound $\mathbf{E}[Y']$ and we obtain $\mathbf{E}[X] \le \mathbf{E}[Y] + \mathbf{E}[Y'] \le 8mn^2/\delta^2$. □

The proof of Corollary 1.3.2 follows immediately from Theorem 1.3.1 and Claim 4 of Lemma 5.4.2.

5.6 Some Probability Theory

The following theorem is variant of Theorem 3.3 for uniformly distributed random variables $X_1, \dots, X_n$, square matrices, and $k = 1$, expressed in the parameter δ that has been introduced in Definition 5.4.1.

Theorem 5.6.1. *Let $X_1, \dots, X_n$ be random variables that are independently and uniformly drawn from the interval $(0,1]$, let $A = [a_1, \dots, a_n] \in \mathbb{R}^{n \times n}$ 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 $I \colon \mathbb{R}^{n-1} \to \{[x, x+\varepsilon] : x \in \mathbb{R}\}$ be a function mapping a tuple $(y_1, \dots, y_{n-1})$ to an interval $I(y_1, \dots, y_{n-1})$ of length ε. Then the probability that Z lies in the interval $I(Y_1, \dots, Y_{n-1})$ is bounded by*

$$\mathbf{Pr}\left[Z \in I(Y_1, \dots, Y_{n-1})\right] \leq \frac{2n\varepsilon}{\delta(a_1, \dots, a_n) \cdot \min_{k \in [n]} \|a_k\|} .$$

Proof. First of all note, that the density f_i of the random variable X_i takes the value 1 on the interval $(0,1]$ and is 0 otherwise. Hence, we can apply the second claim of Theorem 3.3 for $m = n$, $\phi = 1$, $k = 1$, and $\hat{A} = A$ and obtain

$$\mathbf{Pr}\left[Z \in I(Y_1, \dots, Y_{n-1})\right] \leq 2 \cdot \sum_{i=1}^{n} \frac{|\det(A_{n,i})|}{|\det(A)|} \cdot \varepsilon ,$$

where $A_{n,i}$ denotes the $(n-1) \times (n-1)$-submatrix of A obtained from A by removing the last row and column i. Now we bound the fraction $|\det(A_{n,i})|/|\det(A)|$. To do this, consider the equation $Ax = e_n$. We obtain

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

where the first equality is due to Cramer's rule and the second equality is due to Laplace's formula. Hence,

$$\mathbf{Pr}\left[Z \in I(Y_1, \dots, Y_{n-1})\right] \leq 2\varepsilon \cdot \sum_{i \in [n]} |x_i| = 2\varepsilon \cdot \|x\|_1 \leq 2\sqrt{n}\varepsilon \cdot \|x\|_2$$

Now consider the equation $\tilde{A}y = e_n$ for the normalized matrix $\tilde{A} = [N(a_1), \dots, N(a_n)]$ and let $b_1, \dots, b_n$ be the columns of $\tilde{A}^{-1}$. Vector $y = \tilde{A}^{-1}e_n$ is the n^{th} column of the matrix $\tilde{A}^{-1}$, i.e., $y = b_n$. Thus, we obtain $\|y\| \leq \max_{i=1,\dots,n} \|b_i\| \leq \sqrt{n}/\delta(a_1, \dots, a_n)$, where second inequality is due to Claim 1 of Lemma 5.4.2. Due to $A = \tilde{A} \cdot \operatorname{diag}(\|a_1\|, \dots, \|a_n\|)$, we have

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

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

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

Chapter 6

Scheduling Heuristics

In this chapter we study the smoothed performance guarantees of popular local search and greedy algorithms for scheduling: the *jump algorithm*, the *lex-jump algorithm*, and the *list scheduling algorithm*. In Section 6.2, we provide asymptotically matching upper and lower bounds on the smoothed performance guarantees of these algorithms in the environment with unrestricted related machines. In Section 6.3, we show that neither the jump algorithm nor the lex-jump algorithm benefit from smoothing in the setting with restricted machines.

6.1 Notation

Consider an instance I for the scheduling problem with a job set $J = \{1, \dots, n\}$, processing requirements $p_1, \dots, p_n \in [0, 1]$, a set $M = \{1, \dots, m\}$ of machines with speeds $s_{\max} {:=} s_1 \geq \dots \geq s_m {=:} s_{\min} = 1$ (in the case of non-identical machines), and the sets $\mathcal{M}_1, \dots, \mathcal{M}_n \subseteq M$ of allowed machines (in the case of restricted machines). For a schedule $\sigma\colon J \to M$ we denote by $J_i(\sigma) \subseteq J$ the set of jobs assigned to machine i according to σ. The *(total) processing requirement* on a machine i is defined as $\sum_{j \in J_i(\sigma)} p_j$ and the *load* of machine i is defined as $L_i(I, \sigma) = \sum_{j \in J_i(\sigma)} p_{ij}$. The makespan $C_{\max}(I, \sigma)$ of σ can be written as $C_{\max}(I, \sigma) = \max_{i \in M} L_i(I, \sigma)$. The optimal makespan, i.e., the makespan of an optimal schedule is denoted by $C^{\star}_{\max}(I)$. By Jump(I), Lex(I), and List(I) we denote the set of all feasible jump optimal schedules, lex-jump optimal schedules, and list schedules, respectively, according to instance I.

If the instance I is clear from the context, then we simply write $L_i(\sigma)$ instead of $L_i(I, \sigma)$, $C_{\max}(\sigma)$ instead of $C_{\max}(I, \sigma)$, and $C^{\star}_{\max}$ instead of $C^{\star}_{\max}(I)$. If the schedule σ is clear as well, we simplify our notation further to L_i and $C_{\max}$ and we write J_i instead of $J_i(\sigma)$. In Appendix 6.4, the notation is summarized in a table.

6.2 Unrestricted Machines

In this section we establish matching lower and upper bounds for the smoothed performance guarantees of the jump algorithm, the lex-jump algorithm, and the list scheduling algorithm. If all machines have the same speed, then all of these algorithms have a constant worst-case approximation guarantee. Consequently, also the smoothed performance guarantee is constant. The only interesting environment in the unrestricted setting is the one with *related machines* which we are going to study in the remainder.

6.2.1 Jump Optimal Schedules

We show that the smoothed performance guarantee of the jump algorithm grows linearly with the smoothing parameter ϕ and is independent of the number of jobs and machines. In particular, it is constant if the smoothing parameter is constant which it is, for instance, in an average-case analysis with uniform distributions. In proving our results, we make use of the following proposition which follows from Cho and Sahni [CS80].

Proposition 6.2.1. *For any scheduling instance I with m unrestricted related machines and n jobs*

$$\max_{\sigma\in\text{Jump}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} \le \frac{1+\sqrt{4\min\{m,n\}-3}}{2} \le \frac{1}{2}+\sqrt{n}\,.$$

Theorem 6.2.2. *For any ϕ-smooth instance $\mathcal{I}$ with unrestricted and related machines,*

$$\mathbf{E}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\text{Jump}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)}\right] < 5.1\phi + 2.5 = O(\phi)\,.$$

Proof. First note that, if $m > n$, then neither the optimal schedule nor any jump-optimal schedule assign any job to any of the slowest $m-n$ machines. Hence, we can ignore these slowest $m-n$ machines, and therefore we can assume that $m \le n$. We will prove an upper bound on the performance guarantee of jump optimal schedules that decreases when the sum of processing requirements $Q = \sum_{j\in J} p_j$ increases and that is valid for every instance. Then we will argue that for ϕ-smooth instances Q is usually not too small, which yields the theorem.

Let σ denote an arbitrary jump optimal schedule for some arbitrary processing requirements $p_j \in [0,1]$. Moreover, let i be an arbitrary machine, let machine $i_{\max}$ be a critical machine in schedule σ, and let j be a job assigned to machine $i_{\max}$ by schedule σ. By jump optimality of σ it follows that

$$C_{\max}(\sigma) = L_{i_{\max}} \le L_i + \frac{p_j}{s_i} \le L_i + \frac{p_{\max}}{s_i}\,,$$

where $p_{\max}$ denotes the processing requirement of the largest job. The previous inequality yields that $s_i \cdot C_{\max}(\sigma) \le s_i \cdot L_i + p_{\max}$ for all machines $i \in M$. A summation over all machines $i \in M$ yields

$$\begin{aligned}\sum_{i\in M} s_i \cdot C_{\max}(\sigma) &\le \sum_{i\in M} s_i \cdot L_i + \sum_{i\in M} p_{\max} = \sum_{i\in M} s_i \cdot L_i + m \cdot p_{\max} \\ &\le \sum_{i\in M} s_i \cdot L_i + n \qquad (6.1)\end{aligned}$$

because $m \le n$ and $p_{\max} \le 1$. Observing, that $s_i \cdot L_i$ equals the total processing requirement on machine i, it follows that $\sum_{i\in M} s_i \cdot L_i = \sum_{j\in J} p_j = Q$. Dividing Inequality (6.1) by the sum of the machine speeds, we obtain the following upper bound on the makespan of any jump optimal schedule σ:

$$C_{\max}(\sigma) \le \frac{Q}{\sum_{i\in M} s_i} + \frac{n}{\sum_{i\in M} s_i}\,.$$

Using the well-known bound $C^\star_{\max} \ge Q/\sum_{i\in M} s_i$, which stems from the fact that the total processing requirement on any machine i in any optimal schedule is at most $s_i \cdot C^\star_{\max}$, we obtain

$$C_{\max}(\sigma) \le \frac{Q}{\sum_{i\in M} s_i} + \frac{n}{\sum_{i\in M} s_i} \le \left(1 + \frac{n}{Q}\right) \cdot C^\star_{\max}\,.$$

Hence,

$$\max_{\sigma\in\text{Jump}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} \le 1 + \frac{n}{Q}\,. \qquad (6.2)$$

We will see that, with high probability, Q is in the order of $\Omega(n/\phi)$ yielding an approximation factor of $O(\phi)$ in this case. In the other case, we only get a worst-case approximation factor of $O(\sqrt{n})$ from Proposition 6.2.1. However, as the latter case is sufficiently unlikely, we obtain the desired bound of $O(\phi)$ for the smoothed performance guarantee of the jump algorithm.

Let $\mathcal{F}$ denote the failure event that $Q \le \big(n - \sqrt{n \ln n}\big)/(2\phi)$. We define X_j to be independent random variables drawn uniformly from $[0, 1/\phi]$ for all jobd $j \in J$. Then $\mathbf{Pr}[p_j \ge a] \ge \mathbf{Pr}[X_j \ge a]$ for any $a \in \mathbb{R}$. This is due to the fact that all processing requirements p_j are drawn according to probability densities $f_j\colon [0,1] \to [0,\phi]$. Let $X = \sum_{j\in J} X_j$. Observe, that the expected value of X is $n/(2\phi)$. Then it follows that $\mathbf{Pr}[Q \ge a] \ge \mathbf{Pr}[X \ge a]$ for any $a \in \mathbb{R}$. Hence,

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{F}\right] &= \mathbf{Pr}\left[Q \le \frac{n - \sqrt{n \ln n}}{2\phi}\right] \le \mathbf{Pr}\left[X \le \frac{n - \sqrt{n \ln n}}{2\phi}\right] \\ &= \mathbf{Pr}\left[\mathbf{E}\left[X\right] - X \ge \frac{\sqrt{n \ln n}}{2\phi}\right] \le \exp\left(-\frac{\ln n}{2}\right) = \frac{1}{\sqrt{n}}\,, \qquad (6.3)\end{aligned}$$

where the last inequality follows from *Hoeffding's inequality* [Hoe63] (see Theorem 3.2). Now consider the random variable

$$Z = \begin{cases} 1/2 + \sqrt{n} & \text{if event } \mathcal{F} \text{ occurs}, \\ 1 + n/Q & \text{otherwise}, \end{cases}$$

and let $Y = \max_{\sigma \in \text{Jump}(I)} \big(C_{\max}(I,\sigma)/C^{\star}_{\max}(I)\big)$. Due to Proposition 6.2.1 and Inequality (6.2) we have $Y \le Z$. We denote by $\overline{\mathcal{F}}$ the complement of $\mathcal{F}$ and obtain

$$\begin{aligned} \mathop{\mathbf{E}}_{I\sim\mathcal{I}}[Y] &\le \mathop{\mathbf{E}}_{I\sim\mathcal{I}}[Z] \le \mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[Z|\overline{\mathcal{F}}\right] + \mathop{\mathbf{E}}_{I\sim\mathcal{I}}[Z|\mathcal{F}] \cdot \mathop{\mathbf{Pr}}_{I\sim\mathcal{I}}[\mathcal{F}] \\ &\le \left(1 + \frac{2\phi n}{n - \sqrt{n \ln n}}\right) + \frac{1/2 + \sqrt{n}}{\sqrt{n}} \\ &\le 2.5 + \frac{2\phi}{1 - \sqrt{\ln(n)/n}} < 2.5 + 5.1\phi\,. \end{aligned}$$

For the third inequality, we used Inequality (6.3) and $Q > (n - \sqrt{n \ln n})/(2\phi)$ if event $\mathcal{F}$ does not hold. The last inequality holds since

$$\max_{n\in\mathbb{N}} \frac{2}{1 - \sqrt{\ln(n)/n}} < 5.1\,,$$

where the maximum is attained for $n = 3$. This is true since $\ln(x)/x$ takes its unique locale maximum at $x = e$. Thus,

$$\max_{n\in\mathbb{N}} \frac{2}{1 - \sqrt{\ln(n)/n}} = \max\left\{\frac{2}{1 - \sqrt{\ln(2)/2}}, \frac{2}{1 - \sqrt{\ln(3)/3}}\right\} < \max\{4.9, 5.1\} = 5.1\,. \quad \square$$

Corollary 6.2.3. *The average-case performance guarantee of the worst jump optimal schedule for scheduling with unrestricted and related machines is* $\Theta(1)$.

Next, we show that the upper bound on the smoothed performance guarantee provided in Theorem 6.2.2 is tight up to constant factor.

Theorem 6.2.4. *There is a class of ϕ-smooth instances $\mathcal{I}$ with unrestricted and related machines such that*

$$\mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\text{Jump}(I)} \frac{C_{\max}(I,\sigma)}{C^{\star}_{\max}(I)}\right] = \Omega(\phi)\,.$$

Proof. It suffices to consider $\phi > 2$. For such values of ϕ we construct a ϕ-smooth instance $\mathcal{I}$ with $n = m = \lceil 4\phi^2 + 1\rceil$ jobs and machines. Note, that due to Proposition 6.2.1 we need $\Omega(\phi^2)$ jobs and machines in order to construct an instance with a smoothed performance guarantee of $\Omega(\phi)$. Let

$$s_1 = \frac{n-1}{4\phi} \ge \phi > 2 \quad \text{and} \quad s_2 = \ldots = s_n = 1\,.$$

We assume that the processing requirement p_1 is chosen uniformly from the interval $[1-1/\phi, 1]$ while the processing requirements of all other jobs are chosen uniformly from the interval $[0, 1/\phi]$. As we only consider the case $\phi > 2$, job 1 is larger than all the other jobs. In an optimal schedule, job 1 is scheduled on machine 1, and all other machines process exactly one job (see Figure 6.1). Hence,

$$C^\star_{\max} = \max\left\{\frac{p_1}{s_1}, p_2, \dots, p_n\right\} \le \max\left\{\frac{1}{s_1}, \frac{1}{\phi}\right\} = \frac{1}{\phi}\,.$$

We show that, with high probability, there exists a jump optimal schedule σ for which $C_{\max}(\sigma) > 1 - 1/\phi$. In order to find such a schedule σ, we first schedule job 1 on machine 2. Then we consider the remaining jobs one after another and assign unassigned jobs to machine 1 until either $L_1 \in \big[L_2 - 1/(\phi s_1), L_2\big)$, which is the interesting case, or until all jobs are scheduled. Any job that remains unscheduled is then exclusively assigned to one empty machine. Let $\mathcal{E}$ denote the event that $Q_2 := \sum_{j=2}^n p_j \ge s_1$. Note that $\mathbf{E}[Q_2] = (n-1)/(2\phi) = 2s_1$. We will see that event $\mathcal{E}$ holds with high probability with respect to ϕ.

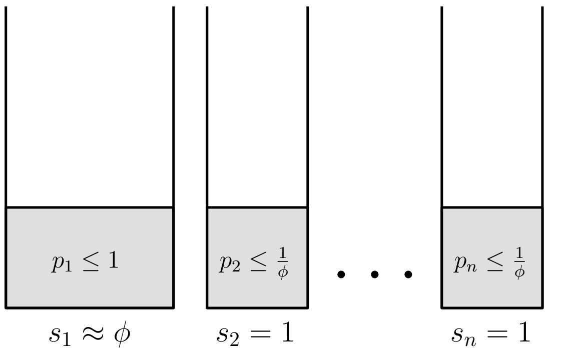
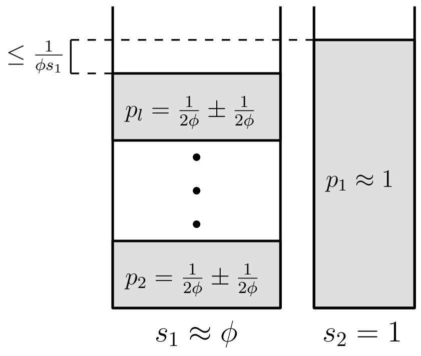


Figure 6.1: Optimal schedule



$\le \frac{1}{\phi s_1}$
$p_l = \frac{1}{2\phi} \pm \frac{1}{2\phi}$
$p_2 = \frac{1}{2\phi} \pm \frac{1}{2\phi}$
$p_1 \approx 1$
$s_1 \approx \phi$
$s_2 = 1$

Figure 6.2: Machines 1 and 2 of schedule σ if event $\mathcal{E}$ occurs

Consider the case that event $\mathcal{E}$ occurs. Then schedule σ is such that $L_1 \in \big[L_2 - 1/(\phi s_1), L_2\big)$ since $Q_2/s_1 \ge 1 \ge p_1 = L_2$ and $p_j \le 1/\phi$ for all jobs $j = 2, \dots, n$ (see Figure 6.2). Now, we argue that schedule σ is jump optimal. First observe that machine 2 defines the makespan since $L_2 > \max\{L_1, p_2/1, \dots, p_n/1\}$. Job 1, which is the only job assigned to that machine, cannot jump to a machine $i > 2$ because these have the same speed as machine 2. Furthermore, it cannot jump to machine 1 because

$$L_1 + \frac{p_1}{s_1} \ge L_2 - \frac{1}{\phi s_1} + \frac{1 - 1/\phi}{s_1} = L_2 + \frac{1 - 2/\phi}{s_1} > L_2$$

as $\phi > 2$. Hence, σ is a jump optimal schedule with

$$\frac{C_{\max}(\sigma)}{C^\star_{\max}} \ge \frac{L_2}{1/\phi} = \frac{p_1}{1/\phi} \ge \frac{1 - 1/\phi}{1/\phi} = \phi - 1\,. \tag{6.4}$$

It remains to determine the probability of event $\mathcal{E}$. Recalling that $\mathbf{E}[Q_2] = 2s_1$, $s_1 = (n-1)/(4\phi)$, and $n \geq 4\phi^2 + 1$, this probability can be bounded by applying Hoeffding's inequality [Hoe63] (see Theorem 3.2) as follows:

$$\begin{aligned}\mathbf{Pr}\left[\overline{\mathcal{E}}\right] &= \mathbf{Pr}\left[Q_2 < s_1\right] = \mathbf{Pr}\left[\mathbf{E}\left[Q_2\right] - Q_2 > s_1\right] \leq \exp\left(\frac{-2s_1^2}{(n-1)/\phi^2}\right) \\ &= \exp\left(\frac{-2\cdot\left((n-1)/(4\phi)\right)^2}{(n-1)/\phi^2}\right) = \exp\left(-\frac{n-1}{8}\right) \leq \exp\left(-\frac{\phi^2}{2}\right).\end{aligned}$$

Let $X = \max_{\sigma\in\mathrm{Jump}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)}$. By applying Inequality (6.4), the smoothed performance guarantee of the jump algorithm can be bounded from below as follows:

$$\begin{aligned}\mathop{\mathbf{E}}_{I\sim\mathcal{I}}[X] &\geq \mathop{\mathbf{E}}_{I\sim\mathcal{I}}[X \mid \mathcal{E}] \cdot \mathop{\mathbf{Pr}}_{I\sim\mathcal{I}}[\mathcal{E}] \geq (\phi-1)\cdot\left(1-\exp\left(-\frac{\phi^2}{2}\right)\right) \\ &= (\phi-1) - (\phi-1)\cdot\exp\left(-\frac{\phi^2}{2}\right) > \phi - 1.14 = \Omega(\phi),\end{aligned}$$

where the last inequality follows because $h(\phi) := (\phi-1)\cdot\exp(-\phi^2/2) < 0.14$ for $\phi > 2$: For such values ϕ we have $h'(\phi)\cdot\exp(\phi^2/2) = -\phi^2+\phi+1 \leq -2\phi+\phi+1 \leq -1$, that is, function h is monotonically decreasing for $\phi \geq 2$ and, hence, $\max_{\phi\geq 2} h(\phi) = h(2) = \exp(-2) < 0.14$. □

6.2.2 Upper Bounds for List Schedules and Lex-jump Optimal Schedules

Although the worst-case bound on the performance guarantee on unrestricted related machines for the list scheduling algorithm is slightly worse than the one for the lex-jump algorithm, we show that the smoothed performance guarantee of both algorithms is $O(\log\phi)$. In the next subsection, we show that this bound is asymptotically tight.

Theorem 6.2.5. *Let $\alpha > 0$ be an arbitrary real. For $\phi \geq 2$ and any ϕ-smooth instance $\mathcal{I}$ with unrestricted and related machines*

$$\mathop{\mathbf{Pr}}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\mathrm{Lex}(I)\cup\mathrm{List}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} \geq \alpha\right] \leq \left(\frac{32\phi}{2^{\alpha/6}}\right)^{n/2}$$

and

$$\mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\mathrm{Lex}(I)\cup\mathrm{List}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)}\right] \leq 18\log_2\phi + 30 = O(\log\phi).$$

Note that the assumption $\phi \geq 2$ in Theorem 6.2.5 is no real restriction as for $\phi \in [1,2)$ any ϕ-smooth instance is a 2-smooth instance. Hence, for these values we can apply all bounds from Theorem 6.2.5 when substituting ϕ by 2. In particular, the expected value is a constant.

Corollary 6.2.6. *The average-case performance guarantee of the worst lex-jump optimal schedule and the worst list schedule for scheduling with unrestricted related machines is a constant.*

In the remainder of this section, we will use the following notation (see also Appendix 6.4). Let $J_{i,j}(\sigma)$ denote the set of all jobs that are scheduled on machine i and have index at most j, that is, $J_{i,j}(\sigma) = J_i(\sigma) \cap \{1, \ldots, j\}$. If σ is clear from the context, then we just write $J_{i,j}$. We start with observing an essential property that both lex-jump optimal schedules and list schedules have in common.

Definition 6.2.7. We call a schedule σ on machines $1, \ldots, m$ with speeds $s_1, \ldots, s_m$ a *near list schedule*, if we can index the jobs in such a way that

$$L_{i'} + \frac{p_j}{s_{i'}} \geq L_i - \sum_{\ell \in J_{i,j-1}(\sigma)} \frac{p_\ell}{s_i} \tag{6.5}$$

for all machines $i' \neq i$ and all jobs $j \in J_i(\sigma)$. With $\mathrm{NL}(I)$ we denote the set of all near list schedules for instance I.

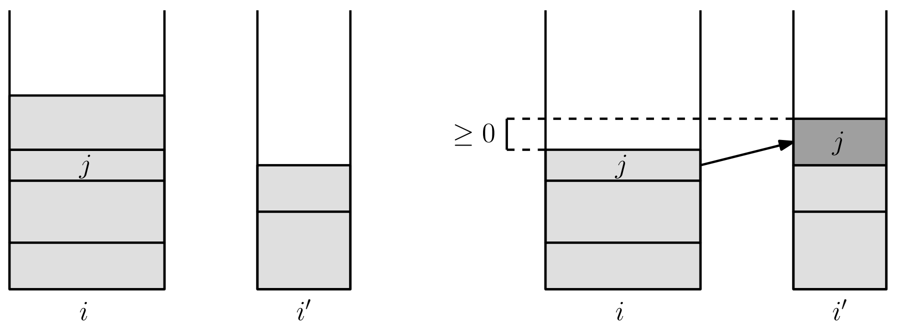
Inequality (6.5) can be interpreted as follows. Assume that the jobs are already indexed correctly and imagine that on each machine the jobs form a stack, ordered from top to bottom ascendingly according to their index. Now, consider an arbitrary job j on machine i (see Figure 6.3a). Inequality (6.5) states that the completion time of job j after removing all jobs above j is minimized on machine i in case only job j is allowed to move (see Figure 6.3b). In particular, the job j on a machine i with the smallest index does not benefit from changing to any other machine because for this job $J_{i,j-1}(\sigma) = \emptyset$ and, hence,

$$L_{i'} + \frac{p_j}{s_{i'}} \geq L_i - \sum_{\ell \in J_{i,j-1}(\sigma)} \frac{p_\ell}{s_i} = L_i\,.$$

Lemma 6.2.8. *For any instance I the relation* $\mathrm{Lex}(I) \cup \mathrm{List}(I) \subseteq \mathrm{NL}(I)$ *holds.*

Note that in general neither $\mathrm{Lex}(I) \subseteq \mathrm{List}(I)$ nor $\mathrm{List}(I) \subseteq \mathrm{Lex}(I)$ holds (see Figure 6.4). Moreover, there also exist near list schedules that are neither in $\mathrm{Lex}(I)$ nor in $\mathrm{List}(I)$ (see Figure 6.4c), that is, near list schedules are a non-trivial generalization of both lex-jump optimal schedules and list schedules.

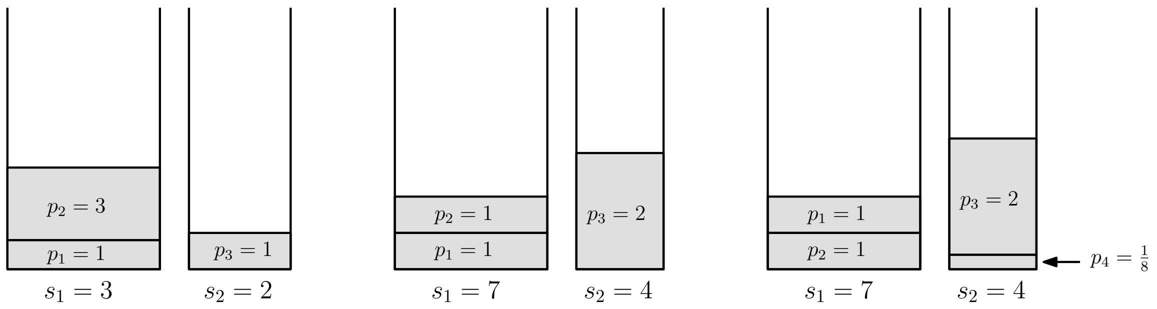
Proof of Lemma 6.2.8. For any schedule $\sigma \in \mathrm{Lex}(I)$, we can index the jobs arbitrarily and, by definition, even the stronger inequality $L_{i'} + p_j/s_{i'} \geq L_i$ holds. For $\sigma \in \mathrm{List}(I)$ we can index the jobs in reverse order in which they appear in the list that was used for list scheduling. That is, the job that is scheduled first is labelled as n, whereas the last job that is scheduled is labelled as 1. Consider an arbitrary job $j \in J_i(\sigma)$ and a machine $i' \neq i$. Let $L'_i, L'_{i'}$ and $L_i, L_{i'}$ denote the loads of machines i and i' before assigning



(a) Jobs on machine i, including job j, visualized as a stack

(b) Job j does not benefit from jumping to machine i'

Figure 6.3: Interpretation of Inequality (6.5)



(a) A list schedule which is not lex-jump optimal

(b) A lex-jump optimal schedule which is no list schedule

(c) A near list schedule which is neither lex-jump optimal nor a list schedule

Figure 6.4: Relationship between Lex(I), List(I), and NL(I)

job j to machine i and the loads of i and i' in the final schedule, respectively. Then $L'_i + p_j/s_i \leq L'_{i'} + p_j/s_{i'}$ because j is assigned to machine i according to list scheduling. Since $L_i = L'_i + \sum_{\ell \in J_{i,j}} p_\ell/s_i$ and $L_{i'} \geq L'_{i'}$, this implies

$$L_{i'} + \frac{p_j}{s_{i'}} \geq L'_{i'} + \frac{p_j}{s_{i'}} \geq L'_i + \frac{p_j}{s_i} = L_i - \sum_{\ell \in J_{i,j-1}} \frac{p_\ell}{s_i}\,. \qquad \square$$

In the remainder, we fix an instance I and consider an arbitrary schedule $\sigma \in \mathrm{NL}(I)$ with appropriate indices of the jobs for which Inequality (6.5) holds. To prove Theorem 6.2.5, we show that in case the ratio of $C_{\max}(I, \sigma)$ over $C^\star_{\max}(I)$ is large, then instance I needs to have many very small jobs (see Corollary 6.2.17). This holds even when the instance I is deterministically picked by some adversary. This observation allows us to prove the main theorem of this subsection by showing that for any ϕ-smooth instance, there are only “few” small jobs in expectation. The latter implies that a large ratio only happens with (exponentially) small probability.

In our proofs, we adopt some of the notation also used by Czumaj and Vöcking [CV07] (see also Appendix 6.4). Given a schedule σ, we set $c = \lfloor C_{\max}(\sigma)/C^{\star}_{\max} \rfloor - 1$. Recall that the machines are ordered such that $s_1 \geq \ldots \geq s_m$. For any integer $k \leq c$ let $H_k = \{1, \ldots, i_k\}$ for

$$i_k = \max \{ i \in M : L_{i'} \geq k \cdot C^{\star}_{\max} \text{ for all } i' \leq i \} \; .$$

That is, $i_k + 1$ is the first machine (the one with the smallest index) whose load is less than $k \cdot C^{\star}_{\max}$, if such a machine exists, and $m + 1$ otherwise. Note that $i_k = m$ for all $k \leq 0$ and, hence, $H_k = M$ for such k (see Figure 6.5). Further, define $R_k = H_k \setminus H_{k+1}$ for all $k \in \{0, \ldots, c-1\}$ and $R_c = H_c$. That is, $(R_c, R_{c-1}, \ldots, R_0)$ is a partition of the set M of machines and $R_{-1} = R_{-2} = \ldots = \emptyset$. Note that this classification always refers to schedule σ even if additionally other schedules are considered.

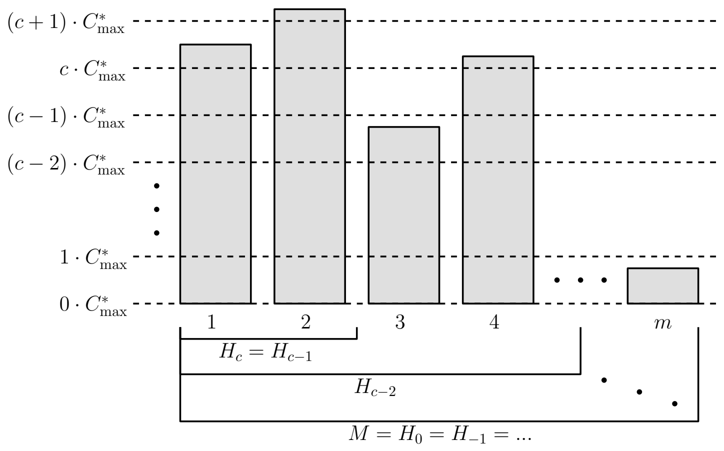


Figure 6.5: Machine classification by Czumaj and Vöcking

We can summarize the relevant properties of this classification as follows:

Property 1. For each machine $i \in H_k$, $L_i \geq k \cdot C^{\star}_{\max}$.

Property 2. Machine $i_k + 1$, if it exists, is the first machine in $M \setminus H_k$, that is, the machine with the least index, and, hence, a fastest machine in $M \setminus H_k$.

Property 3. $L_{i_k+1} < k \cdot C^{\star}_{\max}$ for all $k \in \{1, \ldots, c\}$, and $L_1 < (c+2) \cdot C^{\star}_{\max}$.

As mentioned before, we need to show that there are many small jobs. To do so, we will show that the speeds of the machines in low classes, that is, R_0 and R_1, are exponentially small with respect to the machines in the highest class R_c (Lemma 6.2.14) and that the machines in low classes have to process a high volume (Lemma 6.2.13). We start by showing that the highest class is nonempty.

Lemma 6.2.9. *Machine* 1 *is in class R_c.*

Proof. Let i be a critical machine, that is, a machine that defines the makespan of the schedule σ. If $i = 1$, then we obtain $L_1/C^\star_{\max} = C_{\max}(\sigma)/C^\star_{\max} > c$. Otherwise we apply Inequality (6.5) for the job $j = \min\{\ell \in J_i\}$ with the smallest index on machine i and for machine 1. This yields $L_1 + p_j/s_1 \geq L_i$. Hence,

$$\frac{L_1}{C^\star_{\max}} \geq \frac{L_i}{C^\star_{\max}} - \frac{\frac{p_j}{s_1}}{C^\star_{\max}} \geq \frac{C_{\max}(\sigma)}{C^\star_{\max}} - 1 \geq c\,,$$

where the second inequality is due to the fact that any job can contribute at most $C^\star_{\max}$ to the makespan of a fastest machine. □

Let t and k be integers satisfying $0 \leq t \leq k \leq c$. Several times we will consider the first jobs on some machine $i \in H_k$ which contribute at least $t \cdot C^\star_{\max}$ to the load of machine i. We denote the set of these jobs by $J_{i,\geq t}$. Formally, $J_{i,\geq t} = J_{i,j_i^t}$ for $j_i^t = \min\{j : \sum_{\ell \in J_{i,j}} p_\ell/s_i \geq t \cdot C^\star_{\max}\}$. Consequently, a job $j \in J_i$ belongs to $J_{i,\geq t}$ if and only if $\sum_{\ell \in J_{i,j-1}} p_\ell/s_i < t \cdot C^\star_{\max}$. Using this notation, Lemma 6.2.10 and Corollary 6.2.11 restrict the machines on which a job in $J_{i,\geq t}$ can be scheduled in an optimal schedule.

Lemma 6.2.10. *Let $k_1 > k_2$ and $t \leq k_1$ be positive integers, let $i_1 \in H_{k_1}$ and $i_2 \in M \setminus H_{k_2}$ be machines in H_{k_1} and not in H_{k_2}, respectively, and let $j \in J_{i_1,\geq t}$ be a job on machine i_1. Then the load job j would contribute to machine i_2 is bounded from below by $p_j/s_{i_2} > (k_1 - k_2 - t) \cdot C^\star_{\max}$ (see Figure 6.6).*

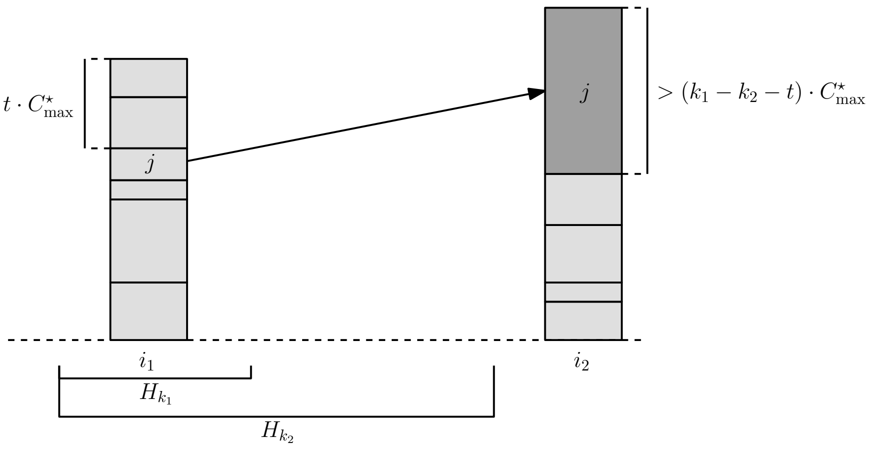

$< t \cdot C^\star_{\max}$
j
j
$> (k_1 - k_2 - t) \cdot C^\star_{\max}$
i_1
i_2
H_{k_1}
H_{k_2}

Figure 6.6: Main property of near list schedules

Proof. We apply Inequality (6.5) for machine i_1, for the first machine i_2' that does not belong to H_{k_2}, and for job j to obtain $L_{i_2'} + p_j/s_{i_2'} \geq L_{i_1} - \sum_{\ell \in J_{i_1,j-1}} p_\ell/s_{i_1}$, which implies $p_j/s_{i_2'} \geq L_{i_1} - L_{i_2'} - \sum_{\ell \in J_{i_1,j-1}} p_\ell/s_{i_1}$. By the choice of the machines i_1 and i_2' and Properties 1 and 3 we obtain $L_{i_1} \geq k_1 \cdot C^\star_{\max}$ and $L_{i_2'} < k_2 \cdot C^\star_{\max}$. Furthermore, $j \in J_{i_1,\geq t}$ yields $\sum_{\ell \in J_{i_1,j-1}} p_\ell/s_{i_1} < t \cdot C^\star_{\max}$. Hence, $p_j/s_{i_2'} > (k_1 - k_2 - t) \cdot C^\star_{\max}$. The claim follows since $s_{i_2'} \geq s_{i_2}$. □

Corollary 6.2.11. *Let $i \in H_k$ be an arbitrary machine and let $t \in \{1, \ldots, k\}$ be an integer. Then in any optimal schedule any job $j \in J_{i,\geq t}$ is assigned to machines from H_{k-t-1}.*

Proof. Assume, for contradiction, that there is a job $j \in J_{i,\geq t}$ that is assigned to a machine $i' \in M \setminus H_{k-t-1}$ by an optimal schedule. In accordance with Lemma 6.2.10 this job causes a load of more than $(k - (k-t-1) - t) \cdot C^\star_{\max} = C^\star_{\max}$ on this machine contradicting the assumption that the considered schedule is optimal. □

Czumaj and Vöcking [CV07] showed that in a lex-jump optimal schedule the speeds of any two machines which are at least two classes apart differ by a factor of at least 2. Aspnes et al. [AAF$^+$97] showed a slightly weaker property for list schedules which carries over to near list schedules.

Lemma 6.2.12. *Let $k \in \{5, \ldots, c\}$ and assume $H_k \neq \emptyset$. The speed of any machine in class H_k is at least twice the speed of any machine in $M \setminus H_{k-4}$.*

Proof. We may assume that $M \setminus H_{k-4} \neq \emptyset$, since otherwise all machines have a load larger than $C^\star_{\max}$ as $H_k \neq \emptyset$. Let $i_0 \in H_k$ and $i_2 \in M \setminus H_{k-4}$ be arbitrary machines and consider the jobs from $\bigcup_{i \in H_k} J_{i,\geq 2}$. If we would assign only these jobs to machines in H_k, then there would be a machine with load at least $2 \cdot C^\star_{\max}$. Consequently, in an optimal schedule at least one job in $\bigcup_{i' \in H_k} J_{i',\geq 2}$ is assigned to some machine $i^\star \in M \setminus H_k$, say job $j \in J_{i_1,\geq 2}$. Since job j contributes at most $C^\star_{\max}$ to the load of machine $i^\star$ in this optimal schedule, this implies $p_j/s_{i^\star} \leq C^\star_{\max}$ and, hence, $p_j/s_{i_0} \leq C^\star_{\max}$ as $s_{i_0} \geq s_{i^\star}$. Due to Lemma 6.2.10, the load that would be contributed by job j on machine i_2 is bounded by $p_j/s_{i_2} > (k - (k-4) - 2) \cdot C^\star_{\max} = 2 \cdot C^\star_{\max}$. The inequality $p_j/s_{i_0} \leq C^\star_{\max}$, which we observed before, yields $s_{i_0} \geq 2 \cdot s_{i_2}$ as claimed in the lemma. □

We want to show that machines in low classes, that is, machines in $R_0 \cup R_1$, have exponentially small speeds (with respect to c) compared to the speeds of the machines in a high class, that is, those in R_c. Lemma 6.2.12 already implies that the machine speeds would double every five classes if no class R_k was empty. Although there might be empty classes R_k, we show that there are not too many of them. To be a bit more precise, we show that the union of two neighbored classes is not empty. This is done in Lemma 6.2.14 which follows from the next lemma.

The machines $i \in H_k$, $k \geq 2$, are overloaded compared to an optimal schedule, even if we just consider the first few jobs $j \in J_{i,\geq t}$ on them (where $t \geq 2$). On the other hand, in Corollary 6.2.11 we showed that in any optimal schedule these jobs are not assigned to machines in much lower classes, that is, to machines from $M \setminus H_{k-t-1}$. Consequently, in any optimal schedule the machines in $H_{k-t-1} \setminus H_k$ consume the current overload of H_k.

Lemma 6.2.13. *Let $t \leq k$ be positive integers. In any optimal schedule the total processing requirement on all machines in $H_{k-t-1} \setminus H_k$ is at least $\sum_{r=k}^{c} \sum_{i \in R_r} (t+r-k-1) \cdot s_i \cdot C^\star_{\max}$.*

Note that Lemma 6.2.13 also holds for the case $t = k$ where $H_{k-t-1} = H_{-1} = M$.

Proof. Applying Corollary 6.2.11 with $h(r) := t + (r-k)$ for arbitrary integers $r \in \{k, \ldots, c\}$ yields that in any optimal schedule $\sigma^\star$ all jobs in $\bigcup_{r=k}^{c} \bigcup_{i \in R_r} J_{i,\geq h(r)}$ are assigned to machines in H_{k-t-1} as $r - h(r) - 1 = k - t - 1$ for any index r. Furthermore, in $\sigma^\star$ the processing requirement on any machine $i \in H_k$ is at most $s_i \cdot C^\star_{\max}$, that is, the machines in $H_{k-t-1} \setminus H_k$ must consume the remainder. Hence, these machines must process jobs with total processing requirement at least

$$\sum_{r=k}^{c} \sum_{i \in R_r} \sum_{\ell \in J_{i,\geq h(r)}} p_\ell - \sum_{r=k}^{c} \sum_{i \in R_r} s_i \cdot C^\star_{\max} \geq \sum_{r=k}^{c} \sum_{i \in R_r} (h(r) - 1) \cdot s_i \cdot C^\star_{\max} \,.$$

This yields the claimed bound as $h(r) - 1 = t + r - k - 1$. □

Some machine classes R_k might be empty. Now we show that this cannot be the case for two consecutive classes.

Lemma 6.2.14. *$H_{k-2} \setminus H_k \neq \emptyset$ for any $k \in \{1, \ldots, c-1\}$.*

Proof. Let i' be a slowest machine in H_k. In any optimal schedule $\sigma^\star$ the processing requirement on any machine $i \in H_{k-2} \setminus H_k$ is at most $s_i \cdot C^\star_{\max} \leq s_{i'} \cdot C^\star_{\max}$. Applying Lemma 6.2.13 for $t = 1$ implies

$$|H_{k-2} \setminus H_k| \cdot s_{i'} \cdot C^\star_{\max} \geq \sum_{r=k}^{c} \sum_{i \in R_r} (r-k) \cdot s_i \cdot C^\star_{\max} \geq \sum_{r=k}^{c} (r-k) \cdot s_{i'} \cdot C^\star_{\max} \cdot |R_r| \,.$$

It follows that $|H_{k-2} \setminus H_k| \geq \sum_{r=k}^{c} (r-k) \cdot |R_r| \geq (c-k) \cdot |R_c| \geq 1$ since $k < c$ and since $R_c \neq \emptyset$ due to Lemma 6.2.9. □

We can now show that machine speeds double every six classes. To be more formal:

Lemma 6.2.15. *Let $0 \leq k_2 \leq k_1 \leq c$ be integers, let i_1 be any machine of R_{k_1} and let $i_2 \in R_{k_2}$. Then $s_{i_1} \geq s_{i_2} \cdot 2^{\lfloor \Delta/6 \rfloor}$ where $\Delta = k_1 - k_2$.*

Proof. We prove the claim by induction. For $\Delta \in \{0, \dots, 5\}$, the claim trivially holds as $s_{i_1} \geq s_{i_2}$. Assume that the claim holds up to some integer $\Delta^\star \geq 5$. We show that it is also true for $\Delta = \Delta^\star + 1 \geq 6$. Note that for such Δ we have $k_1 \geq 6$. According to Lemma 6.2.14 the class $H_{k_1-6} \setminus H_{k_1-4} \subseteq M \setminus H_{k_1-4}$ contains at least one machine. Let i' be the fastest machine in $H_{k_1-6} \setminus H_{k_1-4}$. Then $s_{i'} \geq s_{i_2}$. Lemma 6.2.12 and the induction hypothesis imply $s_{i_1} \geq 2s_{i'}$ and $s_{i'} \geq s_{i_2} \cdot 2^{\lfloor (\Delta-6)/6 \rfloor}$, respectively. Hence, $s_{i_1} \geq s_{i_2} \cdot 2^{\lfloor \Delta/6 \rfloor}$. □

Since the machines in low classes are exponentially slower than the machines in high classes (with respect to c) and as their aggregated total processing requirement in an optimal schedule is large (Lemma 6.2.13), it follows that many jobs have processing requirements exponentially small in c.

Lemma 6.2.16. *Let $i \in M \setminus H_2$ be an arbitrary machine. Then each job j assigned to machine i by an optimal schedule has processing requirement at most $p_j \leq 2^{-c/6+2}$.*

Proof. For $c \leq 12$ the claim is true since we rescale all processing requirements to be at most 1. Assume $c \geq 13$. Consider an optimal schedule $\sigma^\star$ and let j be a job processed on a machine $i \in M \setminus H_2 = R_1 \cup R_0$ according to $\sigma^\star$. Note that $M \setminus H_2 \neq \emptyset$ due to Lemma 6.2.14. Then $p_j/s_i \leq C^\star_{\max}$, that is,

$$p_j \leq s_i \cdot C^\star_{\max} \,. \tag{6.6}$$

To bound $s_i \cdot C^\star_{\max}$, consider the job $j' = \min\{\ell \in J_1(\sigma)\}$ with the smallest index on machine 1 of schedule σ and consider the first machine $i' \in H_{c-3} \setminus H_{c-1} = R_{c-3} \cup R_{c-2}$ which exists due to Lemma 6.2.14 and $c \geq 13$. Applying Inequality (6.5), we obtain $L_{i'}(\sigma) + p_{j'}/s_{i'} \geq L_1(\sigma)$, that is, $p_{j'} \geq s_{i'} \cdot (L_1(\sigma) - L_{i'}(\sigma))$. Since machine 1 belongs to H_c due to Lemma 6.2.9 and since machine i' is the first machine that does not belong to H_{c-1}, we have $L_1(\sigma) \geq c \cdot C^\star_{\max}$ and $L_{i'}(\sigma) < (c-1) \cdot C^\star_{\max}$, which implies $p_{j'} \geq s_{i'} \cdot C^\star_{\max}$. Lemma 6.2.15 yields $s_{i'} \geq s_i \cdot 2^{\lfloor (c-3-1)/6 \rfloor}$. Applying Inequality (6.6) and $p_{j'} \leq 1$ according to our input model we obtain

$$p_j \leq s_i \cdot C^\star_{\max} \leq s_{i'} \cdot C^\star_{\max} \cdot 2^{-\lfloor (c-4)/6 \rfloor} \leq p_{j'} \cdot 2^{-c/6+2} \leq 2^{-c/6+2} \,.$$ □

Corollary 6.2.17. *The processing requirement of at least $n/2$ jobs is at most $2^{-c/6+2}$.*

Proof. Lemma 6.2.13 for $k = t = 2$ implies that the total processing requirement of all jobs assigned to machines from $M \setminus H_2 = H_{-1} \setminus H_2$ according to $\sigma^\star$ is at least $\sum_{i \in H_2} s_i \cdot C^\star_{\max}$ which is an upper bound for the total processing requirement of all jobs assigned to machines in H_2 according to $\sigma^\star$. Since all jobs assigned to machines from $M \setminus H_2$ by an optimal schedule have processing requirement at most $2^{-c/6+2}$ due to Lemma 6.2.16, at least half of the jobs have processing requirement at most $2^{-c/6+2}$. □

Since having many so small jobs is unlikely when the processing requirements have been smoothed, it follows that the approximation factor, which is between $c+1$ and $c+2$, cannot be too high, yielding Theorem 6.2.5.

Proof of Theorem 6.2.5. If $C_{\max}(\sigma)/C^\star_{\max} \geq \alpha$, then at least $n/2$ jobs have processing requirement at most $2^{-\alpha/6+3}$ due to Corollary 6.2.17 and $c = \lfloor C_{\max}(\sigma)/C^\star_{\max} \rfloor - 1 \geq \alpha - 2$. The probability that one specific job is that small is bounded by $\phi \cdot 2^{-\alpha/6+3} = 8\phi \cdot 2^{-\alpha/6}$ in the smoothed input model. Hence, the probability that the processing requirement of at least $n/2$ jobs is at most $2^{-\alpha/6+3}$, is bounded from above by

$$\sum_{k \geq \frac{n}{2}} \binom{n}{k} \left(8\phi \cdot 2^{-\alpha/6}\right)^k \leq \sum_{k \geq \frac{n}{2}} \binom{n}{k} \left(8\phi \cdot 2^{-\alpha/6}\right)^{n/2}$$
$$\leq 2^n \cdot \left(8\phi \cdot 2^{-\alpha/6}\right)^{n/2} = \left(32\phi \cdot 2^{-\alpha/6}\right)^{n/2}.$$

This yields

$$\mathop{\mathbf{Pr}}_{I \sim \mathcal{I}} \left[\max_{\sigma \in \mathrm{NL}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} \geq \alpha \right] \leq \left(\frac{32\phi}{2^{\alpha/6}}\right)^{n/2}.$$

As for $n = 1$ any schedule $\sigma \in \mathrm{NL}(I)$ is optimal, we just consider the case $n \geq 2$. For $k \geq 1$ let $\alpha_k = 6k \log_2 \phi + 30$, that is, $2^{\alpha_k/6} = 32\phi^k$. If $\alpha \geq \alpha_k$, then we obtain

$$\mathop{\mathbf{Pr}}_{I \sim \mathcal{I}} \left[\max_{\sigma \in \mathrm{NL}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} \geq \alpha \right] \leq \mathop{\mathbf{Pr}}_{I \sim \mathcal{I}} \left[\max_{\sigma \in \mathrm{NL}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} \geq \alpha_k \right]$$
$$\leq \left(\phi^{1-k}\right)^{n/2} \leq \phi^{1-k} \leq 2^{1-k}$$

as $\phi \geq 2$. Since $\alpha_{k+1} - \alpha_k = 6 \log_2 \phi$, we obtain

$$\mathop{\mathbf{E}}_{I \sim \mathcal{I}} \left[\max_{\sigma \in \mathrm{NL}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} \right] = \int_0^\infty \mathop{\mathbf{Pr}}_{I \sim \mathcal{I}} \left[\max_{\sigma \in \mathrm{NL}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} \geq \alpha \right] \mathrm{d}\alpha$$
$$\leq \alpha_1 + \sum_{k=1}^\infty \int_{\alpha_k}^{\alpha_{k+1}} \mathop{\mathbf{Pr}}_{I \sim \mathcal{I}} \left[\max_{\sigma \in \mathrm{NL}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} \geq \alpha \right] \mathrm{d}\alpha$$
$$\leq \alpha_1 + 6 \log_2 \phi \cdot \sum_{k=1}^\infty 2^{1-k} = 18 \log_2 \phi + 30. \qquad \square$$

6.2.3 Lower Bounds for List Schedules and Lex-jump Optimal Schedules

In this subsection, we show that the upper bound for the smoothed performance guarantee of the lex-jump algorithm and the list scheduling algorithm, given in Theorem 6.2.5, is tight up to a constant factor. We provide a ϕ-smooth instance for which the worst lex-jump optimal schedule as well as the worst list schedule is worse than the optimal schedule by a factor of $\Omega(\log \phi)$, independent of the processing times' realizations.

Theorem 6.2.18. *There is a class of ϕ-smooth instances $\mathcal{I}$ with unrestricted and related machines such that, for any $I \in \mathcal{I}$,*

$$\max_{\sigma \in \mathrm{Lex}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} = \Omega(\log\phi) \quad \textit{and} \quad \max_{\sigma \in \mathrm{List}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} = \Omega(\log\phi)\,.$$

To prove this theorem, we present a ϕ-smooth instance. In Algorithm 4, we implicitely give a permutation of the jobs such that the list scheduling algorithm, when using this permutation, results in a schedule σ which we will show is also lex-jump optimal. The schedule σ resembles the worst-case example constructed by Czumaj and Vöcking [CV07]: Machines are partitioned into classes indexed by $0, 1, \ldots, r$. We will show that in σ, each machine in class i has a load of approximately i, whereas the optimal makespan is bounded by 3. Hence, we can lower bound the performance guarantee in the order of the number r of classes. For this construction we use $r = \Theta(\log\phi)$ classes instead of $\Theta(\log m/\log\log m)$ classes as Czumaj and Vöcking did.

As scaling of all processing requirements does not change the approximation ratio, for sake of simplicity we do not consider probability densities $f_j\colon [0,1] \to [0,\phi]$ but scaled densities $f'_j\colon [0,2^{r+1}] \to [0,\phi/2^{r+1}]$ for an appropriate integer r. This simplifies many terms.

Let $\phi \geq 4$ and consider an integer $r = \lfloor \log_4 \phi \rfloor \geq 1$, i.e., $\phi \geq 4^r = 2^{2r}$. The machines are partitioned into machine classes M_k for $k = 0, \ldots, r$, such that machine class M_k contains $r!/k!$ machines of speed 2^k. Also the jobs are partitioned into job classes J_ℓ for $\ell = 1, \ldots, r$ such that a job class J_ℓ contains $r!/(\ell-1)!$ jobs each having a processing requirement uniformly drawn from $\left[2^\ell, 2^\ell + 2^{r+1}/\phi\right) \subseteq (0, 2^{r+1})$. Note that the density of this instance is bounded by $\phi/2^{r+1}$ which is valid in the variant of our model that we use in this subsection. The permutation of the jobs is such that the list scheduling algorithm considers the jobs in the same order as Algorithm 4.

Algorithm 4

```
1: for $k = 1, \ldots, r$ do
2:     for $\ell = r, r-1, \ldots, k$ do
3:         Schedule $r!/\ell!$ arbitrary jobs of class $J_\ell$ according to list scheduling.
4:     end for
5: end for
```

Note that for any job class J_ℓ all $\ell \cdot r!/\ell! = r!/(\ell-1)!$ jobs have been scheduled. Let σ be the resulting schedule. First, we show a key property of σ.

Lemma 6.2.19. *For any index $\ell = 1, \ldots, r$ each machine in M_ℓ is assigned exactly ℓ jobs of job class J_ℓ and no other jobs. The machines in M_0 remain empty.*

Proof. Let $\sigma(k,\ell)$ denote the partial schedule after processing Line 3 of iteration (k,ℓ) of Algorithm 4. Within the $(k,\ell)^{th}$ iteration, we call a machine $i \in M_\ell$ *used* if a job of class J_ℓ has already been assigned to i during that iteration. Otherwise, we call machine i *unused*. We show the two claims below inductively and simultaneously. The lemma then follows straightforwardly from the second claim since the last iteration is (r, r).

Claim 1. *During iteration (k, ℓ), $r!/\ell!$ jobs of class J_ℓ are assigned to $r!/\ell!$ distinct machines (i.e., all machines) of class M_ℓ.*

Claim 2. *In the partial schedule $\sigma(k, \ell)$ each machine in class $M_{\ell'}$ is assigned*

$$k' = \begin{cases} k & : \ \ell' \geq \ell\,, \\ \min\{k-1, \ell'\} & : \ \ell' < \ell\,, \end{cases}$$

jobs of class $J_{\ell'}$ and no other jobs.

Figure 6.7 visualizes the partial schedule $\sigma(k, \ell)$. Machine i with speed $s_i = 2^i$ is a representative for all machines in class M_i. With L_i we refer to the current load of machine i and with L'_i to the load of machine i at the end of iteration (k, k), i.e., in the partial schedule $\sigma(k, k)$. In phase (k, ℓ), $r!/\ell!$ jobs of size roughly 2^ℓ are being assigned to the $r!/\ell!$ machines in M_ℓ. All machines in $M_{\ell'}$ for $\ell' > \ell$ just received a job of roughly size $2^{\ell'}$. All machines in $M_{\ell'}$ for $\ell' \in \{k, \dots, \ell-1\}$ will still receive a single job of size roughly $2^{\ell'}$ during iteration k of the outer loop. Figure 6.7 follows from the observations.

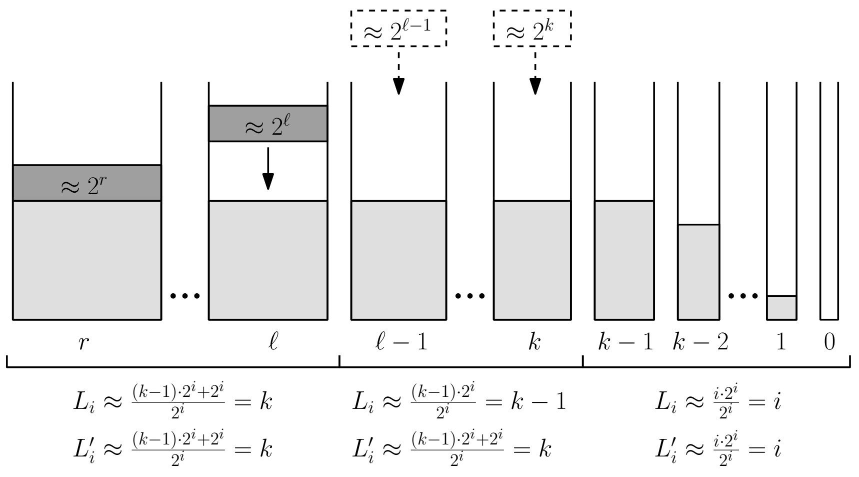

$\approx 2^{\ell-1}$
$\approx 2^k$
$\approx 2^r$
$\approx 2^\ell$
r
ℓ
$\ell-1$
k
$k-1$
$k-2$
1
0
$L_i \approx \frac{(k-1)\cdot 2^i + 2^i}{2^i} = k$
$L'_i \approx \frac{(k-1)\cdot 2^i + 2^i}{2^i} = k$
$L_i \approx \frac{(k-1)\cdot 2^i}{2^i} = k-1$
$L'_i \approx \frac{(k-1)\cdot 2^i + 2^i}{2^i} = k$
$L_i \approx \frac{i\cdot 2^i}{2^i} = i$
$L'_i \approx \frac{i\cdot 2^i}{2^i} = i$

Figure 6.7: The partial schedule $\sigma(k, \ell)$

First, we validate the claims for the first iteration $(1, r)$. As only $r!/r! = 1$ job of class J_r has to be scheduled and since all machines are still empty, the job will be scheduled on the fastest machine which is the single machine in M_r. Hence, both claims hold true for the first iteration. Now, consider an arbitrary iteration (k, ℓ) and assume both claims hold true for all previous iterations. Consider a job $j \in J_\ell$ which needs to be assigned to a machine during iteration (k, ℓ). We show that job j will always be assigned to an unused machine $i \in M_\ell$. To see this, first note that the previous iteration was either $(k, \ell+1)$ or $(k-1, k-1)$.

Let $i \in M_\ell$ be an unused machine. By the second claim, we know that this machine carries $k-1$ jobs of class J_ℓ. Consequently, we can upper bound its load by

$$L_i + \frac{p_j}{s_i} < \frac{k \cdot (2^\ell + 2^{r+1}/\phi)}{2^\ell} = k + \frac{k}{\phi} \cdot 2^{r+1-\ell} \leq k + \frac{\ell}{2^{2r}} \cdot 2^{r+1-\ell} \leq k + \frac{1}{2^r}\,,$$

where we used that $k \leq \ell$, $\phi \geq 2^{2r}$, and $\ell/2^\ell \leq 1/2$ for all integers $\ell \geq 1$.

Consider a machine machine h which is either used (in that case let $\ell' = \ell$) or in class $M_{\ell'}$ for some $\ell' \in \{\ell+1, \ldots, r\}$. By Claim 2, this machine carries k jobs of class $J_{\ell'}$ and thus

$$L_h + \frac{p_j}{s_h} \geq \frac{k \cdot 2^{\ell'} + 2^\ell}{2^{\ell'}} = k + 2^{\ell-\ell'} > k + \frac{1}{2^r} > L_i + \frac{p_j}{s_i}\,.$$

Finally, consider a machine $h \in M_{\ell'}$ for some $\ell' \in \{1, \ldots, \ell-1\}$. Again by Claim 2, it carries $\min\{k-1, \ell'\}$ jobs of class $J_{\ell'}$ and thus

$$\begin{aligned} L_h + \frac{p_j}{s_h} &\geq \frac{\min\{k-1, \ell'\} \cdot 2^{\ell'} + 2^\ell}{2^{\ell'}} = \min\{k-1, \ell'\} + 2^{\ell-\ell'} \\ &\geq (k - \max\{k-\ell', 1\}) + 2^{\max\{k-\ell', 1\}} \geq k+1 > L_i + \frac{p_j}{s_i}\,, \end{aligned}$$

where the second inequality follows from $\ell \geq \max\{k, \ell'+1\}$ and the third inequality follows from $2^i - i \geq 1$ for all positive integers i.

With this complete case analysis we have shown that job j will be assigned to an unused machine $i \in M_\ell$. We conclude that during iteration (k, ℓ), each of the $r!/\ell!$ jobs to be assigned will be assigned to an unused machine in M_ℓ. Note that $|M_\ell| = r!/\ell!$, and hence for each job there always exists such an unused machine. The first claim and the second claim follow immediately. □

Lemma 6.2.20. *Schedule σ is lex-jump optimal.*

Proof. It follows from Lemma 6.2.19 that the load of any machine $i \in M_\ell$ can be bounded by

$$\ell \leq L_i \leq \ell + \ell \cdot \frac{2^{r+1}}{2^\ell \phi} \leq \ell + \frac{\ell}{2^\ell} \cdot \frac{2^{r+1}}{2^{2r}} \leq \ell + \frac{1}{2} \cdot \frac{2^{r+1}}{2^{2r}} < \ell + 1\,.$$

If a job j assigned to machine $i \in M_\ell$ would jump to another machine $i' \in M_{\ell'}$, then

$$L_{i'} + \frac{p_j}{s_{i'}} \geq \ell' + \frac{2^\ell}{2^{\ell'}} = \ell - (\ell - \ell') + 2^{\ell-\ell'} \geq \ell + 1 > L_i\,,$$

where the last inequality follows from $2^k - k \geq 1$ for all integers k. Thus, any job would be worse off by jumping to another machine, and hence schedule σ is lex-jump optimal. □

We conclude this subsection by proving Theorem 6.2.18.

Proof of Theorem 6.2.18. We consider schedule σ constructed above which is both a list schedule and a lex-jump optimal schedule. By Lemma 6.2.19 the load of the single machine in M_r is at least r. Hence, $C_{\max}(\sigma) \geq r$. Now, consider a schedule σ' in which each machine in M_ℓ processes a single job from job class $J_{\ell+1}$, $\ell = 0, \dots, r-1$. The single machine in M_r remains empty. Then the load of any machine $i \in M_\ell$ with job j assigned to it is bounded as follows:

$$L_i = \frac{p_j}{s_i} \leq \frac{2^{\ell+1} + 2^{r+1}/\phi}{2^\ell} \leq 2 + \frac{2^{r+1}}{2^{2r}} = 2 + 2^{1-r} \leq 3\,.$$

Hence, $C^\star_{\max}(I) \leq C_{\max}(I, \sigma') \leq 3$ and $C_{\max}(\sigma)/C^\star_{\max}(I) \geq r/3 = \Omega(r) = \Omega(\log \phi)$. $\square$

6.3 Restricted Machines

In this section, we provide lower bound examples showing that the worst-case performance guarantees of the *jump algorithm* and the *lex-jump algorithm* for all environments with restricted machines are robust against random noise. Our lower bounds are in the order of the worst-case bounds and hold in particular for $\phi = 2$. In our lower bound constructions all processing requirements are chosen uniformly at random from intervals of length 1/2. This means that even with large perturbations the worst-case lower bounds still apply.

6.3.1 Jump Neighborhood on Restricted Machines

Rutten et al. [RRSV12] showed that the makespan of a jump optimal schedule is at most a factor of $1/2 + \sqrt{m - 3/4}$ away from the optimal makespan on restricted identical machines. On restricted related machines they showed that the makespan of a jump optimal schedule is not more than a factor of $1/2 + \sqrt{(m-1) \cdot s_{\max} + 1/4}$ away from the makespan of an optimal schedule, assuming that $s_{\min} = 1$. They provided two examples showing that the bound on identical machines is tight and the one on related machines is tight up to a constant factor. We show that even on ϕ-smooth instances these bounds are tight up to a constant factor.

As in [RRSV12], we construct an example with two job classes and three machine classes. The first machine class consists of only one machine and this machine is the slowest among all machines. The first class of jobs can only be scheduled on machines in the first two classes, whereas the jobs in the second class are allowed on all machines. To construct a bad example, we schedule all jobs in the first class on the slowest machine and use the jobs of the second class to fill the machines in the second machine class so that the schedule will be jump optimal, with high probability.

Theorem 6.3.1. *For every $\phi \geq 2$ there exists a class of ϕ-smooth instances $\mathcal{I}$ on restricted related machines such that*

$$\mathop{\mathbf{E}}_{I \sim \mathcal{I}} \left[\max_{\sigma \in \mathrm{Jump}(I)} \frac{C_{\max}(I, \sigma)}{C^{\star}_{\max}(I)} \right] = \Omega\left(\sqrt{m \cdot s_{\max}}\right),$$

assuming without loss of generality that $s_{\min} = 1$.

Proof. It suffices to show the theorem for $\phi = 2$ and $m \geq 3$. We set $s := s_{\max}/s_{\min} = s_{\max}$. Let $z > 2$ be an arbitrary integer, let

$$m' = m - 2 \geq 1\,, \quad k' = \sqrt{\frac{m'}{s}} \leq \sqrt{m'}\,, \quad \text{and} \quad k = \lceil k' \rceil\,.$$

In the remainder we assume that $\sqrt{m's} \geq 17$. This is possible because we only want to derive an asymptotic bound. We consider the following ϕ-smooth instance $\mathcal{I}$. The set M of machines is partitioned into three classes M_1, M_2, and M_3 such that

$$|M_1| = 1\,, \quad |M_2| = k\,, \quad \text{and} \quad |M_3| = m' - (k-1) > m' - k' \geq 0\,.$$

The machine in M_1 has speed 1, the machines in M_2 have speed $s' = \max\{1, s \cdot k'/k\} \in [1, s]$, and the machines in M_3 have speed s. Let the set J of jobs be partitioned into two subsets $\mathcal{J}_1$ and $\mathcal{J}_2$, consisting of

$$|\mathcal{J}_1| = \lfloor 2zsk' \rfloor \quad \text{and} \quad |\mathcal{J}_2| = \lceil 32zs \cdot (m' - k') \rceil \leq \lceil 32zs \cdot |M_3| \rceil$$

jobs whose processing requirements are independently and uniformly drawn from $[1/2, 1]$ and from $[0, 1/2]$, respectively. The jobs in $\mathcal{J}_1$ are only allowed to be scheduled on the machines in $M_1 \cup M_2$, whereas the jobs in $\mathcal{J}_2$ are allowed to be scheduled on any machine.

First, we construct a schedule σ' to bound the optimal makespan: Use the *list scheduling algorithm* to schedule all jobs in $\mathcal{J}_1$ on the machines in M_2, and all jobs in $\mathcal{J}_2$ on the machines in M_3. Figure 6.8 depicts schedule σ'. Machine i is a representative for all machines in class M_i.

Along the same lines as in [Gra66], it follows that for all machines $i \in M_2$

$$L_i \leq \frac{\frac{\sum_{j \in \mathcal{J}_1} p_j}{|M_2|} + \max_{j \in \mathcal{J}_1} p_j}{s'} \leq \frac{\frac{|\mathcal{J}_1| \cdot 1}{|M_2|} + 1}{s'} \leq \frac{\frac{2zsk'}{k} + 1}{s'} \leq \frac{\frac{2zsk'}{k}}{s \cdot \frac{k'}{k}} + \frac{1}{1} = 2z + 1\,.$$

Similarly, for all machines $i \in M_3$

$$L_i \leq \frac{\frac{\sum_{j \in \mathcal{J}_2} p_j}{|M_3|} + \max_{j \in \mathcal{J}_2} p_j}{s} \leq \frac{\frac{|\mathcal{J}_2| \cdot \frac{1}{2}}{|M_3|} + \frac{1}{2}}{s} \leq \frac{\frac{32zs \cdot |M_3|}{2 \cdot |M_3|} + 1}{s} \leq 16z + 1\,.$$

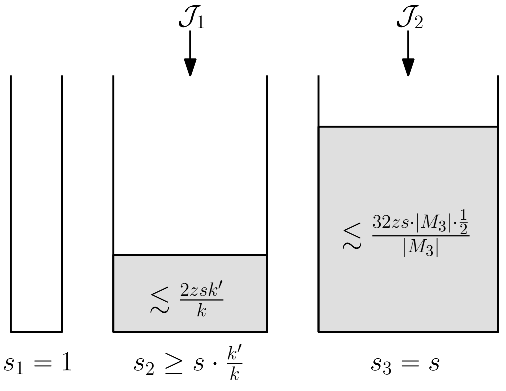

$\mathcal{J}_1$
$\mathcal{J}_2$
$\lesssim \frac{2zsk'}{k}$
$\lesssim \frac{32zs\cdot|M_3|\cdot\frac{1}{2}}{|M_3|}$
$s_1 = 1$
$s_2 \geq s \cdot \frac{k'}{k}$
$s_3 = s$

Figure 6.8: Schedule σ'

Consequently, $C^\star_{\max} \leq C_{\max}(\sigma') \leq 17z$. Before we proceed with constructing a 'bad' jump optimal schedule σ, we observe that

$$s' \leq 2s \cdot k'/k \tag{6.7}$$

due to $1 \leq (\sqrt{m'}+1)/k \leq 2\sqrt{m's}/k = 2s \cdot k'/k$. We construct a jump optimal schedule σ on the ϕ-smooth instance $\mathcal{I}$ whose makespan exceeds zsk' with high probability: Schedule all jobs in $\mathcal{J}_1$ on the single machine in M_1. Then $zsk' - 1 \leq L_1 \leq 2zsk'$. Next, start assigning jobs from $\mathcal{J}_2$ to the machines in M_2 according to the list scheduling algorithm with an arbitrary job permutation, until

(a) either $\mathcal{J}_2$ becomes empty, or until

(b) $L_i \in \big[L_1 - 1/(2s'), L_1\big)$ for all $i \in M_2$. If there remain unscheduled jobs in $\mathcal{J}_2$, then we assign them to the machines in M_3 using the list scheduling algorithm.

Let $Q = \sum_{j\in\mathcal{J}_2} p_j$ be the total processing requirement of all jobs from class $\mathcal{J}_2$ and let $\mathcal{E}$ denote the event that $Q > 4z(sk')^2 = 4zsm'$. Note, that

$$\mathbf{E}[Q] = \frac{|\mathcal{J}_2|}{4} \geq 8zs \cdot (m' - k') = 8zsm' \cdot \left(1 - \frac{k'}{m'}\right) = 8zsm' \cdot \left(1 - \frac{1}{\sqrt{m's}}\right) > 6zsm'$$

as $\sqrt{m's} \geq 17$ by our initial assumption. If event $\mathcal{E}$ occurs, then

$$\sum_{i\in M_2} s' \cdot L_1 \leq |M_2| \cdot \left(2s \cdot \frac{k'}{k}\right) \cdot 2zsk' = 4z(sk')^2 < Q$$

due to Inequality (6.7), i.e., the algorithm will end up in Case (b) because $p_j \leq 1/2$ for any job $j \in \mathcal{J}_2$. This shows that no machine $i \in M_2$ is critical. With the same argument as for the analysis of σ' we can show that the load of any machine $i \in M_3$ is bounded from above by $16z+1 < 17z-1 \leq z \cdot \sqrt{m' \cdot s} - 1 = zsk' - 1 \leq L_1$, i.e., the machine in M_1 is the unique critical machine. As each job on this machine has processing requirement at least 1/2 and due to the property of the loads of the machines in M_2 in Case (b), schedule σ is jump optimal and $C_{\max}(\sigma) = L_1 \geq zsk' - 1$.

It remains to determine the probability $\mathbf{Pr}[\mathcal{E}]$. Applying *Hoeffding's inequality* [Hoe63] (see Theorem 3.2), we obtain

$$\begin{aligned}\mathbf{Pr}\left[\overline{\mathcal{E}}\right] &= \mathbf{Pr}\left[Q \leq 4zsm'\right] \leq \mathbf{Pr}\left[Q - \mathbf{E}[Q] \leq -2zsm'\right] \\ &\leq \exp\left(-\frac{2 \cdot (2zsm')^2}{|\mathcal{J}_2| \cdot \left(\frac{1}{2}\right)^2}\right) \leq \exp\left(-\frac{32z^2s^2m'^2}{32zsm'+1}\right),\end{aligned}$$

which becomes arbitrarily close to 0 when z increases. Hence, for sufficiently large integers z

$$\begin{aligned}\mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\mathrm{Jump}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)}\right] &\geq \mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\mathrm{Jump}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)} \middle| \mathcal{E}\right] \cdot \mathop{\mathbf{Pr}}_{I\sim\mathcal{I}}\left[\mathcal{E}\right] \geq \frac{zsk'-1}{17z} \cdot \frac{17}{18} \\ &\geq \frac{\sqrt{(m-2)\cdot s_{\max}} - \frac{1}{z}}{18} = \Omega\left(\sqrt{m \cdot s_{\max}}\right). \qquad \square\end{aligned}$$

Corollary 6.3.2. *For every $\phi \geq 2$ there exists a class of ϕ-smooth instances $\mathcal{I}$ on restricted identical machines such that*

$$\mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\mathrm{Jump}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)}\right] = \Omega(\sqrt{m}).$$

Remark. In the proof of Theorem 6.3.1 we introduce an arbitrary integer z. We argue that there exists a sufficiently large value for z such that the desired result follows. Choosing an even larger value for z implies that the results above not only hold in expectation but also with high probability.

6.3.2 Lex-jump Optimal Schedules on Restricted Identical Machines

In this subsection, we show that for $\phi \geq 8$ there exist instances on which the approximation ratio of the lex-jump algorithm in the restricted setting is in the same order as the worst-case performance guarantee.

As in Section 6.2.3, we construct an instance with several job classes and machine classes and a bad lex-jump optimal schedule for which the loads of the machines are gradually decreasing with increasing machine class index. By setting the sets $\mathcal{M}_j$ of allowed machines equal to the union of only one or two machine classes and choosing to schedule the jobs on the *wrong* machines, we can enforce that jobs cannot leave the machine class on which they are scheduled in the lex-jump optimal solution, whereas the optimal makespan is still small.

Theorem 6.3.3. *For every $\phi \geq 8$ there exists a class of ϕ-smooth instances $\mathcal{I}$ on restricted identical machines such that*

$$\mathop{\mathbf{E}}_{I \sim \mathcal{I}} \left[\max_{\sigma \in \mathrm{Lex}(I)} \frac{C_{\max}(I, \sigma)}{C^{\star}_{\max}(I)} \right] = \Omega\left(\frac{\log m}{\log \log m} \right) .$$

First, we introduce the ϕ-smooth instance $\mathcal{I}$ for $\phi \geq 8$. Given an integer $k \geq 68$, consider the following recurrence formula:

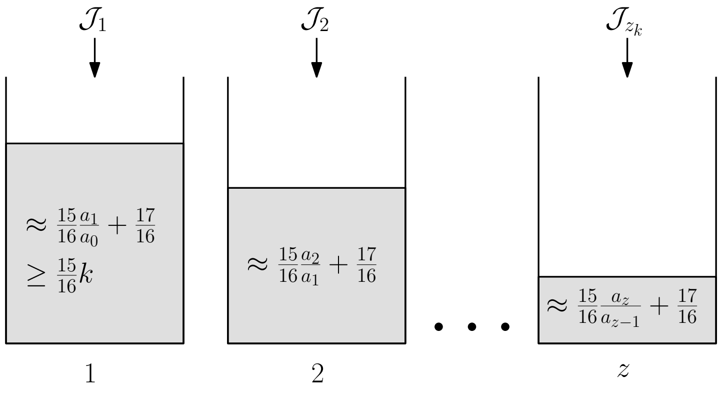
$$a_0 = k^2 , \quad a_1 = k^3 , \quad \text{and} \quad a_h = \left\lceil \left(\frac{a_{h-1}}{a_{h-2}} - \frac{7}{15} \right) \cdot a_{h-1} \right\rceil \ \text{ for } h \geq 2 .$$

The sequence $(a_h)_{h \geq 0}$ is chosen such that $a_h/a_{h-1} \approx a_{h-1}/a_{h-2} - 7/15$, that is, the ratio between two consecutive elements decreases by approximately $7/15$, starting with $a_1/a_0 = k$. Let z be the smallest integer h for which $a_h/a_{h-1} \leq 1$. Due to rounding issues the fraction a_h/a_{h-1} does not decrease exactly by $7/15$ each time. However, the deviation caused by rounding is at most $1/a_{h-1} \leq 1/k^2 \ll 1/15$ for $h \leq z$: The sequence $(a_h)_{h \geq 0}$ is monotonically increasing as long as the ratio a_h/a_{h-1} is at least 1, and hence, $a_{h-1} \geq a_0 = k^2$. Summarizing these observations, $(a_h)_{h=0}^{z-1}$ is a strictly increasing sequence, whereas the sequence $(a_h/a_{h-1})_{h=1}^{z}$ is a decreasing sequence, where the difference between two consecutive elements is at least $7/15 - 1/15 = 2/5$. We will bound the number z from above later in the analysis.

We consider z job classes $\mathcal{J}_1, \ldots, \mathcal{J}_z$ and as many machine classes $M_1, \ldots, M_z$. Each machine class M_h contains $m_h = a_{h-1}$ machines with speed 1 (as we have identical machines). Each job class $\mathcal{J}_h$ consists of two subclasses $\mathcal{J}_h^A$ and $\mathcal{J}_h^B$ of size a_h and of size $b_h = 17 m_h = 17 a_{h-1}$, respectively. We call the jobs in class $\mathcal{J}_h^A$ *large jobs* as they have processing requirements uniformly drawn from $[7/8, 1]$ and can be processed on machines in $M_h \cup M_{h+1}$. As a convention let $M_{z+1} := \emptyset$. Jobs in class $\mathcal{J}_h^B$ are called *small jobs* because they have processing requirements uniformly drawn from $[0, 1/8]$ and can only be processed on machines in M_h.

The schedule $\sigma = \sigma(I)$ for an instance $I \in \mathcal{I}$ is obtained by scheduling the jobs in $\mathcal{J}_h$ on the machines in M_h using the *LPT (longest processing time) algorithm.* This algorithm is a special variant of the *list scheduling algorithm* where the list of jobs is ordered monotonically decreasing with respect to the processing requirements. That is, the largest job is the first one scheduled by the LPT algorithm, whereas the smallest job will be scheduled last. Note, that the LPT algorithm first schedules all jobs from the classes $\mathcal{J}_h^A$ before it considers a job from a class $\mathcal{J}_h^B$. Schedule $\sigma(I)$ is visualized in Figure 6.9. Machine h is a representative for the machines in class M_h. In expectation, the average load of a machine in class M_h is

$$\frac{|\mathcal{J}_h^A| \cdot \frac{15}{16} + |\mathcal{J}_h^B| \cdot \frac{1}{16}}{m_h} = \frac{a_h \cdot \frac{15}{16} + 17a_{h-1} \cdot \frac{1}{16}}{a_{h-1}} = \frac{15}{16}\frac{a_h}{a_{h-1}} + \frac{17}{16}\,.$$



$\mathcal{J}_1$
$\mathcal{J}_2$
$\mathcal{J}_{z_k}$
$\approx \frac{15}{16}\frac{a_1}{a_0} + \frac{17}{16}$
$\geq \frac{15}{16}k$
$\approx \frac{15}{16}\frac{a_2}{a_1} + \frac{17}{16}$
$\approx \frac{15}{16}\frac{a_z}{a_{z-1}} + \frac{17}{16}$
1
2
z

Figure 6.9: Schedule $\sigma(I)$

We show that schedule σ is lex-jump optimal with high probability. To be more specific, we show lex-jump optimality when the total processing requirements $Q_h^A = \sum_{j\in\mathcal{J}_h^A} p_j$ and $Q_h^B = \sum_{j\in\mathcal{J}_h^B} p_j$ of the classes $\mathcal{J}_h^A$ and $\mathcal{J}_h^B$ are close to their expectations for all $h = 1,\dots,z$. Let $\mathcal{E}_h^A$ and $\mathcal{E}_h^B$ denote the events that

$$\left|Q_h^A - \mathbf{E}\left[Q_h^A\right]\right| \leq \frac{m_h}{16} \quad \text{and} \quad \left|Q_h^B - \mathbf{E}\left[Q_h^B\right]\right| \leq \frac{m_h}{32}\,, \quad \text{respectively}\,.$$

Moreover, let $\mathcal{E} = \bigcap_{h=1}^{z}\left(\mathcal{E}_h^A \cap \mathcal{E}_h^B\right)$ denote the event that the events $\mathcal{E}_h^A$ and $\mathcal{E}_h^B$ are simultaneously true for all $h = 1,\dots,z$. By $\overline{\mathcal{E}}_h^A$, $\overline{\mathcal{E}}_h^B$, and $\overline{\mathcal{E}}$ we refer to the complement of $\mathcal{E}_h^A$, $\mathcal{E}_h^B$, and $\mathcal{E}$.

First, we analyze the sequence $a_0, a_1, \dots, a_z$ to obtain bounds for the number z of machine and job classes and for the number m of machines.

Lemma 6.3.4. *For any $h = 1,\dots,z$ the following inequality holds:*

$$\frac{a_h}{a_{h-1}} \leq k - (h-1)\cdot\frac{2}{5}\,.$$

Proof. The claim is true for $h = 1$. By definition of a_h,

$$\frac{a_h}{a_{h-1}} \leq \frac{\left(\frac{a_{h-1}}{a_{h-2}} - \frac{7}{15}\right)\cdot a_{h-1} + 1}{a_{h-1}} \leq \frac{a_{h-1}}{a_{h-2}} - \frac{6}{15} = \frac{a_{h-1}}{a_{h-2}} - \frac{2}{5}$$

for any $h = 2,\dots,z$ as $a_{h-1} \geq a_0 = k^2 \geq 15$. The claim follows by induction. □

Corollary 6.3.5. *The number z of machine and job classes is bounded by $5k/2$.*

Proof. Applying Lemma 6.3.4 for $h = z - 1$ we obtain $1 < a_{z-1}/a_{z-2} \leq k - (z-2) \cdot 2/5$. Hence, $z < (k-1) \cdot 5/2 + 2 < 5k/2$. □

Lemma 6.3.6. *The number m of machines is bounded from above by $\Gamma(k'+3)$ for $k' = \lceil 5k/2 \rceil$, where Γ denotes the gamma function.*

Proof. By induction we show that $a_h \leq k^2 \cdot (2/5)^h \cdot k'!/(k'-h)!$ for any $h = 0, \dots, z-1$. Note that $z \leq 5k/2 \leq k'$ due to Corollary 6.3.5. For $h = 0$ the claim holds because $a_0 = k^2$. For $h \geq 1$ we apply Lemma 6.3.4 to get $a_h/a_{h-1} \leq k - (h-1) \cdot 2/5 \leq (k' - (h-1)) \cdot 2/5$. The induction hypothesis for a_{h-1} yields

$$a_h \leq \frac{2}{5} \cdot (k' - (h-1)) \cdot k^2 \cdot \left(\frac{2}{5}\right)^{h-1} \cdot \frac{k'!}{(k'-(h-1))!} = k^2 \cdot \left(\frac{2}{5}\right)^h \cdot \frac{k'!}{(k'-h)!} \, .$$

Recalling that $m_h = a_{h-1}$ we can bound the number m of machines by observing that

$$\frac{m}{k^2} = \sum_{h=1}^{z} \frac{m_h}{k^2} = \sum_{h=0}^{z-1} \frac{a_h}{k^2} \leq \sum_{h=0}^{z-1} \frac{k'!}{(k'-h)!} \leq k'! \cdot e \, .$$

Hence, $m \leq e \cdot k^2 \cdot k'! \leq (k'+2)! = \Gamma(k'+3)$. □

Lemma 6.3.7. *Event $\overline{\mathcal{E}}$ occurs with probability at most $10k \cdot \exp(-k/2)$.*

Proof. We bound the probability for the events $\overline{\mathcal{E}}_h^A$ and $\overline{\mathcal{E}}_h^B$ to occur. Recalling that $m_h = a_{h-1} \geq a_0 = k^2$, $a_h \leq k \cdot a_{h-1}$ (see Lemma 6.3.4), $b_h = 17 m_h$, and $k \geq 68$ we obtain

$$\begin{aligned}\mathbf{Pr}\left[\overline{\mathcal{E}}_h^A\right] &= \mathbf{Pr}\left[\left|Q_h^A - \mathbf{E}[Q_h^A]\right| > \frac{m_h}{16}\right] \leq 2\exp\left(-\frac{2\left(\frac{m_h}{16}\right)^2}{a_h \cdot \left(\frac{1}{8}\right)^2}\right) \\ &= 2\exp\left(-\frac{a_{h-1}}{a_h} \cdot \frac{a_{h-1}}{2}\right) \leq 2\exp\left(-\frac{a_{h-1}}{2k}\right) \leq 2\exp\left(-\frac{k}{2}\right)\end{aligned}$$

and

$$\begin{aligned}\mathbf{Pr}\left[\overline{\mathcal{E}}_h^B\right] &= \mathbf{Pr}\left[\left|Q_h^B - \mathbf{E}[Q_h^B]\right| > \frac{m_h}{32}\right] \leq 2\exp\left(-\frac{2\left(\frac{m_h}{32}\right)^2}{b_h \cdot \left(\frac{1}{8}\right)^2}\right) \\ &= 2\exp\left(-\frac{m_h}{17 m_h} \cdot \frac{a_{h-1}}{8}\right) \leq 2\exp\left(-\frac{k^2}{136}\right) \leq 2\exp\left(-\frac{k}{2}\right) \, .\end{aligned}$$

Each of the first inequalities stems from *Hoeffding's inequality* [Hoe63] (see Theorem 3.2). A *union bound* yields

$$\mathbf{Pr}\left[\overline{\mathcal{E}}\right] = \mathbf{Pr}\left[\bigcup_{h=1}^{z} \left(\overline{\mathcal{E}}_h^A \cup \overline{\mathcal{E}}_h^B\right)\right] \leq 2z \cdot 2\exp\left(-\frac{k}{2}\right) \leq 10k \cdot \exp\left(-\frac{k}{2}\right)$$

due to Corollary 6.3.5. □

As event $\mathcal{E}$ occurs with high probability and as

$$\mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\mathrm{Lex}(I)}\frac{C_{\max}(I,\sigma)}{C^{\star}_{\max}(I)}\right] \geq \mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\mathrm{Lex}(I)}\frac{C_{\max}(I,\sigma)}{C^{\star}_{\max}(I)}\middle|\,\mathcal{E}\right]\cdot\mathop{\mathbf{Pr}}_{I\sim\mathcal{I}}\left[\mathcal{E}\right],$$

to prove Theorem 6.3.3 it suffices to bound the expected value conditioned on event $\mathcal{E}$ by $\Omega\big((\log m)/(\log\log m)\big)$. Therefore, in the remainder of this section we assume that event $\mathcal{E}$ happens.

Lemma 6.3.8. *The loads of the machines within the same class differ only slightly. In particular, $|L_i - L_{i'}| \leq 1/8$ for any machines $i, i' \in M_h$.*

Proof. Suppose to the contrary that there exist two machines $i, i' \in M_h$ for which $L_i - L_{i'} > 1/8$. Recall that according to the LPT rule all large jobs (from classes $\mathcal{J}_h^A$) will be assigned to the machines before the small jobs (from classes $\mathcal{J}_h^B$) are assigned. After all large jobs have been assigned to the machines in M_h, the difference in load between any two machines in M_h is at most 1 since $p_j \leq 1$ for all jobs j.

Since the processing time of all small jobs is bounded by 1/8, $L_i - L_{i'} > 1/8$ implies that no small job is assigned to machine i nor to any machine that has load at least L_i. Hence, all small jobs are assigned to the machines from class M_h that have load less than L_i. Note that there are at most $m_h - 1$ such machines.

As the difference in load between machine i and any other machine in M_h is at most 1 after all large jobs have been scheduled, the total amount of processing requirements of the small jobs in class M_h is bounded by $Q_h^B \leq (m_h - 1)\cdot 1 < 17m_h/16 - m_h/32 = \mathbf{E}[Q_h^B] - m_h/32$ contradicting the assumption that event $\mathcal{E}_h^B$ holds. □

Lemma 6.3.9. *For any machine $i \in M_h$ the inequality $\big|L_i - \big(\mathbf{E}[Q_h^A] + \mathbf{E}[Q_h^B]\big)/m_h\big| \leq 7/32$ holds, i.e., the load of machine i is close to the expected average machine load in class M_h.*

Proof. By applying the triangle inequality we obtain

$$\begin{aligned}\left|L_i - \frac{\mathbf{E}\left[Q_h^A\right] + \mathbf{E}\left[Q_h^B\right]}{m_h}\right| &\leq \left|L_i - \frac{Q_h^A + Q_h^B}{m_h}\right| + \frac{\left|Q_h^A - \mathbf{E}\left[Q_h^A\right]\right|}{m_h} + \frac{\left|Q_h^B - \mathbf{E}\left[Q_h^B\right]\right|}{m_h} \\ &\leq \left|L_i - \frac{\sum_{i'\in M_h} L_{i'}}{|M_h|}\right| + \frac{1}{16} + \frac{1}{32} \leq \frac{1}{8} + \frac{1}{16} + \frac{1}{32} = \frac{7}{32},\end{aligned}$$

where the second inequality holds since $\mathcal{E}_h^A$ and $\mathcal{E}_h^B$ are true. The third inequality is due to Lemma 6.3.8. □

Lemma 6.3.10. *Schedule σ is lex-jump optimal.*

Proof. We have to show that $L_{i'} + p_j \geq L_i$ holds for any machine $i \in M_h$, any job $j \in J_i$, and any machine $i' \in \mathcal{M}_j$. Let $i \in M_h$ be an arbitrary machine. First, consider the last job j that has been assigned to i. If there is no such job, then we are done. Otherwise, $L_{i'} + p_j \geq L_i$ for any machine $i' \in M_h$ as this job was assigned to machine i by the LPT algorithm. Furthermore, job j is a smallest job on machine i due to the LPT rule. Hence, $L_{i'} + p_{j'} \geq L_i$ for any machine $i' \in M_h$ and any job $j' \in J_i$ assigned to machine i.

For the small jobs on machine i the set of allowed machines equals M_h, so nothing more has to be shown for them. For large jobs j on machine i it remains to show that $L_{i'} + p_j \geq L_i$ for any machine $i' \in M_{h+1}$. Recalling that $a_h = \lceil (a_{h-1}/a_{h-2}) - 7/15) \cdot a_{h-1} \rceil$ for $h \geq 2$, $m_h = a_{h-1}$, and $b_h/m_h = 17$ we observe that

$$\begin{aligned}\frac{\mathbf{E}\left[Q_{h+1}^A\right] + \mathbf{E}\left[Q_{h+1}^B\right]}{m_{h+1}} &= \frac{|\mathcal{J}_{h+1}^A| \cdot \frac{15}{16} + |\mathcal{J}_{h+1}^B| \cdot \frac{1}{16}}{m_{h+1}} = \frac{\frac{15}{16} \cdot a_{h+1} + \frac{1}{16} \cdot b_{h+1}}{m_{h+1}} \\ &= \frac{15}{16} \cdot \frac{a_{h+1}}{a_h} + \frac{1}{16} \cdot \frac{b_{h+1}}{m_{h+1}} \geq \frac{15}{16} \cdot \left(\frac{a_h}{a_{h-1}} - \frac{7}{15}\right) + \frac{1}{16} \cdot \frac{b_h}{m_h} \\ &= \frac{15}{16} \cdot \frac{a_h}{a_{h-1}} + \frac{1}{16} \cdot \frac{b_h}{m_h} - \frac{7}{16} = \frac{\mathbf{E}\left[Q_h^A\right] + \mathbf{E}\left[Q_h^B\right]}{m_h} - \frac{7}{16}\end{aligned}$$

for any $h = 1, \ldots, z-1$. This implies

$$\begin{aligned}L_{i'} + p_j &\geq \frac{\mathbf{E}\left[Q_{h+1}^A\right] + \mathbf{E}\left[Q_{h+1}^B\right]}{m_{h+1}} - \frac{7}{32} + \frac{7}{8} \geq \frac{\mathbf{E}\left[Q_h^A\right] + \mathbf{E}\left[Q_h^B\right]}{m_h} - \frac{7}{16} + \frac{21}{32} \\ &= \frac{\mathbf{E}\left[Q_h^A\right] + \mathbf{E}\left[Q_h^B\right]}{m_h} + \frac{7}{32} \geq L_i\,,\end{aligned}$$

where the first and the last inequality are due to Lemma 6.3.9. □

Finally, we can prove Theorem 6.3.3.

Proof of Theorem 6.3.3. As mentioned before, due to Lemma 6.3.7 it suffices to bound the expected value conditioned on event $\mathcal{E}$. If event $\mathcal{E}$ holds, then schedule $\sigma = \sigma(I)$ is lex-jump optimal (see Lemma 6.3.10), i.e., $\sigma \in \mathrm{Lex}(I)$, and has makespan

$$\begin{aligned}C_{\max} &\geq \max_{i \in M_1} L_i \geq \frac{Q_1^A + Q_1^B}{m_1} \geq \frac{\mathbf{E}[Q_1^A] + \mathbf{E}[Q_1^B]}{m_1} - \frac{\frac{m_1}{16} + \frac{m_1}{32}}{m_1} \\ &= \frac{\frac{15}{16}k^3 + \frac{1}{16} \cdot 17k^2}{k^2} - \frac{3}{32} \geq \frac{15}{16}k\,,\end{aligned}$$

where the third inequality is due to the occurrence of $\mathcal{E}_1^A$ and $\mathcal{E}_1^B$. Now, consider the following schedule σ':

- For $h = 1, \dots, z-1$ spread the jobs of class $\mathcal{J}_h^A$ evenly among the machines in class M_{h+1}. As $|\mathcal{J}_h^A| = a_h = m_{h+1} = |M_{h+1}|$, each machine is assigned exactly one large job.
- Spread the jobs of class $\mathcal{J}_z^A$ evenly among the machines in class M_z. As $|\mathcal{J}_z^A| = a_z \le a_{z-1} = m_z = |M_z|$, each machine is assigned at most one such large job.
- For $h = 1, \dots, z$ spread the jobs of class $\mathcal{J}_h^B$ evenly among the machines in class M_h. As $|\mathcal{J}_h^B| = 17m_h = 17 \cdot |M_h|$, each machine is assigned exactly 17 of these small jobs.

Note that with 'evenly' we refer to the number of jobs on each machine and not to the load. Figure 6.10 shows schedule σ' where each machine h is a representative for all machines in class M_h.

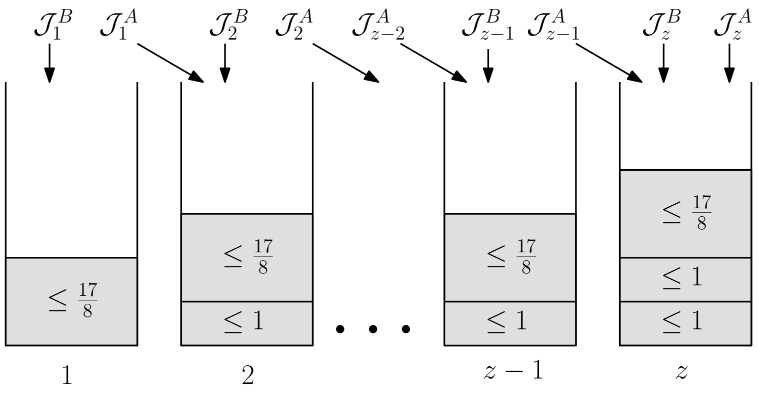


Figure 6.10: Schedule σ'

As each machine contains at most 2 large jobs and 17 small jobs, the makespan of schedule σ' and, hence, $C^\star_{\max}$ is bounded by $2{\cdot}1+17{\cdot}1/8 \le 5$. This implies $C_{\max}(\sigma)/C^\star_{\max} \ge 3k/16 = \Omega(\Gamma^{-1}(m))$ due to Lemma 6.3.6. Hence,

$$\mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[\max_{\sigma\in\mathrm{Lex}(I)} \frac{C_{\max}(I,\sigma)}{C^\star_{\max}(I)}\middle|\,\mathcal{E}\right] \ge \mathop{\mathbf{E}}_{I\sim\mathcal{I}}\left[\frac{C_{\max}(I,\sigma(I))}{C^\star_{\max}(I)}\middle|\,\mathcal{E}\right] = \Omega\left(\Gamma^{-1}(m)\right) = \Omega\left(\frac{\log m}{\log\log m}\right).$$

□

Remark. Lemma 6.3.7 establishes that event $\mathcal{E}$ occurs with high probability. Hence, if we choose k suitably large, then the stated results not only hold in expectation, but also with high probability.

The bound of $\Omega\big((\log m)/(\log\log m)\big)$ of Theorem 6.3.3 carries over to the environment with restricted related machines. In this environment it does, however, not match the known worst-case upper bound of $O\big((\log S)/(\log\log S)\big)$ for $S = \sum_i s_i/s_m$ due to Rutten et al. [RRSV12].

6.4 Table of Notation

In the table below, the notation used in this chapter is summarized.

J	set of jobs $1,\ldots,n$
M	set of machines $1,\ldots,m$
$p_j \in [0,1]$	processing requirement of job j
$s_i \geq 1$	speed of machine i
$s_{\min} = 1$	minimum speed of the machines
$s_{\max}$	maximum speed of the machines
$\mathcal{M}_j \subseteq M$	set of machines on which job j is allowed to be scheduled
$\sigma\colon J \to M$	a schedule
$\sigma^\star$	an optimal schedule
$C_{\max}(\sigma)$	makespan of schedule σ
$C^\star_{\max} = C_{\max}(\sigma^\star)$	optimal makespan
$J_i(\sigma) \subseteq J$	set of jobs assigned to machine i in schedule σ
$L_i(\sigma)$	$:= \sum_{j\in J_i(\sigma)} p_j/s_i$; load of machine i in schedule σ
$J_{i,j}(\sigma)$	$:= J_i(\sigma) \cap \{1,\ldots,j\}$
j_i^t	$:= \min\left\{j :\ \sum_{\ell\in J_{i,j}(\sigma)} p_\ell/s_i \geq t\cdot C^\star_{\max}\right\}$
$J_{i,\geq t}(\sigma)$	$:= J_{i,j_i^t}(\sigma)$
c	$:= \left\lfloor \frac{C_{\max}(\sigma)}{C^\star_{\max}} \right\rfloor - 1$
i_k	$:= \max\{i \in M :\ L_{i'} \geq k\cdot C^\star_{\max}\ \forall\, i' \leq i\}$, assuming $s_1 \geq \ldots \geq s_m$
H_k	$:= \{1,\ldots,i_k\} \subseteq M$
R_k	$:= H_k \setminus H_{k+1}$ for $k = 0,\ldots,c-1$
R_c	$:= H_c$

Chapter 7

Counting Pareto-Optimal Solutions

This chapter is devoted to the study of the number of *Pareto-optimal solutions* of integer linear *multiobjective optimization problems.* In Section 7.1 we show that in the bicriteria case ($d = 1$) the model with zero-preserving perturbations is not more general than the model of ϕ-smooth instances. After introducing some notation in Section 7.2, we present an outline of our approach and our methods in Section 7.3. In Section 7.5 we prove our result about the first moment (Theorem 1.5.1) and in Section 7.6 our results about higher moments (Theorem 1.5.3) of the smoothed number of Pareto-optimal solutions. In Section 7.7 we consider zero-preserving perturbations and prove Theorem 1.5.4. Section 7.8 again considers non-zero preserving perturbations. There the lower bound for the smoothed number of Pareto-optimal solutions for the binary setting stated in Theorem 1.5.2 is derived.

7.1 Zero-Preserving Perturbations in the Bicriteria Case

Let us first of all remark that we can assume that the adversarial objective V^{d+1} is injective. If not, then let $v_1, \dots, v_\ell$ be the values taken by V^{d+1} and let $\Delta = \min_{i \neq j} |v_i - v_j|$. Now, consider an arbitrary injective function $\delta \colon \mathcal{S} \to [0, \Delta)$ and define the new adversarial objective as $W^{d+1}x = V^{d+1}x + \delta(x)$. Obviously, this function is injective and it preserves the order of the solutions in $\mathcal{S}$. That is, if $V^{d+1}x < V^{d+1}y$ for $x, y \in \mathcal{S}$, then also $W^{d+1}x < W^{d+1}y$. Let x be a Pareto optimum with respect to $\mathcal{S}$ and $\{V^1, \dots, V^{d+1}\}$ and let $x_2, \dots, x_m$, $m \geq 1$, be all the other solutions for which $V^k x_i = V^k x$ for any $k \in \{1, \dots, d+1\}$. These are all Pareto optima but, due to our convention, we only count them once. Without loss of generality let x be the solution that minimizes W^{d+1} among these solutions. Then x is also Pareto-optimal with respect to $\mathcal{S}$ and $\{V^1, \dots, V^d, W^{d+1}\}$.

Now we demonstrate by an example that, in the bicriteria case, which was studied in [BV04], zero-preserving perturbations are not more powerful than other perturbations because they can be simulated by the right choice of $\mathcal{S} \subseteq \{0,\dots,\mathcal{K}\}^n$ and the objective function $V^2 \colon \mathcal{S} \to \mathbb{R}$. Assume, for instance, that the adversary has chosen $\mathcal{S}$ and V^2 and that he has decided that the first coefficient V_1^1 of the first objective function should be deterministically set to zero. Also assume without loss of generality that V^2 is injective. We can partition the set $\mathcal{S}$ in classes of solutions that agree in all components except for the first one. That is, two solutions $x \in \mathcal{S}$ and $y \in \mathcal{S}$ belong to the same class if $x_i = y_i$ for all $i \in \{2,\dots,n\}$. All solutions in the same class have the same value in the first objective V^1 as they differ only in the binary variable x_1, whose coefficient has been set to zero. We construct a new set of solutions $\mathcal{S}'$ that contains for every class only the solution with smallest value in V^2. It is easy to see that the number of Pareto-optimal solutions is the same with respect to $\mathcal{S}$ and with respect to $\mathcal{S}'$ because all solutions in $\mathcal{S} \setminus \mathcal{S}'$ are dominated by solutions in $\mathcal{S}'$.

Then we transform the set $\mathcal{S}' \subseteq \{0,\dots,\mathcal{K}\}^n$ into a set $\mathcal{S}'' \subseteq \{0,\dots,\mathcal{K}\}^{n-1}$ by dropping the first component of every solution. Furthermore, we define a function $W^2 \colon \mathcal{S}'' \to \mathbb{R}$ that assigns to every solution $x \in \mathcal{S}''$ the same value that V^2 assigns to the corresponding solution in $\mathcal{S}'$. One can easily verify that the Pareto set with respect to $\mathcal{S}'$ and V^2 is identical with the Pareto set with respect to $\mathcal{S}''$ and W^2. The only difference is that in the latter problem we have eliminated the coefficient that is deterministically set to zero. Such an easy reduction of zero-preserving perturbations to other perturbations does not seem to be possible for $d \geq 2$ anymore.

7.2 Notation

For the sake of simplicity we write $V^t x$ instead of $V^t(x)$, even for the adversarial objective V^{d+1}. With $V^{k_1\dots k_t}x$ we refer to the vector $(V^{k_1}x,\dots,V^{k_t}x)$. In our analysis, we will shift the solutions $x \in \mathcal{S}$ by a certain vector $u \in \{0,\dots,\mathcal{K}\}^n$ and consider the values $V^t \cdot (x-u)$. For the linear objectives we mean the value $V^t x - V^t u$, where $V^t u$ is well-defined even for a shift vector $u \in \{0,\dots,\mathcal{K}\}^n \setminus \mathcal{S}$. For the adversarial objective, however, we define $V^{d+1} \cdot (x-u) := V^{d+1}x$. It should not be confused with $V^{d+1}y$ for $y = x-u$. Throughout this chapter let $\varepsilon > 0$ be a small real for which $1/\varepsilon$ is integral. Let $b = (b_1,\dots,b_d) \in \mathbb{R}^d$ be a vector such that b_k/ε is an integer for any k. We call the set $B = \big\{(y_1,\dots,y_d) \in \mathbb{R}^d : y_k \in (b_k, b_k+\varepsilon] \text{ for any } k\big\}$ an *ε-box* and b *the corner* of B. For a vector $x \in \{-\mathcal{K},\dots,\mathcal{K}\}^n$ the expression $B_V(x)$ denotes the unique ε-box B for which $V^{1\dots d}x \in B$. We call B the ε-box of x and say that x lies in B. By $\mathbb{B}_\varepsilon$ we denote the set of all ε-boxes with a corner $b \in \{-n\mathcal{K}, -n\mathcal{K}+\varepsilon,\dots,n\mathcal{K}-2\varepsilon, n\mathcal{K}-\varepsilon\}^d$. Hence, $|\mathbb{B}_\varepsilon| = (2n\mathcal{K}/\varepsilon)^d$. If all coefficients V_i^k of V are from $[-1,1]$, which is true for any of the models considered in this chapter, and if for any $k = 1,\dots,d$ there is an index i for which $|V_i^k| < 1$, which holds with probability 1 in any of our models, then the ε-box of any vector $x \in \{-\mathcal{K},\dots,\mathcal{K}\}^n$ belongs to $\mathbb{B}_\varepsilon$. Note that all vectors x constructed in this chapter are from $\{-\mathcal{K},\dots,\mathcal{K}\}^n$. Hence, without further explanation we assume that $B_V(x) \in \mathbb{B}_\varepsilon$.

We will extensively use tuples instead of sets. The reason for this is that we are not only interested in certain components of a vector or matrix, but we also want to describe in which order they are considered. This will be clear after the introduction of the following notation. Let n, m be positive integers and let $a_1, \ldots, a_n$, $b_1, \ldots, b_m$ be arbitrary and not necessarily pairwise distinct reals. We define $[n] = (1, \ldots, n)$, $[n]_0 = (0, 1, \ldots, n)$, $|(a_1, \ldots, a_n)| = n$ and $(a_1, \ldots, a_n) \cup (b_1, \ldots, b_m) = (a_1, \ldots, a_n, b_1, \ldots, b_m)$. Note, that always $|(a_1, \ldots, a_n) \cup (b_1, \ldots, b_m)| = n + m$. This is different from sets, where $|\{a_1, \ldots, a_n\} \cup \{b_1, \ldots, b_m\}| < n + m$ if $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_m\}$ have at least one element in common. By $(a_1, \ldots, a_n) \setminus (b_1, \ldots, b_m)$ and $(a_1, \ldots, a_n) \cap (b_1, \ldots, b_m)$ we denote the tuples we obtain by removing all elements that (do not) belong to $(b_1, \ldots, b_m)$ from $(a_1, \ldots, a_n)$. We write $(a_1, \ldots, a_n) \subseteq (b_1, \ldots, b_m)$ if $a_k \in (b_1, \ldots, b_m)$ for any index $k \in [n]$. Let x be a vector and let A be a matrix. By $x|_{i_1 \ldots i_n} = x|_{(i_1, \ldots, i_n)}$ we denote the column vector $(x_{i_1}, \ldots, x_{i_n})^{\mathrm{T}}$, by $A|_{(i_1, \ldots, i_n)}$ we denote the matrix consisting of the rows $i_1, \ldots, i_n$ of matrix A (in this order).

For an index set $I \subseteq [n]$ and a vector $y \in \{0, \ldots, \mathcal{K}\}^n$ let $\mathcal{S}_I(y)$ denote the set of all solutions $z \in \mathcal{S}$ for which $z_i = y_i$ for any index $i \in I$. For the sake of simplicity we also use the notation $\mathcal{S}_I(\hat{y})$ for vectors $\hat{y} \in \{0, \ldots, \mathcal{K}\}^{|I|}$ whose components are labeled as $y_{i_1}, \ldots, y_{i_{|I|}}$ (assuming that $I = (i_1, \ldots, i_{|I|})$) to describe the set $\{z \in \mathcal{S} : z_i = \hat{y}_i \text{ for any } i \in I\}$.

With $\mathbb{I}_n$ we refer to the $n \times n$-identity matrix $\operatorname{diag}(1, \ldots, 1)$ and with $\mathbb{O}_{m \times n}$ to the $m \times n$-matrix whose entries are all zero. If the number of rows and columns are clear, then we drop the indices.

For a set $M \subseteq \mathbb{R}^n$ and a vector $y \in \mathbb{R}^n$ we define $M + y := \{x + y : x \in M\}$, the *Minkowski sum* of M and $\{y\}$.

Definition 7.2.1. Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a set of solutions and let $f_1, \ldots, f_d \colon \mathcal{S} \to \mathbb{R}$ be functions.

1. Let $x, y \in \mathbb{R}^n$ be vectors. We say that x *dominates* y (with respect to $\{f_1, \ldots, f_d\}$), if $f_i(x) \leq f_i(y)$ for any $i \in [d]$ and $f_i(x) < f_i(y)$ for at least one $i \in [d]$. We say that x *dominates* y *strongly* (with respect to $\{f_1, \ldots, f_d\}$), if $f_i(x) < f_i(y)$ for any $i \in [d]$.

2. Let $x \in \mathbb{R}^n$ be a vector. We call x *Pareto-optimal* or a *Pareto optimum* (with respect to $\mathcal{S}$ and $\{f_1, \ldots, f_d\}$), if x is an element of $\mathcal{S}$ and if no solution $y \in \mathcal{S}$ dominates x. We call x *weakly Pareto-optimal* or a *weak Pareto optimum* (with respect to $\mathcal{S}$ and $\{f_1, \ldots, f_d\}$), if x is an element of $\mathcal{S}$ and if no solution $y \in \mathcal{S}$ dominates x strongly.

We focus on Pareto-optimal solutions. The notion of strong dominance and weak Pareto optimality is a mere concept that we need in our analysis of the model with zero-preserving perturbations.

7.3 Outline of Our Approach

To prove our results we adapt and improve methods from the previous analyses by Moitra and O'Donnell [MO11] and by Röglin and Teng [RT09] and combine them in a novel way. Since all coefficients of the linear objective functions lie in the interval $[-1, 1]$, for every solution $x \in \mathcal{S}$ the vector $V^{1 \ldots d}x$ lies in the hypercube $[-n\mathcal{K}, n\mathcal{K}]^d$. The first step is to partition this hypercube into ε-boxes. If ε is very small (exponentially small in n), then it is unlikely that there are two different solutions $x \in \mathcal{S}$ and $y \in \mathcal{S}$ that lie in the same ε-box B unless x and y differ only in positions that are not perturbed in any of the objective functions, in which case we consider them as the same solution. In the remainder of this section we assume that no two solutions lie in the same ε-box. Then, in order to bound the number of Pareto-optimal solutions, it suffices to count the number of non-empty ε-boxes.

In order to prove Theorem 1.5.1 we show that for any fixed ε-box the probability that it contains a Pareto-optimal solution is bounded by $kn^d\phi^d\varepsilon^d$ for a sufficiently large $k = k(\mathcal{K}, d)$. This implies the theorem as the number of ε-boxes is $(2n\mathcal{K}/\varepsilon)^d$. Fix an arbitrary ε-box B. In the following we will call a solution $x \in \mathcal{S}$ a *candidate* if there is a realization of V such that x is Pareto-optimal and lies in B. If there was only a single candidate $x \in \mathcal{S}$, then we could bound the probability that there is a Pareto-optimal solution in B by the probability that this particular solution x lies in B. This probability can easily be bounded from above by $\varepsilon^d\phi^d$. However, in principle, every solution $x \in \mathcal{S}$ can be a candidate and a union bound over all of them leads to a factor of $|\mathcal{S}|$ in the bound, which we have to avoid.

Following the ideas of Moitra and O'Donnell, we divide the draw of the random matrix V into two steps. In the first step some information about V is revealed that suffices to limit the set of candidates to a single solution $x \in \mathcal{S}$. The exact position $V^{1 \ldots d}x$ of this solution is determined in the second step. If the information that is revealed in the two steps is chosen carefully, then there is enough randomness left in the second step to bound the probability that x lies in the ε-box B. In Moitra and O'Donnell's analysis the coefficients in the matrix V are partitioned into two groups. In the first step the first group of coefficients is drawn, which suffices to determine the unique candidate x, and in the second step the remaining coefficients are drawn, which suffices to bound the probability that x lies in B. The second part consists essentially of $d(d+1)/2$ coefficients, which causes the factor of $\phi^{d(d+1)/2}$ in their bound. Since the coefficients from the first and the second draw are independent of each other, Moitra and O'Donnell can apply the *principle of deferred decisions* to complete their analysis.

We improve the analysis by a different choice of how to break the draw of V into two parts. As in the previous analysis, most coefficients are drawn in the first step. Only d^2 coefficients of V are drawn in the second step. However, these coefficients are not left completely random as in [MO11] because after the other coefficients have been drawn there can still be multiple candidates for Pareto-optimal solutions in B. Instead, the randomness is reduced further by drawing $d(d-1)$ linear combinations of these random variables in the first step. These linear combinations have the property that, after they have been drawn, there is a unique candidate x whose position can be described by d linear combinations that are linearly independent of the linear combinations already drawn in the first step. Unlike in the analysis of Moitra and O'Donnell we do not have independent random variables. Even worse, for some realizations of the linear combinations of the first step the conditional probability that x lies in B can be very large. This is why we cannot apply the principle of deferred decisions here.

In [RT09] it was observed that linearly independent linear combinations of independent random variables behave in some respect similar to independent random variables. With this insight one can argue that in the second step there is still enough randomness to bound the probability that x lies in B. While the analysis in [RT09] yields only a bound proportional to $\phi^{d^2}\varepsilon^d$, we prove an improved result for quasiconcave densities that yields the desired bound proportional to $\phi^d\varepsilon^d$.

For analyzing higher moments, it does not suffice to bound the probability that a fixed ε-box contains a Pareto-optimal solution. Instead, in order to bound the c^{th} moment, we sum over all c-tuples $(B_1, \dots, B_c)$ of ε-boxes the probability that all ε-boxes $B_1, \dots, B_c$ simultaneously contain a Pareto-optimal solution. We bound this probability from above by $kn^{cd}\phi^{cd}\varepsilon^{cd}$ for a sufficiently large $k = k(\mathcal{K}, d, c)$. Since there are $(2n\mathcal{K}/\varepsilon)^{cd}$ different c-tuples of ε-boxes, this implies the bound of $f(\mathcal{K}, d, c) \cdot O((n^2\phi)^{cd})$ for the c^{th} moment of the smoothed number of Pareto-optimal solutions.

Let us fix a c-tuple $(B_1, \dots, B_c)$ of ε-boxes. The approach to bound the probability that all of these ε-boxes contain simultaneously a Pareto-optimal solution is similar to the approach for the first moment. We divide the draw of V into two steps. In the first step enough information is revealed to identify for each of the ε-boxes B_i a unique candidate $x_i \in \mathcal{S}$ for a Pareto-optimal solution in B_i. If we do this carefully, then there is enough randomness left in the second step to bound the probability that $V^{1\dots d}x_i \in B_i$ for every $i \in [c]$. Again most coefficients are drawn in the first step and some linear combinations of the other cd^2 coefficients are also drawn in the first step. However, we cannot simply repeat the construction for the first moment independently c times because then there might be (strong) dependencies between the events $V^{1\dots d}x_i \in B_i$ for different i. In order to bound the probability that in the second step all x_i lie in their corresponding ε-boxes B_i, we need to ensure that the events $V^{1\dots d}x_i \in B_i$ are (almost) independent after the information from the first step has been revealed.

The general approach to handle zero-preserving perturbations is closely related to the approach for bounding the first moment for non-zero-preserving perturbations. However, additional complications have to be handled. The main problem is that we cannot easily guarantee anymore that the linear combinations in the second step are linearly independent of the linear combinations revealed in the first step. Essentially, the revealed linear combinations describe the positions of some auxiliary solutions. For non-zero-preserving perturbations revealing this information is not critical as no solution has in any objective function exactly the same value as x. For zero-preserving solutions it can, however, happen that the auxiliary solutions take exactly the same value as x in one of the objective functions. Then there is not enough randomness left in the second step anymore to bound the probability that x lies in this objective in the ε-interval described by the ε-box B.

In the remainder of this section we will present some more details on our analysis. We first present a simplified argument to bound the smoothed number of Pareto-optimal solutions. Afterwards we will briefly discuss which changes to this argument are necessary to bound higher moments and to analyze zero-preserving perturbations.

Smoothed Number of Pareto-optimal Solutions As an important building block in the proof of Theorem 1.5.1 we use an insight from [MO11] about how to test whether a given ε-box contains a Pareto-optimal solution. Let us fix an ε-box $B = (b_1, b_1 + \varepsilon] \times \ldots \times (b_d, b_d + \varepsilon]$ with corner $b = (b_1, \ldots, b_d)$. The following algorithm takes as parameters the matrix V and the ε-box B and it returns a solution $x^{(0)}$.

```
Witness($V, B$)
1: Set $\mathcal{R}_{d+1} = \mathcal{S}$.
2: for $t = d, d-1, \ldots, 0$ do
3:     Set $\mathcal{C}_t = \{z \in \mathcal{R}_{t+1} : V^{1 \ldots t} z \leq b|_{1 \ldots t}\}$.
4:     Set $x^{(t)} = \arg\min\{V^{t+1} z : z \in \mathcal{C}_t\}$.
5:     Set $\mathcal{R}_t = \{z \in \mathcal{R}_{t+1} : V^{t+1} z < V^{t+1} x^{(t)}\}$.
6: end for
7: return $x^{(0)}$
```

The actual Witness function that we use in the proof of Theorem 1.5.1 is more complex because it has to deal with some technicalities. In particular, the case that some set $\mathcal{C}_t$ is empty has to be handled. For the purpose of illustration we ignore these technicalities here and assume that $\mathcal{C}_t$ is never empty. The crucial oberservation that has been made by Moitra and O'Donnell is that if there is a Pareto-optimal solution $x \in \mathcal{S}$ that lies in B, then $x^{(0)} = x$ (assuming that no two solutions lie in the same ε-box). Hence, the solution $x^{(0)}$ returned by the Witness function is the only candidate for a Pareto-optimal solution in B. While this statement and the following reasoning are true for any $d \in \mathbb{N}$, we want to illustrate the case $d = 2$, in which there are one adversarial and two linear objective functions. For this, assume that B contains a single solution x which is Pareto-optimal and that x is very close to the corner b of B which can be assumed if B is very small. Then $V^t z \leq b_t$ is equivalent to $V^t z < V^t x$ for any $t \in [d]$.

Consider the situation depicted in Figure 7.1a. The first and the second objective value of each solution determine a point in the Euclidean plane. The additional value depicted next to this point represents the third objective value of each solution. Let us consider the situation before entering the loop. All points in Figure 7.1a are encircled meaning that $\mathcal{R}_3$ contains all solutions, i.e., $\mathcal{R}_3 = \mathcal{S}$. Now let us analyze the loop. The set $\mathcal{C}_2$ contains all solutions that have smaller first and second objective values than x (gray area in Figure 7.1b). Among these solutions we pick the one with the smallest third objective value and denote it by $x^{(2)}$. Set $\mathcal{R}_2$ contains all solutions with a smaller third objective value (encircled points in Figure 7.1c). Note, that in particular no solution of the gray region is considered anymore. On the other hand, x belongs to $\mathcal{R}_2$ due to Pareto-optimality.

The set $\mathcal{C}_1$ contains all solutions from $\mathcal{R}_2$ that have a smaller first objective value than x (encircled points in the gray area in Figure 7.1d). Among these solutions $x^{(1)}$ is the one with the smallest second objective value. Set $\mathcal{R}_1$ contains all solutions from $\mathcal{R}_2$ with a smaller second objective value (encircled points in Figure 7.1e). This set still contains x, but no points from the gray region.

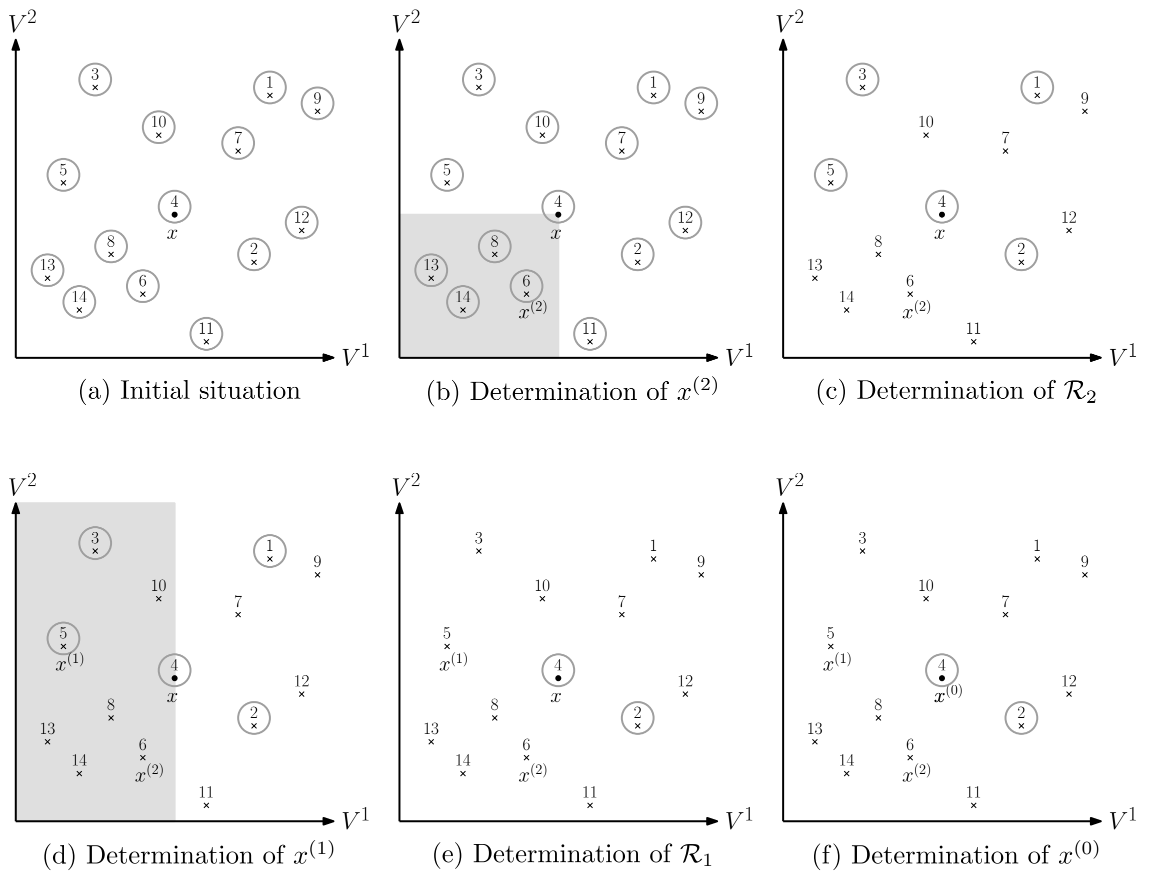
In the final iteration $t = 0$ we obtain $\mathcal{C}_0 = \mathcal{R}_1$ since there is no restriction in the construction of $\mathcal{C}_0$ anymore and $\mathcal{C}_0 \neq \emptyset$ since $x \in \mathcal{R}_1$. Solution $x^{(0)}$ is among the remaining solutions the one with the smallest first objective value (Figure 7.1f). This solution equals x and is now returned.

Our goal is to execute the Witness function and to obtain the solution $x^{(0)}$ without revealing the entire matrix V. We will see that it is indeed possible to divide the draw of V into two steps such that in the first step enough information is revealed to execute the Witness function and such that in the second step there is still enough radnomness left to bound the probability that $x^{(0)}$ lies in B. For this let $I \subseteq [n]$ be a set (or rather a tuple) of indices and assume that we know in advance which values the solutions $x^{(0)}, \dots, x^{(d)}$ take at these indices, i.e., assume that we know $a^{(0)} = x^{(0)}|_I, \dots, a^{(d)} = x^{(d)}|_I$ before executing the Witness function. Then we can reconstruct $x^{(0)}, \dots, x^{(d)}$ without having to reveal the entire matrix V. This can be done by the following algorithm, which gets as additional parameters the set I and the matrix $A = (a^{(0)}, \dots, a^{(d)})$.

```
Witness($V, I, A, B$)
1: Set $\mathcal{R}_{d+1} = \bigcup_{t'=0}^{d} \mathcal{S}_I\big(a^{(t')}\big)$.
2: for $t = d, d-1, \dots, 0$ do
3:     Set $\mathcal{C}_t = \{z \in \mathcal{R}_{t+1} : V^{1\dots t} z \le b|_{1\dots t}\} \cap \mathcal{S}_I\big(a^{(t)}\big)$.
4:     Set $x^{(t)} = \arg\min\{V^{t+1} z : z \in \mathcal{C}_t\}$.
5:     Set $\mathcal{R}_t = \{z \in \mathcal{R}_{t+1} : V^{t+1} z < V^{t+1} x^{(t)}\} \cap \bigcup_{t'=0}^{t-1} \mathcal{S}_I\big(a^{(t')}\big)$.
6: end for
7: return $(x^{(0)}, \dots, x^{(d)})$
```

The additional restriction of the set $\mathcal{R}_{d+1}$ does not change the outcome of the Witness function as all solutions $x^{(0)}, \dots, x^{(d)}$ generated by the first Witness function are contained in the set $\mathcal{R}_{d+1}$ defined in Line 1 of the second Witness function. Similarly one can argue that the additional restrictions in Lines 3 and 5 do not change the outcome of the algorithm because all solutions $x^{(t)}$ generated by the first Witness function satisfy the restrictions

[t]



(a) Initial situation (b) Determination of $x^{(2)}$ (c) Determination of $\mathcal{R}_2$

(d) Determination of $x^{(1)}$ (e) Determination of $\mathcal{R}_1$ (f) Determination of $x^{(0)}$

Figure 7.1: Execution of the Witness function for three objectives

that are made in the second Witness function. Hence, if $a^{(0)} = x^{(0)}|_I, \ldots, a^{(d)} = x^{(d)}|_I$, then both Witness functions generate the same $x^{(0)}$.

We will now discuss how much information about V needs to be revealed in order to execute the second Witness function, assuming that the additional parameters I and A are given. We assume that the coefficients V_i^t are revealed for every $t \in [d]$ and $i \notin I$. For the remaining coefficients only certain linear combinations need to be known in order to be able to execute the Witness function. By carefully looking at the Witness function, one can deduce that for $t \in [d]$ only the linear combinations $V^t|_I \cdot x^{(t)}|_I, \ldots, V^t|_I \cdot x^{(d)}|_I$ and $V^t|_I \cdot (x^{(t-1)} - x^{(0)})|_I, \ldots, V^t|_I \cdot (x^{(t-1)} - x^{(t-2)})|_I$ need to be known. These terms can be viewed as linear combinations of the random variables V_i^t, $t \in [d]$, $i \in I$, with coefficients from $\{-\mathcal{K}, \ldots, \mathcal{K}\}$. In addition to the already fixed random variables V_i^t, $t \in [d]$, $i \notin I$

the following d linear combinations determine the position $V^{1\dots d}x$ of $x = x^{(0)}$:

$$V_I^1 \cdot x^{(0)}|_I, \dots, V_I^d \cdot x^{(0)}|_I \,.$$

A crucial observation our analysis is based upon is that if the vectors $x^{(0)}|_I, \dots, x^{(d)}|_I$ are linearly independent, then also all of the above mentioned linear combinations are linearly independent. In particular, the d linear combinations that determine the position of x cannot be expressed by the other linear combinations. Usually, however, it is not possible to find a subset $I \subseteq [n]$ of indices such that the vectors $x^{(0)}|_I, \dots, x^{(d)}|_I$ are linearly independent. By certain technical modifications of the Witness function we will ensure that there always exists such a set I with $|I| \leq d+1$. Since we do not know the set I and the matrix A in advance, we apply a union bound over all valid choices for these parameters, which yields a factor of $f(\mathcal{K}, d) \cdot O(n^d)$ in the bound for the probability that there exists a Pareto-optimal solution in B.

Röglin and Teng [RT09] observed that the linear independence of the linear combinations implies that even if the linear combinations needed to execute the Witness function are revealed in the first step, there is still enough randomness in the second step to prove an upper bound on the probability that $V^{1\dots d}x$ lies in a fixed ε-box B that is proportional to ε^d. The bound proven in [RT09] is, however, not strong enough to improve Moitra and O'Donnell's result [MO11] because the dependence on ϕ is in the order of $\Theta(\phi^{d^2})$ which is worse than the dependence of $\Theta(\phi^{d(d+1)/2})$ proven by Moitra and O'Donnell. We show that for quasiconcave density functions the dependence in [RT09] can be improved significantly to $\Theta(\phi^d)$, which yields the improved bound of $O(n^{2d}\phi^d)$ in Theorem 1.5.1 for the binary case.

Higher Moments The analysis of higher moments is based on running the Witness function multiple times. As described above, we bound for a fixed c-tuple $(B_1, \dots, B_c)$ of ε-boxes the probability that all of them contain a Pareto-optimal solution. For this, we run the Witness function c times. This way, we get for every $j \in [c]$ a sequence $x^{(j,0)}, \dots, x^{(j,d)}$ of solutions such that $x^{(j,0)}$ is the unique candidate for a Pareto-optimal solution in B_j.

As above, we would like to execute the c calls of the Witness function without having to reveal the entire matrix V. Again if we know for a subset $I \subseteq [n]$ the values that the solutions $x^{(j,t)}$, $j \in [c]$, $t \in [d]$, take at these positions, then we do not need to reveal the coefficients V_i^t with $i \in I$ to be able to execute the calls of the Witness function. As in the case of the first moment, it suffices to reveal some linear combinations of these coefficients.

In order to guarantee that these linear combinations are linearly independent of the linear combinations that determine the positions of the solutions $x^{(j,0)}$, $j \in [c]$, we need to coordinate the calls of the Witness function. Otherwise it might happen, for example, that the linear combinations revealed for executing the first call of the witness function determine already the position of $x^{(2,0)}$, the candidate for a Pareto-optimal solution in B_2. Assume that the first call of the Witness function returns a sequence $x^{(1,0)}, \dots, x^{(1,d)}$ of solutions and that $I_1 \subseteq [n]$ is a set of indices that satisfies the desired property that $x^{(1,0)}|_{I_1}, \dots, x^{(1,d)}|_{I_1}$ are linearly independent. In order to achieve that all solutions generated in the following calls of the Witness function are linearly independent of these linear combinations, we do not start a second independent call of the Witness function, but we restrict the set of feasible solutions first. Instead of choosing $x^{(2,0)}, \dots, x^{(2,d)}$ among all solutions from $\mathcal{S}$, we restrict the set of feasible solutions for the second call of the Witness function to $\mathcal{S}' = S_{I_1}(x^{(2,0)})$. Although we do not know $x^{(2,0)}$ in advance, we can assume to know some of its entries due to a technical trick. Essentially, all solutions generated in call r of the Witness function have to coincide with $x^{(r,0)}$ in all positions that have been selected in one of the previous calls.

This and some additional tricks allow us to ensure that in the end there is a set $I \subseteq [n]$ with $|I| \leq (d+1)c$ such that all vectors $x^{(j,t)}|_I$, $j \in [c]$, $t \in [d]$ are linearly independent. Then we can again use the bound proven in [RT09] to bound the probability that $V^{1\dots d}x^{(j,0)} \in B_j$ simultaneously for every $j \in [c]$ from above by a term proportional to $\varepsilon^{cd}\phi^{cd^2}$. With our improved bound for quasiconcave density functions, we obtain a bound proportional to $\varepsilon^{cd}\phi^{cd}$. Together with a union bound over all valid choices for I and the values $x^{(j,t)}|_I$, $j \in [c]$, $t \in [d]$, we obtain a bound of $kn^{cd}\varepsilon^{cd}\phi^{cd}$ on the probability that all candidates $x^{(j,0)}$ lie in their corresponding ε-boxes for a sufficiently large $k = k(\mathcal{K}, d, c)$. Together with the bound of $O((n\mathcal{K})^{cd}/\varepsilon^{cd})$ for the number of c-tuples $(B_1, \dots, B_c)$ this implies Theorem 1.5.3.

Zero-preserving Perturbations If we use the same Witness function as above, then it can happen that there is a Pareto-optimal solution x in the ε-box B that does not coincide with the solution $x^{(0)}$ returned by the Witness function. This problem occurs, for example, if $V^d \cdot x^{(d-1)} = V^d \cdot x^{(0)}$, which we cannot exclude if we allow zero-preserving perturbations. On the other hand if we knew in advance that $V^d \cdot x^{(d-1)} = V^d \cdot x^{(0)}$, then we could bound the probability of $V^d x^{(0)} \in (b_d, b_d + \varepsilon]$ already after the solution $x^{(d-1)}$ has been generated. Hence, if we were only interested in bounding this probability, we could terminate the Witness function already after $x^{(d-1)}$ has been generated. Instead of terminating the Witness function at this point entirely, we keep in mind that $V^d x^{(0)}$ has already been determined and we restart the Witness function with the remaining objective functions only.

Let us make this a bit more precise. As long as the solutions $x^{(t)}$ generated by the Witness function differ in all objective functions from x, we execute the Witness function without any modification. Only if a solution $x^{(t)}$ is generated that agrees with x in some objective functions, then we deviate from the original Witness function. Let $K \subseteq [d]$ denote the objective functions in which $x^{(t)}$ coincides with x. Then we can bound at this point the probability that $V^t \cdot x \in (b_t, b_t + \varepsilon]$ simultaneously for all $t \in K$. In order to also deal with the other objectives $t \notin K$, we restart the Witness function. In this restart, we ignore all objective functions in K and we execute the Witness function as if only objectives $t \notin K$ were present. Additionally we restrict in the restart the set of feasible solutions to those that coincide in the objectives $t \in K$ with x, i.e., to $\{y \in \mathcal{S} : V^t \cdot y = V^t \cdot x \text{ for all } t \in K\}$. With similar techniques as in the analysis of higher moments we ensure that different restarts lead to linearly independent linear combinations.

This exploits that every Pareto-optimal solution x is also Pareto-optimal with respect to only the objective functions V^t with $t \notin K$ if the set $\mathcal{S}$ is restricted to solutions that agree with x in all objective functions V^t with $t \in K$. This property guarantees that whenever the Witness function is restarted, x is still a Pareto-optimal solution with respect to the restricted solution set and the remaining objective functions.

It can happen that we have to restart the Witness function d times before a unique candidate for a Pareto-optimal solution in B is identified. As in each of these restarts at most d solutions are generated, the total number of solutions that is generated can increase from $d+1$, as in the case of non-zero-preserving perturbations, to roughly d^2. The set $I \subseteq [n]$ of indices restricted to which these solutions are linearly independent has a cardinality of at most d^3. The reason for this increase is that we have to choose more indices to obtain linear independence due to the fixed zeros. Taking a union bound over all valid choices of I, of the values that the generated solutions take at these positions, and the possibilities when and due to which objectives the restarts occur, yields Theorem 1.5.4. This theorem relies again on the result about linearly independent linear combinations of independent random variables from [RT09] and its improved version for quasiconcave densities that we show in this thesis.

Lower Bound For the construction of a lower bound example for the model without zero-preserving perturbations we consider the following generalization of the knapsack problem which we call *restricted multi-profit knapsack problem*. Here, we are given n objects $a_1, \dots, a_n$, each with a weight w_i and a profit vector $p_i \in \mathbb{R}^d$ for an integer $d \geq 1$. By a vector $s \in \{0,1\}^n$ we describe which objects to put into the knapsack. In this variant of the knapsack problem we are additionally given a set $\mathcal{S} \subseteq \{0,1\}^n$ that encodes which combinations of objects are allowed to be put into the knapsack. We want to simultaneously minimize the total weight and maximize all total profits of the objects. Thus, our *multiobjective optimization problem*, which we denote by $K_{\mathcal{S}}(\{a_1, \dots, a_n\})$, can be written as

$$\textbf{minimize } \textstyle\sum_{i=1}^n w_i \cdot s_i \text{ and } \textbf{maximize } \sum_{i=1}^n (p_i)_j \cdot s_i \text{ for all } j \in [d]$$
$$\textbf{subject to } (s_1, \dots, s_n) \text{ is a feasible solution from } \mathcal{S}.$$

For $\mathcal{S} = \{0,1\}^n$ we simply write $K(\{a_1, \dots, a_n\})$ instead of $K_{\mathcal{S}}(\{a_1, \dots, a_n\})$. First of all note, that the restricted multi-profit knapsack problem only serves as a tool to construct a lower bound example. There is no obvious practical application behind it. Secondly, we break with the convention that we want to simultaneously minimize all objectives. We can easily transform the restricted multi-profit knapsack problem into an equivalent pure multiobjective minimization problem. However, we find that maximizing positive profits is much more intuitive than minimizing negative costs. Of course, the notion of *domination* has to be adapted accordingly.

The idea of our lower bound construction is as follows: Assume that we have already chosen weights and profits for the objects $a_1, \dots, a_n$ as well as a set $\mathcal{S}$. Let W be the total weight and P_i be the i^{th} total profit of these objects. For each dimension $i \in [d]$ we introduce an additional object b_i with a weight that exceeds W. The i^{th} profit of b_i exceeds P_i, whereas all of its other profits are (approximately) zero. As the set of feasible solutions for this new instance we now consider the set $\mathcal{S}' = \mathcal{S} \times \{0,1\}^d$, i.e., the choice of the objects $a_1, \dots, a_n$ is restricted as before, whereas the objects $b_1, \dots, b_d$ can be put into the knapsack arbitrarily. The crucial point of adding the objects $b_1, \dots, b_d$ is, that a solution $(x_1, \dots, x_n, y_1, \dots, y_d)$ is Pareto-optimal for the new instance if and only if $(x_1, \dots, x_n)$ is Pareto-optimal for the old instance. Hence, instead of a Pareto-optimal solution $(x_1, \dots, x_n)$ we now have the Pareto-optimal solutions $(x_1, \dots, x_n, y_1, \dots, y_d)$ for any choice $(y_1, \dots, y_d) \in \{0,1\}^d$. That is, we created 2^d clones of the the original Pareto set. This is what we call the *copy step.*

We repeat this copy step several times to increase the size of the Pareto set. Due to this, the profits of the additional objects grow exponentially and finally exceed 1, which is the limit according to our input model. To deal with inadmissibly high profits, we break the large objects into smaller objects whose profits are again in the interval $[0,1]$. As these small objects together are supposed to behave like the large object, we adapt the set of solutions in such a way that we only allow to choose all or none of them. This is what we call the *split step.*

The more copy steps we perform, the larger the Pareto set becomes. On the other hand, each copy step also creates more and larger objects. These objects have to be broken into many small objects, which increases the number of objects used for the instance. As we want to maximize the size of the Pareto set as a function of the number of objects, there is a trade-off between creating a large Pareto set and creating not too many objects. This problem and the question how to choose the initial objects and their profits will be handled in the analysis.

7.4 Properties of (Weak) Pareto-optimal Solutions

In this section we will identify the main properties of (*weakly*) *Pareto-optimal solutions* that lay the foundation for all variants of the Witness function. In the model without zero-preserving perturbations we only need properties of Pareto optima. In the model with zero-preserving perturbations, however, much more work has to be done and there we need the notion of weak Pareto optimality.

We start with an observation that is valid for both Pareto-optimal solutions and weak Pareto-optimal solutions.

Proposition 7.4.1. *Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a set of solutions, let $f_1, \dots, f_d \colon \mathcal{S} \to \mathbb{R}$ be functions, let $x^\star$ be a (weak) Pareto optimum with respect to $\mathcal{S}$ and $\{f_1, \dots, f_d\}$, and let $\mathcal{S}' \subseteq \mathcal{S}$ be a subset of solutions that contains $x^\star$. Then $x^\star$ is (weakly) Pareto-optimal with respect to $\mathcal{S}'$ and $\{f_1, \dots, f_d\}$.*

The core idea of the Witness functions is given by the following lemma.

Lemma 7.4.2. *Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a set of solutions, let $f_1, \dots, f_{t+1} \colon \mathcal{S} \to \mathbb{R}$, $t \geq 1$, be functions, and let $x^\star$ be a weak Pareto optimum with respect to $\mathcal{S}$ and $\{f_1, \dots, f_{t+1}\}$. We consider the set $\mathcal{C} \subseteq \mathcal{S}$ of solutions that dominate $x^\star$ strongly with respect to $\{f_1, \dots, f_t\}$.*

(I) If $\mathcal{C} = \emptyset$, then $x^\star$ is weakly Pareto-optimal with respect to $\mathcal{S}$ and $\{f_1, \dots, f_t\}$.

(II) If $\mathcal{C} \neq \emptyset$, then $f_{t+1}(x^\star) \leq \min_{x \in \mathcal{C}} f_{t+1}(x) =: \hat{f}$. Furthermore, if $f_{t+1}(x^\star) < \hat{f}$, then $x^\star$ is weakly Pareto-optimal with respect to $\mathcal{R} := \{x \in \mathcal{S} : f_{t+1}(x) < \hat{f}\}$ and $\{f_1, \dots, f_t\}$.

Proof. Claim (I) of Lemma 7.4.2 holds due to the definition of weak Pareto optimality. Let us consider Claim (II). If the inequality $f_{t+1}(x^\star) \leq \hat{f}$ does not hold, then $\hat{x} = \arg\min_{x \in \mathcal{C}} f_{t+1}(x)$ dominates $x^\star$ strongly with respect to $\{f_1, \dots, f_{t+1}\}$. This is a contradiction since $x^\star$ is weakly Pareto-optimal with respect to $\mathcal{S}$ and $\{f_1, \dots, f_{t+1}\}$.

Now let us show that $x^\star$ is weakly Pareto-optimal with respect to $\mathcal{R}$ and $\{f_1, \dots, f_t\}$ if $f_{t+1}(x^\star) < \hat{f}$. The condition ensures that $x^\star \in \mathcal{R}$. Assume, for the contrary, that there exists a $y \in \mathcal{R}$ that dominates $x^\star$ strongly with respect to $\{f_1, \dots, f_t\}$. Since $\mathcal{R} \subseteq \mathcal{S}$, this implies $y \in \mathcal{C}$. Due to $y \in \mathcal{R}$ we obtain the contradiction $f_{t+1}(y) < \hat{f} \leq f_{t+1}(y)$, where the second inequality follows from the definition of $\hat{f}$ and $y \in \mathcal{C}$. □

If the functions $f_1, \dots, f_t$ in Lemma 7.4.2 are injective, we can also obtain a statement about Pareto optima.

Corollary 7.4.3. *Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a set of solutions, let $f_1, \dots, f_{t+1} \colon \mathcal{S} \to \mathbb{R}$, $t \geq 1$, be functions, where $f_1, \dots, f_t$ are injective, and let $x^\star$ be a Pareto optimum with respect to $\mathcal{S}$ and $\{f_1, \dots, f_{t+1}\}$. We consider the set $\mathcal{C} \subseteq \mathcal{S}$ of solutions that dominate $x^\star$ strongly with respect to $\{f_1, \dots, f_t\}$.*

(I) If $\mathcal{C} = \emptyset$, then $x^\star$ is Pareto-optimal with respect to $\mathcal{S}$ and $\{f_1, \dots, f_t\}$.

(II) If $\mathcal{C} \neq \emptyset$, then $f_{t+1}(x^\star) < \min_{x \in \mathcal{C}} f_{t+1}(x) =: \hat{f}$. Furthermore, $x^\star$ is Pareto-optimal with respect to $\mathcal{R} := \{x \in \mathcal{S} : f_{t+1}(x) < \hat{f}\}$ and $\{f_1, \dots, f_t\}$.

Proof. First of all we observe that a solution $y \in \mathcal{S}$ dominates $x^\star$ with respect to $\{f_1, \dots, f_t\}$ if and only if y dominates $x^\star$ strongly with respect to $\{f_1, \dots, f_t\}$. This is due to the injectivity of the functions $f_1, \dots, f_t$. Consequently, Claim (I) follows from the definition of Pareto optimality. Let us consider Claim (II). Assume for the contrary that $\hat{f} \le f_{t+1}(x^\star)$. In this case, the solution $\hat{x} = \arg\min_{x \in \mathcal{C}} f_{t+1}(x)$ would dominate $x^\star$ with respect to $\{f_1, \dots, f_{t+1}\}$ contradicting the assumption that $x^\star$ is Pareto-optimal. Hence, $f_{t+1}(x^\star) < \hat{f}$.

Due to Lemma 7.4.2, $x^\star$ is weakly Pareto-optimal with respect to $\mathcal{R}$ and $\{f_1, \dots, f_t\}$ because any Pareto optimum is also a weak Pareto optimum. As these functions are injective, $x^\star$ is even Pareto-optimal with respect to $\mathcal{R}$ and $\{f_1, \dots, f_t\}$. □

For the model with zero-preserving perturbations we need one more lemma that allows us to handle non-injectivity appropriately.

Lemma 7.4.4. *Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a set of solutions, let $f_1, \dots, f_{t+1} \colon \mathcal{S} \to \mathbb{R}$, $t \ge 1$, be functions, and let $x^\star$ be a Pareto optimum with respect to $\mathcal{S}$ and $\{f_1, \dots, f_{t+1}\}$. Furthermore, let $K \subseteq [t+1]$ be a tuple of indices and let $\mathcal{S}'$ be a subset of $\{x \in \mathcal{S} : f_k(x) = f_k(x^\star)$ for all $k \in K\}$. Then $x^\star$ is Pareto-optimal with respect to $\mathcal{S}'$ and $\{f_k : k \in [t+1] \setminus K\}$.*

Proof. Assume, for the contrary, that $x^\star$ is not Pareto-optimal. Then there exists a solution $y \in \mathcal{S}'$ such that y dominates $x^\star$ with respect to $\{f_k : k \in [t+1] \setminus K\}$. Since $f_k(y) = f_k(x^\star)$ for all $k \in K$, solution y also dominates $x^\star$ with respect to $\{f_1, \dots, f_{t+1}\}$. This contradicts the assumption that $x^\star$ is Pareto-optimal. □

7.5 Smoothed Number of Pareto-optimal Solutions

Throughout this section we consider the model without zero-preserving perturbations. To prove Theorem 1.5.1 we assume w.l.o.g. that $n \ge d+1$ and consider the function given as Algorithm 5 which we call the Witness function. It is very similar to the one suggested by Moitra and O'Donnell, but with an additional parameter I. This parameter is a tuple of forbidden indices: it restricts the set of indices we are allowed to choose from. For the analysis of the smoothed number of Pareto-optimal solutions we will set $I = ()$. The parameter becomes important in the next section when we analyze higher moments.

Let us give some remarks about the Witness function. Note, that $\mathcal{C}_0 = \mathcal{R}_1$ since $V^{1\dots t} z < V^{1\dots t} x$ is no restriction if $t = 0$. In Line 6 ties are broken by taking the lexicographically first solution $x^{(t)}$. For $t \ge 1$ the index i_t in Line 8 exists because $V^1 x^{(t)} < V^1 x$ which implies $x^{(t)} \ne x$.

Unless stated otherwise, we assume that the OK-*event* occurs. That means that $|V^k \cdot (y - z)| \ge \varepsilon$ for every $k \in [d]$ and for any two distinct solutions $y \ne z \in \mathcal{S}$ and that for any $k \in [d]$ there is an index $i \in [n]$ such that $|V_i^k| < 1$. The first property

Algorithm 5 Witness(V, x, I)

```
1: Set $I_{d+1} = I$.
2: Set $\mathcal{R}_{d+1} = \mathcal{S}_{I_{d+1}}(x)$.
3: for $t = d, d-1, \dots, 0$ do
4:     Set $\mathcal{C}_t = \{z \in \mathcal{R}_{t+1} : V^{1\dots t}z < V^{1\dots t}x\}$.
5:     if $\mathcal{C}_t \neq \emptyset$ then
6:         Set $x^{(t)} = \arg\min\{V^{t+1}z : z \in \mathcal{C}_t\}$.
7:         if $t = 0$ then return $x^{(t)}$
8:         Set $i_t = \min\{i \in [n] : x^{(t)}_{i_t} \neq x_{i_t}\}$.
9:         Set $I_t = I_{t+1} \cup (i_t)$.
10:        Set $\mathcal{R}_t = \{z \in \mathcal{R}_{t+1} : V^{t+1}z < V^{t+1}x^{(t)}\} \cap \mathcal{S}_{I_t}(x)$.
11:    else
12:        Set $i_t = \min([n] \setminus I_{t+1})$.
13:        Set $I_t = I_{t+1} \cup (i_t)$.
14:        Set $x^{(t)}_i = \begin{cases} \min(\{0, \dots, \mathcal{K}\} \setminus \{x_i\}) & \text{if } i = i_t\,, \\ x_i & \text{otherwise}\,. \end{cases}$
15:        Set $\mathcal{R}_t = \mathcal{R}_{t+1} \cap \mathcal{S}_{I_t}(x)$.
16:    end if
17: end for
18: return $(\perp, \dots, \perp)$
```

ensures, amongst others, that there is a unique $\arg\min$ in Line 6 and that the functions $V^1, \dots, V^d$ are injective. The latter property, which holds with probability 1, ensures that $B_V(x) \in \mathbb{B}_\varepsilon$ for any vector $x \in \{-\mathcal{K}, \dots, \mathcal{K}\}^n$. Later we will see that the OK-event occurs with sufficiently high probability.

Before we start to analyze the Witness function, let us discuss the differences between the function Witness(V, B) described in Section 7.3 and the function Witness(V, x, I) given as Algorithm 5. As described in Section 7.3 for the illustrative case $d = 2$, the parameters B and x play exactly the same role if $B = B_V(x)$ assuming that the OK-event holds. As stated earlier, the additional parameter I in the function Witness(V, x, I) has no meaning for the analysis of the first moment. To prove Theorem 1.5.1, we simply set it to the empty tuple.

In the remainder of this section we only consider the case that x is Pareto-optimal, that I is an arbitrary index tuple that contains pairwise distinct indices, and that the number $|I|$ of indices contained in I is at most $n - (d + 1)$. This ensures that the indices $i_0, \dots, i_d$ exist.

Lemma 7.5.1. *The call* Witness(V, x, I) *returns the vector* $x^{(0)} = x$.

Proof. We show the following claim by induction on t.

Claim 3. *For any $t \in [d+1]$, solution x is Pareto-optimal with respect to $\mathcal{R}_t$ and $\{V^1, \dots, V^t\}$.*

If Claim 3 holds, then for $t = 1$ we obtain that x is Pareto-optimal with respect to $\mathcal{R}_1$ and $\{V^1\}$. In particular, $x \in \mathcal{R}_1 = \mathcal{C}_0 \neq \emptyset$, i.e., $x^{(0)} = x$. This solution will be returned in iteration $t = 0$.

Note, that the functions $V^1, \dots, V^d$ are injective due to the assumption that the OK-event occurs. This allows us to apply Corollary 7.4.3. Recalling that $x \in \mathcal{S}_{I'}(x)$ for any index tuple I', Claim 3 is true for $t = d+1$ by assumption and due to Proposition 7.4.1.

Now let us assume that the claim holds for some value $t+1$ and consider set $\mathcal{C}_t$. We distinguish between two cases. If $\mathcal{C}_t = \emptyset$, then $\mathcal{R}_t = \mathcal{R}_{t+1} \cap \mathcal{S}_{I_t}(x)$ and the claim follows from the induction hypothesis and from Corollary 7.4.3, Claim (I), and Proposition 7.4.1. If $\mathcal{C}_t \neq \emptyset$, then $\mathcal{R}_t = \mathcal{R}'_t \cap \mathcal{S}_{I_t}(x)$ for $\mathcal{R}'_t = \{z \in \mathcal{R}_{t+1} : V^{t+1}z < V^{t+1}x^{(t)}\}$. Hence, the claim follows from the induction hypothesis and from Corollary 7.4.3, Claim (II), and Proposition 7.4.1. □

At a first glance it seems odd to compute a solution x by calling a function with x as parameter. However, we will see that not all information about x is required to execute the call Witness(V, x, I). To be a bit more precise, the indices $i_0, \dots, i_d$ and the entries at the positions $i \in I \cup (i_0, \dots, i_d)$ of the vectors $x^{(t)}$ constructed during the execution of the Witness function suffice to simulate the execution of Witness without knowing x completely (see Lemma 7.5.4). We will call these information a *certificate* (see Definition 7.5.2). For technical reasons we will assume that we have some additional information about x.

For our purpose it is not necessary to know how to obtain the required information about x to reconstruct it. It suffices to know that the set of possible certificates is sufficiently small (see Lemma 7.5.7) and that for at least one of them the simulation of the execution of Witness returns x (see Lemma 7.5.4). This is one crucial property which will help us to bound the expected number of Pareto-optimal solutions.

Definition 7.5.2. Let $x^{(0)}, \dots, x^{(d)}$ be the vectors and I_1 be the index tuple constructed during the call Witness(V, x, I) and set $i_0 = \min\big([n] \setminus I_1\big)$ and $I_0 = I_1 \cup (i_0)$. We call the pair (I_0, A_0) for $A_0 = \big[x^{(d)}, \dots, x^{(0)}\big]$ the *(V, I)-certificate* of x. The pair (I_0, A) for $A = A_0|_{I_0}$ is called the *restricted (V, I)-certificate* of x. We call a pair (I', A') a *(restricted) I-certificate*, if there exist a realization V for which $\mathrm{OK}(V)$ is true and a Pareto-optimal solution $x \in \mathcal{S}$ such that (I', A') is the (restricted) (V, I)-certificate of x. By $\mathscr{C}(I)$ we denote the set of all restricted I-certificates.

For the analysis of the first moment we only need restricted I-certificates. Our analysis of higher moments requires more knowledge about the vectors $x^{(t)}$ than just the values x_i for $i \in I_0$. The additional indices are, however, depending on further calls of the Witness function which we do not know a priori. This is why we have to define two types of certificates. For the sake of reusability we formulate some statements more general than necessary for this section.

Lemma 7.5.3. *Let V be an arbitrary realization where $\mathrm{OK}(V)$ is true, let x be a Pareto-optimal solution with respect to $\mathcal{S}$ and V, and let (I_0, A) be the restricted (V, I)-certificate of x. Then $I_0 = (j_1, \dots, j_{|I|+d+1})$ consists of pairwise distinct indices and*

$$A = \begin{bmatrix} x_{j_1} & \dots & x_{j_{|I|}} & \overline{x_{j_{|I|+1}}} & * & \dots & * \\ \vdots & & \vdots & x_{j_{|I|+1}} & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \ddots & \overline{x_{j_{|I|+d}}} & * \\ x_{j_1} & \dots & x_{j_{|I|}} & x_{j_{|I|+1}} & \dots & x_{j_{|I|+d}} & x_{j_{|I|+d+1}} \end{bmatrix}^{\mathrm{T}} \in \{0, \dots, \mathcal{K}\}^{(|I|+d+1)\times(d+1)} ,$$

where each '$*$'-entry can be an arbitrary value from $\{0, \dots, \mathcal{K}\}$ independently of the other '$*$'-entries and where $\overline{z}$ for $z \in \{0, \dots, \mathcal{K}\}$ can be an arbitrary value from $\{0, \dots, \mathcal{K}\} \setminus \{z\}$.

Proof. Lemma 7.5.1 implies that the last column of A equals $x^{(0)}_{I_0} = x|_{I_0}$. Hence, it suffices to consider the first d columns of A. Note, that $I = (j_1, \dots, j_{|I|})$ and $j_{|I|+1}, \dots, j_{|I|+d+1} = i_d, \dots, i_0$. The construction of the sets $\mathcal{R}_t$ yields $\mathcal{R}_t \subseteq \mathcal{S}_{I_t}(x)$ (see Lines 2, 10, and 15). Index i_t is always chosen such that $i_t \notin I_{t+1}$: If it is constructed in Line 8, then $x^{(t)}_{i_t} \neq x_{i_t}$. Since in that case we have $x^{(t)} \in \mathcal{C}_t \subseteq \mathcal{R}_{t+1} \subseteq \mathcal{S}_{I_{t+1}}(x)$, index i_t cannot be an element of I_{t+1}. In Line 12, index i_t is explicitely constructed such that $i_t \notin I_{t+1}$. The same argument holds for index i_0. Hence, the indices of I_0 are pairwise distinct.

Now, consider the column of A corresponding to vector $x^{(t)}$ for $t \in [d]$. If $\mathcal{C}_t = \emptyset$, then the form of the column follows directly from the construction of $x^{(t)}$ in Line 14 and from the fact that the indices of I_0 are pairwise distinct. If $\mathcal{C}_t \neq \emptyset$, then $x^{(t)} \in \mathcal{C}_t \subseteq \mathcal{R}_{t+1} \subseteq \mathcal{S}_{I_{t+1}}(x)$, i.e., $x^{(t)}$ agrees with x in all indices $i \in I_{t+1}$. By the choice of i_t in Line 8 we get $x^{(t)}_{i_t} \in \{0, \dots, \mathcal{K}\} \setminus \{x_{i_t}\}$. This concludes the proof. □

Let (I_0, A_0) be the (V, I)-certificate of x and let $J \supseteq I_0$ be a tuple of pairwise distinct indices. We consider the following variant of the Witness function given as Algorithm 6 that uses information about x given by the index tuple J, the matrix $A = A_0|_J$ with columns $a^{(d)}, \dots, a^{(0)}$, a shift vector u and the ε-box $B = B_V(x - u)$ instead of vector x itself. The meaning of the shift vector will become clear when we analyze the probability of certain events. We will see that not all information about V needs to be revealed to execute the new Witness function, i.e., we have some randomness left which we can use later. With the choice of the shift vector we can control which information has to be revealed for executing the Witness function.

Lemma 7.5.4. *Let (I_0, A_0) be the (V, I)-certificate of x, let $J \supseteq I_0$ be an arbitrary tuple of pairwise distinct indices, let $A = A_0|_J$, let $u \in \{0, \dots, \mathcal{K}\}^n$ be an arbitrary vector, and let $B = B_V(x - u)$. Then the call* Witness(V, J, A, B, u) *returns vector x.*

Before we give a formal proof of Lemma 7.5.4 we try to give some intuition for it. Instead of considering the whole set $\mathcal{S}$ of solutions we restrict it to vectors that look like

Algorithm 6 Witness(V, J, A, B, u)

```
1: Let $b$ be the corner of $B$.
2: Set $\mathcal{R}_{d+1} = \bigcup_{s=0}^{d} \mathcal{S}_J\big(a^{(s)}\big)$.
3: for $t = d, d-1, \ldots, 0$ do
4:     Set $\mathcal{C}_t = \{z \in \mathcal{R}_{t+1} :\ V^{1\ldots t} \cdot (z - u) \leq b|_{1\ldots t}\} \cap \mathcal{S}_J\big(a^{(t)}\big)$.
5:     if $\mathcal{C}_t \neq \emptyset$ then
6:         Set $x^{(t)} = \arg\min\{V^{t+1}z :\ z \in \mathcal{C}_t\}$.
7:         if $t = 0$ then return $x^{(t)}$
8:         Set $\mathcal{R}_t = \{z \in \mathcal{R}_{t+1} :\ V^{t+1}z < V^{t+1}x^{(t)}\} \cap \bigcup_{s=0}^{t-1} \mathcal{S}_J\big(a^{(s)}\big)$.
9:     else
10:        Set $x^{(t)} = (\perp, \ldots, \perp)$.
11:        Set $\mathcal{R}_t = \mathcal{R}_{t+1} \cap \bigcup_{s=0}^{t-1} \mathcal{S}_J\big(a^{(s)}\big)$.
12:    end if
13: end for
14: return $x^{(0)}$
```

the vectors we want to reconstruct in the next rounds, i.e., we intersect the current set with the set $\bigcup_{s=0}^{t-1} \mathcal{S}_J\big(a^{(s)}\big)$ in round t. That way we only deal with subsets of the original sets, but we do not lose the vectors we want to reconstruct since $J \supseteq I_0$. This restriction to the essential candidates of solutions allows us to execute this variant of the Witness function with only partial information about V.

Proof. Let $\mathcal{R}'_t$, $\mathcal{C}'_t$, and $x'^{(t)}$ denote the sets and vectors constructed during the execution of the call Witness(V, J, A, B, u) and let $\mathcal{R}_t$, $\mathcal{C}_t$, and $x^{(t)}$ denote the sets and vectors constructed during the execution of call Witness(V, x, I). We prove the following claims simultaneously by induction.

Claim 4. *$\mathcal{R}'_t \subseteq \mathcal{R}_t$ for any $t \in [d+1]$.*

Claim 5. *$x'^{(t)} = x^{(t)}$ for any $t \in [d]_0$ for which $\mathcal{C}_t \neq \emptyset$.*

Claim 6. *$x^{(s)} \in \mathcal{R}'_t$ for any $t \in [d+1]$ and for any $s \in [t-1]_0$ for which $\mathcal{C}_s \neq \emptyset$.*

With these claims Lemma 7.5.4 follows immediately: Since $x^{(0)} = x$ and $\mathcal{C}_0 \neq \emptyset$ due to Lemma 7.5.1, we obtain $x'^{(0)} = x^{(0)}$ (Claim 5). Hence, the call Witness(V, J, A, B, u) returns $x'^{(0)} = x$.

Let us first focus on the shift vector u and compare Line 4 of the first Witness function (Algorithm 5) with Line 4 of the second Witness function (Algorithm 6). The main difference is that in the first version we have the restriction $V^{1\ldots t}z < V^{1\ldots t}x$, whereas in the second version we seek for solutions z for which $V^{1\ldots t} \cdot (z - u) \leq b|_{1\ldots t}$. As b is the corner of the ε-box $B = B_V(x - u)$, these restrictions are equivalent for solutions $z \in \mathcal{S}$.

This is due to the equivalences

$$V^{1\dots t} \cdot (z-u) \le b|_{1\dots t} \iff V^{1\dots t} \cdot (z-u) < V^{1\dots t} \cdot (x-u) \iff V^{1\dots t} z < V^{1\dots t} x\,.$$

The first equivalence is due to the occurrence of the OK-event. Now, we prove the statements by downward induction over t. Let $t = d+1$. Lemma 7.5.3 yields $a^{(s)}\big|_I = x|_I$ for any $s \in [d]_0$, i.e., $\bigcup_{s=0}^{d} \mathcal{S}_J\big(a^{(s)}\big) \subseteq \mathcal{S}_I(x)$ because $I \subseteq I_0 \subseteq J$. Consequently, $\mathcal{R}'_{d+1} \subseteq \mathcal{R}_{d+1}$ (Claim 4). Consider an arbitrary index $s \in [(d+1)-1]_0$ for which $\mathcal{C}_s \neq \emptyset$. Then $x^{(s)} \in \mathcal{C}_s \subseteq \mathcal{R}_{s+1} \subseteq \mathcal{S}$ (see Line 6) and, thus, $x^{(s)} \in \mathcal{S}_J\big(a^{(s)}\big)$. Hence, $x^{(s)} \in \mathcal{R}'_{d+1}$ (Claim 6).

For the induction step let $t \le d$. By the observation above we have $\mathcal{C}'_t = \{z \in \mathcal{R}'_{t+1} : V^{1\dots t} z < V^{1\dots t} x\} \cap \mathcal{S}_J\big(a^{(t)}\big)$ and $\mathcal{C}_t = \{z \in \mathcal{R}_{t+1} : V^{1\dots t} z < V^{1\dots t} x\}$. Since $\mathcal{R}'_{t+1} \subseteq \mathcal{R}_{t+1}$, we obtain $\mathcal{C}'_t \subseteq \mathcal{C}_t$. We first consider the case $\mathcal{C}_t = \emptyset$ which implies $\mathcal{C}'_t = \emptyset$ and $t \ge 1$ in accordance with Lemma 7.5.1 since $\mathcal{C}_0 \neq \emptyset$. Then $\mathcal{R}'_t = \mathcal{R}'_{t+1} \cap \bigcup_{s=0}^{t-1} \mathcal{S}_J\big(a^{(s)}\big)$ and $\mathcal{R}_t = \mathcal{R}_{t+1} \cap \mathcal{S}_{I_t}(x)$. According to Lemma 7.5.3, all vectors $x^{(0)}, \dots, x^{(t-1)}$ agree with x on the indices $i \in I_t$ as $I_t \subseteq I_0 \subseteq J$. Thus, $\bigcup_{s=0}^{t-1} \mathcal{S}_J\big(a^{(s)}\big) \subseteq \mathcal{S}_{I_t}(x)$. As $\mathcal{R}'_{t+1} \subseteq \mathcal{R}_{t+1}$ due to the induction hypothesis, Claim 4, we obtain $\mathcal{R}'_t \subseteq \mathcal{R}_t$ (Claim 4). For Claim 5 nothing has to be shown here. Let $s \in [t-1]_0$ be an index for which $\mathcal{C}_s \neq \emptyset$. Then $x^{(s)} \in \mathcal{R}'_{t+1}$ by Claim 6 of the induction hypothesis, $x^{(s)} \in \mathcal{S}_J\big(a^{(s)}\big)$, and consequently $x^{(s)} \in \mathcal{R}'_t$ (Claim 6).

Finally, let us consider the case $\mathcal{C}_t \neq \emptyset$. Claim 6 of the induction hypothesis yields $x^{(t)} \in \mathcal{R}'_{t+1}$. Since $x^{(t)} \in \mathcal{S}_J\big(a^{(t)}\big)$ and $V^{1\dots t} x^{(t)} < V^{1\dots t} x$, also $x^{(t)} \in \mathcal{C}'_t$ and, thus, $\mathcal{C}'_t \neq \emptyset$. Hence, $x'^{(t)} = x^{(t)}$ as $\mathcal{C}'_t \subseteq \mathcal{C}_t$ (Claim 5). The remaining claims have only to be validated if $t \ge 1$. Then $\mathcal{R}'_t = \{z \in \mathcal{R}'_{t+1} : V^{t+1} z < V^{t+1} x^{(t)}\} \cap \bigcup_{s=0}^{t-1} \mathcal{S}_J\big(a^{(s)}\big)$, because $x'^{(t)} = x^{(t)}$, and $\mathcal{R}_t = \{z \in \mathcal{R}_{t+1} : V^{t+1} z < V^{t+1} x^{(t)}\} \cap \mathcal{S}_{I_t}(x)$. With the same argument used for the case $\mathcal{C}_t = \emptyset$ we obtain $\mathcal{R}'_{t+1} \cap \bigcup_{s=0}^{t-1} \mathcal{S}_J\big(a^{(s)}\big) \subseteq \mathcal{R}_{t+1} \cap \mathcal{S}_{I_t}(x)$ and, hence, $\mathcal{R}'_t \subseteq \mathcal{R}_t$ (Claim 4). Consider an arbitrary index $s \in [t-1]_0$ for which $\mathcal{C}_s \neq \emptyset$. Then $x^{(s)} \in \mathcal{C}_s \subseteq \mathcal{R}_{s+1} \subseteq \mathcal{R}_t$. In particular, $V^{t+1} x^{(s)} < V^{t+1} x^{(t)}$ (see Line 10) and, hence, $V^{t+1} x^{(s)} < V^{t+1} x'^{(t)}$ because $x'^{(t)} = x^{(t)}$. Furthermore, $x^{(s)} \in \mathcal{R}'_{t+1}$ due to the induction hypothesis, Claim 6, and $x^{(s)} \in \mathcal{S}_J\big(a^{(s)}\big)$. Consequently, $x^{(s)} \in \mathcal{R}'_t$ (Claim 6). $\square$

As mentioned earlier, with the shift vector u we control which information of V has to be revealed to execute the call $\mathsf{Witness}(V, J, A, B, u)$. While Lemma 7.5.4 holds for any vector u, we have to choose u carefully for our probabilistic analysis to work. We will see that the choice $u^\star = u^\star(J, A)$, given by

$$u^\star_i = \begin{cases} |x_i - 1| & \text{if } i = i_0\,, \\ x_i & \text{if } i \in J \setminus (i_0)\,, \\ 0 & \text{otherwise}\,, \end{cases} \tag{7.1}$$

is appropriate since $x_i - u^\star_i = 0$ for all $i \in J \setminus (i_0)$ and $|x_{i_0} - u^\star_{i_0}| = 1$ (cf. Lemma 7.5.9). Recall, that i_0 is the index that has been added to I_1 in the definition of the (V, I)-

certificate to obtain I_0 and note, that $u^\star \in \{0, \dots, \mathcal{K}\}^n$. Moreover, the values x_i are given in the last column of A for any index $i \in J$ (see Lemma 7.5.3). Hence, if (I_0, A_0) is the (V, I)-certificate of x, then vector $u^\star$ can be defined when a tuple $J \supseteq I_0$ and the matrix $A = A_0|_J$ are known; we do not have to know the solution x itself.

For bounding the number of Pareto-optimal solutions consider the functions $\chi_{I_0,A,B}(V)$ parameterized by an arbitrary restricted I-certificate (I_0, A), and an arbitrary ε-box $B \in \mathbb{B}_\varepsilon$, defined as follows: $\chi_{I_0,A,B}(V) = 1$ if the call $\mathsf{Witness}(V, I_0, A, B, u^\star(I_0, A))$ returns a solution $x' \in \mathcal{S}$ for which $B_V\big(x' - u^\star(I_0, A)\big) = B$, and $\chi_{I_0,A,B}(V) = 0$ otherwise.

Corollary 7.5.5. *Assume that* OK(V) *is true. Then the number* PO(V) *of Pareto-optimal solutions is at most* $\sum_{(I_0,A)\in\mathscr{C}(I)} \sum_{B\in\mathbb{B}_\varepsilon} \chi_{I_0,A,B}(V)$.

Proof. Let x be a Pareto-optimal solution, let (I_0, A) be the restricted (V, I)-certificate of x, and let $B = B_V\big(x - u^\star(I_0, A)\big) \in \mathbb{B}_\varepsilon$ be the ε-box of $x - u^\star(I_0, A)$. Due to Lemma 7.5.4, the call $\mathsf{Witness}(V, I_0, A, B, u^\star(I_0, A))$ returns vector x. Hence, $\chi_{I_0,A,B}(V) = 1$. It remains to show that this assignment $x \mapsto (I_0, A, B)$ given in the previous lines is injective. Otherwise we would count the occurence of two distinct Pareto-optimal solutions x_1 and x_2 only once in the sum stated in Corollary 7.5.5.

Let x_1 and x_2 be distinct Pareto-optimal solutions and let $(I_0^{(1)}, A_1)$ and $(I_0^{(2)}, A_2)$ be the restricted (V, I)-certificates of x_1 and x_2, respectively. If $(I_0^{(1)}, A_1) \neq (I_0^{(2)}, A_2)$, then x_1 and x_2 are mapped to distinct triplets. Otherwise, $u^\star(I_0^{(1)}, A_1) = u^\star(I_0^{(2)}, A_2)$ and, hence, $B_V\big(x_1 - u^\star(I_0^{(1)}, A_1)\big) \neq B_V\big(x_2 - u^\star(I_0^{(2)}, A_2)\big)$ because OK(V) is true and $x_1 \neq x_2$. Consequently, also in this case x_1 and x_2 are mapped to distinct triplets. □

Corollary 7.5.5 immediately implies a bound on the expected number of Pareto-optimal solutions.

Corollary 7.5.6. *The expected number of Pareto-optimal solutions is bounded by*

$$\mathbf{E}_V[\mathrm{PO}(V)] \leq \sum_{(I_0,A)\in\mathscr{C}(I)} \sum_{B\in\mathbb{B}_\varepsilon} \mathbf{Pr}_V[E_{I_0,A,B}] + (\mathcal{K}+1)^n \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}(V)}\right],$$

where $E_{I_0,A,B}$ *denotes the event that the call* $\mathsf{Witness}(V, I_0, A, B, u^\star(I_0, A))$ *returns a vector* x' *for which* $B_V\big(x' - u^\star(I_0, A)\big) = B$.

Proof. By applying Corollary 7.5.5, we obtain

$$\begin{aligned}
&\mathbf{E}_V\left[\mathrm{PO}(V)\right] \\
&= \mathbf{E}_V\left[\mathrm{PO}(V) \mid \mathrm{OK}(V)\right] \cdot \mathbf{Pr}_V\left[\mathrm{OK}(V)\right] + \mathbf{E}_V\left[\mathrm{PO}(V) \mid \overline{\mathrm{OK}(V)}\right] \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}(V)}\right] \\
&\leq \mathbf{E}_V\left[\sum_{(I_0,A)\in\mathscr{C}(I)} \sum_{B\in\mathbb{B}_\varepsilon} \chi_{I_0,A,B}(V) \,\middle|\, \mathrm{OK}(V)\right] \cdot \mathbf{Pr}_V\left[\mathrm{OK}(V)\right] + |\mathcal{S}| \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}(V)}\right]
\end{aligned}$$

$$\leq \mathbf{E}_V\left[\sum_{(I_0,A)\in\mathscr{C}(I)}\sum_{B\in\mathbb{B}_\varepsilon}\chi_{I_0,A,B}(V)\right]+(\mathcal{K}+1)^n\cdot\mathbf{Pr}_V\left[\overline{\mathrm{OK}(V)}\right]$$
$$=\sum_{(I_0,A)\in\mathscr{C}(I)}\sum_{B\in\mathbb{B}_\varepsilon}\mathbf{Pr}_V\left[E_{I_0,A,B}\right]+(\mathcal{K}+1)^n\cdot\mathbf{Pr}_V\left[\overline{\mathrm{OK}(V)}\right]. \qquad \square$$

We will see that the first term of the sum in Corollary 7.5.6 can be bounded independently of ε and that the limit of the second term tends to 0 for $\varepsilon \to 0$. First of all, we analyze the size of the restricted certificate space.

Lemma 7.5.7. *The size of the restricted certificate space $\mathscr{C}(I)$ for $I = ()$ is bounded by $|\mathscr{C}(I)| \leq (\mathcal{K}+1)^{(d+1)^2}n^d$.*

Proof. If $\mathrm{OK}(V)$ is true and if x is Pareto-optimal with respect to V, then exactly d indices $i_1,\dots,i_d$ are created during the execution of the call $\mathsf{Witness}(V,x,I)$. The index i_0 is determined deterministicly depending on the indices $i_1,\dots,i_d$. Matrix A of any restricted I-certificate (I_0,A) is a $(d+1)\times(d+1)$-matrix with entries from $\{0,\dots,\mathcal{K}\}$. Hence, the number of possible restricted I-certificates is bounded by $(\mathcal{K}+1)^{(d+1)^2}n^d$. $\square$

Let us now fix an arbitrary I-certificate (I_0,A_0), a tuple $J \supseteq I_0$, and an ε-box $B \in \mathbb{B}_\varepsilon$. We want to analyze the probability $\mathbf{Pr}_V\left[E_{J,A,B}\right]$ where $A = A_0|_J$. By V_J and $V_{\overline{J}}$ we denote the part of the matrix $V^{1\dots d}$ that belongs to the indices $i \in J$ and to the indices $i \notin J$, respectively. We apply the principle of deferred decisions and assume that $V_{\overline{J}}$ is fixed as well, i.e., we will only exploit the randomness of V_J.

As motivated above, the call $\mathsf{Witness}(V,J,A,B,u)$ can be executed without the full knowledge of V_J. To formalize this, we introduce matrices Q_k that describe the linear combinations of V_J^k that suffice to be known:

$$Q_k = \left[p^{(d)},\dots,p^{(k)},p^{(k-2)}-p^{(k-1)},\dots,p^{(0)}-p^{(k-1)}\right] \in \{-\mathcal{K},\dots,\mathcal{K}\}^{|J|\times d} \qquad (7.2)$$

for $p^{(t)} = p^{(t)}(J,A,u) = a^{(t)} - u|_J$ where $a^{(t)}$ are the columns of matrix $A = \left[a^{(d)},\dots,a^{(0)}\right]$ and $t \in [d]_0$. Note, that the matrices $Q_k = Q_k(J,A,u)$ depend on the pair (J,A) and on the vector u.

Lemma 7.5.8. *Let $u \in \{0,\dots,\mathcal{K}\}^n$ be an arbitrary shift vector and let U and W be two realizations of V such that $U_{\overline{J}} = W_{\overline{J}}$ and $U_J^k \cdot q = W_J^k \cdot q$ for any index $k \in [d]$ and any column q of the matrix $Q_k(J,A,u)$. Then the calls* $\mathsf{Witness}(U,J,A,B,u)$ *and* $\mathsf{Witness}(W,J,A,B,u)$ *return the same result.*

Lemma 7.5.8 states that for different realizations U_J and W_J of V_J the modified $\mathsf{Witness}$ function outputs the same result. Actually, in the proof we will even see that the complete execution of both calls is identical. This means, that solution x is already determined if these realizations are known. However, there is still randomness left in the objective values $V^1x,\dots,V^dx$ which allows us to bound the probability that x falls into box B (see Corollary 7.5.11).

Proof. We fix an index $k \in [d]$ and analyze which information of V^k_J is required for the execution of the call $\mathsf{Witness}(V, J, A, B, u)$. For the execution of Line 4 we need to know $V^k \cdot (z - u)$ for solutions $z \in \mathcal{S}_J\big(a^{(t)}\big)$ in all rounds $t \geq k$. Since we can assume $V^k_{\overline{J}}$ to be known, this means that $V^k_J \cdot \big(z|_J - u|_J\big) = V^k_J \cdot \big(a^{(t)} - u|_J\big) = V^k_J \cdot p^{(t)}$ must be revealed. For $t \geq k$ vector $p^{(t)}$ is a column of Q_k. The execution of Line 6 does not require further information about V^k_J: The only iteration where we might need information about V^k_J is iteration $t = k - 1$. However, as $\mathcal{C}_t \subseteq \mathcal{S}_J\big(a^{(t)}\big)$, we obtain

$$x^{(t)} = \arg\min\{V^{t+1} z : \, z \in \mathcal{C}_t\} = \arg\min\{V^{t+1}_{\overline{J}} z|_{\overline{J}} : \, z \in \mathcal{C}_t\}$$

because all solutions $z \in \mathcal{C}_t$ agree on the entries with indices $i \in J$. Since $V^{t+1}_{\overline{J}} = V^k_{\overline{J}}$ is known, $x^{(t)}$ can be determined without any further information. Note, that this does not imply that $V^{t+1} x^{(t)}$ is already specified.

It remains to consider Line 8. Only in round $t = k - 1$ we need information about V^k. In that round it suffices to know $V^k_J \cdot \big(z|_J - x^{(t)}|_J\big)$ for any solution $z \in \bigcup_{s=0}^{t-1} \mathcal{S}_J\big(a^{(s)}\big)$. Hence, for $z \in \mathcal{S}_J\big(a^{(s)}\big)$, $s \in [t-1]_0 = [k-2]_0$, the linear combinations

$$V^k_J \cdot \big(z|_J - x^{(t)}|_J\big) = V^k_J \cdot \Big(\big(a^{(s)} - u|_J\big) - \big(a^{(k-1)} - u|_J\big)\Big) = V^k_J \cdot \big(p^{(s)} - p^{(k-1)}\big)$$

must be revealed. For $s \in [k-2]_0$, vector $p^{(s)} - p^{(k-1)}$ is a column of Q_k.

As U and W agree on all necessary information, both calls return the same result. □

We will now see why $u^\star = u^\star(J, A)$ defined in Equation (7.1) is a good shift vector.

Lemma 7.5.9. *Let $Q = \big[\hat{p}^{(d)}, \dots, \hat{p}^{(0)}\big]$ where $\hat{p}^{(t)} = p^{(t)}\big(J, A, u^\star(J, A)\big)\big|_{I_0}$. Then*

$$|Q| = \begin{bmatrix} 0 & \dots & 0 & + & * & \dots & * \\ \vdots & & \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \ddots & + & * \\ 0 & \dots & 0 & 0 & \dots & 0 & 1 \end{bmatrix}^{\mathrm{T}} \in \{0, \dots, \mathcal{K}\}^{(|I|+d+1)\times(d+1)} ,$$

where $|Q|$ denotes the matrix Q' for which $q'_{ij} = |q_{ij}|$. Each '$*$'-entry can be an arbitrary value from $\{0, \dots, \mathcal{K}\}$ independently of the other '$*$'-entries. Similarly, each '$+$'-entry can be an arbitrary value from $\{1, \dots, \mathcal{K}\}$ independently of the other '$+$'-entries.

Proof. Let $I_0 = (j_1, \dots, j_{|I|+d+1})$, i.e., $i_0 = j_{|I|+d+1}$. According to Lemma 7.5.3 and the construction of vector $u^\star$ in Equation (7.1) we obtain

$$Q = \begin{bmatrix} x_{j_1} & \cdots & \cdots & x_{j_1} \\ \vdots & & & \vdots \\ x_{j_{|I|}} & \cdots & \cdots & x_{j_{|I|}} \\ \overline{x_{j_{|I|+1}}} & x_{j_{|I|+1}} & \cdots & x_{j_{|I|+1}} \\ * & \ddots & \ddots & \vdots \\ \vdots & \ddots & \overline{x_{j_{|I|+d}}} & x_{j_{|I|+d}} \\ * & \cdots & * & x_{j_{|I|+d+1}} \end{bmatrix} - \begin{bmatrix} x_{j_1} & \cdots & x_{j_1} \\ \vdots & & \vdots \\ x_{j_{|I|}} & \cdots & x_{j_{|I|}} \\ x_{j_{|I|+1}} & \cdots & x_{j_{|I|+1}} \\ \vdots & & \vdots \\ x_{j_{|I|+d}} & \cdots & x_{j_{|I|+d}} \\ |x_{j_{|I|+d+1}} - 1| & \cdots & |x_{j_{|I|+d+1}} - 1| \end{bmatrix}.$$

The claim follows since $|a-b| \leq \mathcal{K}$, $\overline{a} - a \neq 0$, and $\big|a - |a-1|\big| = 1$ for any $a, b \in \{0, \ldots, \mathcal{K}\}$. □

Lemma 7.5.10. *For any $k \in [d]$ the columns of the matrix $Q_k\big(J, A, u^\star(J,A)\big)$ and the vector $p^{(0)}$ are linearly independent.*

Proof. Let $\hat{p}^{(t)} = p^{(t)}\big|_{I_0}$ for any $t \in [d]_0$. It suffices to show that the columns of the submatrix $\hat{Q}_k = Q_k\big|_{I_0}$ and the vector $\hat{p}^{(0)}$ are linearly independent. Consider the matrix $Q = \big[\hat{p}^{(d)}, \ldots, \hat{p}^{(0)}\big]$. Due to Lemma 7.5.9 the last $d+1$ rows of Q form a lower triangular matrix and the entries on the principal diagonal are from the set $\{-\mathcal{K}, \ldots, \mathcal{K}\} \setminus \{0\}$. Consequently, the vectors $\hat{p}^{(t)}$ are linearly independent. As these vectors are the same as the columns of matrix $\hat{Q}_1$ plus vector $\hat{p}^{(0)}$ (see Equation 7.2), the claim holds for $k=1$. Now let $k \geq 2$. We consider an arbitrary linear combination of the columns of $\hat{Q}_k$ and the vector $\hat{p}^{(0)}$ and show that it is zero if and only if all coefficients are zero.

$$\begin{aligned} 0 &= \sum_{t=k}^{d} \lambda_t \cdot \hat{p}^{(t)} + \sum_{t=0}^{k-2} \lambda_t \cdot \big(\hat{p}^{(t)} - \hat{p}^{(k-1)}\big) + \mu \cdot \hat{p}^{(0)} \\ &= \sum_{t=k}^{d} \lambda_t \cdot \hat{p}^{(t)} + \sum_{t=1}^{k-2} \lambda_t \cdot \hat{p}^{(t)} - \left(\sum_{t=0}^{k-2} \lambda_t\right) \cdot \hat{p}^{(k-1)} + (\lambda_0 + \mu) \cdot \hat{p}^{(0)}. \end{aligned}$$

As the vectors $\hat{p}^{(t)}$ are linearly independent, we first get $\lambda_t = 0$ for $t \in [d] \setminus \{k-1\}$, which yields $\lambda_0 = 0$ due to $\sum_{t=0}^{k-2} \lambda_t = 0$ and, finally, $\mu = 0$ because of $\lambda_0 + \mu = 0$. This concludes the proof. □

Corollary 7.5.11. *For an arbitrary restricted I-certificate (I_0, A) the probability of the event $E_{I_0,A,B}$ is bounded by*

$$\mathbf{Pr}_V\left[E_{I_0,A,B}\right] \leq (2\gamma\mathcal{K})^{\gamma-d}\phi^\gamma\varepsilon^d$$

for $\gamma = d(d+1)$, and by

$$\mathbf{Pr}_V\left[E_{I_0,A,B}\right] \leq 2^d(\gamma\mathcal{K})^{\gamma-d}\phi^d\varepsilon^d$$

if all densities are quasiconcave.

Proof. Event $E_{I_0,A,B}$ occurs if the output of the call $\mathsf{Witness}(V, I_0, A, B, u^\star(I_0, A))$ is a vector x' for which $B_V\big(x' - u^\star(I_0, A)\big) = B$. We apply the principle of deferred decisions and assume that $V|_{\overline{I_0}}$ is arbitrarily fixed. Now let us further assume that the linear combinations of $V^k_{I_0}$ given by the columns of matrix $Q_k = Q_k(I_0, A, u^\star(I_0, A))$ are known for any $k \in [d]$. That is, for some fixed values we consider all realizations of V for which the linear combinations of $V^k_{I_0}$ given by the columns of Q_k equal these values. In accordance with Lemma 7.5.8, vector x' is therefore already determined, i.e., it is the same for all realizations of V that are still under consideration.

The equality $B_V\big(x' - u^\star(I_0, A)\big) = B$ holds if and only if

$$V^k \cdot \big(x' - u^\star(I_0, A)\big) = V^k_{\overline{I_0}} \cdot \big(x' - u^\star(I_0, A)\big)\big|_{\overline{I_0}} + V^k_{I_0} \cdot \big(x' - u^\star(I_0, A)\big)\big|_{I_0} \in (b_k, b_k + \varepsilon]$$

holds for any $k \in [d]$, where $b = (b_1, \dots, b_d)$ is the corner of B. Since

$$\big(x' - u^\star(I_0, A)\big)\big|_{I_0} = a^{(0)} - u^\star(I_0, A)|_{I_0} = p^{(0)}$$

for the vector $p^{(0)} = p^{(0)}\big(I_0, A, u^\star(I_0, A)\big)$, this is equivalent to the event that

$$V^k_{I_0} \cdot p^{(0)} \in (b_k, b_k + \varepsilon] - V^k_{\overline{I_0}} \cdot \big(x' - u^\star(I_0, A)\big)\big|_{\overline{I_0}} =: C_k\,,$$

where C_k is an interval of length ε depending on x' and hence on the linear combinations of V_{I_0} given by the matrices Q_k. By C we denote the d-dimensional hypercube $C = \prod_{k=1}^d C_k$ with side length ε defined by the intervals C_k.

For any $k \in [d]$ let $Q'_k \in \{-\mathcal{K}, \dots, \mathcal{K}\}^{|I_0| \times (d+1)}$ be the matrix consisting of the columns of Q_k and the vector $p^{(0)}$. These matrices form the diagonal blocks of the matrix

$$Q' = \begin{bmatrix} Q'_1 & \mathbb{O} & \dots & \mathbb{O} \\ \mathbb{O} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbb{O} \\ \mathbb{O} & \dots & \mathbb{O} & Q'_d \end{bmatrix} \in \{-\mathcal{K}, \dots, \mathcal{K}\}^{d\cdot(|I|+d+1) \times d\cdot(d+1)}\,.$$

Lemma 7.5.10, applied for $J = I_0$, implies that matrix Q' has full rank. We permute the columns of Q' to obtain a matrix Q whose last d columns belong to the last column of one of the matrices Q'_k. That is, the vectors $(p^{(0)}, 0^{|I_0|}, \dots, 0^{|I_0|}), \dots, (0^{|I_0|}, \dots, 0^{|I_0|}, p^{(0)})$ form the last d columns of Q'. Let the rows of Q be labeled by $Q_{j_1,1}, \dots, Q_{j_m,1}, \dots, Q_{j_1,d}, \dots, Q_{j_m,d}$ assuming that $I_0 = (j_1, \dots, j_m)$. We introduce random variables $X_{j,k} = V^k_j$, $j \in I_0$, $k \in [d]$, labeled in the same fashion as the rows of Q. Event $E_{I_0,A,B}$ holds if and only if the d linear combinations of the variables $X_{j,k}$ given by the last d columns of Q fall into the d-dimensional hypercube C depending on the linear combinations of the variables $X_{j,k}$ given by the remaining columns of Q. The claim follows by applying Theorem 7.9.1 for the matrix Q^{T} and $k = d$ and due to the fact that the number of columns of Q is $\gamma = d\cdot(d+1)$. Hence, $\mathbf{Pr}_V\,[E_{I_0,A,B}] \le (2\gamma\mathcal{K})^{\gamma-d}\phi^\gamma\varepsilon^d$ in general and $\mathbf{Pr}_V\,[E_{I_0,A,B}] \le 2^d(\gamma\mathcal{K})^{\gamma-d}\phi^d\varepsilon^d$ if all densities are quasiconcave. □

Proof of Theorem 1.5.1. We begin the proof by showing that the OK-event is likely to happen. For any index $t \in [d]$ and any solutions $x \neq y \in \mathcal{S}$ the probability that $\left|V^t x - V^t y\right| \leq \varepsilon$ is bounded by $2\phi\varepsilon$. To see this, choose one index $i \in [n]$ such that $x_i \neq y_i$ and apply the principle of deferred decisions by fixing all coefficients V_j^t for $j \neq i$. Then the value V_i^t must fall into an interval of length $2\varepsilon/|x_i - y_i| \leq 2\varepsilon$. The probability for this is bounded from above by $2\varepsilon\phi$. A union bound over all indices $t \in [d]$ and over all pairs $(x, y) \in \mathcal{S} \times \mathcal{S}$ for which $x \neq y$ yields $\mathbf{Pr}_V\left[\overline{\text{OK}(V)}\right] \leq 2(\mathcal{K}+1)^{2n} d\phi\varepsilon$.

For general densities we set $s = (2\gamma\mathcal{K})^{\gamma-d}\phi^{\gamma}$, for quasiconcave densities we set $s = 2^d(\gamma\mathcal{K})^{\gamma-d}\phi^d$, where $\gamma = d \cdot (d+1)$. With $I = ()$ we obtain

$$\begin{aligned}
\mathbf{E}_V\left[\text{PO}(V)\right] &\leq \sum_{(I_0,A)\in\mathscr{C}(I)} \sum_{B\in\mathbb{B}_\varepsilon} \mathbf{Pr}_V\left[E_{I_0,A,B}\right] + (\mathcal{K}+1)^n \cdot \mathbf{Pr}_V\left[\overline{\text{OK}(V)}\right] \\
&\leq \sum_{(I_0,A)\in\mathscr{C}(I)} \sum_{B\in\mathbb{B}_\varepsilon} s \cdot \varepsilon^d + (\mathcal{K}+1)^n \cdot 2(\mathcal{K}+1)^{2n} d\phi\varepsilon \\
&= |\mathscr{C}(I)| \cdot |\mathbb{B}_\varepsilon| \cdot s \cdot \varepsilon^d + 2(\mathcal{K}+1)^{3n} d\phi\varepsilon \\
&\leq (\mathcal{K}+1)^{(d+1)^2} n^d \cdot \left(\frac{2n\mathcal{K}}{\varepsilon}\right)^d \cdot s \cdot \varepsilon^d + 2(\mathcal{K}+1)^{3n} d\phi\varepsilon \\
&= 2^d(\mathcal{K}+1)^{(d+1)^2}\mathcal{K}^d n^{2d} \cdot s + 2(\mathcal{K}+1)^{3n} d\phi\varepsilon\,.
\end{aligned}$$

The first inequality is due to Corollary 7.5.6. The second inequality is due to Corollary 7.5.11. The third inequality stems from Lemma 7.5.7. Since this bound is true for arbitrarily small $\varepsilon > 0$, we obtain

$$\mathbf{E}_V\left[\text{PO}(V)\right] \leq 2^d(\mathcal{K}+1)^{(d+1)^2}\mathcal{K}^d n^{2d} \cdot s\,.$$

Substituting s and γ by their definitions yields

$$\begin{aligned}
\mathbf{E}_V\left[\text{PO}(V)\right] &\leq 2^d(\mathcal{K}+1)^{(d+1)^2}\mathcal{K}^d n^{2d} \cdot (2d(d+1)\mathcal{K})^{d(d+1)-d}\phi^{d(d+1)} \\
&= (\mathcal{K}+1)^{2(d+1)^2} \cdot O\big(n^{2d}\phi^{d(d+1)}\big)
\end{aligned}$$

for general densities and

$$\begin{aligned}
\mathbf{E}_V\left[\text{PO}(V)\right] &\leq 2^d(\mathcal{K}+1)^{(d+1)^2}\mathcal{K}^d n^{2d} 2^d (d(d+1)\mathcal{K})^{d(d+1)-d}\phi^d \\
&= (\mathcal{K}+1)^{2(d+1)^2} \cdot O\big(n^{2d}\phi^d\big)
\end{aligned}$$

for quasiconcave densities. □

7.6 Higher Moments

As in Section 7.5, we consider the model without zero-preserving perturbations. The basic idea behind our analysis of higher moments is the following: If the OK-event occurs, then we can count the c^{th} power of the number $\mathrm{PO}(V)$ of Pareto-optimal solutions by counting all c-tuples $(B_1, \dots, B_c)$ of ε-boxes where each ε-box B_i contains a Pareto-optimal solution x_i. We can bound this value as follows: First, call $\mathsf{Witness}(V, x_1, ())$ to obtain a vector x'_1 and consider the index tuple $I_0^{(1)}$ that contains all indices created in this call and one additional index. In the second step, call $\mathsf{Witness}(V, x_2, I_0^{(1)})$ to obtain a vector x'_2 and consider the tuple $I_0^{(2)}$ consisting of the indices of $I_0^{(1)}$, the indices created in this call, and one additional index. Now, call $\mathsf{Witness}(V, x_3, I_0^{(2)})$ and so on. For the call $\mathsf{Witness}(V, x, I)$ to be well-defined, in Section 7.5 we assumed $|I| \leq n - (d+1)$. Consequently, here we have to ensure that $|I_0^{(c-1)}| \leq n - (d+1)$, i.e., $n \geq c \cdot (d+1)$. We can assume this for fixed integers c and d because all of our results are presented in O-notation.

If $(x_1, \dots, x_c)$ is a tuple of Pareto-optimal solutions with $V^{1\dots d}x_i \in B_i$ for $i \in [c]$, then $(x'_1, \dots, x'_c) = (x_1, \dots, x_c)$ due to Lemma 7.5.1. As in the analysis of the first moment, we use the variant of the $\mathsf{Witness}$ function that uses certificates of the vectors x_ℓ instead of the vectors itself to simulate the calls. Hence we can reuse several statements of Section 7.5.

Unless stated otherwise, let V be a realization such that $\mathrm{OK}(V)$ is true and fix arbitrary solutions $x_1, \dots, x_c \in \mathcal{S}$ with $V^{1\dots d}x_i \in B_i$ for $i \in [c]$ that are Pareto-optimal with respect to V.

Definition 7.6.1. Let $I_0^{(0)} = ()$ and let $(I_0^{(\ell)}, A_0^{(\ell)})$ be the $(V, I_0^{(\ell-1)})$-certificate of x_ℓ defined in Definition 7.5.2, $\ell = 1, \dots, c$. We call the pair (I, A) for $I = I_0^{(c)}$, $A = (A^{(1)}, \dots, A^{(c)})$, and $A^{(\ell)} = A_0^{(\ell)}\big|_I$, the (*restricted*) *$V$-certificate* of $(x_1, \dots, x_c)$. We call a pair (I'_0, A') a *c-certificate*, if there is a realization V for which $\mathrm{OK}(V)$ is true and if there are Pareto-optimal solutions $x_1, \dots, x_c \in \mathcal{S}$ such that (I'_0, A') is the V-certificate of $(x_1, \dots, x_c)$. By $\mathscr{C}_c$ we denote the set of all c-certificates.

Note, that $I_0^{(0)} \subseteq \dots \subseteq I_0^{(c)}$ and $|I_0^{(\ell)}| = |I_0^{(\ell-1)}| + d + 1$ for $\ell \in [c]$. We now consider the functions $\chi_{I,A,\vec{B}}(V)$, parameterized by an arbitrary c-certificate $(I, A) \in \mathscr{C}_c$ and a vector $\vec{B} = (B_1, \dots, B_c) \in \mathbb{B}_\varepsilon^c$ of ε-boxes, which is defined as follows: $\chi_{I,A,\vec{B}}(V) = 1$ if for any $\ell \in [c]$ the call $\mathsf{Witness}(V, I, A^{(\ell)}, B_\ell, u^\star(I, A^{(\ell)}))$ returns a solutions x'_ℓ such that $B_V\big(x'_\ell - u^\star(I, A^{(\ell)})\big) = B_\ell$, and $\chi_{I,A,\vec{B}}(V) = 0$ otherwise. Recall that the vector $u^\star = u^\star(I, A^{(\ell)})$ is defined in Equation 7.1.

Corollary 7.6.2. *Assume that* $\mathrm{OK}(V)$ *is true. Then the c^{th} power of the number* $\mathrm{PO}(V)$ *of Pareto-optimal solutions is at most* $\sum\limits_{(I,A)\in\mathscr{C}_c} \sum\limits_{\vec{B}\in\mathbb{B}_\varepsilon^c} \chi_{I,A,\vec{B}}(V)$.

Proof. The c^{th} power of the number $\mathrm{PO}(V)$ of Pareto-optimal solutions equals the number of c-tuples $(x_1, \dots, x_c)$ of Pareto-optimal solutions. Let $(x_1, \dots, x_c)$ be such a c-tuple, let (I, A) be the V-certificate of $(x_1, \dots, x_c)$, and let $B_\ell = B_V\big(x_\ell - u^\star(I, A^{(\ell)})\big) \in \mathbb{B}_\varepsilon$. Due to Lemma 7.5.4, $\mathsf{Witness}(V, I, A^{(\ell)}, B_\ell, u^\star(I, A^{(\ell)}))$ returns vector x_ℓ for any $\ell \in [c]$. Hence,

$\chi_{I,A,\vec{B}}(V) = 1$ for $\vec{B} = (B_1, \ldots, B_c)$. As in the proof of Corollary 7.5.5 we have to show that this assignment $(x_1, \ldots, x_c) \mapsto (I, A, \vec{B})$ is injective.

Let $(x_1, \ldots, x_c)$ and $(y_1, \ldots, y_c)$ be distinct c-tuples of Pareto-optimal solutions, i.e., there is an index $\ell \in [c]$ such that $x_\ell \neq y_\ell$, and let (I_1, A_1) and (I_2, A_2) be their V-certificates. If $(I_1, A_1) \neq (I_2, A_2)$, then both tuples are maped to distinct triplets. Otherwise, $u^\star(I_1, A_1^{(\ell)}) = u^\star(I_2, A_2^{(\ell)})$ and, thus, $B_V\big(x_\ell - u^\star(I_1, A_1^{(\ell)})\big) \neq B_V\big(y_\ell - u^\star(I_2, A_2^{(\ell)})\big)$ since $\mathrm{OK}(V)$ holds and $x_\ell \neq y_\ell$. Consequently, also in this case $(x_1, \ldots, x_c)$ and $(y_1, \ldots, y_c)$ are mapped to distinct triplets. $\square$

Corollary 7.6.2 immediately implies a bound on the c^{th} moment of the number of Pareto-optimal solutions.

Corollary 7.6.3. *The c^{th} moment of the number of Pareto-optimal solutions is bounded by*

$$\mathbf{E}_V\left[\mathrm{PO}^c(V)\right] \leq \sum_{(I,A)\in\mathscr{C}_c} \sum_{\vec{B}\in\mathbb{B}_\varepsilon^c} \mathbf{Pr}_V\left[E_{I,A,\vec{B}}\right] + (\mathcal{K}+1)^{cn} \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}(V)}\right],$$

where $E_{I,A,\vec{B}}$ denotes the event that $\chi_{I,A,\vec{B}}(V) = 1$.

We omit the proof since it is exactly the same as the one of Corollary 7.5.6.

Lemma 7.6.4. *The size of the certificate space is bounded by $|\mathscr{C}_c| \leq (\mathcal{K}+1)^{c^2(d+1)^2} n^{cd}$.*

Proof. Let (I, A) be an arbitrary c-certificate. Each matrix $A^{(\ell)}$ is a $|I| \times (d+1)$-matrix with entries from $\{0, \ldots, \mathcal{K}\}$. Tuple I can be written as $I = (i_d^{(1)}, \ldots, i_0^{(1)}, \ldots, i_d^{(c)}, \ldots, i_0^{(c)})$, created by c successive calls of the $\mathsf{Witness}$ function, where the indices $i_0^{(\ell)}$ are chosen deterministically in Definition 7.5.2. Since $|I| = c \cdot (d+1)$ the claim follows. $\square$

Corollary 7.6.5. *For an arbitrary c-certificate (I, A) and an arbitrary vector $\vec{B} \in \mathbb{B}_\varepsilon^c$ of ε-boxes the probability of the event $E_{I,A,\vec{B}}$ is bounded by*

$$\mathbf{Pr}_V\left[E_{I,A,\vec{B}}\right] \leq (2\gamma\mathcal{K})^{\gamma-cd}\phi^\gamma\varepsilon^{cd}$$

for $\gamma = cd(d+1)$, and by

$$\mathbf{Pr}_V\left[E_{I,A,\vec{B}}\right] \leq 2^{cd}(\gamma\mathcal{K})^{\gamma-cd}\phi^{cd}\varepsilon^{cd}$$

if all densities are quasiconcave.

Proof. For indices $k \in [d]$ and $\ell \in [c]$ consider the matrices $Q_k\big(I, A^{(\ell)}, u_\ell^\star\big)$ for $u_\ell^\star = u^\star(I, A^{(\ell)})$ defined in Equation (7.2). In accordance with Lemma 7.5.8, the output of the call $\mathsf{Witness}(V, I, A^{(\ell)}, B_\ell, u_\ell^\star)$ is determined if $V_{\overline{I}}$ and the linear combinations $V_I^k \cdot q$ for any index $k \in [d]$ and any column q of the matrix $Q_k^{(\ell)} = Q_k(I, A^{(\ell)}, u_\ell^\star)$ are given. With the same argument as in the proof of Corollary 7.6.5 event $E_{I,A,\vec{B}}$ occurs if and only if $V_I \cdot \left[p^{(\ell,1)}, \dots, p^{(\ell,d)}\right]$ falls into some d-dimensional hypercube C_ℓ with side length ε depending on the linear combinations $V_I \cdot Q_k^{(\ell)}$. In this notation, $p^{(\ell,t)}$ is short for $p^{(t)}(I, A^{(\ell)}, u_\ell^\star)$.

Now, consider the matrix

$$Q'_k = \left[Q_k^{(1)}, p^{(1,k)}, \dots, Q_k^{(c)}, p^{(c,k)}\right] \in \{-\mathcal{K}, \dots, \mathcal{K}\}^{|I| \times c \cdot (d+1)} .$$

Note, that $|I| = c \cdot (d+1) = \gamma/d$. Due to Lemma 7.5.9, Q'_k is a lower block triangular matrix, due to Lemma 7.5.10 the columns of $\left[Q_k^{(\ell)}, p^{(\ell,k)}\right]$ are linearly independent. Hence, matrix Q'_k is an invertible matrix and the same holds for the block diagonal matrix

$$Q' = \begin{bmatrix} Q'_1 & \mathbb{O} & \dots & \mathbb{O} \\ \mathbb{O} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbb{O} \\ \mathbb{O} & \dots & \mathbb{O} & Q'_d \end{bmatrix} \in \{-\mathcal{K}, \dots, \mathcal{K}\}^{\gamma \times \gamma} .$$

We permute the columns of Q' to obtain a matrix Q where the last cd columns belong to the columns $p^{(1,1)}, \dots, p^{(1,d)}, \dots, p^{(c,1)}, \dots, p^{(c,d)}$. We assume the rows of Q to be labeled by $Q_{j_1,1}, \dots, Q_{j_m,1}, \dots, Q_{j_1,d}, \dots, Q_{j_m,d}$ where $I = (j_1, \dots, j_m)$ and introduce random variables $X_{j,k} = V_j^k$, $j \in I$, $k \in [d]$, indexed the same way as the rows of Q. Event $E_{I,A,\vec{B}}$ holds if and only if the cd linear combinations of the variables $X_{j,k}$ given by the last cd columns of Q fall into the cd-dimensional hypercube $C = \prod_{\ell=1}^{c} C_\ell$ with side length ε depending on the linear combinations of the variables $X_{j,k}$ given by the remaining columns of Q. The claim follows by applying Theorem 7.9.1 for the matrix Q^{T} and $k = cd$ and due to the fact that the number of columns of Q is γ. □

Proof of Theorem 1.5.3. In the proof of Theorem 1.5.1 we showed that the probability that the OK-event does not hold is bounded by $2(\mathcal{K}+1)^{2n} d\phi\varepsilon$. For $\gamma = cd(d+1)$, we set $s = (2\gamma\mathcal{K})^{\gamma - cd}\phi^\gamma$ for general densities and $s = 2^{cd}(\gamma\mathcal{K})^{\gamma - cd}\phi^{cd}$ in the case of quasiconcave density functions. Then we obtain

$$\begin{aligned} \mathbf{E}_V\left[\mathrm{PO}^c(V)\right] &\le \sum_{(I,A) \in \mathscr{C}_c} \sum_{\vec{B} \in \mathbb{B}_\varepsilon^c} \mathbf{Pr}_V\left[E_{I,A,\vec{B}}\right] + (\mathcal{K}+1)^{cn} \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}(V)}\right] \\ &\le \sum_{(I,A) \in \mathscr{C}_c} \sum_{\vec{B} \in \mathbb{B}_\varepsilon^c} s \cdot \varepsilon^{cd} + (\mathcal{K}+1)^{cn} \cdot 2(\mathcal{K}+1)^{2n} d\phi\varepsilon \\ &= |\mathscr{C}_c| \cdot |\mathbb{B}_\varepsilon^c| \cdot s \cdot \varepsilon^{cd} + 2(\mathcal{K}+1)^{(c+2)n} d\phi\varepsilon \\ &\le (\mathcal{K}+1)^{c^2(d+1)^2} n^{cd} \cdot \left(\frac{2n\mathcal{K}}{\varepsilon}\right)^{cd} \cdot s \cdot \varepsilon^{cd} + 2(\mathcal{K}+1)^{(c+2)n} d\phi\varepsilon \\ &= 2^{cd}(\mathcal{K}+1)^{c^2(d+1)^2} \mathcal{K}^{cd} n^{2cd} \cdot s + 2(\mathcal{K}+1)^{(c+2)n} d\phi\varepsilon . \end{aligned}$$

The first inequality is due to Corollary 7.6.3. The second inequality is due to Corollary 7.6.5. The third inequality stems from Lemma 7.6.4. Since this bound is true for arbitrarily small $\varepsilon > 0$, we obtain

$$\mathbf{E}_V\left[\mathrm{PO}^c(V)\right] \leq 2^{cd}(\mathcal{K}+1)^{c^2(d+1)^2}\mathcal{K}^{cd}n^{2cd} \cdot s\,.$$

Substituting s and γ by their definitions yields

$$\begin{aligned}\mathbf{E}_V\left[\mathrm{PO}^c(V)\right] &\leq 2^{cd}(\mathcal{K}+1)^{c^2(d+1)^2}\mathcal{K}^{cd}n^{2cd} \cdot (2cd(d+1)\mathcal{K})^{cd(d+1)-cd}\phi^{cd(d+1)} \\ &= (\mathcal{K}+1)^{(c+1)^2(d+1)^2} \cdot O\big((n^{2d}\phi^{d(d+1)})^c\big)\end{aligned}$$

for general densities and

$$\begin{aligned}\mathbf{E}_V\left[\mathrm{PO}^c(V)\right] &\leq 2^{cd}(\mathcal{K}+1)^{c^2(d+1)^2}\mathcal{K}^{cd}n^{2cd} \cdot 2^{cd}(cd(d+1)\mathcal{K})^{cd(d+1)-cd}\phi^{cd} \\ &= (\mathcal{K}+1)^{(c+1)^2(d+1)^2} \cdot O\big((n^{2d}\phi^{d})^c\big)\end{aligned}$$

for quasiconcave densities. □

The proof of Theorem 1.5.3 yields that

$$\mathbf{E}_V\left[\mathrm{PO}^c(V)\right] \leq s_c := 2^{c(d+1)^2}(cd(d+1))^{cd^2}(\mathcal{K}+1)^{(c+1)^2(d+1)^2}n^{2cd}\phi^{c\beta}$$

for $\beta = d(d+1)$ in general and $\beta = d$ for quasiconcave densities. With the following Corollary we bound the probability that $\mathrm{PO}(V)$ exceeds a certain multiple of s_1. We obtain a significantly better concentration bound than the one we would obtain by applying *Markov's inequality* for the first moment.

Corollary 7.6.6. *The probability that the number of Pareto-optimal solutions is at least $\lambda \cdot s_1$ for some $\lambda \geq 1$ is bounded by*

$$\mathbf{Pr}_V\left[\mathrm{PO}(V) \geq \lambda \cdot s_1\right] \leq \left(\frac{1}{\lambda}\right)^{\frac{1}{2}\cdot\left\lfloor\frac{\log_{\mathcal{K}+1}\lambda}{4(d+1)^2}\right\rfloor}.$$

Proof. Let $c^\star$ be the real for which $(\mathcal{K}+1)^{2c^\star(d+1)^2} = \lambda^{1/2}$, i.e., $c^\star = (\log_{\mathcal{K}+1}\lambda)/(4(d+1)^2)$. Observing that $c \leq 2^c \leq (\mathcal{K}+1)^c$ for any $c \in \mathbb{R}$ and setting $c = \lfloor c^\star \rfloor$ yields

$$\begin{aligned}\mathbf{Pr}_V\left[\mathrm{PO}(V) \geq \lambda \cdot s_1\right] &= \mathbf{Pr}_V\left[\mathrm{PO}^c(V) \geq \lambda^c \cdot s_1^c\right] \\ &= \mathbf{Pr}_V\left[\mathrm{PO}^c(V) \geq \frac{\lambda^c \cdot s_1^c}{\mathbf{E}_V\left[\mathrm{PO}^c(V)\right]} \cdot \mathbf{E}_V\left[\mathrm{PO}^c(V)\right]\right] \\ &\leq \frac{\mathbf{E}_V\left[\mathrm{PO}^c(V)\right]}{\lambda^c \cdot s_1^c} \leq \frac{s_c}{\lambda^c \cdot s_1^c} \\ &= \frac{2^{c(d+1)^2}(cd(d+1))^{cd^2}(\mathcal{K}+1)^{(c+1)^2(d+1)^2}n^{2cd}\phi^{c\beta}}{\lambda^c \cdot 2^{c(d+1)^2}(d(d+1))^{cd^2}(\mathcal{K}+1)^{4c(d+1)^2}n^{2cd}\phi^{c\beta}}\end{aligned}$$

$$
\begin{aligned}
&= \frac{c^{cd^2}(\mathcal{K}+1)^{(c-1)^2(d+1)^2}}{\lambda^c} \le \left(\frac{c^{(d+1)^2}(\mathcal{K}+1)^{c(d+1)^2}}{\lambda}\right)^c \\
&\le \left(\frac{(\mathcal{K}+1)^{c(d+1)^2}(\mathcal{K}+1)^{c(d+1)^2}}{\lambda}\right)^c \le \left(\frac{(\mathcal{K}+1)^{2c^\star(d+1)^2}}{\lambda}\right)^c \\
&= \left(\frac{1}{\lambda}\right)^{c/2} = \left(\frac{1}{\lambda}\right)^{\frac{1}{2}\cdot\left\lfloor\frac{\log_{\mathcal{K}+1}\lambda}{4(d+1)^2}\right\rfloor} .
\end{aligned}
$$

The first inequality is Markov's inequality (see Theorem 3.1). The second inequality only holds if $c \geq 1$. However, for $c = 0$ the inequality $\mathbf{Pr}_V\left[\mathrm{PO}(V) \geq \lambda \cdot s_1\right] \leq \lambda^{-c/2}$ is trivially true. □

7.7 Zero-Preserving Perturbations

In this section we consider the model with zero-preserving perturbations. To prove Theorem 1.5.4 we will first show that we can concentrate on a special class of instances.

Lemma 7.7.1. *Without loss of generality in each objective function except for the adversarial one there are more than $(d+1)^3$ perturbed coefficients, i.e., coefficients that are not deterministically set to zero.*

Proof. For an index $k \in [d]$ let P_k be the tuple of indices i for which V_i^k is a perturbed coefficient, i.e., a coefficient which is not set to zero deterministically. Let K be the tuple of indices k for which $|P_k| \leq (d+1)^3$, let $P = \bigcup_{k \in K} P_k$, and consider the decomposition of $\mathcal{S}$ into subsets of solutions $\mathcal{S}_v = \{x \in \mathcal{S} : \, x|_P = v\}$, $v \in \{0, \ldots, \mathcal{K}\}^{|P|}$. Let $x \in \mathcal{S}_v$ be an arbitrary solution. If x is Pareto-optimal with respect to $\mathcal{S}$ and $\{V^1, \ldots, V^{d+1}\}$, then x is also Pareto-optimal with respect to $\mathcal{S}_v$ and $\{V^k : \, k \in [d+1] \setminus K\}$ due to Lemma 7.4.4. As all remaining objective functions V^k, $k \in [d+1] \setminus K$, have more than $(d+1)^3$ perturbed coefficients, the instance with these objective functions and $\mathcal{S}_v$ as set of feasible solutions has the desired form and we can apply Theorem 1.5.4 for each of these instances. Since we now have $(\mathcal{K}+1)^{|P|} \leq (\mathcal{K}+1)^{|K| \cdot (d+1)^3}$ instances, each having $d - |K|$ linear and one adversarial objective, we can bound the number of Pareto-optimal solutions by

$$(\mathcal{K}+1)^{|K|\cdot(d+1)^3} \cdot c(d-|K|) \cdot (\mathcal{K}+1)^{(d-|K|+1)^5} \cdot n^{\alpha(d-|K|)} \cdot \phi^{\beta(d-|K|)}$$

from above, where c is the factor and α and β are the exponents of n and ϕ in the bound stated in Theorem 1.5.4. These functions depend on the number of linear objectives and whether the densities are quasiconcave or not. Since they are monotonically increasing, we can bound the number of Pareto-optima simply by

$$(\mathcal{K}+1)^{|K|\cdot(d+1)^3} \cdot c(d) \cdot (\mathcal{K}+1)^{(d-|K|+1)^5} \cdot n^{\alpha(d)} \cdot \phi^{\beta(d)} .$$

Hence, it suffices to show that $(\mathcal{K}+1)^{|K|\cdot(d+1)^3+(d-|K|+1)^5} \le (\mathcal{K}+1)^{(d+1)^5}$. This is equivalent to showing that $b \cdot a^3 + (a-b)^5 \le a^5$ for $b = |K|$ and $a = d+1$. Note, that $0 \le b = |K| \le d = a-1$. By a chain of equivalences we obtain

$$\begin{aligned} b \cdot a^3 + (a-b)^5 \le a^5 \iff a^3 &\le \frac{1}{b} \cdot (a^5 - (a^5 - 5a^4b + 10a^3b^2 - 10a^2b^3 + 5ab^4 - b^5) \\ \iff a^3 &\le 5a^4 - 10a^3b + 10a^2b^2 - 5ab^3 + b^4 \\ &= 5a \cdot (a^3 - 2a^2b + 2ab^2 - b^3) + b^4 \\ &= 5a(a-b) \cdot (a^2 - ab + b^2) + b^4 =: f(a,b)\,. \end{aligned}$$

Applying the inequality $ab \le (a+b)^2/4$ yields

$$f(a,b) \ge 5a(a-b) \cdot \left(a^2 - \frac{(a+b)^2}{4} + b^2\right) \ge 5a \cdot \left(\frac{a^2}{2} + \frac{(a-b)^2}{4} + \frac{b^2}{2}\right) \ge \frac{5}{2}a^3 \ge a^3\,.$$

This concludes the proof. □

Lemma 7.7.2. *Without loss of generality for every $i \in [n]$ exactly one of the coefficients $V_i^1, \dots, V_i^d$ is perturbed, whereas the others are deterministically set to zero.*

Proof. We first show how to decrease the number of indices i for which $V_i^1, \dots, V_i^d$ is perturbed to at most one. For this, let $\mathcal{S}' = \{(x, x, \dots, x) : x \in \mathcal{S}\} \subseteq \{0, \dots, \mathcal{K}\}^{dn}$ be the set of feasible solutions that contains for every $x \in \mathcal{S}$ the solution $x^d \in \{0, \dots, \mathcal{K}\}^{dn}$ that consists of d copies of x. For $k \in [d]$ we define a linear objective function $W^k \colon \mathcal{S}' \to \mathbb{R}$ in which all coefficients W_i^k with $i \notin I_k := \{(k-1)n+1, \dots, kn\}$ are deterministically set to zero. The coefficients $W_{kn-n+1}^k, \dots, W_{kn}^k$ are chosen as the coefficients $V_1^k, \dots, V_n^k$, i.e., either randomly according to a density f_i^k or zero deterministically. The objective function W^{d+1} maps every solution $x^d \in \mathcal{S}'$ to $V^{d+1}x$. The instance consisting of $\mathcal{S}'$ and the objective functions $W^1, \dots, W^{d+1}$ has the desired property that every variable appears in at most one of the objective functions $W^1, \dots, W^d$ and it has the same smoothed number of Pareto-optimal solutions as the instance consisting of $\mathcal{S}$ and the objective functions $V^1, \dots, V^{d+1}$. For every $i \in [dn]$ for which none of the coefficients $W_i^1, \dots, W_i^d$ is perturbed we can eliminate the corresponding variable from $\mathcal{S}'$.

This shows that any ϕ-smooth instance with $\mathcal{S} \subseteq \{0, \dots, \mathcal{K}\}^n$ can be transformed into another ϕ-smooth instance with $\mathcal{S} \subseteq \{0, \dots, \mathcal{K}\}^\ell$ with $\ell \le dn$ in which every variable appears in exactly one objective function and that has the same smoothed number of Pareto-optimal solutions. As the bound proven in Theorem 1.5.4 depends polynomially on the number of variables, we lose only a constant factor (with respect to n, ϕ, and $\mathcal{K}$) by going from $\mathcal{S} \subseteq \{0, \dots, \mathcal{K}\}^n$ to $\mathcal{S}' \subseteq \{0, \dots, \mathcal{K}\}^{dn}$. This constant is hidden in the O-notation. □

In the remainder of this chapter we focus on instances having the structure described in Lemma 7.7.1 and Lemma 7.7.2. Then $(P_1, \dots, P_d)$ is a partition of $[n]$, where P_t denotes the tuple of indices i for which V_i^t is perturbed.

We consider the variant of the Witness function given as Algorithm 7, referred to as the $\mathsf{Witness}_0$ function. The $\mathsf{Witness}_0$ function gets as parameters besides the usual V, x, and I, a set $K \subseteq [d]$ of indices of objective functions and a call number $r \in \mathbb{N}$. In a call of the $\mathsf{Witness}_0$ function only the adversarial objective function V^{d+1} and the objective functions V^t with $t \in K$ are considered. The set of solutions is restricted to solutions that agree with x in all positions P_k with $k \notin K$. Additionally, as in the Witness function, only solutions are considered that agree with x in all positions $i \in I$. By the right choice of I, we can avoid choosing an index multiple times in different calls of the $\mathsf{Witness}_0$ function. The parameter r simply corresponds to the number of the current call of the $\mathsf{Witness}_0$ function. The $\mathsf{Witness}_0$ function always returns some subset of $\mathcal{S}$.

In the following we use the term *round* to denote an iteration of the for-loop starting in Line 5. Now let us give some remarks about the $\mathsf{Witness}_0$ function. As a convention, we set $\bigcap_{k \in ()} \mathcal{S}_{P_k}(x) = \mathcal{S}$ (cf. Line 3). This is only important in the case where $K = [d]$, i.e., in the first call.

As in the Witness function, if round $t = 0$ is reached in a certain call r (this does not have to be the case), then we obtain $\mathcal{C}_0^{(r)} = \mathcal{R}_1^{(r)}$ since $V^{k_1 \dots k_t} z < V^{k_1 \dots k_t} x$ (see Line 6) is no restriction for $t = 0$. For the definition of $x^{(r,t)}$ in Line 8, ties are broken by taking the lexicographically first solution. Though we did the same in the Witness function, it is much more important here. In the model without zero-preserving perturbations the functions $V^1, \dots, V^d$ are injective with probability 1. If this is the case, then no ties have to be broken. In the model with zero-preserving perturbations, the functions $V^1, \dots, V^d$ can be non-injective with probability 1: If there are two distinct solutions $x, y \in \mathcal{S}$ for which $x|_{P_k} = y|_{P_k}$, then $V^k x = V^k y$.

The index r_k defined in Line 13 is the number of the last call in which the objective function V^k has been considered. The index t_r defined in Line 22 is the number of the round in call number r of $\mathsf{Witness}_0$ in which the next recursive call of $\mathsf{Witness}_0$ was made. We will see that, if the last call of the $\mathsf{Witness}_0$ function is the call with number $r^\star + 1$, then $r_k \in [r^\star]$ for each $k \in [d]$ and there is at least one index $k \in [d]$ for which $r_k = r^\star$, i.e., the objective function V^k has been considered until the end. Furthermore, the indices t_r are defined for $r = 1, \dots, r^\star$. For the simulation of the Witness function information about the solutions $x^{(t)}$ and the indices i_t are required. For the simulation of the $\mathsf{Witness}_0$ function we additionally need the values r_k and t_r to know when to make a new recursive call (in round $t = t_r$ in the call with number r) and which objectives to consider (objective V^k will be considered in the call with number r if and only if $r \leq r_k$).

In Line 15 it is always possible to find an index $i \in P_k$ on which the current vector $x^{(r,t)}$ and x disagree because this line is only reached if $k \in K_{\mathrm{neq}}$, i.e., if $x^{(r,t)}|_{P_k} \neq x|_{P_k}$. In order for Line 27 to be feasible, we have to guarantee that $P_k \setminus I \neq ()$. This follows since we assumed $|P_k| > (d+1)^3 > d(d+1)$ in accordance with Lemma 7.7.1 and because there are at most d calls of $\mathsf{Witness}_0$ with non-empty K with at most $d+1$ rounds each, and in each round at most one index from P_k is added to I. Note, that it would be more precise

Algorithm 7 $\mathsf{Witness}_0(V, x, K, r, I)$

```
Let $K$ be the form $K = (k_1, \dots, k_{d_r})$.
Set $k_{d_r+1} = d + 1$.
Set $\mathcal{R}^{(r)}_{d_r+1} = \mathcal{S}_I(x) \cap \bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x)$.
**if** $d_r = 0$ **then return** $\mathcal{R}^{(r)}_{d_r+1}$
**for** $t = d_r, d_r - 1, \dots, 0$ **do**
    Set $\mathcal{C}^{(r)}_t = \{z \in \mathcal{R}^{(r)}_{t+1} :\ V^{k_1 \dots k_t} z < V^{k_1 \dots k_t} x\}$.
    **if** $\mathcal{C}^{(r)}_t \neq \emptyset$ **then**
        Set $x^{(r,t)} = \arg\min\{V^{k_{t+1}} z :\ z \in \mathcal{C}^{(r)}_t\}$.
        Let $K_{\text{eq}} \subseteq K$ be the tuple of indices $k$ for which $x^{(r,t)}|_{P_k} = x|_{P_k}$.
        Set $K_{\text{neq}} = K \setminus K_{\text{eq}}$.
        **for** $k \in K$ **do**
            **if** $k \in K_{\text{eq}}$ **then**
                Set $r_k = r$.
            **else**
                Set $i_k = \min\{i \in P_k :\ x^{(r,t)}_i \neq x_i\}$.
                $I \hookleftarrow I \cup (i_k)$
            **end if**
        **end for**
        **if** $K_{\text{eq}} = ()$ **then**
            Set $\mathcal{R}^{(r)}_t = \{z \in \mathcal{R}^{(r)}_{t+1} :\ V^{k_{t+1}} z < V^{k_{t+1}} x^{(r,t)}\} \cap \mathcal{S}_I(x)$.
        **else**
            Set $t_r = t$.
            **return** $\mathsf{Witness}_0(V, x, K_{\text{neq}}, r + 1, I)$
        **end if**
    **else**
        **for** $k \in K$ **do**
            Set $i_k = \min(P_k \setminus I)$.
            $I \hookleftarrow I \cup (i_k)$.
        **end for**
        Set $x^{(r,t)}_i = \begin{cases} \min(\{0, \dots, \mathcal{K}\} \setminus \{x_i\}) & \text{if } i \in \{i_{k_1}, \dots, i_{k_{d_r}}\}, \\ x_i & \text{otherwise}. \end{cases}$
        Set $\mathcal{R}^{(r)}_t = \mathcal{R}^{(r)}_{t+1} \cap \mathcal{S}_I(x)$.
    **end if**
**end for**
**return** $\emptyset$
```

to introduce the notation $i_k^{(r,t)}$ rather than i_k (cf. Line 15 and Line 27). Furthermore, we could also write $I_k^{(r,t)}$ instead of I. For the sake of readability we decided to drop these additional indices and refer to index i_k and tuple I of round t of call r in our proofs.

Before we analyze the $\mathsf{Witness}_0$ function, let us discuss similarities and differences to the $\mathsf{Witness}$ function. The initial call $\mathsf{Witness}_0(V, x, [d], 1, I)$ is very similar to the initial call $\mathsf{Witness}(V, x, I)$. All objectives $V^1, \ldots, V^{d+1}$ are considered. Furthermore, $d_1 = d$ and $\mathcal{R}_{d+1}^{(1)} = \mathcal{S}_I(x)$. Line 4 can be ignored in this call since $d_1 = d \geq 1$. Also the loop of the $\mathsf{Witness}_0$ function is very similar to the loop of the $\mathsf{Witness}$ function. The sets $\mathcal{C}_t^{(r)}$ and $\mathcal{R}_t^{(r)}$ and the solution $x^{(r,t)}$ are defined the same way as the sets $\mathcal{C}_t$ and $\mathcal{R}_t$ and the solution $x^{(t)}$ in the $\mathsf{Witness}$ function.

However, there are two main differences to the $\mathsf{Witness}$ function in the body of the loop. In the model with zero-preserving perturbations it is possible that $V^k x^{(r,t)} = V^k x$ for some of the indices k. This happens, if $x^{(r,t)}|_{P_k} = x|_{P_k}$ (otherwise, it happens with probability 0) and is a fundamental issue. If we would proceed running the loop as we do it in the $\mathsf{Witness}$ function, then we might lose the crucial property that the function returns $\{x\}$ if x is Pareto-optimal. The tuple K_{eq} contains the problematic indices k for which $x^{(r,t)}|_{P_k} = x|_{P_k}$. If $K_{\text{eq}} = ()$, then we proceed more or less as we did in the $\mathsf{Witness}$ function (see Line 15 and Line 20). As discussed above, the case $K_{\text{eq}} \neq ()$ has to be treated differently. In this case we make use of Lemma 7.4.4 which implies that, if x is Pareto-optimal, then it is also Pareto-optimal with respect to $\bigcap_{k \in K_{\text{eq}}} \mathcal{S}_{P_k}(x)$ and $\{V^k : k \in K_{\text{neq}} \cup (d+1)\}$ (cf. Line 23 of the current call and Line 3 of the next call).

The second difference due to another issue with zero-preserving perturbations can be sketched as follows. In each round t of the $\mathsf{Witness}$ function one index i_t is chosen. Since in the model without zero-preserving perturbations all coefficients are perturbed, we can then exploit the randomness in the coefficients $V_{i_t}^1, \ldots, V_{i_t}^d$. In the model with zero-preserving perturbations under the assumption given by Lemma 7.7.2, for each index $i \in [n]$ exactly one of the coefficients $V_i^1, \ldots, V_i^d$ is perturbed while the others are zero. Hence, we chose one index $i_k \in P_k$ for each objective V^k to ensure that we have one perturbed coefficient $V_{i_k}^k$ per objective. These indices i_k are chosen only for $k \in K_{\text{neq}}$ (see Line 15). This is due to the fact that for our analysis to work we need the additional property that $x_{i_k}^{(r,t)} \neq x_{i_k}$ which is impossible for $k \in K_{\text{eq}}$ by the definition of K_{eq} and the requirement $i_k \in P_k$. However, as from now on we do not consider the objectives V^k for $k \in K_{\text{eq}}$ anymore, we do not have to choose indices i_k for $k \in K_{\text{eq}}$.

In the remainder of this section we only consider the case that x is Pareto-optimal. Unless stated otherwise, we consider the case that the OK_0-*event* occurs. This means that $|V^k \cdot (y - z)| \geq \varepsilon$ for every $k \in [d]$ and for any two solutions $y, z \in \mathcal{S}$ for which $y|_{P_k} \neq z|_{P_k}$. We will later see that the OK_0-event occurs with sufficiently high probability.

Lemma 7.7.3. *The call* $\mathsf{Witness}_0(V, x, [d], 1, ())$ *returns the set* $\{x^{(r^\star, t_{r^\star})}\} = \{x\}$*, where* $r^\star = \max\{r_1, \ldots, r_d\}$*.*

Lemma 7.7.3 was also stated in [BR12] (Lemma 25) but Claim 1 of the proof was not correct. Here, we rely on the concept of weak Pareto-optimality (see Definition 7.2.1) and its properties (Lemma 7.4.2) since we cannot guarantee x to be Pareto-optimal in any round. However, the Pareto-optimality will be given at the beginning of any call to the $\mathsf{Witness_0}$ function.

Proof. It suffices to show the following claim.

Claim 7. *Consider an arbitrary call* $\mathsf{Witness_0}(V, x, K, r, I)$*. If* $K \neq ()$*, then this call results in another call to the* $\mathsf{Witness_0}$ *function (and does not terminate in Line 34).*

If Claim 7 is true, then there will be recursive calls until the call $\mathsf{Witness_0}(V, x, (), r, I)$. This call immediately returns the set $\mathcal{R}^{(r)}_{d_r+1}$ in Line 4. Since $[d] \setminus () = [d]$, we obtain $\mathcal{R}^{(r)}_{d_r+1} = \mathcal{S}_I(x) \cap \bigcap_{k \in [d]} \mathcal{S}_{P_k}(x) = \mathcal{S}_{[n]}(x) = \{x\}$. Now consider the call with number $r-1$ and the round t_{r-1} in this call in which $\mathsf{Witness_0}(V, x, (), r, I)$ has been called. In this round, $K_{\text{eq}} \neq ()$ since Line 23 is reached. Hence, there is at least one index $k \in K_{\text{eq}}$, and for these indices, r_k is set to r in Line 13. Now, as the next call is of the form $\mathsf{Witness_0}(V, x, (), r, I)$, this implies $K_{\text{neq}} = ()$, i.e., all values r_k for $k \in [d]$ have been set by now and, thus, $r^\star = \max\{r_1, \dots, r_d\} = r - 1$, i.e., the number of the call we currently consider.

Consider the solution $x^{(r^\star, t_{r^\star})}$ defined in Line 8 and let K be the tuple of call $r^\star$. Since $K_{\text{neq}} = ()$ this implies $K_{\text{eq}} = K$. Hence, $x^{(r^\star, t_{r^\star})}|_{P_k} = x|_{P_k}$ for any $k \in K$ by definition of K_{eq} in Line 9. On the other hand, $x^{(r^\star, t_{r^\star})} \in \mathcal{C}^{(r^\star)}_{t_{r^\star}} \subseteq \mathcal{R}^{(r^\star)}_{t_{r^\star}+1} \subseteq \mathcal{R}^{(r^\star)}_{d_{r^\star}+1} \subseteq \bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x)$. The first inclusion is due to the definition of $\mathcal{C}^{(r^\star)}_{t_{r^\star}}$ in Line 6. The second inclusion is due to the observation that always $\mathcal{R}^{(r)}_{t+1} \subseteq \mathcal{R}^{(r)}_{t+2} \subseteq \dots \subseteq \mathcal{R}^{(r)}_{d_r+1}$ due to the construction of the sets $\mathcal{R}^{(r)}_t$ in Line 20 and Line 31. The construction of $\mathcal{R}^{(r^\star)}_{d_{r^\star}+1}$ in Line 3 yields the third inclusion. Hence, $x^{(r^\star, t_{r^\star})}|_{P_k} = x|_{P_k}$ for any $k \in [d] \setminus K$, and according to the previous observations, even for all $k \in K$. Consequently, $x^{(r^\star, t_{r^\star})} = x$. Summarizing all these results, we obtain that the call $\mathsf{Witness_0}(V, x, [d], 1, ())$ ends up in the call $\mathsf{Witness_0}(V, x, (), r, I)$ for some index tuple I which immediately returns the set $\mathcal{R}^{(r)}_{d_r+1} = \{x\} = \{x^{(r^\star, t_{r^\star})}\}$.

It remains to prove Claim 7. For this, consider an arbitrary call $\mathsf{Witness_0}(V, x, K, r, I)$ where $K \neq ()$. First we show the following claim by induction on t.

Claim 8. *In each round t that is reached, x is weakly Pareto-optimal with respect to* $\mathcal{R}^{(r)}_{t+1}$ *and* $\{V^{k_1}, \dots, V^{k_{t+1}}\}$*.*

To begin with, consider $t = d_r$. As x is Pareto-optimal with respect to $\mathcal{S}$ and $\{V^1, \dots, V^{d+1}\}$, x is also Pareto-optimal with respect to $\bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x)$ and $\{V^k : k \in K \cup (d+1)\} = \{V^{k_1}, \dots, V^{k_{d_r+1}}\}$ due to Lemma 7.4.4. Consequently, x is also Pareto-optimal with respect to $\mathcal{R}^{(r)}_{d_r+1} = \mathcal{S}_I(x) \cap \bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x)$ and $\{V^{k_1}, \dots, V^{k_{d_r+1}}\}$ due to

Proposition 7.4.1. Note, that this property is even stronger than weak Pareto-optimality. We will need this strong version in the induction step when $t = d_r - 1$.

Now consider a round $t \leq d_r - 1$ that is reached and assume that the induction hypothesis is true for $t+1$. We consider round $t+1$ where $\mathcal{R}^{(r)}_{t+1}$ is defined, and distinguish between two cases. If $\mathcal{C}^{(r)}_{t+1} = \emptyset$, then x is weakly Pareto-optimal with respect to $\mathcal{R}^{(r)}_{t+1} = \mathcal{R}^{(r)}_{t+2} \cap \mathcal{S}_I(x)$ and $\{V^{k_1}, \dots, V^{k_{t+1}}\}$ due to the induction hypothesis, Lemma 7.4.2, Claim (I), and Proposition 7.4.1. Let us consider the more interesting case $\mathcal{C}^{(r)}_{t+1} \neq \emptyset$. Since round t is reached, there is no call of the $\mathsf{Witness_0}$ function in round $t+1$, i.e., $K_{\text{eq}} = ()$ in round $t+1$. The induction hypothesis and Lemma 7.4.2, Claim (II) yield $V^{k_{t+2}}x \leq V^{k_{t+2}}x^{(r,t+1)}$.

We will show that even $V^{k_{t+2}}x < V^{k_{t+2}}x^{(r,t+1)}$. For this, we assume for the contrary that $V^{k_{t+2}}x = V^{k_{t+2}}x^{(r,t+1)}$ and distinguish between the cases $t = d_r - 1$ and $t < d_r - 1$. If $t = d_r - 1$, then we obtain $V^{k_{d_r+1}}x = V^{k_{d_r+1}}x^{(r,d_r)}$ and $V^{k_1 \dots k_{d_r}}x^{(r,d_r)} < V^{k_1 \dots k_{d_r}}x$ since $x^{(r,d_r)} \in \mathcal{C}^{(r)}_{d_r}$. Hence, $x^{(r,d_r)} \in \mathcal{R}^{(r)}_{d_r+1}$ dominates x with respect to $\{V^{k_1}, \dots, V^{k_{d_r+1}}\}$ which contradicts the fact that x is Pareto-optimal with respect to $\mathcal{R}^{(r)}_{d_r+1}$ and $\{V^{k_1}, \dots, V^{k_{d_r+1}}\}$. If $t < d_r - 1$, then $V^{k_{t+2}}x = V^{k_{t+2}}x^{(r,t+1)}$ implies $x|_{P_{k_{t+2}}} = x^{(r,t)}|_{P_{k_{t+2}}}$ as we assume that the OK_0-event occurs. Consequently, $k_{t+2} \in K_{\text{eq}}$ in round $t+1$, which contradicts the previous observation that $K_{\text{eq}} = ()$ in that round. This concludes the proof of Claim 8.

To finish the proof of Claim 7, let us assume that there is no further call to the $\mathsf{Witness_0}$ function until round 0, i.e., we reach round 0. In accordance with Claim 8, x is weakly Pareto-optimal with respect to $\mathcal{R}^{(r)}_1$ and $\{V^{k_1}\}$. Now let us consider round $t = 0$. We obtain $\mathcal{C}^{(r)}_0 = \mathcal{R}^{(r)}_1$ since there are no restrictions in this round. Consequently, $\mathcal{C}^{(r)}_0 \neq \emptyset$ because $x \in \mathcal{R}^{(r)}_1$. The solution $x^{(r,0)}$ minimizes V^{k_1} among all solutions from $\mathcal{C}^{(r)}_0$. On the other hand, $x^{(r,0)}$ cannot dominate x strongly. Hence, $V^{k_1}x^{(r,0)} = V^{k_1}x$, i.e., $x^{(r,0)}|_{P_{k_1}} = x|_{P_{k_1}}$ as we assumed that the OK_0-event occurs. Therefore, $k_1 \in K_{\text{eq}}$, i.e., $K_{\text{eq}} \neq ()$, and thus, the $\mathsf{Witness_0}$ function is called again in Line 23. □

Like for the simple $\mathsf{Witness}$ function, we show that it is enough to know some information about the run of the $\mathsf{Witness_0}$ function to reconstruct the solution x. As before, we call this data the *certificate* of x.

Definition 7.7.4. Let $r_1, \dots, r_d$ and $t_1, \dots, t_{r^\star}$ for $r^\star = \max\{r_1, \dots, r_d\}$ be the indices and $x^{(r,t)}$ be the vectors constructed during the execution of the call $\mathsf{Witness_0}(V, x, [d], 1, ())$. Furthermore, consider the tuple I at the moment when the last call to the $\mathsf{Witness_0}$ function terminates. The pair $(I^\star, A)$ for $I^\star = I \cup \left(i^\star_1, \dots, i^\star_d\right)$, $i^\star_k = \min(P_k \setminus I)$, and $A = \left[x^{(1,d_1)}, \dots, x^{(1,t_1)}, \dots, x^{(r^\star,d_{r^\star})}, \dots, x^{(r^\star,t_{r^\star})}\right]\big|_{I^\star}$, is called the *$V$-certificate* of x. We label the columns of A by $a^{(r,t)}$. Moreover, we call a pair (I', A') a *certificate* if there is some realization V such that $\text{OK}_0(V)$ is true and if there exists a Pareto-optimal solution $x \in \mathcal{S}$ such that (I', A') is the V-certificate of x. By $\mathscr{C}$ we denote the set of all certificates.

We assume that the indices r_k and t_r (and hence also the indices d_r) are implicitly encoded in a given certificate. Later we will take these indices into consideration again when we count the number of possible certificates.

Lemma 7.7.5. *Let V be a realization for which $\mathrm{OK}_0(V)$ is true and let $(I^\star, A)$ be a V-certificate of some Pareto-optimal solution x. Moreover, let the matrix A be of the form $A = \left[a^{(1,d_1)}, \dots, a^{(1,t_1)}, \dots, a^{(r^\star,d_{r^\star})}, \dots, a^{(r^\star,t_{r^\star})}\right]$. For a fixed index $k \in [d]$ consider the matrix $M = \left[a^{(1,d_1)}, \dots, a^{(1,t_1)}, \dots, a^{(r_k,d_{r_k})}, \dots, a^{(r_k,t_{r_k})}\right]\big|_J$ for $J = I^\star \cap P_k =: (j_1, \dots, j_m)$. Then M is of the form*

$$M = \begin{bmatrix} \overline{x_{j_1}} & x_{j_1} & \dots & x_{j_1} \\ * & \ddots & \ddots & \vdots \\ \vdots & \ddots & \overline{x_{j_{m-1}}} & x_{j_{m-1}} \\ * & \dots & * & x_{j_m} \end{bmatrix} \in \{0, \dots, \mathcal{K}\}^{|J| \times |J|} ,$$

where each '$*$'-entry can be an arbitrary value from $\{0, \dots, \mathcal{K}\}$ independently of the other '$*$'-entries and where $\overline{z}$ for $z \in \{0, \dots, \mathcal{K}\}$ can be an arbitrary value from $\{0, \dots, \mathcal{K}\} \setminus \{z\}$.

Proof. Consider the call $\mathsf{Witness}_0(V, x, [d], 1, ())$ and the resulting subsequent recursive calls $\mathsf{Witness}_0(V, x, K, r, I)$. By definition of r_k we have $r \le r_k \iff k \in K$ (see Line 13, Line 10, and Line 23). In each call where $r \le r_k$ one vector $x^{(r,t)}$ is constructed in each round t. Also, in each round except for the last round t_{r_k} of the r_k^{th} call, when $k \in K_{\text{eq}}$ for the first and the last time, one index $i \in P_k$ is chosen and added to I. Since J consists of the chosen indices $i \in P_k$ and the additional index $i_k^\star$, matrix M is a square matrix.

We first consider the last column of M. As $x^{(r_k,t_{r_k})}$ is the last vector constructed before k is removed from K, index k must be an element of K_{eq} in round t_{r_k} of call r_k, i.e., $x^{(r_k,t_{r_k})}\big|_{P_k} = x|_{P_k}$. Hence, the last column of M has the claimed form because $J \subseteq P_k$.

Now consider the remaining columns of M. Due to the construction of the set $\mathcal{R}_t^{(r)}$ in Line 3, Line 20, and Line 31, all vectors $z \in \mathcal{R}_t^{(r)}$ agree with x in the previously chosen indices i. As in the case $\mathcal{C}_t^{(r)} \neq \emptyset$ vector $x^{(r,t)}$ is an element of $\mathcal{C}_t^{(r)} \subseteq \mathcal{R}_{t+1}^{(r)}$ and in the case $\mathcal{C}_t^{(r)} = \emptyset$ vector $x^{(r,t)}$ is constructed appropriately, the upper triangle of M, excluding the principal diagonal, has the claimed form. The form of the principal diagonal follows from the choice of index $i \in P_k$: In Line 15 we chose i such that $x_i^{(r,t)} \neq x_i$, in Line 30 we construct $x^{(r,t)}$ explicitely such that $x_i^{(r,t)} \neq x_i$. □

Like in the model without zero-preserving perturbations we now consider a variant of the $\mathsf{Witness}_0$ function given as Algorithm 8 which gets as additional parameters the V-certificate of x, a shift vector $u \in \{0, \dots, \mathcal{K}\}^n$, the ε-box $B = B_V(x - u)$, and a set $\mathcal{S}'$ of solutions that are still under consideration. Recall, that at the beginning of any call to the original $\mathsf{Witness}_0$ function the set of solutions that have still to be considered is restricted to a subset of $\bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x)$ (see Line 3). The huge amount of information that is necessary to restrict the current set of solutions that extemely is not given by the V-certificate of x. Thus, we keep track of this set of remaining solutions by passing it as a parameter. We will see how to update this set without too much knowledge about x (cf. Line 13 of Algorithm 8).

Algorithm 8 $\mathsf{Witness}_0(V, K, r, I^\star, A, \mathcal{S}', B, u)$

```
1:  Let $K$ be of the form $K = (k_1, \dots, k_{d_r})$.
2:  Set $k_{d_r+1} = d + 1$.
3:  Let $b$ be the corner of $B$.
4:  if $d_r = 0$ then return $\mathcal{S}'$
5:  Set $\mathcal{R}^{(r)}_{d_r+1} = \mathcal{S}' \cap \bigcup_{s=t_r}^{d_r} \mathcal{S}_{I^\star}\big(a^{(r,s)}\big)$.
6:  for $t = d_r, d_r - 1, \dots, 0$ do
7:      Set $\mathcal{C}^{(r)}_t = \{z \in \mathcal{R}^{(r)}_{t+1} : V^{k_1 \dots k_t} \cdot (z - u) \leq b|_{k_1 \dots k_t}\} \cap \mathcal{S}_{I^\star}\big(a^{(r,t)}\big)$.
8:      if $\mathcal{C}^{(r)}_t \neq \emptyset$ then
9:          Set $x^{(r,t)} = \arg\min\{V^{k_{t+1}} z : z \in \mathcal{C}^{(r)}_t\}$.
10:         if $t = t_r$ then
11:             Let $K_{\mathrm{eq}} \subseteq K$ be the tuple of indices $k$ for which $r_k = r$.
12:             Set $K_{\mathrm{neq}} = K \setminus K_{\mathrm{eq}}$.
13:             return $\mathsf{Witness}_0\big(V, K_{\mathrm{neq}}, r + 1, I^\star, A, \mathcal{S}' \cap \bigcap_{k \in K_{\mathrm{eq}}} \mathcal{S}_{P_k}\big(x^{(r,t)}\big), B, u\big)$
14:         else
15:             Set $\mathcal{R}^{(r)}_t = \{z \in \mathcal{R}^{(r)}_{t+1} : V^{k_{t+1}} z < V^{k_{t+1}} x^{(r,t)}\} \cap \bigcup_{s=t_r}^{t-1} \mathcal{S}_{I^\star}\big(a^{(r,s)}\big)$.
16:         end if
17:     else
18:         Set $x^{(r,t)} = (\perp, \dots, \perp)$.
19:         Set $\mathcal{R}^{(r)}_t = \mathcal{R}^{(r)}_{t+1} \cap \bigcup_{s=t_r}^{t-1} \mathcal{S}_{I^\star}\big(a^{(r,s)}\big)$.
20:     end if
21: end for
22: return $\emptyset$
```

It is important to break ties in Line 9 the same way as we did in the original $\mathsf{Witness}_0$ function, i.e., we take the lexicographically first solution.

Lemma 7.7.6. *Let $(I^\star, A)$ be the V-certificate of x, let $u \in \{0, \dots, \mathcal{K}\}^n$ be an arbitrary vector, and let $B = B_V(x - u)$. Then the call $\mathsf{Witness}_0(V, [d], 1, I^\star, A, \mathcal{S}, B, u)$ returns $\{x\}$.*

Before we give a formal proof of Lemma 7.7.6 we try to give some intuition for it. As for the simple variant of the $\mathsf{Witness}$ function we restrict the set of solutions to vectors that look like the vectors we want to reconstruct in the next rounds of the current call, i.e., we intersect the current set with the set $\bigcup_{s=t_r}^{t-1} \mathcal{S}_{I^\star}\big(a^{(r,s)}\big)$ in round t. That way we only deal with subsets of the original sets, but we do not lose the vectors we want to reconstruct. In order to reconstruct the vectors, we need more information than in the simple variant:

we need to know in which rounds the recursive calls of $\mathsf{Witness}_0$ are made, in each call we need to know which objective functions V^k must not be considered anymore, and for each of these objective functions we need to know the vector $x|_{P_k}$. The information when the recursive calls are made and which objective functions must not be considered anymore is given in the certificate: The variable t_r contains the round number when the recursive call is made. The index r_k contains the number of the call where index k has to be removed from K. Hence, index k is removed in the $t_{r_k}^{\text{th}}$ round of call r_k. If we can reconstruct K_{eq} and the vector $x^{(r,t)}$ in the round where we make the recursive call, then we can also reconstruct the bits of x at indices $i \in P_k$ for any index $k \in K_{\text{eq}}$ because $x|_{P_k} = x^{(r,t)}|_{P_k}$ for these indices k (cf. Line 13).

Proof. We compare the execution of the call $\mathsf{Witness}_0(V, x, [d], 1, ())$ with the execution of the call $\mathsf{Witness}_0(V, [d], 1, I^\star, A, \mathcal{S}, B, u)$ and show the following claim by induction on r.

Claim 9. *If there is a call of the form* $\mathsf{Witness}_0(V, x, K, r, I)$ *during the execution of the call* $\mathsf{Witness}_0(V, x, [d], 1, ())$, *then there is a call* $\mathsf{Witness}_0(V, K, r, I^\star, A, \mathcal{S}', B, u)$ *for* $\mathcal{S}' = \bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x)$ *during the execution of the call* $\mathsf{Witness}_0(V, [d], 1, I^\star, A, \mathcal{S}, B, u)$.

If Claim 9 is true, then the correctness of Lemma 7.7.6 follows: In accordance with Lemma 7.7.3, the call $\mathsf{Witness}_0(V, x, [d], 1, ())$ returns the set $\{x\} \neq \emptyset$. Hence, there is a call of the form $\mathsf{Witness}_0(V, x, K, r, I)$ for $K = ()$ (see Line 4). Due to Claim 9, there must be also a call of the form $\mathsf{Witness}_0(V, (), r, I^\star, A, \mathcal{S}', B, u)$ for $\mathcal{S}' = \bigcap_{k \in [d] \setminus ()} \mathcal{S}_{P_k}(x) = \{x\}$. This set is immediately returned in Line 4.

Let us prove Claim 9. Recalling that $\bigcap_{k \in ()} \mathcal{S}_{P_k}(x) = \mathcal{S}$ it holds for $r = 1$. Now let us consider an arbitrary call $r+1$ and assume that Claim 9 holds for r. That is, we can assume that there are calls of the form $\mathsf{Witness}_0(V, x, K, r, I)$ and $\mathsf{Witness}_0(V, K, r, I^\star, A, \mathcal{S}', B, u)$ for $\mathcal{S}' = \bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x)$. We show that both calls are executed essentially the same way. Formally, we prove the following claims by induction on t, where $\mathcal{R}_t'^{(r)}$, $\mathcal{C}_t'^{(r)}$, $x'^{(r,t)}$, K'_{eq}, and K'_{neq} refer to the sets, vectors, and tuples from the call $\mathsf{Witness}_0(V, K, r, I^\star, A, \mathcal{S}', B, u)$.

Claim 10. $\mathcal{R}_t'^{(r)} \subseteq \mathcal{R}_t^{(r)}$ *for any* $t \in \{t_r + 1, \dots, d_r + 1\}$.

Claim 11. $x'^{(r,t)} = x^{(r,t)}$ *for any* $t \in \{t_r, \dots, d_r\}$ *for which* $\mathcal{C}_t^{(r)} \neq \emptyset$.

Claim 12. $x^{(r,s)} \in \mathcal{R}_t'^{(r)}$ *for any* $t \in \{t_r + 1, \dots, d_r + 1\}$ *and for any* $s \in \{t_r, \dots, t - 1\}$ *for which* $\mathcal{C}_s^{(r)} \neq \emptyset$.

The induction step of the proof of Claim 9 follows from these claims: Let us assume that there is a call of the form $\mathsf{Witness}_0(V, x, \hat{K}, r + 1, \hat{I})$. By the definition of t_r, this call is executed in round t_r of call r. Consequently, $x^{(r,t_r)} \in \mathcal{C}_{t_r}^{(r)} \neq \emptyset$. Applying Claim 12 for $s = t_r$ and $t = t_r + 1$, we obtain $x^{(r,t_r)} \in \mathcal{R}_{t_r+1}'^{(r)}$. As $x^{(r,t_r)} \in \mathcal{C}_{t_r}^{(r)}$, the inequalities $V^{k_1 \dots k_{t_r}} x^{(r,t_r)} < V^{k_1 \dots k_{t_r}} x$ hold, which are equivalent to $V^{k_1 \dots k_{t_r}} \cdot (x^{(r,t_r)} - u) \leq b|_{k_1 \dots k_{t_r}}$ due to the occurence of the OK_0-event. Furthermore, $x^{(r,t_r)} \in \mathcal{S}_{I^\star}(a^{(r,t_r)})$ by the definition of $a^{(r,t_r)}$. Hence, $x^{(r,t_r)} \in \mathcal{C}'^{(r)}_{t_r}$ (see Line 7), i.e., $\mathcal{C}'^{(r)}_{t_r} \neq \emptyset$. Moreover, $x'^{(r,t_r)} = x^{(r,t_r)}$ in accordance with Claim 11. In round t_r of the call $\mathsf{Witness_0}(V, K, r, I^\star, A, \mathcal{S}', B, u)$ Line 11 is reached. By the definition of the values r_k we obtain $K'_{\text{eq}} = K_{\text{eq}}$, and hence, $x'^{(r,t_r)}|_{P_k} = x^{(r,t_r)}|_{P_k} = x|_{P_k}$ for any $k \in K'_{\text{eq}} = K_{\text{eq}}$ due to the definition of K_{eq}. In Line 13, there is a call of the form $\mathsf{Witness_0}(V, K'_{\text{neq}}, r+1, I^\star, A, \mathcal{S}' \cap \bigcap_{k \in K'_{\text{eq}}} \mathcal{S}_{P_k}(x^{(r,t_r)}), B, u)$. The correctness of Claim 9 follows because

$$K'_{\text{neq}} = K \setminus K'_{\text{eq}} = K \setminus K_{\text{eq}} = K_{\text{neq}} = \hat{K}\,,$$

where $\hat{K}$ is the parameter from the call $\mathsf{Witness_0}(V, x, \hat{K}, r+1, \hat{I})$, and because

$$\begin{aligned}\mathcal{S}' \cap \bigcap_{k \in K'_{\text{eq}}} \mathcal{S}_{P_k}(x^{(r,t_r)}), B, u) &= \bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x) \cap \bigcap_{k \in K'_{\text{eq}}} \mathcal{S}_{P_k}(x^{(r,t_r)}) \\ &= \bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x) \cap \bigcap_{k \in K'_{\text{eq}}} \mathcal{S}_{P_k}(x) \\ &= \bigcap_{k \in [d] \setminus K'_{\text{neq}}} \mathcal{S}_{P_k}(x)\,.\end{aligned}$$

Let us finish the proof of Lemma 7.7.6 by proving the Claims 10, 11, and 12 by downward induction on t. For the beginning, consider $t = d_r + 1$. We have $\mathcal{R}'^{(r)}_{d_r+1} = \mathcal{S}' \cap \bigcup_{s=t_r}^{d_r} \mathcal{S}_{I^\star}(a^{(r,s)})$ for $\mathcal{S}' = \bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x)$ and $\mathcal{R}^{(r)}_{d_r+1} = \mathcal{S}_I(x) \cap \bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x)$. Due to the construction of the vectors $x^{(r,s)}$ and the definition of $a^{(r,s)}$, $a^{(r,s)}|_I = x^{(r,s)}|_I = x|_I$ for any $s = t_r, \dots, d_r$ (see Lemma 7.7.5). The inclusion $I^\star \supseteq I$ yields $\mathcal{S}_{I^\star}(a^{(r,s)}) \subseteq \mathcal{S}_I(a^{(r,s)}) = \mathcal{S}_I(x)$ for any $s = t_r, \dots, d_r$. Consequently, $\mathcal{R}'^{(r)}_{d_r+1} \subseteq \mathcal{R}^{(r)}_{d_r+1}$. For Claim 11 nothing has to be shown in the initial step $t = d_r + 1$ of the induction. Let us consider Claim 12 and let $s \in \{t_r, \dots, d_r\}$ be an arbitrary index for which $\mathcal{C}^{(r)}_s \neq \emptyset$. Then

$$x^{(r,s)} \in \mathcal{C}^{(r)}_s \subseteq \mathcal{R}^{(r)}_{s+1} \subseteq \mathcal{R}^{(r)}_{d_r+1} \subseteq \bigcap_{k \in [d] \setminus K} \mathcal{S}_{P_k}(x)\,.$$

Furthermore, $x^{(r,s)} \in \mathcal{S}_J(a^{(r,s)})$ for any index tuple J due to the definition of $a^{(r,s)}$. Consequently, $x^{(r,s)} \in \mathcal{R}'^{(r)}_{d_r+1}$.

For the induction step let $t \leq d_r$. Due to the occurence of the OK_0-event and the fact that $B = B_V(x-u)$, we obtain

$$\begin{aligned}\mathcal{C}'^{(r)}_t &= \{z \in \mathcal{R}'^{(r)}_{t+1} : V^{k_1 \dots k_t} \cdot (z - u) \leq b|_{k_1 \dots k_t}\} \cap \mathcal{S}_{I^\star}(a^{(r,t)}) \\ &= \{z \in \mathcal{R}'^{(r)}_{t+1} : V^{k_1 \dots k_t} z < V^{k_1 \dots k_t} x\} \cap \mathcal{S}_{I^\star}(a^{(r,t)})\end{aligned}$$

and $\mathcal{C}^{(r)}_t = \{z \in \mathcal{R}^{(r)}_{t+1} : V^{k_1 \dots k_t} z < V^{k_1 \dots k_t} x\}$. Since $\mathcal{R}'^{(r)}_{t+1} \subseteq \mathcal{R}^{(r)}_{t+1}$, we obtain $\mathcal{C}'^{(r)}_t \subseteq \mathcal{C}^{(r)}_t$. First, we consider the case $\mathcal{C}^{(r)}_t = \emptyset$ which implies $\mathcal{C}'^{(r)}_t = \emptyset$ and $t \geq t_r + 1$. The inequality follows from the fact that in round t_r the $\mathsf{Witness}_0(f)$ unction is called (Line 23 of Algorithm 7) which implies $\mathcal{C}_{t_r}^{(r)} \neq \emptyset$. In this case, $\mathcal{R}_t'^{(r)} = \mathcal{R}_{t+1}'^{(r)} \cap \bigcup_{s=t_r}^{t-1} \mathcal{S}_{I^\star}(a^{(r,s)})$ and $\mathcal{R}_t^{(r)} = \mathcal{R}_{t+1}^{(r)} \cap \mathcal{S}_I(x)$, where I is the updated index tuple I. Due to the construction of the vectors $x^{(r,s)}$ (see Lemma 7.7.5) and the definition of the vectors $a^{(r,s)}$, we know that $a^{(r,s)}|_I = x^{(r,s)}|_I = x|_I$ for any $s = t_r, \dots, t-1$. As $I^\star \supseteq I$, this implies $\bigcup_{s=t_r}^{t-1} \mathcal{S}_{I^\star}(a^{(r,s)}) \subseteq \bigcup_{s=t_r}^{t-1} \mathcal{S}_I(a^{(r,s)}) = \mathcal{S}_I(x)$. As $\mathcal{R}_{t+1}'^{(r)} \subseteq \mathcal{R}_{t+1}^{(r)}$ in accordance with the induction hypothesis, Claim 10, we obtain $\mathcal{R}_t'^{(r)} \subseteq \mathcal{R}_t^{(r)}$. For Claim 11 nothing has to be shown here. Let $s \in \{t_r, \dots, t-1\}$ be an arbitrary index for which $\mathcal{C}_s^{(r)} \neq \emptyset$. Then $x^{(r,s)} \in \mathcal{R}_{t+1}'^{(r)}$ by Claim 12 of the induction hypothesis, $x^{(r,s)} \in \mathcal{S}_{I^\star}(a^{(r,s)})$, and consequently $x^{(r,s)} \in \mathcal{R}_t'^{(r)}$.

Let us finally consider the case $\mathcal{C}_t^{(r)} \neq \emptyset$. Claim 12 of the induction hypothesis yields $x^{(r,t)} \in \mathcal{R}_{t+1}'^{(r)}$. Since $x^{(r,t)} \in \mathcal{S}_{I^\star}\big(a^{(r,t)}\big)$ and $V^{k_1 \dots k_t} x^{(r,t)} < V^{k_1 \dots k_t} x$, also $x^{(r,t)} \in \mathcal{C}_t'^{(r)}$ and, thus, $\mathcal{C}_t'^{(r)} \neq \emptyset$. Hence, $x'^{(r,t)} = x^{(r,t)}$ as $\mathcal{C}_t'^{(r)} \subseteq \mathcal{C}_t^{(r)}$. Claim 10 and Claim 12 have only to be validated if $t \geq t_r$, i.e., we can assume that $K_{\text{eq}} = ()$. Then $\mathcal{R}_t'^{(r)} = \{z \in \mathcal{R}_{t+1}'^{(r)} : V^{k_{t+1}} z < V^{k_{t+1}} x^{(r,t)}\} \cap \bigcup_{s=t_r}^{t-1} \mathcal{S}_{I^\star}(a^{(r,s)})$, because $x'^{(r,t)} = x^{(r,t)}$, and $\mathcal{R}_t^{(r)} = \{z \in \mathcal{R}_{t+1}^{(r)} : V^{k_{t+1}} z < V^{k_{t+1}} x^{(r,t)}\} \cap \mathcal{S}_I(x)$. With the same argument used for the case $\mathcal{C}_t^{(r)} = \emptyset$ we obtain $\mathcal{R}_{t+1}'^{(r)} \cap \bigcup_{s=t_r}^{t-1} \mathcal{S}_{I^\star}(a^{(r,s)}) \subseteq \mathcal{R}_{t+1}^{(r)} \cap \mathcal{S}_I(x)$ and, hence, $\mathcal{R}_t'^{(r)} \subseteq \mathcal{R}_t^{(r)}$. Consider an arbitrary index $s \in \{t_r, \dots, t-1\}$ for which $\mathcal{C}_s^{(r)} \neq \emptyset$. Then $x^{(r,s)} \in \mathcal{C}_s^{(r)} \subseteq \mathcal{R}_{s+1}^{(r)} \subseteq \mathcal{R}_t^{(r)}$. In particular, $V^{k_{t+1}} x^{(r,s)} < V^{k_{t+1}} x^{(r,t)}$ (see Line 20) and, hence, $V^{k_{t+1}} x^{(r,s)} < V^{k_{t+1}} x'^{(r,t)}$ because $x'^{(r,t)} = x^{(r,t)}$. Furthermore, $x^{(r,s)} \in \mathcal{R}_{t+1}'^{(r)}$ due to the induction hypothesis, Claim 12, and $x^{(r,s)} \in \mathcal{S}_{I^\star}(a^{(r,s)})$. Consequently, $x^{(r,s)} \in \mathcal{R}_t'^{(r)}$. □

By the choice of the vector u we can control which information about V has to be known in order to be able to execute the call $\mathsf{Witness}_0(V, [d], 1, I^\star, A, \mathcal{S}, B, u)$. While Lemma 7.7.6 is correct for any choice of $u \in \{0, \dots, \mathcal{K}\}^n$, we have to choose u carefully in order for the following probabilistic analysis to work. Later we will see that $u^\star = u^\star(I^\star, A)$, given by

$$u_i^\star = \begin{cases} |x_i - 1| & \text{if } i \in (i_1^\star, \dots, i_d^\star)\,, \\ x_i & \text{if } i \in I^\star \setminus (i_1^\star, \dots, i_d^\star)\,, \\ 0 & \text{otherwise}\,, \end{cases} \in \{0, \dots, \mathcal{K}\}^n \tag{7.3}$$

is well-suited for our purpose. Recall, that $i_k^\star \in P_k$ are the indices that have been added to I in the definition of the V-certificate to obtain $I^\star$. Furthermore, x_i is given by the last column of A for any index $i \in I^\star$ (cf. Lemma 7.7.3). Hence, vector $u^\star$ can be defined with the information that is contained in the V-certificate of x; we do not have to know the vector x itself.

In the next step, we bound the number of Pareto-optimal solutions. For this, consider the following function $\chi_{I^\star,A,B}(V)$, parameterized by an arbitrary certificate $(I^\star, A) \in \mathscr{C}$ and an arbitrary ε-box $B \in \mathbb{B}_\varepsilon$, that is defined as follows: $\chi_{I^\star,A,B}(V) = 1$ if the call

$\mathsf{Witness_0}(V, [d], 1, I^\star, A, \mathcal{S}, B, u^\star(I^\star, A))$ returns a set $\{x'\}$ such that $B_V\big(x' - u^\star(I^\star, A)\big) = B$, and $\chi_{I^\star,A,B}(V) = 0$ otherwise.

Corollary 7.7.7. *Assume that* $\mathrm{OK}_0(V)$ *is true. Then the number* $\mathrm{PO}(V)$ *of Pareto-optimal solutions is at most* $\sum\limits_{(I^\star,A)\in\mathscr{C}} \sum\limits_{B\in\mathbb{B}_\varepsilon} \chi_{I^\star,A,B}(V)$.

Proof. Let x be a Pareto-optimal solution, let $(I^\star, A)$ be the V-certificate of x, and let $B = B_V\big(x - u^\star(I^\star, A)\big) \in \mathbb{B}_\varepsilon$. Due to Lemma 7.7.6, $\mathsf{Witness_0}(V, [d], 1, I^\star, A, \mathcal{S}, B, u^\star(I^\star, A))$ returns $\{x\}$. Hence, $\chi_{I^\star,A,B}(V) = 1$. It remains to show that this function $x \mapsto (I^\star, A, B')$ defined within the previous lines is injective. Let x_1 and x_2 be distinct Pareto-optimal solutions and let $(I_1^\star, A_1)$ and $(I_2^\star, A_2)$ be the V-certificates of x_1 and x_2, respectively. If $(I_1^\star, A_1) \neq (I_2^\star, A_2)$, then x_1 and x_2 are mapped to distinct triplets. Otherwise, $u^\star(I_1^\star, A_1) = u^\star(I_2^\star, A_2)$ and, hence, $B_V\big(x_1 - u^\star(I_1^\star, A_1)\big) \neq B_V\big(x_2 - u^\star(I_2^\star, A_2)\big)$ because $\mathrm{OK}_0(V)$ is true. Consequently, also in this case x_1 and x_2 are mapped to distinct triplets. □

Corollary 7.7.7 immediately implies a bound on the expected number of Pareto-optimal solutions.

Corollary 7.7.8. *The expected number of Pareto-optimal solutions is bounded by*

$$\mathbf{E}_V\left[\mathrm{PO}(V)\right] \le \sum_{(I^\star,A)\in\mathscr{C}} \sum_{B\in\mathbb{B}_\varepsilon} \mathbf{Pr}_V\left[E_{I^\star,A,B}\right] + (\mathcal{K}+1)^n \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}_0(V)}\right]$$

where $E_{I^\star,A,B}$ *denotes the event that the call* $\mathsf{Witness_0}(V, [d], 1, I^\star, A, \mathcal{S}, B, u^\star(I^\star, A))$ *returns a set* $\{x'\}$ *such that* $B_V\big(x' - u^\star(I^\star, A)\big) = B$.

Proof. By applying Corollary 7.7.7, we obtain

$$\begin{aligned}
&\mathbf{E}_V\left[\mathrm{PO}(V)\right] \\
&= \mathbf{E}_V\left[\mathrm{PO}(V) \,\middle|\, \mathrm{OK}_0(V)\right] \cdot \mathbf{Pr}_V\left[\mathrm{OK}_0(V)\right] + \mathbf{E}_V\left[\mathrm{PO}(V) \,\middle|\, \overline{\mathrm{OK}_0(V)}\right] \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}_0(V)}\right] \\
&\le \mathbf{E}_V\left[\sum_{(I^\star,A)\in\mathscr{C}} \sum_{B\in\mathbb{B}_\varepsilon} \chi_{I^\star,A,B}(V) \,\middle|\, \mathrm{OK}_0(V)\right] \cdot \mathbf{Pr}_V\left[\mathrm{OK}_0(V)\right] + |\mathcal{S}| \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}_0(V)}\right] \\
&\le \mathbf{E}_V\left[\sum_{(I^\star,A)\in\mathscr{C}} \sum_{B\in\mathbb{B}_\varepsilon} \chi_{I^\star,A,B}(V)\right] + (\mathcal{K}+1)^n \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}_0(V)}\right] \\
&= \sum_{(I^\star,A)\in\mathscr{C}} \sum_{B\in\mathbb{B}_\varepsilon} \mathbf{Pr}_V\left[E_{I^\star,A,B}\right] + (\mathcal{K}+1)^n \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}_0(V)}\right] .
\end{aligned}$$

□

We will see that the first term of the sum in Corollary 7.7.8 can be bounded independently of ε and that the second term tends to 0 for $\varepsilon \to 0$. First of all, we analyze the size of the certificate space.

Lemma 7.7.9. *The size of the certificate space is bounded by*

$$|\mathscr{C}| = (\mathcal{K}+1)^{(d^2+d)(d^3+d^2+d)} \cdot O\big(n^{d^3+d^2}\big)\,.$$

Proof. Consider the execution of the call $\mathsf{Witness_0}(V, x, [d], 1, ())$, in which, among others, the indices $r_1, \ldots, r_d$ are defined, and let $r^\star = \max\{r_1, \ldots, r_d\}$ be the maximum of these indices. Including the call with number 1, there can be at most d calls to the $\mathsf{Witness_0}$ function except for the call with number $r^\star + 1$ that terminates due to $d_{r^\star+1} = 0$. This is because in each of the other calls at least one index $k \in [d]$ is removed from the tuple K. Hence, $r_1, \ldots, r_d \in [d]$. Consequently, there are at most d^d possibilities for these numbers. In the r^{th} call, the round number t_r is an element of $[d_r]_0 \subseteq [d]_0$, and hence, there are at most $(d+1)^{r^\star} \leq (d+1)^d$ possibilities to choose round numbers $t_1, \ldots, t_{r^\star}$. In each round, at most d indices i are added to the tuple I. As there are at most d calls and at most $d+1$ rounds each call, tuple I contains at most $d^2 \cdot (d+1)$ indices in total. Hence, there are at most $\sum_{k=1}^{d^2(d+1)} n^k \leq d^2 \cdot (d+1) \cdot n^{d^2 \cdot (d+1)}$ choices for I. Once I is fixed, also the indices in $I^\star \setminus I$ are fixed because the indices added to I in Definition 7.7.4 are determined by I. The tuple $I^\star$ contains $|I| + d \leq d^3 + d^2 + d$ indices. In each call r, at most $d+1$ vectors $x^{(r,t)}$ are generated. Hence, matrix A has at most $d \cdot (d+1)$ columns and at most $d^3 + d^2 + d$ rows. This yields the claimed bound

$$\begin{aligned}
|\mathscr{C}| &\leq d^d \cdot (d+1)^d \cdot d^2 \cdot (d+1) \cdot n^{d^2(d+1)} \cdot (\mathcal{K}+1)^{d(d+1)\cdot(d^3+d^2+d)} \\
&\leq 2^{d+1} \cdot d^{2d+3} \cdot n^{d^3+d^2} \cdot (\mathcal{K}+1)^{(d^2+d)(d^3+d^2+d)} \\
&= (\mathcal{K}+1)^{(d^2+d)(d^3+d^2+d)} \cdot O\big(n^{d^3+d^2}\big)\,.
\end{aligned}$$

□

In the next step we analyze how much information of V is required in order to perform the call $\mathsf{Witness_0}(V, [d], 1, I^\star, A, \mathcal{S}, B, u)$. We will see that V does not need to be revealed completely and that some randomness remains even after the necessary information to perform the call has been revealed. This is the key observation for analyzing the probability $\mathbf{Pr}_V\left[E_{I^\star,A,B}\right]$. For this, let V be an arbitrary realization, i.e., we do not condition on the OK_0-event anymore, and fix an index $k \in [d]$. Only the indices $i \in P_k$ are relevant for function V^k. We set $I^\star_k = I^\star \cap P_k$ and apply the principle of deferred decisions by assuming that the coefficients of V^k belonging to indices $i \notin I^\star_k$ are fixed arbitrarily. We denote this part of V^k by $V^k_{\overline{I^\star_k}}$ and concentrate on the remaining part of V^k which we denote by $V^k_{I^\star_k}$. By the construction of index r_k we know that only in the calls $r = 1, \ldots, r_k$ information about the function V^k must be available as in all subsequent calls this function is not considered anymore.

Since in the r^{th} call we consider only vectors that agree with one of the vectors $x^{(r,t_r)}, \dots, x^{(r,d_r)}$ (see Line 5) in the indices $i \in I^\star$, it suffices to know all linear combinations $V^k_{I^\star_k} \cdot \big(a^{(r,t)}|_{I^\star_k} - u|_{I^\star_k}\big)$. However, in call r_k we would reveal too much information about x such that there would be no randomness left. Therefore, we reveal those linear combinations for all calls $r \in [r_k - 1]$ and analyze call r_k more detailed like we did in the case of non-zero-preserving perturbations:

Consider call number r_k and let $j_k \in [d_{r_k}]$ be the index for which $k_{j_k} = k$ in round r_k, i.e., V^k is the j_k^{th} objective function in K in round r_k . For this call the inequality $t_{r_k} \leq j_k - 1$ holds. This is due to the fact that $x^{(r_k,t_{r_k})}\big|_{P_k} = x|_{P_k}$, i.e., $V^k x^{(r_k,t_{r_k})} = V^k x$, since this is the round when k is removed from tuple K. On the other hand, $x^{(r_k,t_{r_k})} \in \mathcal{C}^{(r_k)}_{t_{r_k}}$, which means that $V^{k_s} x^{(r_k,t_{r_k})} < V^{k_s} x$ for all indices $s = 1, \dots, t_{r_k}$, where $k_1, \dots, k_{d_{r_k}}$ denote the indices tuple K consists of in round r_k. Hence, $t_{r_k} < j_k$ as $k = k_{j_k}$.

There are only three lines where information about V is required: Line 7, Line 9, and Line 15. For Line 7 the values $V^k_{I^\star_k} \cdot \big(a^{(r_k,t)}|_{I^\star_k} - u|_{I^\star_k}\big)$ from round $t = d_{r_k}$ down to round j_k are needed. In Line 9 no additional information about $V^k_{I^\star}$ is needed since all considered vectors agree on indices $i \in I^\star_k$ with each other. For Line 15 only in round $j_k - 1$ the values $V^k_{I^\star_k} \cdot \big(a^{(r_k,s)}|_{I^\star_k} - a^{(r_k,j_k-1)}|_{I^\star_k}\big)$ for $s = t_{r_k}, \dots, j_k - 2$ are required. We write the linear combinations of $V^k_{I^\star_k}$ of the calls $r = 1, \dots, r_k - 1$ and of call r_k into the matrices P_k and Q_k, respectively, and obtain

$$P_k = \big[a^{(1,d_1)} - u|_{I^\star}, \dots, a^{(1,t_1)} - u|_{I^\star}, \dots, a^{(r_k-1,d_{r_k-1})} - u|_{I^\star}, \dots, a^{(r_k-1,t_{r_k-1})} - u|_{I^\star}\big]\big|_{I^\star_k}$$

and

$$Q_k = \big[a^{(r_k,d_{r_k})} - u|_{I^\star}, \dots, a^{(r_k,j_k)} - u|_{I^\star}, a^{(r_k,j_k-2)} - a^{(r_k,j_k-1)}, \dots, a^{(r_k,t_{r_k})} - a^{(r_k,j_k-1)}\big]\big|_{I^\star_k} .$$

Using the notation $p^{(r,t)}_k = a^{(r,t)}\big|_{I^\star_k} - u|_{I^\star_k}$ we can write both matrices as

$$\begin{aligned} P_k &= \Big[p^{(1,d_1)}_k, \dots, p^{(1,t_1)}_k, \dots, p^{(r_k-1,d_{r_k-1})}_k, \dots, p^{(r_k-1,t_{r_k-1})}_k\Big]\Big|_{I^\star_k} \quad \text{and} \\ Q_k &= \Big[p^{(r_k,d_{r_k})}_k, \dots, p^{(r_k,j_k)}_k, p^{(r_k,j_k-2)}_k - p^{(r_k,j_k-1)}_k, \dots, p^{(r_k,t_{r_k})}_k - p^{(r_k,j_k-1)}_k\Big]\Big|_{I^\star_k} . \end{aligned}$$

Note that the matrices $P_k = P_k(I^\star, A, u)$ and $Q_k = Q_k(I^\star, A, u)$ depend, among others, on the choice of u. Matrix Q_k has $(d_{r_k} - j_k + 1) + ((j_k - 2) - t_{r_k} + 1) = d_{r_k} - t_{r_k}$ columns, matrix P_k has

$$\sum_{r=1}^{r_{k-1}} (d_r - t_r + 1) = \sum_{r=1}^{r_k} (d_r - t_r + 1) - (d_{r_k} - t_{r_k} + 1) = |I^\star_k| - (d_{r_k} - t_{r_k} + 1)$$

columns. The last equation is due to the fact that in each call $r < r_k$ in each round one index i_k is chosen. In call $r = r_k$ one index i_k is chosen in each round $t > t_{r_k}$. The equation follows since $I^\star_k$ contains one index more than the number of indices i_k that are chosen during the execution of the $\mathsf{Witness_0}$ function (see Definition 7.7.4). Moreover, observe that all entries of P_k and Q_k are from $\{-\mathcal{K}, \dots, \mathcal{K}\}$.

Corollary 7.7.10. *Let $u \in \{0, \dots, \mathcal{K}\}^n$ be an arbitrary shift vector, let $(I^\star, A) \in \mathscr{C}$ be an arbitrary certificate, and let V and W be two realizations for which $V^k_{\overline{I^\star_k}} = W^k_{\overline{I^\star_k}}$ and $V^k_{I^\star_k} \cdot q = W^k_{I^\star_k} \cdot q$ for any index $k \in [d]$ and any column q of one of the matrices $P_k(I^\star, A, u)$ and $Q_k(I^\star, A, u)$. Then the calls $\mathsf{Witness_0}(V, [d], 1, I^\star, A, \mathcal{S}, B, u)$ and $\mathsf{Witness_0}(W, [d], 1, I^\star, A, \mathcal{S}, B, u)$ return the same result for any ε-box $B \in \mathbb{B}_\varepsilon$.*

In the remainder of this section we assume that $V^k_{\overline{I^\star_k}}$ and the ε-box B are fixed. In accordance with Corollary 7.7.10, the output of the call $\mathsf{Witness_0}(V, [d], 1, I^\star, A, \mathcal{S}, B, u)$ is determined if the linear combinations of $V^k_{I^\star_k}$ given by the columns of the matrices P_k and Q_k are fixed arbitrarily, i.e., it does not depend on the remaining randomness in the coefficients. We are interested in the event $E_{I^\star,A,B}$, i.e., in the event that the output is a set $\{x'\}$ such that $V^{1\dots d} \cdot \big(x' - u^\star(I^\star, A)\big) \in B$. Since $(I^\star, A)$ is a V-certificate of x for some V and x, the output is always of the form $\{x'\}$ due to Lemma 7.7.6. Hence, event $E_{I^\star,A,B}$ occurs if and only if for all indices k the relation $V^k_{I^\star_k} \cdot \big(x' - u^\star(I^\star, A)\big)\big|_{I^\star_k} \in C_k$ holds for some interval C_k of length ε that depends on the linear combinations of $V_{I^\star_\ell}$ given by P_ℓ and Q_ℓ for all indices $\ell \in [d]$.

Lemma 7.7.11. *For any fixed index $k \in [d]$ the columns of matrix $P_k\big(I^\star, A, u^\star(I^\star, A)\big)$, of matrix $Q_k\big(I^\star, A, u^\star(I^\star, A)\big)$, and the vector $p_k^{(r_k, t_{r_k})}$ are linearly independent.*

Proof. Consider the square matrix $\hat{Q}_k$ consisting of the vectors $p_k^{(r,t)}$, for $r \in \{1, \dots, r_k\}$ and $t \in \{t_r, \dots, d_r\}$. Matrix $\hat{Q}_k$ can be obtained from the matrix M of Lemma 7.7.5 by subtracting the vector $u^\star|_{I^\star_k}$ from any of its columns. Due to Lemma 7.7.5 and due to the construction of $u^\star = u^\star(I^\star, A)$ (see Equation 7.3) matrix $\hat{Q}_k$ is a lower triangular matrix and the elements of the principal diagonal are from the set $\{-\mathcal{K}, \dots, \mathcal{K}\} \setminus \{0\}$. This is because $x_i - u^\star_i = x_i - |x_i - 1| \in \{-1, 1\}$ for $i = i^\star_k$ and $\overline{x_i} - u^\star_i = \overline{x_i} - x_i \neq 0$ for any $i \in I^\star_k \setminus (i^\star_k)$, where $\overline{z}$ for $z \in \{0, \dots, \mathcal{K}\}$ represents an arbitrary value from $\{0, \dots, \mathcal{K}\}$ not equal to z. Hence, the vectors $p_k^{(r,t)}$ are linearly independent.

The columns of matrix Q_k and vector $p_k^{(r_k, t_{r_k})}$ are linear combinations of the vectors $p_k^{(r_k, t_{r_k})}, \dots, p_k^{(r_k, d_{r_k})}$, whereas the columns of matrix P_k are the remaining columns of matrix $\hat{Q}_k$. As the vectors $p_k^{(r,t)}$ are linearly independent, it suffices to show that the columns of matrix Q_k and vector $p_k^{(r_k, t_{r_k})}$ are linearly independent. For this, we consider an arbitrary linear combination of the columns of matrix Q_k and vector $p_k^{(r_k, t_{r_k})}$ and show that it is zero if and only if all coefficients are zero. For sake of simplicity, we drop the index k in the remainder of this proof and write r, j, and $p^{(r,t)}$ instead of r_k, j_k, and $p_k^{(r_k,t)}$, respectively.

$$\sum_{t=j}^{d_r} \lambda_t \cdot p^{(r,t)} + \sum_{t=t_r}^{j-2} \lambda_t \cdot \left(p^{(r,t)} - p^{(r,j-1)}\right) + \mu \cdot p^{(r,t_r)} = 0\,.$$

If $t_r = j-1$, then this equation is equivalent to

$$\sum_{t=t_r+1}^{d_r} \lambda_t \cdot p^{(r,t)} + \mu \cdot p^{(r,t_r)} = 0\,.$$

Therefore, all coefficients are zero due to the linear independence of the vectors $p^{(r,t)}$. If $t_r < j-1$, which is the only case remaining due to previous observations, then the equation is equivalent to

$$\sum_{t=j}^{d_r} \lambda_t \cdot p^{(r,t)} + \sum_{t=t_r+1}^{j-2} \lambda_t \cdot p^{(r,t)} - \left(\sum_{t=t_r}^{j-2} \lambda_t\right) \cdot p^{(r,j-1)} + (\lambda_{t_r} + \mu) \cdot p^{(r,t_r)} = 0\,.$$

The linear independence of the vectors $p^{(r,t)}$ implies $\lambda_t = 0$ for $t \in \{t_r+1, \dots, j-2\} \cup \{j, \dots, d_r\}$, $\sum_{t=t_r}^{j-2} \lambda_t = 0$, and $\lambda_{t_r} + \mu = 0$. Consequently, also $\lambda_{t_r} = 0$ and, thus, $\mu = 0$. That is, all coefficients are zero. In both cases, the linear independence of the columns of Q_k and the vector $p^{(r,t_r)}$ follows. $\square$

Corollary 7.7.12. *Let $(I^\star, A) \in \mathscr{C}$ be an arbitrary certificate and let $B \in \mathbb{B}_\varepsilon$ be an arbitrary ε-box. Then*

$$\mathbf{Pr}_V\left[E_{I^\star,A,B}\right] \le (2\gamma\mathcal{K})^{\gamma-d}\phi^\gamma\varepsilon^d$$

for $\gamma = d^3 + d^2 + d$, and

$$\mathbf{Pr}_V\left[E_{I^\star,A,B}\right] \le 2^d(\gamma\mathcal{K})^{\gamma-d}\phi^d\varepsilon^d$$

if all densities are quasiconcave.

Proof. For a fixed index $k \in [d]$ we write the columns of matrix P_k, of matrix Q_k, and vector $p_k^{(r_k,t_{r_k})}$ into one matrix $Q'_k \in \{-\mathcal{K}, \dots, \mathcal{K}\}^{|I_k^\star| \times |I_k^\star|}$ (the number of columns is $\big(|I_k^\star| - (d_{r_k} - t_{r_k} + 1)\big) + (d_{r_k} - t_{r_k}) + 1 = |I_k^\star|$ due to previous observations) and consider the matrix

$$Q' = \begin{bmatrix} Q'_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & Q'_d \end{bmatrix} \in \{-\mathcal{K}, \dots, \mathcal{K}\}^{|I^\star| \times |I^\star|}\,.$$

This matrix has full rank due to Lemma 7.7.11. Now we permute the columns of Q' to obtain a matrix Q whose last d columns belong to the last column of one of the matrices Q_k. That is, the last d columns are $(p_1^{(r_1,t_{r_1})}, 0^{|I_2^\star|}, \dots, 0^{|I_d^\star|}), \dots, (0^{|I_1^\star|}, \dots, 0^{|I_{d-1}^\star|}, p_d^{(r_d,t_{r_d})})$. For any $k \in [d]$ and any index $i_k \in I^\star_k$ let $X_i = V^k_i$ be the i^{th} coefficient of V^k. Event $E_{I^\star,A,B}$ holds if and only if the d linear combinations of the variables X_i given by the last d columns of Q fall into a d-dimensional hypercube C depending on the linear combinations of the variables X_i given by the remaining columns. The claim follows by applying Theorem 3.3 for matrix $A = Q'^{\mathrm{T}}$ and due to the fact that $|I^\star| \leq \gamma$ (see proof of Lemma 7.7.9). $\square$

Proof of Theorem 1.5.4. We begin the proof by showing that the OK_0-event is likely to happen. For any index $t \in [d]$ and any solutions $x, y \in \mathcal{S}$ for which $x|_{P_t} \neq y|_{P_t}$ the probability that $\left|V^t x - V^t y\right| \leq \varepsilon$ is bounded by $2\phi\varepsilon$. To see this, choose one index $i \in P_t$ for which $x_i \neq y_i$ and apply the principle of deferred decisions by fixing all coefficients V^t_j for $j \neq i$ arbitrarily. Then the value V^t_i must fall into an interval of length $2\varepsilon/|x_i - y_i| \leq 2\varepsilon$. The probability for this is bounded by $2\phi\varepsilon$. A union bound over all indices $t \in [d]$ and over all pairs $(x, y) \in \mathcal{S} \times \mathcal{S}$ for which $x|_{P_t} \neq y|_{P_t}$ yields $\mathbf{Pr}_V\left[\overline{\mathrm{OK}_0(V)}\right] \leq 2(\mathcal{K}+1)^{2n} d\phi\varepsilon$.

For $\gamma = d^3 + d^2 + d$, we set $s = (2\gamma\mathcal{K})^{\gamma-d}\phi^\gamma = \mathcal{K}^{\gamma-d} \cdot O(\phi^\gamma)$ for general density functions and $s = 2^d(\gamma\mathcal{K})^{\gamma-d}\phi^d = \mathcal{K}^{\gamma-d} \cdot O(\phi^d)$ if all density functions are quasiconcave. Then we obtain

$$\begin{aligned}
\mathbf{E}_V\left[\mathrm{PO}(V)\right] &\leq \sum_{(I^\star,A)\in\mathscr{C}} \sum_{B\in\mathbb{B}_\varepsilon} \mathbf{Pr}_V\left[E_{I^\star,A,B}\right] + (\mathcal{K}+1)^n \cdot \mathbf{Pr}_V\left[\overline{\mathrm{OK}_0(V)}\right] \\
&\leq \sum_{(I^\star,A)\in\mathscr{C}} \sum_{B\in\mathbb{B}_\varepsilon} s \cdot \varepsilon^d + (\mathcal{K}+1)^n \cdot 2(\mathcal{K}+1)^{2n} d\phi\varepsilon \\
&= |\mathscr{C}| \cdot |\mathbb{B}_\varepsilon| \cdot s \cdot \varepsilon^d + (\mathcal{K}+1)^n \cdot 2(\mathcal{K}+1)^{2n} d\phi\varepsilon \\
&= (\mathcal{K}+1)^{(d^2+d)(d^3+d^2+d)} \cdot O\Big(n^{d^3+d^2}\Big) \cdot \left(\frac{2n\mathcal{K}}{\varepsilon}\right)^d \cdot s \cdot \varepsilon^d + 2(\mathcal{K}+1)^{3n} d\phi\varepsilon \\
&= (\mathcal{K}+1)^{(d^2+d)(d^3+d^2+d)+d} \cdot O\Big(n^{d^3+d^2+d}\Big) \cdot s + 2(\mathcal{K}+1)^{3n} d\phi\varepsilon\,.
\end{aligned}$$

The first inequality is due to Corollary 7.7.8. The second inequality is due to Corollary 7.7.12. The third inequality stems from Lemma 7.7.9. Since this bound is true for arbitrarily small $\varepsilon > 0$, we obtain

$$\mathbf{E}_V\left[\mathrm{PO}(V)\right] = (\mathcal{K}+1)^{(d^2+d)(d^3+d^2+d)+d} \cdot O\Big(n^{d^3+d^2+d}\Big) \cdot s\,.$$

Substituting s and γ by their definitions yields

$$\begin{aligned}
\mathbf{E}_V\left[\mathrm{PO}(V)\right] &= (\mathcal{K}+1)^{(d^2+d)(d^3+d^2+d)+d} \cdot O\Big(n^{d^3+d^2+d}\Big) \cdot \mathcal{K}^{d^3+d^2+d-d} \cdot O\Big(\phi^{d^3+d^2+d}\Big) \\
&= (\mathcal{K}+1)^{(d^2+d+1)(d^3+d^2+d)} \cdot O\Big((n\phi)^{d^3+d^2+d}\Big) \\
&= (\mathcal{K}+1)^{(d+1)^5} \cdot O\Big((n\phi)^{d^3+d^2+d}\Big)
\end{aligned}$$

for general densities and

$$\mathbf{E}_V\left[\mathrm{PO}(V)\right] = (\mathcal{K}+1)^{(d^2+d)(d^3+d^2+d)+d} \cdot O\Big(n^{d^3+d^2+d}\Big) \cdot \mathcal{K}^{\gamma-d} \cdot O\big(\phi^d\big)$$

$$= (\mathcal{K}+1)^{(d^2+d+1)(d^3+d^2+d)} \cdot O\big(n^{d^3+d^2+d}\phi^d\big)$$
$$= (\mathcal{K}+1)^{(d+1)^5} \cdot O\big(n^{d^3+d^2+d}\phi^d\big)$$

for quasiconcave densities. □

7.8 A Lower Bound

In this section we present a lower bound example for the bound stated in Theorem 1.5.2. This example is an instance for the restricted multi-profit knapsack problem. Without loss of generality we assume that $\phi > d$. The initial instance, which will be successively modified during the *copy step* and the *split step*, is an instance with n_0 objects $a_1, \dots, a_{n_0}$ with a total weight $W < 1$ and profit vectors from $[0, 1/\phi]^d$, and a set $\mathcal{S}_0 \subseteq \{0,1\}^{n_0}$ of allowed object combinations.

Starting with the initial instance, we perform n_1 copy steps to increase the size of the Pareto set. For this, consider n_1 groups $i \in [n_1]$, each consisting of d objects b_{ij}, $j \in [d]$, with weight $w_i = (d+1)^{i-1}$ and a profit vector $p_{ij} \in P_{ij}$, where P_{ij} is the Cartesian product $P_{ij} = \prod_{\ell=1}^{d} P_{ij}^{(\ell)}$ of the intervals

$$P_{ij}^{(\ell)} = \begin{cases} \left[m_i - \frac{\lceil m_i \rceil}{\phi}, m_i\right] & \text{if } \ell = j\,, \\ \left[0, \frac{\lceil m_i \rceil}{\phi}\right] & \text{otherwise}\,. \end{cases}$$

The value m_i has to be chosen in such a way that the jth profit of object b_{ij} is guaranteed to be larger than the jth total profit of all objects from groups $i' \leq i$, excluding object b_{ij}, and the objects $a_1, \dots, a_{n_0}$. If this is the case, then no solution that does not use object b_{ij} nor objects from groups $i' > i$ can dominate any solution that uses object b_{ij}. We choose m_i as follows.

$$m_i = \left(\frac{2\phi}{\phi - d}\right)^{i-1} \cdot \frac{n_0 + 2d}{\phi - d} - \frac{d}{\phi - d}\,,$$

i.e., $m_i \approx 2^{i-1} n_0/\phi$ increases exponentially with i. The ℓth profit $(p_{ij})_\ell$ of object b_{ij} for $\ell \neq j$ is very small compared to its jth profit. The intuition behind this choice is that using object b_{ij} should only lead to a gain in the jth profit, but leave the remaining profits nearly unchanged.

Lemma 7.8.1. *For any group $i \in [n_1]$ the following inequality holds:*

$$\frac{n_0}{\phi} + \sum_{k=1}^{i-1} \left(m_k + d \cdot \frac{m_k + 1}{\phi}\right) + (d-1) \cdot \frac{m_i + 1}{\phi} \leq m_i - \frac{m_i + 1}{\phi}\,.$$

Proof. Note, that the inequality is equivalent to

$$\frac{n_0}{\phi} + \sum_{k=1}^{i-1} \left(m_k + d \cdot \frac{m_k + 1}{\phi} \right) \leq m_i - d \cdot \frac{m_i + 1}{\phi} = \frac{\phi - d}{\phi} \cdot m_i - \frac{d}{\phi} \,.$$

We show the claim by induction. For $i = 1$ we obtain

$$\frac{\phi - d}{\phi} \cdot m_i - \frac{d}{\phi} = \left(\frac{\phi - d}{\phi} \cdot \frac{n_0 + 2d}{\phi - d} - \frac{d}{\phi} \right) - \frac{d}{\phi} = \frac{n_0}{\phi} = \frac{n_0}{\phi} + \sum_{k=1}^{i-1} \left(m_k + d \cdot \frac{m_k + 1}{\phi} \right) \,.$$

Assume that the claim holds for some integer i. Then

$$\begin{aligned}
&\frac{n_0}{\phi} + \sum_{k=1}^{i} \left(m_k + d \cdot \frac{m_k + 1}{\phi} \right) \\
&= \frac{n_0}{\phi} + \sum_{k=1}^{i-1} \left(m_k + d \cdot \frac{m_k + 1}{\phi} \right) + \left(m_i + d \cdot \frac{m_i + 1}{\phi} \right) \\
&\leq m_i - d \cdot \frac{m_i + 1}{\phi} + \left(m_i + d \cdot \frac{m_i + 1}{\phi} \right) = 2m_i \,.
\end{aligned}$$

The inequality is due to the induction hypothesis. On the other hand,

$$\begin{aligned}
\frac{\phi - d}{\phi} \cdot m_{i+1} - \frac{d}{\phi} &= \left(\frac{\phi - d}{\phi} \cdot \left(\frac{2\phi}{\phi - d} \right)^i \cdot \frac{n_0 + 2d}{\phi - d} - \frac{d}{\phi} \right) - \frac{d}{\phi} \\
&= \frac{\phi - d}{\phi} \cdot \frac{2\phi}{\phi - d} \cdot \left(\frac{2\phi}{\phi - d} \right)^{i-1} \cdot \frac{n_0 + 2d}{\phi - d} - \frac{2d}{\phi} \\
&= 2 \left(m_i + \frac{d}{\phi - d} \right) - \frac{2d}{\phi} \geq 2m_i \,.
\end{aligned}$$

□

Lemma 7.8.1 immediately implies that the intervals $P_{ij}^{(\ell)}$ are subsets of $[0, \infty)$. This is intuitively clear but nevertheless important because we want to scale them down to subintervals of $[0, 1]$ later.

Corollary 7.8.2. *For any $i \in [n_1]$ the inequality $m_i - \lceil m_i \rceil / \phi \geq 0$ holds.*

Proof. It suffices to show that $m_i - (m_i + 1)/\phi \geq 0$ for any $i \in [n_1]$. Note, that

$$m_i = \left(\frac{2\phi}{\phi - d} \right)^{i-1} \cdot \frac{n_0 + 2d}{\phi - d} - \frac{d}{\phi - d} \geq \frac{n_0 + 2d}{\phi - d} - \frac{d}{\phi - d} \geq 0$$

for any $i \in [n_1]$. Consequently,

$$m_i - \frac{m_i + 1}{\phi} \geq m_i - d \cdot \frac{m_i + 1}{\phi} \geq \frac{n_0}{\phi} + \sum_{k=1}^{i-1} \left(m_k + d \cdot \frac{m_k + 1}{\phi} \right) \geq 0 \,,$$

where the second inequality is in accordance with Lemma 7.8.1. □

With Lemma 7.8.1 we now have the technical support for proving that the copy step indeed clones the Pareto set.

Lemma 7.8.3. *Let $\mathcal{S}_1 = \mathcal{S}_0 \times \{0,1\}^{n_1 d}$ and let $\mathcal{P}_0$ and $\mathcal{P}_1$ denote the Pareto set of the restricted multi-profit knapsack instance $K_{\mathcal{S}_0}(\{a_i : i \in [n_0]\})$ and $K_{\mathcal{S}_1}(\{a_i : i \in [n_0]\} \cup \{b_{ij} : i \in [n_1], j \in [d]\})$, respectively. Then $\mathcal{P}_1 \supseteq \mathcal{P}_0 \times \{0,1\}^{n_1 d}$.*

Proof. Let $(y_1, \dots, y_{n_1 d}) \in \{0,1\}^{n_1 d}$ be arbitrary and let $(x_1, \dots, x_{n_0}) \in \mathcal{S}_0$ be Pareto-optimal with respect to $K_{\mathcal{S}_0}(\{a_i : i \in [n_0]\})$. We have to show that the solution z defined as $z = (x_1, \dots, x_{n_0}, y_1, \dots, y_{n_1 d}) \in \mathcal{S}_1$ is Pareto-optimal for $K_{\mathcal{S}_1}(\{a_i : i \in [n_0]\} \cup \{b_{ij} : i \in [n_1], j \in [d]\})$. For this, consider an arbitrary solution $\hat{z} = (\hat{x}_1, \dots, \hat{x}_{n_0}, \hat{y}_1, \dots, \hat{y}_{n_1}) \in \mathcal{S}_1$. We show that $\hat{z}$ does not dominate z. For this, we partition the bits $n_0 + 1, \dots, n_0 + n_1 d$ into n_1 block containing d bits each. The bits of Block i describe which objects of Group i to put into the knapsack.

Let i be the largest number k for which solution z and solution $\hat{z}$ do not use exactly the same objects from Group k. If no such group exists, then set $i = 0$. Now we distinguish between three cases. If $i = 0$, then $\hat{z}$ dominates z if and only if $(\hat{x}_1, \dots, \hat{x}_{n_0})$ dominates $(x_1, \dots, x_{n_0})$. This cannot happen because $(x_1, \dots, x_{n_0})$ is Pareto-optimal. In the remainder of this proof we assume that $i \geq 1$.

If $\hat{z}$ uses more objects from Group i than z, then the difference in weight between $\hat{z}$ and z in the objects of Group i is at least w_i. On the other hand, the absolute value of the difference in weight between $\hat{z}$ and z in the objects $a_1, \dots, a_{n_0}$ and the objects from groups $k < i$ is at most

$$W + \sum_{k=1}^{i-1} d \cdot w_k = W + d \cdot \sum_{k=1}^{i-1} (d+1)^{k-1} < 1 + d \cdot \frac{(d+1)^{i-1} - 1}{(d+1) - 1} = (d+1)^{i-1} = w_i \,.$$

Thus, solution $\hat{z}$ is heavier than solution z. The only case that remains to be considered is the case that z uses at least as many objects from Group i as $\hat{z}$. Then there must be an object b_{ij} used by solution z but not by solution $\hat{z}$. Then the j^{th} profit of z is at least $m_i - \lceil m_i \rceil / \phi > m_i - (m_i + 1)/\phi$, whereas the j^{th} profit of $\hat{z}$ is at most

$$\frac{n_0}{\phi} + \sum_{k=1}^{i-1} \left(m_k + d \cdot \frac{m_k + 1}{\phi} \right) + (d-1) \cdot \frac{m_i + 1}{\phi} \,.$$

The first term is an upper bound for the j^{th} total profit of the objects $a_1, \dots, a_{n_0}$, each term of the sum is an upper bound for the j^{th} total profit of the objects of Group k for some $k \leq i - 1$, and the last term is an upper bound for the j^{th} total profit of the objects $d - 1$ objects $b_{ij'}$, $j' \neq j$, from Group i. Lemma 7.8.1 finishes the proof. □

As already mentioned in the outline, the objects profits of Group i can be as large as m_i, which is not valid if $m_i > 1$. Hence, in the split step we break each object b_{ij} of

Group i into $k_i := \lceil m_i \rceil$ objects $b_{ij}^{(1)}, \dots, b_{ij}^{k_i}$ with weights $\tilde{w}_i = w_i/k_i$ and profit vectors from $\prod_{\ell=d}^{d} \tilde{P}_{ij}^{(\ell)}$, where

$$\tilde{P}_{ij}^{(\ell)} = \begin{cases} \left[\frac{m_i}{k_i} - \frac{1}{\phi}, \frac{m_i}{k_i}\right] & \text{if } \ell = j\,, \\ \left[0, \frac{1}{\phi}\right] & \text{otherwise}\,. \end{cases}$$

Note, that after this split step all these intervals have a length of $1/\phi$ and that their right boundaries are at most 1. In combination with Corollary 7.8.2 this implies that all these intervals are subintervals of $[0,1]$. For an integer $k \geq 1$ let $\psi_k \colon \{0,1\}^d \to \{0,1\}^{k \cdot d}$ denote the function that maps a vector $(x_1, \dots, x_d)$ to the vector $(x_1, \dots, x_1, \dots, x_d, \dots, x_d)$, which contains exactly k copies of each value x_i. In the split step we have to ensure that either all objects $b_{ij}^{(1)}, \dots, b_{ij}^{k_i}$ are chosen, which resembles choosing object b_{ij}, or none of them. This can be managed by defining the set of solutions appropriately. Formally, we set

$$\tilde{\mathcal{S}}_1 = \mathcal{S}_0 \times \prod_{i=1}^{n_1} \left\{ \psi_{k_i}(x) \,:\, x \in \{0,1\}^d \right\} \subseteq \{0,1\}^{n_0} \times \prod_{i=1}^{n_1} \{0,1\}^{k_i \cdot d} = \{0,1\}^{n_0 + d \cdot \sum_{i=1}^{n_1} k_i}\,.$$

Our previous considerations immediately yield the following corollary.

Lemma 7.8.4. *Let $\mathcal{P}_0$ denote the Pareto set of $K_{\mathcal{S}_0}(\{a_i \,:\, i \in [n_0]\})$. Then the instance $K_{\tilde{\mathcal{S}}_1}(\{a_i \,:\, i \in [n_0]\} \cup \{b_{ij}^{(\ell)} \,:\, i \in [n_1], j \in [d], \ell \in [k_i]\})$ consists of*

$$n_0 + d \cdot n_1 + d \cdot \frac{n_0 + 2d}{\phi + d} \cdot \left(\frac{2\phi}{\phi - d}\right)^{n_1}$$

objects and the Pareto set has a size of at least $|\mathcal{P}_0| \cdot 2^{n_1 d}$.

Proof. The bound on the size of the Pareto set follows from Lemma 7.8.3 and the fact that the instance $K_{\tilde{\mathcal{S}}_1}(\{a_i \,:\, i \in [n_0]\} \cup \{b_{ij}^{(\ell)} \,:\, i \in [n_1], j \in [d], \ell \in [k_i]\})$ with split objects resembles the instance $K_{\mathcal{S}_1}(\{a_i \,:\, i \in [n_0]\} \cup \{b_{ij} \,:\, i \in [n_1], j \in [d]\})$ with the original objects. The number of objects is

$$n_0 + d \cdot \sum_{i=1}^{n_1} k_i \leq n_0 + d \cdot \sum_{i=1}^{n_1} (m_i + 1) = n_0 + d \cdot n_1 + d \cdot \sum_{i=1}^{n_1} m_i$$

and

$$\begin{aligned} \sum_{i=1}^{n_1} m_i &\leq \frac{n_0 + 2d}{\phi - d} \cdot \sum_{i=1}^{n_1} \left(\frac{2\phi}{\phi - d}\right)^{i-1} \leq \frac{n_0 + 2d}{\phi - d} \cdot \frac{\left(\frac{2\phi}{\phi - d}\right)^{n_1}}{\frac{2\phi}{\phi - d} - 1} \\ &= \frac{n_0 + 2d}{\phi - d} \cdot \frac{\phi - d}{\phi + d} \cdot \left(\frac{2\phi}{\phi - d}\right)^{n_1} = \frac{n_0 + 2d}{\phi + d} \cdot \left(\frac{2\phi}{\phi - d}\right)^{n_1}\,. \end{aligned}$$

□

The following lemma is due to Beier and Vöcking [BV04] (for $d = 1$) and Goyal and Rademacher [GR12] (for $d \geq 2$).

Lemma 7.8.5 ([BV04, GR12]). *Let $d \geq 1$ be a fixed integer. For any integer $n_0 \geq 1$ there exists a random instance for the restricted multi-profit knapsack problem with n_0 objects $a_1, \ldots, a_{n_0}$ with a total weight of $W < 1$ and profit vectors uniformly drawn from $[0, 1/\phi]^d$ as well as a set $\mathcal{S}_0 \subseteq \{0,1\}^{n_0}$ such that the expected number of Pareto-optimal solutions of $K_{\mathcal{S}_0}(\{a_i : i \in [n_0]\})$ is $\Omega(n_0^{f(d)})$. The function f is defined as $f(1) = 2$ and $f(d) = d - 1.5$ for $d \geq 2$.*

Lemma 7.8.5 suggests how to choose the initial instance. Lemma 7.8.4 states how many objects we create in the copy and in the split step to obtain a certain number of Pareto-optimal solutions. It remains to balance both numbers by choosing n_0 and n_1 appropriately.

Proof of Theorem 1.5.2. Let n be an arbitrary positive integer. Without loss of generality we assume that $n \geq 8d$ and $\phi \geq 2d$. For the moment let us assume that $\phi/d \leq 2^{n/(4d)-1}$. This is the interesting case leading to the first term in the minimum in Theorem 1.5.2. Let

$$\hat{n}_1 = \log_2\left(\frac{\phi}{d}\right) \in \left[1, \frac{n-4d}{4d}\right] \quad \text{and} \quad \hat{n}_0 = \frac{n - 2d \cdot \left(\frac{\phi}{d}\right)^h - d \cdot \hat{n}_1}{1 + \left(\frac{\phi}{d}\right)^h}$$

for $h = h(\phi, d) = \log_2(\phi/(\phi - d))$. In Lemma 7.10.2 we show that for $\phi > d$ the function $\phi \mapsto (\phi/d)^h$ takes only values in $[1, 2]$. This implies $\hat{n}_0 \geq (n - 4d - (n-4d)/4)/3 = (n-4d)/4 \geq d$ since $n \geq 8d$. Now we consider the random restricted multi-profit knapsack problem from Lemma 7.8.5 with $n_0 := \lfloor \hat{n}_0 \rfloor \geq 1$ objects and perform $n_1 := \lfloor \hat{n}_1 \rfloor \geq 1$ copy steps and, subsequently, a split step. In accordance with Lemma 7.8.4, the number N of objects of this new instance is bounded by

$$\begin{aligned} N &\leq n_0 + d \cdot n_1 + d \cdot \frac{n_0 + 2d}{\phi + d} \cdot \left(\frac{2\phi}{\phi - d}\right)^{n_1} \leq \hat{n}_0 + d \cdot \hat{n}_1 + d \cdot \frac{\hat{n}_0 + 2d}{\phi} \cdot \left(\frac{2\phi}{\phi - d}\right)^{\hat{n}_1} \\ &= \hat{n}_0 + d \cdot \hat{n}_1 + (\hat{n}_0 + 2d) \cdot \frac{d}{\phi} \cdot \left(\frac{\phi}{d}\right)^{\log_2\left(\frac{2\phi}{\phi - d}\right)} = \hat{n}_0 + d \cdot \hat{n}_1 + (\hat{n}_0 + 2d) \cdot \left(\frac{\phi}{d}\right)^h \\ &= \hat{n}_0 \cdot \left(1 + \left(\frac{\phi}{d}\right)^h\right) + 2d \cdot \left(\frac{\phi}{d}\right)^h + d \cdot \hat{n}_1 = n\,. \end{aligned}$$

The second equality follows from $\log_2(2\phi/(\phi - d)) - 1 = \log_2(\phi/(\phi - d)) = h$, the last equality from the definition of $\hat{n}_0$. The bound on N implies that the new instance consists of at most n objects. Due to Lemma 7.8.4 and Lemma 7.8.5 the expected number of Pareto-optimal solutions of this instance is

$$\Omega\left(n_0^{f(d)} \cdot 2^{n_1 d}\right) = \Omega\left(\hat{n}_0^{f(d)} \cdot 2^{\hat{n}_1 d}\right) = \Omega\left(\left(\frac{n-4d}{4}\right)^{f(d)} \cdot \left(\frac{\phi}{d}\right)^d\right) = \Omega\left(n^{f(d)} \phi^d\right).$$

In the case $\phi/d > 2^{n/(4d)-1}$ we construct the same instance as above, but for the maximum density $\phi' = d \cdot 2^{n/4d-1} \in [2d, \phi)$. Consequently, $\hat{n}_1 = n/(4d) - 1$. As above, the expected size of the Pareto set is $\Omega\big(n^{f(d)} \cdot 2^{n/4-d}\big) = \Omega(2^{\Theta(n)})$. □

7.9 Some Probability Theory

The following theorem is variant of Theorem 3.3 for integer matrices whose entries are bounded by $\mathcal{K}$.

Theorem 7.9.1. *Let $m \le n$ be integers and let $X_1, \ldots, X_n$ be independent random variables, each with a probability density function $f_i \colon [-1,1] \to [0,\phi]$, let $A \in \{-\mathcal{K}, \ldots, \mathcal{K}\}^{m \times n}$ be a matrix of rank m, let $k \in [m-1]$ be an integer, let $(Y_1, \ldots, Y_{m-k}, Z_1, \ldots, Z_k)^{\mathrm{T}} = A \cdot (X_1, \ldots, X_n)^{\mathrm{T}}$ be the linear combinations of $X_1, \ldots, X_n$ given by A, and let C be a function mapping a tuple $(y_1, \ldots, y_{m-k}) \in \mathbb{R}^{m-k}$ to a hypercube $C(y_1, \ldots, y_{m-k}) \subseteq \mathbb{R}^k$ with side length ε. Then*

$$\mathbf{Pr}\left[(Z_1, \ldots, Z_k) \in C(Y_1, \ldots, Y_{m-k})\right] \le (2m\mathcal{K})^{m-k} \phi^m \varepsilon^k \,.$$

If all densities f_i are quasiconcave, then even the stronger bound

$$\mathbf{Pr}\left[(Z_1, \ldots, Z_k) \in C(Y_1, \ldots, Y_{m-k})\right] \le 2^k (m\mathcal{K})^{m-k} \phi^k \varepsilon^k$$

holds.

Proof. Let $\hat{A}$ be an arbitrary full-rank $m \times m$-submatrix of A. Applying Theorem 3.3, for general densities we obtain

$$\mathbf{Pr}\left[(Z_1, \ldots, Z_k) \in C(Y_1, \ldots, Y_{m-k})\right] \le \frac{(2m\alpha)^{m-k}}{|\det(\hat{A})|} \cdot \phi^m \varepsilon^k \,,$$

where α denotes the maximum absolute value of the entries of $\hat{A}$. As the entries of A are from $\{-\mathcal{K}, \ldots, \mathcal{K}\}$, we obtain $\alpha \le \mathcal{K}$ and $|\det(\hat{A})| \ge 1$. Consequently, the probability that $(Z_1, \ldots, Z_k)$ falls into the random hypercube $C(Y_1, \ldots, Y_{m-k})$ can be bounded by $\mathbf{Pr}\left[(Z_1, \ldots, Z_k) \in C(Y_1, \ldots, Y_{m-k})\right] \le (2m\mathcal{K})^{m-k} \phi^m \varepsilon^k$. For *quasiconcave* densities, Theorem 3.3 yields

$$\mathbf{Pr}\left[(Z_1, \ldots, Z_k) \in C(Y_1, \ldots, Y_{m-k})\right] \le 2^k \cdot \sum_I \frac{|\det(\hat{A}_I)|}{|\det(\hat{A})|} \cdot \phi^k \varepsilon^k \,,$$

where the sum runs over all tuples $I = (i_1, \ldots, i_k)$ for which $1 \le i_1 < \ldots < i_k \le m$. For such a tuple I, matrix $\hat{A}_I$ is the $(m-k) \times (m-k)$-submatrix of $\hat{A}$ that is obtained from $\hat{A}$ by removing the last k rows and the columns with the numbers $i \in I$. Each matrix $\hat{A}_I$ is from $\{-\mathcal{K}, \ldots, \mathcal{K}\}^{(m-k)\times(m-k)}$. Hence, $|\det(\hat{A}_I)| \le (m-k)! \cdot \mathcal{K}^{m-k}$. Furthermore, there are $\binom{m}{k}$ tuples $(i_1, \ldots, i_k)$ for which $1 \le i_1 < \ldots < i_k \le m$. This implies

$$\begin{aligned} \mathbf{Pr}\left[(Z_1, \ldots, Z_k) \in C(Y_1, \ldots, Y_{m-k})\right] &\le 2^k \cdot \binom{m}{k} \cdot (m-k)! \cdot \mathcal{K}^{m-k} \cdot \phi^k \varepsilon^k \\ &\le 2^k \cdot m^{m-k} \cdot \mathcal{K}^{m-k} \cdot \phi^k \varepsilon^k \\ &= 2^k (m\mathcal{K})^{m-k} \phi^k \varepsilon^k . \end{aligned}$$

□

7.10 Some Analysis

In this section we prove a technical claim that is required for constructing the lower bound example in Section 7.8. In the proof we apply the following well-known theorem.

Theorem 7.10.1 (*Rolle's theorem*). *Let $a < b$ be reals and let $f\colon [a,b] \to \mathbb{R}$ be a continuous function that is differentiable on the interval (a,b). If $f(a) = f(b)$, then there exists a point $\xi \in (a,b)$ for which $f'(\xi) = 0$.*

Lemma 7.10.2. *For positive reals $\phi > d$ and $h = \log_2(\phi/(\phi - d))$ the chain of inequalities $1 \le (\phi/d)^h \le 2$ holds.*

Proof. With $x = \phi/d - 1 > 0$ we can restate these inequalities as $1 \le (1+x)^{\log_2(1+1/x)} \le 2$, which is equivalent to $0 \le \hat{f}(x) \le 1$ for $\hat{f}(x) = \log_2(1+x) \cdot \log_2(1+1/x)$. The first inequality is obviously true. In the remainder of this proof we show that $\hat{f}(x)$ is maximal for $x = 1$, which is equivalent to showing that $f(x) = \ln(1+x) \cdot \ln(1+1/x)$ is maximal for $x = 1$. This implies $\hat{f}(x) \le \hat{f}(1) = 1$ for all $x > 0$. Due to $f(x) = f(1/x)$ it suffices to focus on values $x \ge 1$. As f is continuously differentiable, we only have to show that $x = 1$ is the unique positive real x for which $f'(x) = 0$. If this is true, then f and $\hat{f}$ are monotonic on $(0,1]$ and on $[1,\infty)$. Due to $\hat{f}(3) = \hat{f}(1/3) = 2 \cdot \log_2(4/3) = \log_2(16/9) < 1 = \hat{f}(1)$ it follows that $\hat{f}$ takes its maximum at $x = 1$. The derivative of f is

$$\begin{aligned} f'(x) &= \frac{1}{1+x} \cdot \ln(1+1/x) + \ln(1+x) \cdot \frac{1}{1+1/x} \cdot \left(-\frac{1}{x^2}\right) \\ &= \frac{x \cdot \ln(1+1/x) - \ln(1+x)}{x \cdot (1+x)} . \end{aligned}$$

Scaling with $x \cdot (1+x) \cdot (1+1/x)$ and with $x \cdot (1+x)$, respectively, yields that $f'(x) = 0$ is equivalent to

$$(1+x) \cdot \ln(1+1/x) = (1+1/x) \cdot \ln(1+x) \tag{7.4}$$

and to

$$x \cdot \ln(1+1/x) = \ln(1+x) \tag{7.5}$$

These equalities hold for $x = 1$. Now assume that there is another solution $x > 1$. We can state Equation (7.4) as $b \ln a = a \ln b$ for $1 < a := 1 + 1/x < 1 + x =: b$. A chain of equivalent transformations yields

$$b \ln a = a \ln b \iff \ln b + \ln \ln a = \ln a + \ln \ln b \iff g(a) = g(b)$$

for $g(y) = \ln y - \ln \ln y$ for any real $y > 1$. As g is differentiable on $[a, b]$, Theorem 7.10.1 implies that there is a real $\xi \in (a, b)$ for which $g'(\xi) = 0$. Since

$$g'(y) = \frac{1}{y} - \frac{1}{\ln y} \cdot \frac{1}{y} = \frac{1}{y} \cdot \left(1 - \frac{1}{\ln y}\right) ,$$

we obtain $\xi = e$, and thus, $1 + x = b > \xi = e$. This is a contradiction to Equation (7.5) as

$$x \cdot \ln(1 + 1/x) \leq x \cdot \ln(\exp(1/x)) = 1 < \ln(1 + x) .$$

Hence, $x = 1$ is the only real for which $f'(x) = 0$. $\square$

Chapter 8

Conclusions and Open Problems

We have studied the successive shortest path algorithm for the minimum-cost flow problem and the jump algorithm, the lex-jump algorithm, and the list scheduling algorithm for scheduling with makespan minimization. These algorithms are well-understood in theory, but there is a huge gap between the worst-case bounds and practical observations. We have narrowed this gap by applying the framework of smoothed analysis, which benefits from the advantages of both worst-case analysis and average-case analysis. With a smoothed analysis we have also been able to show that integer optimization problems with multiple linear objectives do usually not admit large Pareto sets.

The insights we gained from the analysis of the successive shortest path algorithm suggested how to modify the shadow vertex method, a variant of the simplex method, in order to find a short path between two given vertices of a polyhedron. This observation gives hope that, in future, smoothed analyses will not only deepen the understanding of the behavior of algorithms on real-life instances, but also reveal typical properties of these instances which can be exploited algorithmically. This would be an important step from the analysis of algorithms towards the design of efficient algorithms bringing theory and practice closer together.

In the following sections we discuss open questions concerning the algorithms and problems discussed in this thesis.

8.1 The Successive Shortest Path Algorithm

We showed that the successive shortest path (SSP) algorithm has a polynomial running time in our smoothed input model. Initially, we motivated the study of the SSP algorithm with the comparison to the minimum-mean cycle canceling (MMCC) algorithm, which is faster in theory, but slower in practice. It would be nice to have a smoothed lower bound for the running time of the MMCC algorithm that exceeds the smoothed upper bound for the running time of the SSP algorithm. This could be seen as an explanation why the

SSP algorithm outperforms the MMCC algorithm on practical instances. Rösner [Rös14] has already excluded the possibility that we can derive a significantly better smoothed upper bound for the SSP algorithm if $\phi = \Omega(n)$. In general, the best we can hope for is an improvement by a factor of $\max\{1, n/\phi\}$.

The SSP algorithm is not the fastest algorithm for solving the minimum-cost flow problem. Experimental studies (see [KK12]) show that the cost scaling algorithm due to Goldberg and Tarjan [GT90] and the network simplex algorithm due to Kelly and O'Neill [KO91] perform significantly better. It is an intriguing open problem to support this observation theoretically by a smoothed analysis.

As a by-product we obtained that the shadow vertex method has expected polynomial running time on any maximum-flow linear program if it draws a vector $c \in [0,1]^{|E|}$ uniformly at random as the second vector for the plane the maximum-flow polytope is projected onto. In general, we believe that it is an interesting question to study whether the strong assumption in Spielman and Teng's [ST04] and Vershynin's [Ver09] smoothed analysis that all coefficients in the constraints are perturbed is necessary. In particular, we find it an interesting open question to characterize for which class of linear programs it suffices to perturb only the coefficients in the objective function or just the projection in the shadow vertex method to obtain polynomial smoothed running time.

8.2 Finding Short Paths on Polyhedra

We considered the problem of finding a short path between two given vertices of a polyhedron and bounded the expected length of the path computed by the shadow vertex method from above. This bound depends on the dimension of the polyhedron, the number of constraints used for its definition, and a parameter δ describing the flatness of the polyhedron's vertices. Whereas in general this parameter can be arbitrarily small, it is polynomially bounded if the coefficients of the linear constraints form a totally unimodular matrix.

The shadow vertex method we studied for finding short paths on polyhedra is randomized. It would be interesting to know whether this variant can be derandomized. Another intriguing question is whether we can dispose of the dependence on δ for general polyhedra. This would resolve the polynomial Hirsch conjecture. Unfortunately, without modifications to the algorithm this is not possible: If the given vertices of the polyhedron are very flat, then there is nearly no randomness left for drawing the vectors that define the plane on which the polyhedron is projected. This would imply that the shadow vertex method for solving linear programs has polynomial running time. However, Murty [Mur80] and Goldfarb [Gol83, Gol94] showed that the shadow vertex method requires an exponential number of iterations on certain instances.

Furthermore, it is open whether our results carry over to the simplex method with shadow vertex pivot rule for solving linear programs. If we only consider Phase II where we are given a vertex of the polyhedron defined by the linear program, then we are in a very similar situation. The only difference is that we are given one vector of the plane of projection instead of another vertex of the polyhedron. This reduces the amount of randomness we can exploit. To avoid this, we could add random noise to the linear objective, either implicitly by applying smoothed analysis or explicitly by perturbing the objective's coefficients. In the latter case we have to ensure that the perturbations are sufficiently small such that optimal solutions remain optimal. There are instances for which this cannot be guaranteed, but there might be a chance to deal with these instances. It seems to be much harder to handle Phase I of the simplex method. Here, we consider a different polyhedron and it is not clear whether the flatness of its vertices can also be bounded depending on the parameter δ of the original polyhedron.

8.3 Scheduling Heuristics

We studied the performance guarantees of the jump algorithm and the lex-jump algorithm for different scheduling variants. For all variants with restricted machines the lower bounds turned out to be rather robust against random noise, not only in expectation, but even with high probability. We have also shown that the situation looks much better for unrestricted machines where we obtained performance guarantees of $\Theta(\phi)$ and $\Theta(\log \phi)$ for the jump and lex-jump algorithm, respectively. The latter bound also holds for the list scheduling algorithm and for the price of anarchy of routing on parallel links.

There are several interesting directions of research and we view our results only as a first step towards fully understanding local search and greedy algorithms in the framework of smoothed analysis. For example, we have only perturbed the processing requirements, and it might be the case that the worst-case bounds for the restricted scheduling variants break down if also the sets $\mathcal{M}_j$ are to some degree random.

Another interesting question is the following: We have always looked at the worst local optimum since we do not know which local optimum is reached. It might, however, be the case that the local optima reached in practice are better than the worst local optimum. It would be interesting to study the quality of the local optimum reached under some reasonable assumptions on how exactly the local search algorithms work.

8.4 Counting Pareto-Optimal Solutions

We studied the number of Pareto optima of optimization problems with several linear objectives over a set of integer vectors in the framework of smoothed analysis. For quasi-concave densities our bound is better than the bound of Moitra and O'Donnell [MO11]. Additionally, we could show that the exponents of the dimension n of the integer vectors and ϕ are in the right order. For higher moments our bound is significantly better than the bound of Röglin and Teng [RT09].

Maybe even more interesting are our results for the model of zero-preserving perturbations suggested by Spielman and Teng [ST04] and Beier and Vöcking [BV06]. For this model we proved the first non-trivial bound on the smoothed number of Pareto-optimal solutions. We showed that this bound can be used to analyze multiobjective optimization problems with polynomial and even more general objective functions. Furthermore, our result implies that the smoothed running time of the algorithm proposed by Berger et al. [BRvdZ11] to compute a path trade in a routing network is polynomially bounded for any constant number of autonomous systems. We believe that there are many more such applications of our result in the area of multiobjective optimization.

There are several interesting open questions. First of all it would be interesting to find asymptotically tight bounds for the smoothed number of Pareto-optimal solutions. There is still a gap between our upper bound of $O(n^{2d}\phi^d)$ for quasiconcave ϕ-smooth instances and our lower bound of $\Omega(n^{d-1.5}\phi^d)$. Only for the case $d = 1$ we have a matching lower bound.

Especially for zero-preserving perturbations there is still a lot of work to do. We conjecture that our techniques can be extended to also bound higher moments of the smoothed number of Pareto optima for ϕ-smooth instances with zero-preserving perturbations. However, we feel that even our bound for the first moment is too pessimistic as we do not have a lower bound showing that setting coefficients to zero can lead to larger Pareto sets. It would be very interesting to either prove a lower bound that shows that zero-preserving perturbations can lead to larger Pareto sets than non-zero-preserving perturbations or to prove a better upper bound for zero-preserving perturbations.

Bibliography

[AAF$^+$97] James Aspnes, Yossi Azar, Amos Fiat, Serge A. Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3):486–504, 1997.

[AART06] Baruch Awerbuch, Yossi Azar, Yossi Richter, and Dekel Tsur. Tradeoffs in worst-case equilibria. *Theoretical Computer Science*, 361:200–209, 2006.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows - theory, algorithms and applications.* Prentice Hall, 1993.

[Ang06] Eric Angel. A survey of approximation results for local search algorithms. In *Efficient Approximation and Online Algorithms*, volume 3484 of *LNCS*, pages 30–73. Springer, 2006.

[BCMR13] Tobias Brunsch, Kamiel Cornelissen, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the successive shortest path algorithm. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1180–1189, 2013.

[BDE$^+$12] 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.

[Bei04] René Beier. *Probabilistic Analysis of Discrete Optimization Problems.* PhD thesis, Universität des Saarlandes, 2004.

[BG60] Robert G. Busacker and Paul J. Gowen. A procedure for determining a family of miminum-cost network flow patterns. Technical Report Technical Paper 15, Operations Research Office, 1960.

[BLMS$^+$06] Luca Becchetti, Stefano Leonardi, Alberto Marchetti-Spaccamela, Guido Schäfer, and Tjark Vredeveld. Average case and smoothed competitive analysis for the multi-level feedback algorithm. *Mathematics of Operations Research*, 31(3):85–108, 2006.

[Bor86] Karl Heinz Borgwardt. *A probabilistic analysis of the simplex method.* Springer, New York, NY, USA, 1986.

[BR11] Tobias Brunsch and Heiko Röglin. Lower bounds for the smoothed number of Pareto optimal solutions. In *Proceedings of the 8th Annual Conference on Theory and Applications of Models of Computation (TAMC)*, pages 416–427, 2011.

[BR12] Tobias Brunsch and Heiko Röglin. Improved smoothed analysis of multiobjective optimization. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 407–426, 2012.

[BR13] 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), Part I*, pages 279–290, 2013.

[BRRV11] Tobias Brunsch, Heiko Röglin, Cyriel Rutten, and Tjark Vredeveld. Smoothed performance guarantees for local search. In *Proceedings of the 19th Annual European Symposium on Algorithms (ESA)*, pages 772–783, 2011.

[BRV07] René Beier, Heiko Röglin, and Berthold Vöcking. The smoothed number of Pareto optimal solutions in bicriteria integer optimization. In *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 53–67, 2007.

[BRvdZ11] André Berger, Heiko Röglin, and Ruben van der Zwaan. Path trading: Fast algorithms, smoothed analysis, and hardness results. In *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA)*, pages 43–53, 2011.

[BV04] René Beier and Berthold Vöcking. Random knapsack in expected polynomial time. *Journal of Computer and System Sciences*, 69(3):306–329, 2004.

[BV06] René Beier and Berthold Vöcking. Typical properties of winners and losers in discrete optimization. *SIAM Journal on Computing*, 35(4):855–881, 2006.

[BvdHS06] Graham Brightwell, Jan van den Heuvel, and Leen Stougie. A linear bound on the diameter of the transportation polytope. *Combinatorica*, 26(2):133–139, 2006.

[CM85] H. William Corley and I. Douglas Moon. Shortest paths in networks with vector weights. *Journal of Optimization Theory and Application*, 46(1):79–86, 1985.

[CS80] Yookun Cho and Sartaj Sahni. Bounds for list schedules on uniform processors. *SIAM Journal on Computing*, 9:91–103, 1980.

[CV07] Artur Czumaj and Berthold Vöcking. Tight bounds for worst-case equilibria. *ACM Transactions on Algorithms*, 3(1):article 4, 2007.

[Dan63] George B. Dantzig. *Linear programming and extensions.* Rand Corporation Research Study. Princeton University Press, 1963.

[DF94] Martin E. Dyer and Alan M. Frieze. Random walks, totally unimodular matrices, and a randomised dual simplex algorithm. *Mathematical Programming*, 64:1–16, 1994.

[DG97] Ali Dasdan and Rajesh K. Gupta. Faster maximum and minimum mean cycle algorithms for system-performance analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17:889–899, 1997.

[Ehr99] Matthias Ehrgott. Integer solutions of multicriteria network flow problems. *Investigacao Operacional*, 19:229–243, 1999.

[Ehr05] Matthias Ehrgott. *Multicriteria Optimization.* Springer, 2005.

[EK72] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

[ERV07] Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst case and probabilistic analysis of the 2-opt algorithm for the TSP. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1295–1304, 2007.

[Ets13] Michael Etscheid. Probabilistische Analyse der Qualität einfacher Scheduling-Algorithmen. Bachelor's thesis (in German), University of Bonn, 2013.

[FF62] Lester R. Ford, Jr. and Delbert R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[FH79] Greg Finn and Ellis Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT*, 19:312–320, 1979.

[Ful61] Delbert R. Fulkerson. An out-of-kilter algorithm for minimal cost flow problems. *Journal of the SIAM Society*, 9(1):18–27, 1961.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co., 1979.

[GK07] Celia A. Glass and Hans Kellerer. Parallel machine scheduling with job assignment restrictions. *Naval Research Logistics*, 54(3):250–257, 2007.

[GLLK79] Ronald L. Graham, Eugene L. Lawler, Jan K. Lenstra, and Alexander H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[Gol83] Donald Goldfarb. Worst-case complexity of the shadow vertex simplex algorithm. Technical report, Columbia University, New York, May 1983.

[Gol94] Donald Goldfarb. On the complexity of the simplex method. In *Advances in Optimization and Numerical Analysis*, volume 275 of *Mathematics and Its Applications*, pages 25–38. Springer, 1994.

[GR12] Navin Goyal and Luis Rademacher. Lower bounds for the average and smoothed number of Pareto optima. In *Proceedings of the 32nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 58–69, 2012.

[Gra66] Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[GT89] Andrew V. Goldberg and Robert E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM*, 36(4):873–886, 1989.

[GT90] Andrew V. Goldberg and Robert E. Tarjan. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

[Han80] Pierre Hansen. Bicriterion path problems. In *Multiple Criteria Decision Making: Theory and Applications*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, pages 109–127, 1980.

[Hoe63] Wassilij Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[HS88] Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for machine scheduling on uniform processors: using the dual approximation approach. *SIAM Journal on Computing*, 17:539–551, 1988.

[HS10] Martin Hoefer and Alexander Souza. Tradeoffs and average-case equilibria in selfish routing. *ACM Transactions on Computation Theory*, 2(1):article 2, 2010.

[Iri60] Masao Iri. A new method for solving transportation-network problems. *Journal of the Operations Research Society of Japan*, 3(1,2):27–87, 1960.

[Jac55] James R. Jackson. Scheduling a production line to minimize maximum tardiness. Technical report, University of California, Los Angeles, 1955.

[Jew62] William S. Jewell. Optimal flow through networks. *Operations Research*, 10(4):476–499, 1962.

[Joh54] Selmer M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68, 1954.

[KK92] Gil Kalai and Daniel J. Kleitman. A quasi-polynomial bound for the diameter of graphs of polyhedra. *Bulletin of the AMS*, 26(2):315–316, 1992.

[KK12] Zoltán Király and Péter Kovács. Efficient implementations of minimum-cost flow algorithms. *Acta Universitatis Sapientiae, Informatica*, 4(1):67–118, 2012.

[Kle67] Morton Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14(3):205–220, 1967.

[KO91] Damian J. Kelly and Garrett M. O'Neill. The minimum cost flow problem and the network simplex solution method. Master's thesis, University College Dublin, 1991.

[KV07] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 4th edition, 2007.

[KW67] Victor Klee and David W. Walkup. The d-step conjecture for polyhedra of dimension $d < 6$. *Acta Mathematica*, 117:53–78, 1967.

[KW00] Kathrin Klamroth and Margaret M. Wiecek. Dynamic programming approaches to the multiple criteria knapsack problem. *Naval Research Logistics*, 47(1):57–76, 2000.

[Li06] Chung-Lun Li. Scheduling unit-length jobs with machine eligibility restrictions. *European Journal of Operational Research*, 174:1325–1328, 2006.

[LL08] Joseph Y. T. Leung and Chung-Lun Li. Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, 116:251–262, 2008.

[MAK07] Wil P. A. J. Michiels, Emile H. L. Aarts, and Jan H. M. Korst. *Theoretical Aspects of Local Search*. Springer, 2007.

[MG98] Adli Mustafa and Mark Goh. Finding integer efficient solutions for bicriteria and tricriteria network flow problems using dinas. *Computers & Operations Research*, 25(2):139–157, 1998.

[MHW01] Matthias Müller-Hannemann and Karsten Weihe. Pareto shortest paths is often feasible in practice. In *Proceedings of the 5th International Workshop on Algorithm Engineering (WAE)*, pages 185–198, 2001.

[Min60] George J. Minty. Monotone networks. In *Proceedings of the Royal Society of London A*, pages 194–212, 1960.

[MO11] Ankur Moitra and Ryan O'Donnell. Pareto optimal solutions for smoothed analysts. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 225–234, 2011.

[MR11] Bodo Manthey and Heiko Röglin. Smoothed analysis: Analysis of algorithms beyond worst case. *it - Information Technology*, 53(6):280–286, 2011.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing - Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[Mur80] Katta G. Murty. Computational complexity of parametric linear programming. *Mathematical Programming*, 19(1):213–219, 1980.

[Nad89] Denis Naddef. The Hirsch conjecture is true for (0, 1)-polytopes. *Mathematical Programming*, 45:109–110, 1989.

[NU69] George L. Nemhauser and Zev Ullmann. Discrete dynamic programming and capital allocation. *Management Science*, 15(9):494–505, 1969.

[OLL08] Jinwen Ou, Joseph Y.-T. Leung, and Chung-Lun Li. Scheduling parallel machines with inclusive set restrictions. *Naval Research Logistics*, 55(4):328–338, 2008.

[Orl84] James B. Orlin. Genuinely polynomial simplex and non-simplex algorithms for the minimum cost flow problem. Technical report, Sloan School of Management, MIT, Cambridge, MA, 1984. Technical Report No. 1615-84.

[Orl93] James B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993.

[Orl97] James B. Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78(2):109–129, 1997.

[RG94] Tomasz Radzik and Andrew V. Goldberg. Tight bounds on the number of minimum-mean cycle cancellations and related results. *Algorithmica*, 11(3):226–242, 1994.

[Rös14] Clemens Rösner. Smoothed analysis of the SSP algorithm and local search. Master's thesis, University of Bonn, 2014.

[RRSV12] Cyriel Rutten, Diego Recalde, Petra Schuurman, and Tjark Vredeveld. Performance guarantees of jump neighborhoods on restricted related parallel machines. *Operations Research Letters*, 40:287–291, 2012.

[RT09] Heiko Röglin and Shang-Hua Teng. Smoothed analysis of multiobjective optimization. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 681–690, 2009.

[Rut13] Cyriel Rutten. *The Power of Simple Scheduling Policies.* PhD thesis, Maastricht University, 2013.

[SA00] Anders J. V. Skriver and Kim Allan Andersen. A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research*, 27(6):507–524, 2000.

[San10] Francisco Santos. A counterexample to the Hirsch conjecture. *CoRR*, abs/1006.2814, 2010.

[Smi56] Wayne E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1–2):59–66, 1956.

[SS05] Guido Schäfer and Naveen Sivadasan. Topology matters: Smoothed competitiveness of metrical task systems. *Theoretical Computer Science*, 341(1–3):3–14, 2005.

[SST06] Arvind Sankar, Daniel A. Spielman, and Shang-Hua Teng. Smoothed analysis of the condition numbers and growth factors of matrices. *SIAM Journal on Matrix Analysis Applications*, 28(2):446–476, 2006.

[ST04] 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.

[ST09] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, 52(10):76–84, 2009.

[SV07] Petra Schuurman and Tjark Vredeveld. Performance guarantees of local search for multiprocessor scheduling. *Informs Journal on Computing*, 19(1):52–63, 2007.

[Tar85] Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–256, 1985.

[Ver09] Roman Vershynin. Beyond Hirsch conjecture: Walks on random polytopes and smoothed complexity of the simplex method. *SIAM Journal on Computing*, 39(2):646–678, 2009.

[Vöc07] Berthold Vöcking. Selfish load balancing. In *Algorithmic Game Theory*, chapter 20. Cambridge University Press, New York, NY, USA, 2007.

[Vyg02] Jens Vygen. On dual minimum cost flow algorithms. *Mathematical Methods of Operations Research*, 56(1):101–126, 2002.

[Zad73] Norman Zadeh. A bad network problem for the simplex method and other minimum cost flow algorithms. *Mathematical Programming*, 5(1):255–266, 1973.

Index