Probabilistic Analysis of Discrete Optimization Problems

DISSERTATION

zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von
René Beier

Saarbrücken 2004

Dekan:	Prof. Dr. Jörg Eschmeier
Vorsitzender:	Prof. Dr. Raimund Seidel
Erstgutachter:	Prof. Dr. Berthold Vöcking
Zweitgutachter:	Prof. Dr. Kurt Mehlhorn
Akademischer Beisitzer:	Dr. Ernst Althaus

Tag des Kolloquiums: 10. September 2004

Abstract

We investigate the performance of exact algorithms for hard optimization problems under random inputs. In particular, we prove various structural properties that lead to two general average-case analyses applicable to a large class of optimization problems.

In the first part we study the size of the Pareto curve for binary optimization problems with two objective functions. Pareto optimal solutions can be seen as trade-offs between multiple objectives. While in the worst case, the cardinality of the Pareto curve is exponential in the number of variables, we prove polynomial upper bounds for the expected number of Pareto points when at least one objective function is linear and exhibits sufficient randomness. Our analysis covers general probability distributions with finite mean and, in its most general form, can even handle different probability distributions for the coefficients of the objective function. We apply this result to the constrained shortest path problem and to the knapsack problem. There are algorithms for both problems that can enumerate all Pareto optimal solutions very efficiently, so that our polynomial upper bound on the size of the Pareto curve implies that the expected running time of these algorithms is polynomial as well. For example, we obtain a bound of $O(n^4)$ for uniformly random knapsack instances, where n denotes the number of available items.

In the second part we investigate the performance of knapsack core algorithms, the predominant algorithmic concept in practice. The idea is to fix most variables to the values prescribed by the optimal fractional solution. The reduced problem has only polylogarithmic size on average and is solved using the Nemhauser/Ullmann algorithm. Applying the analysis of the first part, we can prove an upper bound of $O(n\,\mathrm{polylog}\,n)$ on the expected running time. Furthermore, we extend our analysis to a harder class of random input distributions. Finally, we present an experimental study of knapsack instances for various random input distributions. We investigate structural properties including the size of the Pareto curve and the integrality gap and compare the running time between different implementations of core algorithms.

The last part of the thesis introduces a semi-random input model for constrained binary optimization problems, which enables us to perform a smoothed analysis for a large class of optimization problems while at the same time taking care of the combinatorial structure of individual problems. Our analysis is centered around structural properties, called *winner*, *loser*, and *feasibility gap*. These gaps describe the sensitivity of the optimal solution to slight perturbations of the input and can be used to bound the necessary accuracy as well as the complexity for solving an instance. We exploit the gaps in form of an adaptive rounding scheme increasing the accuracy of calculation until the optimal solution is found. The strength of our techniques is illustrated by applications to various NP-hard optimization problems for which we obtain the first algorithms with polynomial average-case/smoothed complexity.

Kurzzusammenfassung

Wir untersuchen mit mathematischen Methoden die Effizienz von Algorithmen für schwierige Optimierungsprobleme auf zufälligen Eingaben. Im Zentrum steht die Analyse struktureller Eigenschaften zufälliger Probleminstanzen, die im Weiteren benutzt werden, um Laufzeitschranken für das Lösen verschiedener Probleme zu beweisen. Im ersten Teil der Arbeit betrachten wir binäre Optimierungsprobleme mit zwei Zielfunktionen. Wir können zeigen, dass die erwartete Anzahl der Pareto-optimalen Lösungen polynomiell beschränkt ist, wenn mindestens eine Zielfunktion linear ist und ihre Koeffizienten zufällig oder zufällig perturbiert sind. Wir wenden dieses Ergebnis auf zwei Optimierungsprobleme an: das Kürzeste-Wege-Problem mit Nebenbedingungen und das Rucksackproblem. Für beide Probleme können die Pareto-optimalen Lösungen effizient aufgezählt werden, so dass die obere Schranke für die Anzahl der Pareto-optimalen Lösungen eine polynomielle erwartete Laufzeit für diese Algorithmen impliziert. Im zweiten Teil der Arbeit untersuchen wir *Core*-Algorithmen für das Rucksackproblem, für die wir eine erwartete Laufzeit von $O(n \operatorname{polylog} n)$ für uniform zufällige Rucksackinstanzen beweisen können. Im letzten Teil der Arbeit wird ein semi-zufälliges Eingabemodell für binäre Optimierungsprobleme vorgestellt. Es ermöglicht eine probabilistische Analyse für eine große Klasse von Optimierungsproblemen und berücksichtigt gleichzeitig die zugrunde liegende kombinatorische Struktur der einzelnen Probleme. Wir definieren *Polynomial Smoothed-Complexity*, angelehnt an *Polynomial Average-Case Complexity*, eine Klassifizierung für die Komplexität von Problemen unter zufälligen Eingaben. Für die von uns betrachtete Klasse von Optimierungsproblemen können wir eine genaue Charakterisierung der Probleme geben, die in diese Komplexitätsklasse fallen. Ein binäres Optimierungsproblem hat *Polynomial Smoothed-Complexity* genau dann, wenn es einen Algorithmus für dieses Problem gibt, dessen Laufzeit im schlechtesten Fall pseudopolynomiell im stochastischen und polynomiell im restlichen Teil der Eingabe ist.

Acknowledgements

First and foremost, I am indebted to my *Doktorvater* Berthold Vöcking, who guided my research in the last two and a half years. I have learned a lot from his official and private lessons on chance and probability and I truly enjoyed our fruitful collaboration. Thank you also for sharing your thoughts about many non-scientific issues as well as for providing accommodation on an overpriced island resort.

I feel very fortunate for being a member of Kurt Mehlhorn's Algorithms and Complexity Group at the Max-Planck Institut für Informatik. I have always appreciated the excellent research conditions and the extraordinary freedom that I was granted during my studies. In particular, I would like to thank Kurt Mehlhorn for being an admirable leader and a fine teacher, Peter Sanders for many valuable suggestions and discussions, Naveen Sivadasan and Guido Schäfer, for sharing the office, books, coffee, time and the secretes of LaTeX, and the countless other people who have helped me over the past years.

I would like to thank the Deutsche Forschungsgesellschaft for their financial support granted within the Graduiertenkolleg Informatik at the Universität des Saarlandes. I would also like to use this opportunity to express my gratitude to my former math teacher Annette Hardegen, to whom I owe my four years stay at a specialized school for Mathematics, Natural Sciences, and Engineering in Erfurt.

Finally, I would like to thank Imke for sharing her life and her love with me and for being a wonderful partner, mother and friend.

Contents

Introduction

A major objective in designing new algorithms is efficiency. The most common theoretical approach to measure the performance of algorithms is the worst-case analysis, which takes into account only the worst possible performance that an algorithm exhibits on any problem instance. On the one hand, this measure provides a very strong guaranty desirable in many applications, as it makes a robust statement about all problem instances, which clearly cannot be obtained by any experimental study. Besides, estimating upper bounds for the running time, memory usage, number of I/O accesses etc. is usually an easy task. On the other hand, a single upper bound cannot truly reflect the overall performance of an algorithm. This becomes especially apparent when the performance of the algorithm differs significantly between individual instances of a problem and the worst-case instances are rare. A quite extreme example in that respect is the knapsack problem, which will be of more concern in this thesis. There exist algorithms for the knapsack problem with a probably exponential worst-case running time, as one would expect for an NP-complete problem. On random instances, however, these algorithms usually exhibit a linear running time, a performance that could not be achieved even for many problems known to be solvable in polynomial time in the worst-case. The reason for this apparent inconsistency is that almost all knapsack instances are easy to solve such that a randomly chosen instance is most likely an easy one. This example shows that the significance of the worst-case measure is very limited for a comprehensive evaluation of algorithms, especially if one is interested in the typical performance on real world problems.

A quite natural approach to remedy this deficit is to take the average performance over all instances as new measure, which is the underlying idea of average-case analysis. More precisely, given a probability distribution on the set of instances, the performance of an algorithm becomes a random variable which is subsequently investigated, for example, by bounding its expectation. Random instances are usually easier to solve as shown for various problems like sorting, finding shortest paths or matching. Although the average case analysis can reveal a lot about the performance of an algorithm or even the complexity of the problem at hand, it is hard to draw sensible conclusions for the performance in practice. The reason is that real world instances are not totally random but usually have some special structure and properties. Notice, for example, that file names are no random strings, road networks are no random graphs and the execution times of jobs are not uniformly distributed. One might argue that this is in principle no problem of the model but just a matter of choosing the right distribution on the set of instances. The underlying distributions, however, are usually not known, and even if so, and putting aside the technical difficulties, it would be a tedious work to prove bounds for every such distribution.

A relatively new measure is provided by *smoothed analysis*, which combines elements of worst case

and average-case analysis. As in worst-case analysis, one takes the maximum over all instances. But instead of directly measuring the performance of each individual instance, one uses the expected performance under a small random perturbation of the original instance. The performance is then measured in terms of the size of the instance and the magnitude of the perturbation. Let us explain the original idea in more detail. Spielman and Teng [ST01] investigated the Simplex algorithm, the standard technique for solving linear programs. The Simplex algorithm is a typical example of an algorithm that is known to perform well in practice but exhibits an exponential running time in the worst case. Recall, that a linear program is completely specified by the objective function, a constraint matrix, and the right hand side. Assume we are given an arbitrary (worst-case) linear program where all row vectors of the constraint matrix have norm at most 1. Perturbing the instance is done by adding a Gaussian random variable with standard deviation σ and mean 0 to each entry in the constraint matrix. The "smoothed" performance is the expected performance of the algorithm on the perturbed instance, where the expectation is over the random perturbation. The parameter σ specifies the magnitude of the perturbations. For $\sigma \to 0$, we do not perturb the original instance and consequently obtain the standard worst-case analysis. For very large σ, the random perturbation swamps out the input values of the original instance and one obtains average-case analysis. Using smoothed analysis we can describe the transition between these two extremes. Spielman and Teng proved that the Simplex algorithm with a certain pivot-rule has a "smoothed" running time bound that is polynomial in $1/\sigma$ and the size of the problem instance.

Smoothed analysis describes the average performance on instances in a parameterized neighborhood of the worst-case instances. For algorithms with equal performance on all problem instances with the same size, worst-case, average-case and smoothed analysis should yield the same result. Assume the worst-case and average-case measure differ significantly. If the worst-case instances are clustered together, then also the smoothed performance is bad. If, however, the worst-case instances are "isolated events" in instance space one would expect a rapid improvement of the running time bound with an increasing size of the neighborhood. In other words, if the smoothed measure is low, worst case instances are not robust under small changes.

The underlying idea of smoothed analysis, that is, smoothening the worst-case by considering the expected performance in the neighborhood of each instance, has been adapted to other problems like shortest path and sorting [BBM03] as well as to other measures like the condition number [DST03] and the competitive ratio for online problems [BLMS$^+$03, SS04]. Obviously, the definition of the neighborhood is crucial. It is supposed to capture the notion of similar instances. Ideally, we would expect the neighborhood of a real-world instance to contain instances that have also some fair chance to be encountered in practice. For real valued input variables, the most natural neighborhood is the neighborhood on the real line, which finds a reasonable justification, as data from the real world are usually inherently noisy. However, if the instance space is a finite set of combinatorial objects, a sensible definition of neighborhood, resp. perturbation scheme, is often less obvious.

Is is easy to argue that smoothed analysis cannot completely explain the performance of algorithms in practice either. It provides, however, a mathematically rigorous theory that is not restricted to individual problems. Smoothed analysis makes the assumption that inputs are subject to noise, circumstance or randomness. This is an assumption that is valid in many practical problem domains. While it is unrea-

sonable to assume that one can develop a model of inputs subject to noise that precisely models practice, one can try to come close and then reason by analogy.

Which problems allow fast algorithms under small perturbations? This question leads to a new field of complexity theory. Recall that the worst-case measure is the basis for the standard complexity theory, one of the most fundamental achievements in theoretical computer science. An integral part of this theory is the classification of algorithmic problems according to their complexity, most relevant the classes P and NP. Problems in class P, which allow a polynomial time algorithm, are considered efficiently solvable, whereas NP-hard problems are considered intractable, as it is common believe that they cannot be solved in subexponential time. All these statements rely on the worst-case measure. In Chapter 3 we will define a class of efficiently solvable problems under small perturbations. These are all problems which have polynomial smoothed complexity, an adaption of polynomial average-case complexity with the additional requirement that the running time should be polynomially bounded not only in the size of the problem instance but also in the magnitude of the perturbation. This complexity measure is slightly weaker than expected polynomial running time, but is a more robust measure, as it is independent of the used machine model. Notice the correspondence to ε-smoothed complexity, proposed in [ST01].

At this points let us remark that the smoothed analysis result on the Simplex algorithm by Spielman and Teng, although it is an outstanding result, hardly allows any conclusion with respect to the complexity of problems. The fact that a problem can be formulated as a linear program and that the Simplex algorithm solves perturbed linear programs in polynomial time says nothing about the smoothed complexity of the problem at hand. The reason is that the linear program formulation of almost every problem has a special structure. Since the analysis of Spielman and Teng assumes that every entry in the constraint matrix is perturbed independently, this structure will be destroyed. Hence, the perturbed instance is almost surely not an instance of the original problem any more. As an example, consider zero entries, which are most likely perturbed to non-zero values. In other words, the expected running time bounded in the analysis is over some larger probability space in which the set of instances of the original problem usually have probability measure zero.

Smoothed Analysis Framework

We will extend the smoothed analysis model introduced by Spielman and Teng. In particular, we will not necessarily perturb all input numbers but only those that are explicitly marked as stochastic. The domain of those marked input numbers is restricted to $[0,1]$ or $[-1,1]$, depending on whether the domain should be non-negative or also include negative numbers. Then a random perturbation slightly changes the marked input numbers by adding an independent random number to each of them. These random numbers are drawn according to a specified family of probability distributions satisfying the following conditions. Let $f : \mathbb{R} \to \mathbb{R}_{\geq 0}$ be any piecewise continuous density function such that $\sup_s(f(s)) = 1$ and $E = \int |s| f(s) ds$ is finite, that is, the random variable described by f has a finite expected absolute value. The function f is called the *perturbation model*. Observe that the smallest value for E is $1/4$ attained by the uniform distribution in the interval $[-1/2, 1/2]$. For $\phi \neq 1$, we define f_ϕ by scaling f,

that is, $f_\phi(s) = \phi f(s\phi)$, for every $s \in \mathbb{R}$. This way, the density parameter of f_ϕ is ϕ. The expected absolute value of the density function f_ϕ is E/ϕ. We obtain ϕ-*perturbations* according to the perturbation model f by adding an independent random variable with density function f_ϕ to each stochastic input number. For example, one obtains the Gaussian perturbation model used by Spielman and Teng [ST01] by choosing f to be the Gaussian density with standard deviation $(2\pi)^{-1/2}$. The perturbation model f is not restricted to density functions that are symmetric around 0. Such a requirement would not be very meaningful as the adversarial choice of the input numbers allows arbitrary shifts of the distribution in either case. Furthermore, if the domain of the input numbers is non-negative, then we can ensure that also the perturbed numbers are non-negative by choosing a perturbation model f with non-negative domain, e.g. the uniform distribution over $[0,1]$. In the seminal paper on smoothed analysis [ST01] the running time is described in terms of the standard deviation σ. In contrast, we describe the running time in terms of the density parameter ϕ. For the Gaussian and the uniform distribution these two parameters are closely related; in both cases, ϕ is proportional to $\frac{1}{\sigma}$.

Our Results

In this thesis we prove that various NP-hard binary optimization problems can be solved in polynomial time under random perturbations. This includes the (multidimensional) knapsack problem, the constrained shortest path problem, general 0/1 integer programming with a constant number of constraints, the general assignment problem with a constant number of constraints and the problem of scheduling to minimize the weighted number of tardy jobs. Our probabilistic analysis is not restricted to the model of smoothed analysis, but we use this framework to illustrate our results. The heart of our analyses are certain structural properties, which appear in many problems, so that our analysis is applicable to a quite general class of problems.

In Chapter 1 we investigate the size of the Pareto curve for binary optimization problems with two objective functions. For most problems the number of Pareto points can be exponential in the number of variables. We show that the expected number of Pareto points is polynomially bounded if at least one objective function is linear and its coefficients exhibits sufficient randomness. The other function is assumed to be adversarial. Our analysis covers general probability distributions for the coefficients of the linear objective function. Each coefficient can follow its individual probability distribution. Hence, our analysis covers the framework of smoothed analysis introduced above. Our bound is parameterized by ϕ, which is the maximum over the density functions of the different probability distributions, and μ, the maximum expected value of these distributions. We prove that the expected size of the Pareto curve is $O(n^4\phi\mu)$. Furthermore, better bounds for specific distributions are presented as well as a lower bound of $\Omega(n^2)$. Besides the fact that this is an interesting result in its own, we apply this bound to algorithms for the knapsack problem and the constrained shortest path problem. These algorithms enumerate all Pareto optimal solutions very efficiently. In particular, the influence of the number of enumerated solutions on the running time is only linear. Hence, the generality of our bounds for the size of the Pareto curve transfers directly to the running time bound of these algorithms.

The result for the knapsack problem is subsequently used in Chapter 2 to analyze knapsack core

algorithms, the predominant algorithmic concept in practice. We present an average-case analysis proving an upper bound of $O(n \operatorname{polylog} n)$ for the expected running time of a core algorithm on uniformly random knapsack instances, i.e., instances where the weights and profits are chosen independently uniformly at random from $[0,1]$. We exploit the fact that for these instances the expected integrality gap is only $O((\log n)^2/n)$, such that the core problem contains only $O(\log^2 n)$ items on average. We extend this result to δ-correlated instances, a harder class of input distributions for the knapsack problem, where we obtain a similar bound.

An experimental study complements the knapsack chapter. We compare all our theoretical finding to the results obtained in various experiments. We investigate structural properties that play an important role for our analyses, like integrality gap and the average number of the Pareto optimal knapsack fillings. Furthermore we compare the performance of different core algorithms on various random input distributions.

In Chapter 3 we present a smoothed/average-case analysis for a large class of binary optimization problems. The core of our analysis are three structural properties called *winner*, *loser* and *feasibility gap*. The *winner gap* describes the difference in the objective value between the best and the second best solution. We generalize the well-known Isolating Lemma and prove that for linear objective functions with random or randomly perturbed coefficients, the winner gap is usually lower bounded by a polynomial in $1/(n\phi)$, where n and ϕ denote the number of variables and the density parameter of the probability distributions. The *loser* and *feasibility gap* apply to linear constraints with random or randomly perturbed coefficients. Consider a single constraint $w^T x \leq t$. The feasibility gap is the slack of the optimal solution with respect to that constraint. The *losers* are those solutions that have a better objective function value than the optimal solution, but they are infeasible because of the considered constraint. The *loser gap* is defined to be the minimal amount by which a loser (except for the solution 0^n) exceeds the constraint threshold. We extend the *loser* and *feasibility gap* to multiple constraints and prove in the *Separating Lemma* that these gaps are usually lower bounded by a polynomial in $1/(n\phi k)$, where k denotes the number of constraints, n and ϕ are defined as above. As a consequence of the large gaps, we can round the coefficients in the random linear function and the random linear constraints to a logarithmic number of bits without changing the optimal solution. These gap properties are applied in a smoothed analysis for binary optimization problems. For this purpose, a semi-random input model is introduced that allows to perturb only parts of the input while other parts are assumed to be adversarial. We allow to perturb linear objective functions as well as linear constraints that define the set of feasible solutions. This semi-random input model allows to take care for the combinatorial structure of individual problems such that the perturbed input is still an instance of the considered problem. We extend the range of possible applications by allowing zero perturbations, that is, we allow to specify individual coefficients of the perturbed objective function or constraint to be fixed to zero.

Well-behaved Random Variables

In several analyses in this thesis we will introduce auxiliary random variables and prove various "upper bounds" on their density function. This term needs some clarification, as the density of a continuous

variable is not uniquely defined. A continuous random variable X is defined by its *distribution* $F_X(t) = \mathbf{Pr}[X \leq t]$. In general, the *density* f_X is any non-negative function satisfying $F_X(t) = \int_{-\infty}^{t} f_X(s)ds$. Observe that the integrand is not uniquely determined. It might be redefined on any set of points of measure 0 without affecting the integral. We say that a continuous random variable X is *well-behaved* if its distribution function F_X is piecewise differentiable. In this case, X admits a piecewise continuous density function f_X, which at all of its continuous points corresponds to the derivative of F_X. As usual, we ignore the trifling indeterminacy in the definition of f_X and refer to f_X as *the* density of X. In particular, the *supremum of the density of* X refers solely to the supremum over the points at which f_X is continuous, and we say that the density is *bounded* if there exists $b \in \mathbb{R}$, such that $f_X(s) \leq b$, for every point $s \in \mathbb{R}$ at which f_X is continuous. Throughout this thesis, $[n]$ denotes $\{1, \ldots, n\}$.

Bibliographic Notes

Many parts of this thesis have been published before. The proof of the main theorem in Chapter 1 appeared as a specific version for the knapsack problem in [BV03, BV04c] together with the application to the Nemhauser/Ullmann algorithm. In this thesis we generalized the theorem to arbitrary bicriteria optimization problems with binary variables and added the application to the constrained shortest path problem. The analysis of knapsack core algorithms in Chapter 2 has been published in [BV04b]. The experimental study in the same chapter appeared as a short version in [BV04a]. Finally, the results of Chapter 3 have been published in [BV04d].

Chapter 1

The Pareto Curve for Bicriteria Problems

When making decisions in the real world, people usually have to take many objectives into account. Contributing to the difficulty of live, the different objectives are most likely conflicting and the final decision is commonly called a trade-off. In theoretical computer science, optimization problems are usually modeled with only one objective, which allows to compare the quality of solutions easily in a mathematical exact way. This leads to a natural notion for the optimality of solutions. When dealing with multiple objectives, solutions can be incomparable since they can dominate each other in different objectives. The notion of Pareto optimality is based on a partial order among the solutions. A solution is called Pareto optimal, if it is not dominated by any other solution, that is, if there is no other solution that is better in at least one objective and not worse in any of the other objectives. Naturally, Pareto optimal solutions are the candidates for a trade-off. For many common optimization problems the number of Pareto optimal solutions can be exponential in the number of variables. We study this issue under a probabilistic setting. We are able to prove that for binary optimization problems with two objective functions the expected number of Pareto optimal solutions is only polynomial in the number of variables, provided that one objective function is linear and its coefficients exhibit sufficient randomness.

1.1 Multiobjective Combinatorial Optimization

Optimization problems can be characterized in terms of instances, feasible solutions and objective functions. An optimization problem (e.g. finding the shortest path in a graph) has a (usually infinite) set of instances. Each instance can be described by a set of feasible solutions (in our example all s-t paths) and an objective function (in our example the sum of the edge length). In general we call an optimization problem combinatorial if solutions are described by variables that take only integer values.

We are interested in the class of problems with binary variables and linear objective function, that is, the variables can take only value 0 or 1, and the objective is a linear function $c^T x$, $c \in \mathbb{R}^n$, over the vector x of binary variables. We call these problems *binary optimization problems*. Hence, an instance of a *binary optimization problem* of size n is completely described by

1. a solution set $\mathcal{S} \subseteq \{0,1\}^n$, i.e., a set of 0/1 vectors, and

2. an objective function $f : \mathcal{S} \Rightarrow \mathbb{R}$, usually a linear function $c^T x$.

At first, the restriction to binary variables might look as a strong limitation, but many well-known problems fall into this category as the following list of examples demonstrates.

- Graph problems with a binary variable for each node: Shortest Path problem, Maximum Matching, Minimum Spanning Tree, Minimum Cut, Traveling Salesperson problem, etc.
- Graph problems with a binary variable for each node: The (weighted) Independent Set Problem, the (weighted) Clique Problem, the (weighted) Vertex Cover Problem, the (Weighted) Dominating Set Problem, etc.
- The Facility location problem: A binary variable is introduced for each facility and for each possible facility-costumer pair.
- the Set Covering Problem, the General Assignment Problem, various Network Flow and Packing problems and many more.

We will frequently use the following different view on this class of problems. Consider a problem instance with n variables. Each solution is described by an assignment of the variables, i.e., a 0/1 vector of length n, or, equivalently, by the subset of the variables that are set to 1. This way, the set of feasible solutions correspond to a collection of subsets over an n-element ground set G. Each element in $i \in G$ is assigned a value $f(i)$, which, in case this element becomes part of the solution, contributes to the objective value. Hence, the objective value for a subset $S \subseteq G$ is given by $\sum_{i \in S} f(i)$. As an example consider the problem of computing the shortest s,t path in a graph $G = (V,E)$. The ground set is the set of edges E and feasible solutions are those subsets of edges that form elementary s,t paths. Given a length function $l : E \to \mathbb{R}_{>0}$, the objective value of a path $P \subseteq E$ is $\sum_{e \in P} l(e)$.

For some applications it is natural to define more than one objective function. As an example consider the problem of vehicle routing modeled as a shortest path problem in a graph. Besides the length of a route one might also be interested in the time it takes a vehicle to drive along this route. Other interesting criteria might be fuel consumption, probability of traffic jam, etc. Formally, we have $k > 1$ objective functions $f_j : G \to \mathbb{R}$, for $1 \leq j \leq k$ and the j-th objective value of a solution $S \subseteq G$ is given by $\sum_{i \in S} f_j(i)$. There are several ways to deal with multiple objectives. A common approach, especially in theoretical computer science, is to define bounds on all but one objective and to optimize the remaining single objective. This way, solutions not satisfying the given bounds are rendered infeasible. A classical example is the constrained shortest path problem [Zie01]. By imposing strict bounds on even a single objective, the problems usually become NP-hard, although the single criterion version of the problem is in P. The NP-hardness proof usually uses a reduction from the knapsack problem. A drawback of this method is the arbitrariness in fixing the bounds on the different objectives. Even the smallest change in the bounds can lead to a significant different objective value or even prohibit any feasible solution at all. This way, one might miss interesting solutions by placing an unfortunate bound. Therefore, another idea attracted more and more interest, the notion of Pareto optimality which is based on a domination concept

usually attributed to Weingartner and Ness [WN67]. Informally, a solution dominates another solution if it is better or equally good in all objectives. Clearly, dominated solutions are inferior and, therefore, suboptimal in any sensible definition of optimality for multiple criteria. A solution that is not dominated by any other solution is called *Pareto optimal* or *undominated*, that is, there is no other solution that is better in all objectives. The concept of Pareto optimality captures the intuitive notion of a trade-off. Let us define the domination concept more formally for binary optimization problems.

Definition 1.1. *Given an instance of a binary optimization problem, let $\mathcal{S} \subseteq 2^G$ denote the set of all feasible solutions. Let $f_1, \ldots, f_k$ denote linear objective functions. For any two solutions $S, T \in \mathcal{S}$ we say S dominates T if*

1. *$f_i(S) \leq f_i(T)$ for all objectives f_i to be minimized, and*
2. *$f_i(S) \geq f_i(T)$ for all objectives f_i to be maximized, and*
3. *there exists some f_i with $f_i(S) \neq f_i(T)$.*

We call a solution $S \in \mathcal{S}$ **Pareto optimal** *or* **undominated** *over $\mathcal{S}$, if there is no other solution $T \in \mathcal{S}$, that dominates S. The set of k-vectors $\{(f_1(S), \ldots, f_k(S)) \mid S \in \mathcal{F}$ is Pareto optimal$\}$ is called* **Pareto points** *or* **Pareto curve** *of $\mathcal{S}$.*

There is a general agreement in the multiobjective optimization community that the Pareto curve is the right solution concept when dealing with multiple criteria (see [Ehr00, EG02]). We distinguish between the number of Pareto optimal solutions and the number of Pareto points, since the two quantities can differ significantly when multiple solutions are mapped to a single Pareto point. We call two solutions equivalent, if they map to the same Pareto point, i.e., if they agree in all objectives. Even if there might be multiple equivalent optimal solutions, it usually suffices to compute any optimal solution. (Computing all optimal solutions is even for single criterion problems much more complex.) Therefore, we concentrate on the complexity of the Pareto curve rather than the set of Pareto optimal solutions. In many publications on multiobjective optimization, however, this distinction is sometimes blurred for the sake of a short presentation. Notice, that for random coefficients drawn according to a continuous probability distribution, with probability 1, no two solutions have the same objective value. This is different for discrete probability distributions, where the finite domain (or expectation) is responsible for the pseudo-polynomial upper bound on the number of Pareto points.

Before we turn to the size of the Pareto curve, let us mention another common approach to deal with multiple objectives. The idea is to combine all objectives into a single objective function using a linear combination. Each objective f_i is assigned a weight λ_i, where λ_i is negative for objectives f_i to be maximized, and positive for objectives to be minimized. This way, minimizing $\{\sum_{i=1}^{k} \lambda_i f_i(S) \mid S \in \mathcal{F}\}$ takes all objectives into account. Observe that every solution that is optimal w.r.t. to a linear combination of the objectives is also Pareto optimal, provided all weights λ_i are non-zero. The converse statement is not true, that is, there exist Pareto optimal solutions that are not optimal w.r.t. any linear combination of objectives. In fact, this gives a criterion to categorize Pareto optimal solutions. We call a Pareto optimal solution *supported* if there exist weights $\lambda_1, \ldots, \lambda_k$ such that S is optimal w.r.t. a linear combination with

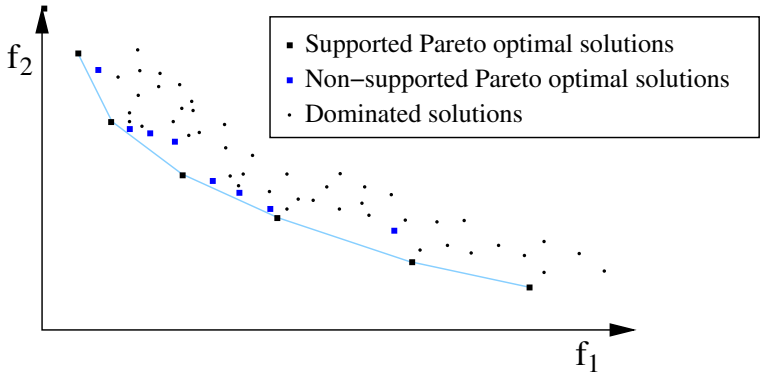

Figure 1.1: Example for a set of solutions and two objective functions f_1 and f_2, both to be minimized. Points, dark and light squares represent dominated solutions, supported and unsupported Pareto optimal solutions, respectively.

coefficients $\lambda_1,\ldots,\lambda_k$. Otherwise, solutions are called *non-supported*. This definition is illustrated in Figure 1.1. In the k-space of objectives, the supported Pareto optimal solutions lie on the boundary of the convex hull $\mathcal{CH}\{\big(f_1(S),\ldots,f_k(S)\big) \mid S \in \mathcal{F}\}$, whereas non-supported Pareto optimal solutions are strictly inside the convex hull. The reason is that fixing coefficients $\lambda_1,\ldots,\lambda_k$ corresponds to fixing an optimization direction in the k-space of objectives. Hence, optimal solutions are extreme points of the convex hull.

We elaborate on this topic since the distinction between supported and non-supported solutions leads to interesting insights and has important consequences in complexity theoretic terms. For a comprehensive survey on available literature on this topic we refer to the book by Ehrgott and Gandibleux [EG02]. Computing supported Pareto points is usually much easier than computing non-supported ones. The definition of supported and non-supported Pareto points yields already a generic algorithm for computing the corner points of the convex hull, e.g., all supported Pareto points for which there is a linear combination of objectives such that the optimal Pareto point is unique. As a subroutine we only require an algorithm that solves the single objective variant of the problem, which allows us to find extreme points on the convex hull. Also the number of supported Pareto points is usually small compared with the number of non-supported Pareto points. As an example, consider the bicriteria version of the knapsack problem. The number of supported Pareto points is $n+1$, provided that the profit-to-weight ratios of all items are distinct. In contrast, the number of non-supported Pareto points is considerably larger (see Section 2.5.2).

1.1.1 Cardinality and Approximations of the Pareto Curve

In the worst case, the number of Pareto optimal solution can be exponential, as shown for various optimization problems like spanning tree, shortest path, assignment and the traveling salesperson problem (see [Han80], [SP91], [EP92], [HR94]). As an example, consider the bicriteria minimal spanning tree problem [GR96]. Each edge $e \in E$ of a graph $G = (V,E)$ is assigned a weight w_e and a cost c_e. By choosing $w_e \in [0,1]$ and $c_e = 1 - l_e$, each spanning tree T has weight $w(T) = \sum_{e\in T} w_e$ and cost

$c(T) = (|V| - 1) - w(T)$, as all spanning trees have $|V| - 1$ edges. Hence, all spanning trees are Pareto optimal and, furthermore, they are even supported Pareto optimal.

Hence, computing the Pareto curve is intractable for these problems under worst case inputs. When a problem is computationally too complex to solve, the standard approach is to give up the requirement of computing optimal solutions and instead aim at close-to-optimal solutions. Following this idea, Papadimitriou and Yannakakis [PY00] investigate the complexity of an approximate Pareto curve P_ε, that is, a set of solutions such that for every other solution $T \in \mathcal{S}$ there is a solution $S \in P_\varepsilon$ such that S dominates T or is within factor of ε in all objectives. This way, every solution is "almost" dominated by some solution in P_ε. They show for a quite general class of multiobjective optimization problems that there always exists an ε-Pareto curve of small size, i.e., polynomial in $1/\varepsilon$ and the problem size. Furthermore, they can characterize the class of problems for which an ε-Pareto curve can be constructed in polynomial time by proving equivalence to following the Gap-Problem: Given a vector of bounds $b_1, \ldots, b_k$ (one bound for each of the k objective functions), either return a solution that is in each objective better than the corresponding bound b_i, or state, that there is no solution that is better than the given bounds by a factor $1 + \varepsilon$. Papadimitriou and Yannakakis show that there exists an FPTAS that computes an ε-Pareto curve if the exact version of the single objective problem can be solved in pseudo-polynomial time. The exact version of an optimization problem asks for a solution with objective value exactly b, for some given b. The existence of a pseudo-polynomial algorithm for the exact version of an optimization problem is in fact a condition we will use in Chapter 3 to prove polynomial smoothed complexity for constrained binary optimization problems with stochastic constraints. Several binary optimization problems allow such algorithms, e.g. minimum spanning tree, shortest path and matching.

Despite the large number of Pareto points in the worst case, it has been observed in practice that for typical instances the number of Pareto points grows only moderate with the problem size. We will support this observation by presenting a smoothed/average-case analysis for binary optimization problems with two objective functions. In contrast to the work of Papadimitriou and Yannakakis we consider the exact Pareto curve but rely on stochastic assumptions on the objective functions. Furthermore, we do not provide general means to compute such a Pareto curve. This remains a problem specific task. For the knapsack problem and the constrained shortest path problem we will describe algorithms that compute the Pareto curve efficiently.

1.2 The Main Theorem

In this section we present our main theorem that bounds the expected number of Pareto points for binary optimization problems with two objective functions. The random input model is quite general and not restricted to a particular input distribution. We allow one of the objectives to be an arbitrary function $w : \mathcal{S} \to \mathbb{R}$. The second objective $p : \mathcal{S} \to \mathbb{R}$ is assumed to be a linear function $p(S) = \sum_{i \in S} p_i$ over the set of solutions $\mathcal{S}$. For historical reasons (we first developed our analysis for the bicriteria version of the knapsack problem) we will occasionally refer to $w(\cdot)$ and $p(\cdot)$ as the weight and profit functions and call $p_1, \ldots, p_n$ the profits of elements. We now state our main theorem.

Theorem 1.2. *Let $\mathcal{S} \subseteq 2^{[n]}$ be an arbitrary collection of subsets of $[n]$. Let $w : \mathcal{S} \to \mathbb{R}$ be an arbitrary function and $p(S) = \sum_{i \in S} p_i$ a linear function over $\mathcal{S}$ to be optimized simultaneously. The direction of optimization (minimization or maximization) can be specified arbitrarily for each of the two functions. The coefficients $p_1, \ldots, p_n$ are assumed to be independent random variables. Let q denote the number of Pareto points over $\mathcal{S}$.*

a) If $p_1, \ldots, p_n$ are chosen according to the uniform distribution over $[0,1]$, then $E[q] = O(n^3)$.

b) For $i \in [n]$, let p_i be a random variable with tail function $T_i : \mathbb{R}_{\geq 0} \to [0,1]$. Define $\mu_i = \mathbf{E}[p_i]$ and let α_i be an appropriate positive real number satisfying $\operatorname{slope}_{T_i}(x) \leq \alpha_i$ for every $x \geq 0$, $i \in [n]$. Let $\alpha = \max_{i \in [n]} \alpha_i$ and $\mu = \max_{i \in [n]} \mu_i$. Then

$$\mathbf{E}[q] \leq \left(\sum_{i \in [n]} \mu_i \right) \cdot \left(\sum_{i \in [n]} \alpha_i \right) + 1 \leq \alpha \mu n^2 + 1.$$

c) For every $i \in [n]$, let p_i be a non-negative random variable with density function $f_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Suppose $\mu = \max_{i \in [n]} (\mathbf{E}[p_i])$ and $\phi = \max_{i \in [n]} \left(\sup_{x \in \mathbb{R}_{\geq 0}} f_i(x) \right)$. Then $\mathbf{E}[q] = O(\phi \mu n^4)$.

First observe that the theorem makes no assumptions on the first objective function w, i.e., it is valid for any adversarial choice for w. In particular, it holds for any random choice for w, provided that w and p are stochastically independent. The three parts of the theorem bound the expected number of Pareto points for different classes of probability distributions for p. Each part is proven in a separate Lemma (Lemma 1.5, 1.7 and 1.10, in this order). Part (c) of the Theorem is in fact the most general bound that covers the probability distributions of part (a) and (b). It uses two parameters, the expectation μ of the distribution and the density parameter ϕ. The product $\phi\mu$ is in fact a consistent measure to describe the influence of the probability distribution on the number of Pareto points by the following argument. Scaling the function p by some factor $1/c > 0$ does not affect the set of Pareto points as the order of subsets that is prescribed by the function p in unchanged. The corresponding probability distribution has density function $f'(s) = cf(c\phi)$, expectation $\mu' = \mu/c$ and density parameter $\phi' = c\phi$. Hence, the product $\phi\mu$ is unaffected too by scaling. The smallest value for $\phi\mu$ is $1/2$ attained by the uniform distribution.

Let us normalize the specified distributions for the random variables $p_1, \ldots, p_n$ by multiplying all variables with $\frac{1}{\mu}$. This way, the maximum expected value over all variables is 1 and the upper bound on the expected number of Pareto points simplifies to $\mathbf{E}[q] = O(\phi n^4)$, for appropriate ϕ. Under this normalization, the maximum density ϕ can be seen as a parameter describing how much randomness is available. For $\phi \to \infty$, the randomness in the specification of the function p can go to zero and, in this case, an adversary can specify an input for an appropriate problem such that the expected number of Pareto points is exponential. In contrast, if ϕ is bounded from above by some constant term, then the described instances inhabit a high degree of randomness and the expected number of Pareto points is polynomial. In particular, the influence of the parameter ϕ on the complexity of the Pareto curve is at most linear. Experimental results for the bicriteria version of the knapsack problem (Section 2.5.2) suggest that the ϕ term is tight, i.e., the influence is indeed linear. Concerning the n^4 term, we believe that we lost a factor n^2 in the analysis and the real bound should be $O(\phi\mu n^2)$.

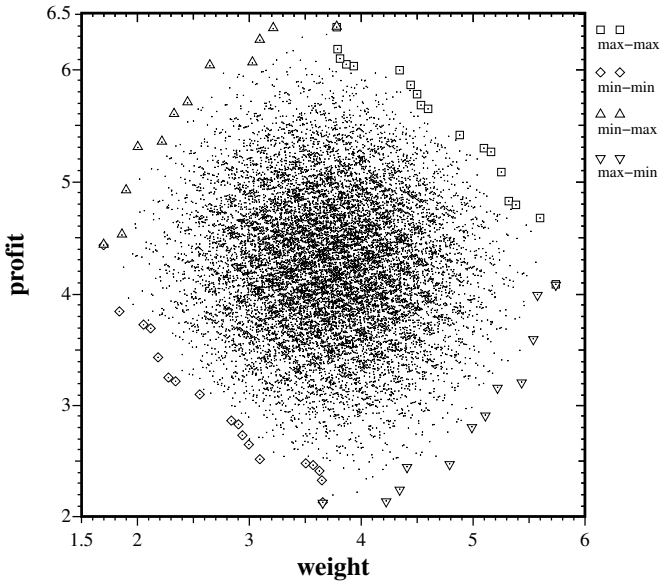

Figure 1.2: Points correspond to 8-element subsets over a ground set with 16 elements. These subsets are plotted according to two linear objective functions, called weight and profit. Weight and profit of each ground element have been chosen uniformly at random from $[0,1]$. The different markers indicate Pareto optimality for points according to different directions of optimization. The density of points in the "center" is exponentially larger than the density along the Pareto curve.

For the uniform distribution, $\phi\mu = 1/2$. Hence, part (c) of the theorem gives an upper bound of $O(n^4)$ for the uniform distribution, which is a factor n worse than the more specific bound in the first part of the theorem. As the distribution can be scaled, the result in part (a) actually holds for any uniform distribution with domain $[0,a]$, $a \in \mathbb{R}_{>0}$.

Part (b) of the theorem applies to the exponential and other long-tailed distributions. A distribution has a long tail if its tail function, under appropriate normalization, can be bounded from below by the tail of an exponential distribution. Only distributions with infinite domain can be long-tailed. For a more detailed discussion as well as the definition of the *slope* of a tail function we refer to Section 1.2.2. Let us only mention two examples for long-tailed distributions: the exponential distribution defined by tail function $T(x) = e^{-ax}$, and the Pareto distribution defined by tail function $T(x) = (1+x)^{-a}$, for some parameter $a > 0$. By Corollary 1.8 and 1.9, if all p_i follow the same distribution, then the corresponding bounds on the expected number of Pareto points are n^2 and $\frac{a}{a-1}\, n^2$, respectively. This is a factor n^2 smaller than the bound that can be obtained by applying part (c) of the theorem for general distributions. In fact, these results are tight. In Section 1.3 we provide a corresponding lower bound. Let us stress the fact that the result for long-tailed distributions plays a much more important role than simply providing better bounds. It is the basis for our analysis of general distributions.

Figure 1.2 illustrates the main theorem. It shows an example solution set and the corresponding Pareto points for different directions of optimization. The bound for general distributions can immediately be applied to our smoothed analysis framework.

Corollary 1.3. *Let $\mathcal{S} \subseteq 2^{[n]}$ be an arbitrary collection of subsets of $[n]$. Let $w : \mathcal{S} \to \mathbb{R}$ be an arbitrary function and $p(S) = \sum_{i \in S} p_i$ a linear function over $\mathcal{S}$ to be optimized simultaneously. The direction of optimization (minimization or maximization) can be specified arbitrarily for each of the two functions. Assume we perturb only the coefficients $p_1, \ldots, p_n$ according to our smoothed analysis framework. For any fixed perturbation model f with non-negative domain the expected number of Pareto points over $\mathcal{S}$ is $O(\phi n^4 + n^4)$.*

Proof. Recall, that our smoothed analysis framework restricts the adversarial choice for the coefficients p_i to $[0,1]$ since we assume a non-negative domain for those coefficients. The maximum expected value of the distribution functions for the coefficients is $1 + E/\phi$, where $E = \int |s| f(s) ds$ is a constant depending on the perturbation model f. Hence, the expected number of Pareto points is upper bounded by $O(\phi \mu n^4) = O(\phi(1 + E/\phi n^4) = O(\phi n^4 + n^4)$. □

Pareto Optimality Based on Explicit Subset Ordering

To prove Theorem 1.2 we use a slightly different definition of Pareto optimality. Instead of the domination concept based on two objective functions $w(\cdot)$ and $p(\cdot)$ we assume only one function $p(\cdot)$ together with an explicit ordering of all solutions in $\mathcal{S}$, i.e., a fixed sequence $S_1, \ldots, S_m$. We adapt the domination concept accordingly. Assume we want to maximize $p(\cdot)$. Then for all $u \in [m]$,

$$S_u \text{ is Pareto optimal} \Leftrightarrow \forall v \in [u-1] : p(S_v) < p(S_u) \quad .$$

The definition for minimization is analog. The explicit subset ordering $S_1, \ldots, S_m$ replaces the order that is implicitly given by the function $w(\cdot)$. The main difference is that two solutions can have the same function value under $w(\cdot)$, whereas the rank of a solution in $S_1, \ldots, S_m$ is always unique. As a consequence, the domination concept based on the explicit subset ordering might exhibit additional Pareto optimal solutions. This happens, if two solutions that have the same value under $w(\cdot)$ but different values under $p(\cdot)$ become both Pareto optimal under the explicit subset ordering, whereas only one of them would be Pareto optimal according to the domination concept based on the two functions $w(\cdot)$ and $p(\cdot)$. However, every Pareto optimal solution w.r.t. the functions $w(\cdot)$ and $p(\cdot)$ is also Pareto optimal w.r.t. function $p(\cdot)$ and any explicit subset order $S_1, \ldots, S_m$ that complies with the order given by $w(\cdot)$, where ties can be broken arbitrarily.

Minimization vs. Maximization

The direction of optimization has a great influence on the complexity of problems (e.g., finding shortest or longest path in a graph). However, for our analysis of the number of Pareto points there exists a close relation between maximization and minimization via complementary solution sets. For any set $S \subseteq [n]$, define the complementary set $\bar{S} = [n] \setminus S$.

Observation 1.4. *Assuming a linear function $p(S) = \sum_{i \in S} p_i$, then for all $u \in [m]$,*

$$S_u \text{ is Pareto optimal over } S_1, \ldots, S_m \text{ w.r.t. minimization of } p(\cdot) \tag{1.1}$$
$$\Leftrightarrow \quad \bar{S}_u \text{ is Pareto optimal over } \bar{S}_1, \ldots, \bar{S}_m \text{ w.r.t. maximization of } p(\cdot) \tag{1.2}$$

This can be seen as follows. Equation 1.1 is equivalent to

$$\begin{aligned} \forall v \in [u-1] \ : \quad & \sum_{i \in S_v} p_i \ > \ \sum_{i \in S_u} p_i \\ \Leftrightarrow \forall v \in [u-1] \ : \quad & \sum_{i \in [n] \setminus \bar{S}_v} p_i \ > \ \sum_{i \in [n] \setminus \bar{S}_u} p_i \\ \Leftrightarrow \forall v \in [u-1] \ : \quad & \sum_{i \in \bar{S}_v} p_i \ < \ \sum_{i \in \bar{S}_u} p_i \ , \end{aligned}$$

which is equivalent to Equation 1.2. Since we allow arbitrary solution sets and assume a linear function $p(\cdot)$ it suffices to provide only a proof for maximization of $p(\cdot)$ as this implies the same bound for minimization.

1.2.1 The Uniform Distribution

In this section we assume that the coefficients $p_1, \ldots, p_n$ are real numbers chosen uniformly at random from $[0,1]$. We subsequently assume a maximization problem. By Observation 1.4, the following lemma holds for the minimization problem as well.

Lemma 1.5. *Let $S_1, \ldots, S_m$ be an arbitrary but fixed sequence of subsets of $[n]$. Suppose the coefficients $p_1, \ldots, p_n$ are chosen according to the uniform distribution over $[0,1]$. Let q denote the number of Pareto points over $S_1, \ldots, S_m$. Then $E[q] = O(n^3)$.*

Proof. First observe that, without loss of generality, we can assume for any $i \neq j \in [m]$, $S_i \subset S_j \Rightarrow i < j$. In other words, no subset of S_j succeeds S_j in the given order. Assume there is such a subset $S_i \subset S_j$ with $i > j$. Since the coefficients $p_1, \ldots, p_n$ are non-negative, $p(S_i) \leq p(S_j)$. As $i > j$, the set S_i can never become Pareto optimal, irrespectively of the random choice for the coefficients $p_1, \ldots, p_n$. Therefore, we could remove the set S_i from the sequence $S_1, \ldots, S_m$ to be analyzed without effecting the result.

For any $2 \leq u \leq m$, define $\Delta_u = \max_{v \in [u]} p(S_v) - \max_{v \in [u-1]} p(S_v) \geq 0$. Observe that S_1 is always Pareto optimal. For all $u \geq 2$, S_u is Pareto optimal if and only if $\Delta_u > 0$. These definitions are illustrated in Figure 1.3. The following lemma shows that the expected increase in profit at Pareto optimal sets is $\Omega(n^{-2})$.

Lemma 1.6. *For every $u \in \{2, \ldots, m\}$, $\mathbf{E}[\Delta_u \,|\, \Delta_u > 0] \geq \dfrac{1}{32n^2}$.*

Proof. Fix $u \in \{2, \ldots, m\}$. Observe that

$$\mathbf{E}[\Delta_u \,|\, \Delta_u > 0] \geq \mathbf{Pr}\left[\Delta_u \geq \tfrac{1}{16n^2} \,\middle|\, \Delta_u > 0\right] \cdot \tfrac{1}{16n^2}.$$

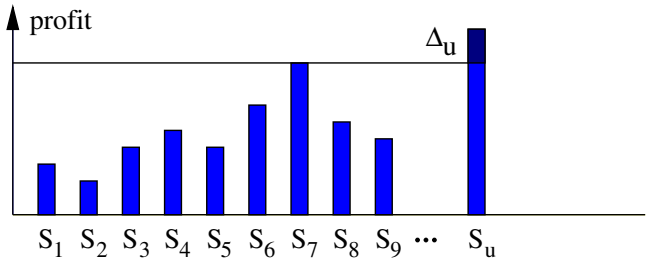

Figure 1.3: Sequence of subsets $S_1,\ldots,S_m$. The random variable Δ_u gives the increase in profit at subset u compared with the most profitable subset left of u. If a subset v is dominated then $\Delta_v = 0$.

Hence, it suffices to show $\mathbf{Pr}\left[\Delta_u \geq \frac{1}{16n^2} \,\middle|\, \Delta_u > 0\right] \geq \frac{1}{2}$. For every $v \in [u-1]$, define $X_v = S_u \setminus S_v$ and $Y_v = S_v \setminus S_u$. It holds

$$\begin{aligned}\mathbf{Pr}\left[\Delta_u \geq \tfrac{1}{16n^2} \,\middle|\, \Delta_u > 0\right] &= \mathbf{Pr}\left[\forall v : p(S_u) \geq p(S_v) + \tfrac{1}{16n^2} \middle| \forall v : p(S_u) > p(S_v)\right] \\ &= \mathbf{Pr}\left[\forall v : p(X_v) \geq p(Y_v) + \tfrac{1}{16n^2} \middle| \forall v : p(X_v) \geq p(Y_v)\right], \qquad (1.3)\end{aligned}$$

where the universal quantifier ranges over all elements $v \in [u-1]$. Since we consider continuous probability distributions, the relaxation of the strict inequality in the conditioning part does not effect the probability. Without loss of generality, $S_u = [k]$. We distinguish two classes of random variables, namely $\{p_1,\ldots,p_k\}$ and $\{p_{k+1},\ldots,p_n\}$. Observe that the X_v's are subsets of the first class and the Y_v's are subsets of the second class. For a moment, let us assume that the variables in the second class are fixed arbitrarily. We investigate the variables in the first class under this assumption. Regardless of how the variables in the second class are fixed, it is unlikely that one of the variables in the first class is much smaller than n^{-1}. In particular,

$$\begin{aligned}\mathbf{Pr}\left[\exists j \in [k] : p_j \leq \tfrac{1}{4n} \mid \forall v : p(X_v) \geq p(Y_v)\right] &\leq \sum_{j\in[k]} \mathbf{Pr}\left[p_j \leq \tfrac{1}{4n} \mid \forall v : p(X_v) \geq p(Y_v)\right] \\ &\leq \sum_{j\in[k]} \mathbf{Pr}\left[p_j \leq \tfrac{1}{4n}\right] = \frac{k}{4n} \leq \frac{1}{4}\,.\end{aligned}$$

From now on, we assume $p_j \geq \frac{1}{4n}$, for every $j \in [k]$. Let $L_v = p(X_v)$. Because of our assumption $S_i \subset S_j \Rightarrow i < j$, for any $i \neq j \in [m]$, each set X_v contains at least one element. Therefore, $L_v \geq \frac{1}{4n}$, for every $v \in [u-1]$. In the following we will assume that the L_v's are fixed in a way satisfying this property but, otherwise, arbitrarily. (We will not consider the variables $p_1,\ldots,p_k$ anymore.) Under our assumption on the L_v's, we analyze $\mathbf{Pr}\left[\Delta_u < \frac{1}{16n^2} \,|\, \Delta_u > 0\right]$. In order to compensate for the case when our assumption fails, we prove that this probability is at most $\frac{1}{4}$ instead of $\frac{1}{2}$. In particular, using the definition of the L_v's to rewrite Equation 1.3, we prove

$$\mathbf{Pr}\left[\forall v : p(Y_v) \leq L_v - \frac{1}{16n^2} \,\middle|\, \forall v : p(Y_v) \leq L_v\right] \geq \frac{3}{4}\,,$$

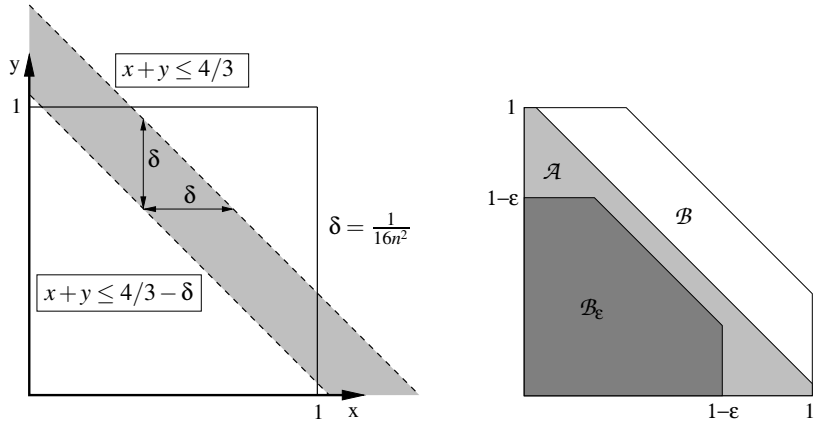

Figure 1.4: Two-dimensional example for polytopes $\mathcal{A}, \mathcal{B}$ and $\mathcal{B}_\varepsilon$ with $\mathcal{B} = \{(x,y) \in [0,1]^2 \mid x+y \leq 4/3\}$.

for arbitrarily fixed $L_v \geq \frac{1}{4n}$, where the probability refers solely to the random choices for the variables $p_{k+1}, \ldots, p_n$. Let us now switch to a geometric interpretation. Consider the following $(n-k)$-dimensional polytopes.

$$\mathcal{A} = \left\{ (p_{k+1} \times \ldots \times p_n) \in [0,1]^{n-k} \;\middle|\; \forall v : p(Y_v) \leq L_v - \tfrac{1}{16n^2} \right\},$$
$$\mathcal{B} = \left\{ (p_{k+1} \times \ldots \times p_n) \in [0,1]^{n-k} \mid \forall v : p(Y_v) \leq L_v \right\} .$$

Figure 1.4 illustrates the definition of these two polytopes. Clearly, $\mathcal{A} \subseteq \mathcal{B}$. As we investigate the uniform distribution,

$$\mathbf{Pr}\left[\Delta_u \geq \frac{1}{16n^2} \;\middle|\; \Delta_u > 0 \right] = \frac{vol(\mathcal{A} \cap \mathcal{B})}{vol(\mathcal{B})} = \frac{vol(\mathcal{A})}{vol(\mathcal{B})},$$

with $vol(\cdot)$ specifying the volume of the corresponding polytopes. In terms of these volumes, we have to show $vol(\mathcal{A}) \geq \frac{3}{4} vol(\mathcal{B})$.

At first view, it might seem that the ratio $vol(\mathcal{A})/vol(\mathcal{B})$ depends on the number of facets of these polytopes. This number, however, can be exponential, since facets correspond to subsets of $[n]$. Fortunately, however, the following argument shows that the ratio between the two volumes can be estimated in terms of the number of dimensions rather than the number of facets. The idea is to shrink the polytope $\mathcal{B}$ uniformly over all dimensions until the shrunken polytope is contained in polytope $\mathcal{A}$. For $\varepsilon \in [0,1]$, we define

$$\mathcal{B}_\varepsilon = \left\{ (p_{k+1} \times \ldots \times p_n) \in [0, 1-\varepsilon]^{(n-k)} \mid \forall v : p(Y_v) \leq (1-\varepsilon) L_v \right\}.$$

Obviously, $\mathcal{B} = \mathcal{B}_0$ and, in general, $\mathcal{B}_\varepsilon$ can be obtained by shrinking $\mathcal{B}$ in each dimension by a factor of $1-\varepsilon$. As the number of dimensions is $n-k$, it holds $vol(\mathcal{B}_\varepsilon) = (1-\varepsilon)^{n-k}\, vol(\mathcal{B})$.

To ensure that $\mathcal{B}_\varepsilon$ is contained in $\mathcal{A}$, we have to choose ε satisfying $(1-\varepsilon)L_v \leq L_v - \frac{1}{16n^2}$ for all $v \in [u-1]$. For $\varepsilon \geq \frac{1}{4n}$, the above equation is always satisfied, because $L_v \geq \frac{1}{4n}$. Thus setting $\varepsilon = \frac{1}{4n}$

implies $\mathcal{B}_\varepsilon \subseteq \mathcal{A}$. As a consequence,

$$vol(\mathcal{A}) \geq vol(\mathcal{B}_\varepsilon) = (1-\varepsilon)^{(n-k)}\, vol(\mathcal{B}) \geq (1-\varepsilon(n-k))\, vol(\mathcal{B}) \geq \tfrac{3}{4}\, vol(\mathcal{B}) \ ,$$

which completes the proof of Lemma 1.6. □

The lemma above shows that at every Pareto optimal set the expected increase in profit is at east $\frac{1}{32n^2}$. On the other hand, the expected profit of the set $[n]$ is $n/2$ as each individual element has expected profit $1/2$. It might be intuitively clear that this implies that the expected number of Pareto optimal sets is at most $16n^3$. The following calculation proves this statement in a formal way. It holds

$$\max_{u\in[m]} p(S_u) = S_1 + \sum_{u=2}^{m} \Delta_u \ ,$$

therefore, $p([n]) \geq \sum_{u=2}^{m} \Delta_u$. As $\mathbf{E}[p_i] = 1/2$, for each $i \in [n]$, we have

$$\begin{aligned} n/2 = \mathbf{E}[p([n])] &\geq \sum_{u=2}^{m} \mathbf{E}[\Delta_u] \\ &= \sum_{u=2}^{m} \mathbf{Pr}[\Delta_u > 0] \cdot \mathbf{E}[\Delta_u \mid \Delta_u > 0] \\ &\geq \sum_{u=2}^{m} \mathbf{Pr}[\Delta_u > 0] \cdot \frac{1}{32n^2} \ . \end{aligned}$$

Consequently,

$$\mathbf{E}[q] = 1 + \sum_{u=2}^{m} \mathbf{Pr}[\Delta_u > 0] \leq 16n^3 + 1 \ .$$

The additional 1 is due to the set S_1, which is always Pareto optimal. Thus Lemma 1.5 is shown. □

1.2.2 Long-tailed Distributions

One can classify continuous probability distributions by comparing their tails with the tail of the exponential distribution. In principle, if the tail function of a distribution can be lower-bounded by the tail function of the exponential function, then we say the distribution has a "long tail", and if the tail function can be upper-bounded by the exponential tail function, then we talk about "short tails". In this section, we investigate the expected number of Pareto points under long-tailed distributions. In fact, we can prove a slightly better bound for these distributions than for the short-tailed uniform distribution. Moreover, our analysis can handle heterogeneous distributions. We want to point out that the results we prove for the long-tailed distributions are important tools in the subsequent analysis for general probability distributions.

We need to define the term "long-tailed distribution" more formally. Of special interest for us is the behavior of the tail function under a logarithmic scale. Given any continuous probability distribution with density function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, the tail function $T : \mathbb{R}_{\geq 0} \to [0,1]$ is defined by $T(t) = \int_t^\infty f(x)dx$. We define the *slope* of tail function T at $x \in \mathbb{R}_{\geq 0}$ to be the first derivative of the function $-\ln(T(\cdot))$

at x, i.e., $\text{slope}_T(x) = -[\ln(T(x)]'$. For example, the tail function of the exponential distribution with parameter λ is $T(x) = \exp(-\lambda x)$ so that the slope of this function is $\text{slope}_T(x) = \lambda$, for every $x \geq 0$. In general, $\text{slope}_T(x)$ is a not necessarily continuous function with non-negative real values. The tail of a continuous probability distribution is defined to be *long* if there exists $\alpha > 0$ such that $\text{slope}_T(x) \leq \alpha$, for every $x \in \mathbb{R}_{\geq 0}$.

According to this definition, the exponential distribution has long tails. However, the uniform distribution over $[0,1]$ (or any other interval) does not have long tails because $\text{slope}_T(x) = 1/(1-x)$, which grows to ∞ for $x \to 1$. Observe that any distribution with a bounded domain cannot have long tails. A typical example for a distribution with long tails is the Pareto distribution. The tail function of the Pareto distribution with parameter $a > 0$ is $T(x) = (1+x)^{-a}$. These tails are long because $\text{slope}_T(x) = [a\ln(1+x)]' = \frac{a}{1+x} \leq a$, for every $x \geq 0$.

We assume that the coefficients $p_1, \ldots, p_n$ are chosen independently at random according to possibly different long-tailed distributions with finite mean.

Lemma 1.7. *For $i \in [n]$, let coefficient p_i be a random variable with tail function $T_i : \mathbb{R}_{\geq 0} \to [0,1]$. Define $\mu_i = \mathbf{E}[p_i]$ and let α_i be an appropriate positive real number satisfying $\text{slope}_{T_i}(x) \leq \alpha_i$ for every $x \geq 0$, $i \in [n]$. Let $\alpha = \max_{i\in[n]} \alpha_i$ and $\mu = \max_{i\in[n]} \mu_i$. Let $S_1, \ldots, S_m$ be an arbitrary but fixed sequence of subsets of $[n]$ and let q denote the number of Pareto optimal set over $S_1, \ldots, S_m$. Then*

$$\mathbf{E}[q] \leq \left(\sum_{i\in[n]} \mu_i\right) \cdot \left(\sum_{i\in[n]} \alpha_i\right) + 1 \leq \alpha\mu n^2 + 1.$$

Proof. We use an approach similar to the proof of Lemma 1.5. The bounds that we can prove, however, are slightly better, as we can exploit the Markovian properties of the exponential distribution lower-bounding the long tailed distributions under consideration.

As in the proof of Lemma 1.5 and for the same reasons, we assume, without loss of generality, that for any $i \neq j \in [m]$, $S_i \subset S_j \Rightarrow i < j$. Fix $u \in [m]$, $u \geq 2$. For every $v \in [u-1]$, define $X_v = S_u \setminus S_v$ and $Y_v = S_v \setminus S_u$. If S_u is Pareto optimal, then the expected increase in profit is $\mathbf{E}[\Delta_u \,|\, \Delta_u > 0]$ corresponding to

$$\begin{aligned} &\mathbf{E}\left[\min_{v\in[u-1]} (p(X_v) - p(Y_v)) \;\middle|\; \min_{v\in[u-1]} (p(X_v) - p(Y_v)) > 0\right] \\ &= \int_0^\infty \mathbf{Pr}[\forall v : p(X_v) - p(Y_v) \geq t \mid \forall v : p(X_v) \geq p(Y_v)]\, dt \ . \end{aligned}$$

Let $k = |S_u|$ be the number of elements in S_u. Without loss of generality, assume $S_u = [k]$. Observe that $X_v \subseteq [k]$ and $Y_v \subseteq [n] \setminus [k]$, for every $v \in [u-1]$. Our next goal is to isolate the random variables $p_1, \ldots, p_k$. For this purpose we partition the set $[u-1]$ into disjoint groups $G_1, \ldots, G_k$ satisfying the following property: $\forall j \in [k] : v \in G_j \Rightarrow j \in X_v$. There is always such a partitioning since none of the X_v, $v \in [u-1]$, is the empty set which is a consequence of our assumption $S_i \subset S_j \Rightarrow i < j$, for any $i \neq j \in [m]$.

Let $E_j(t)$ denote the event $\forall v \in G_j : p(X_v) \geq p(Y_v) + t$. Then

$$\begin{aligned}\mathbf{E}[\Delta_u \,|\, \Delta_u > 0] &= \int_0^\infty \mathbf{Pr}\left[\bigwedge_{i\in[k]} E_i(t) \;\middle|\; \bigwedge_{i\in[k]} E_i(0)\right] dt \\ &= \int_0^\infty \prod_{j=1}^{k} \mathbf{Pr}\left[E_j(t) \;\middle|\; \bigwedge_{i=1}^{j-1} E_i(t) \wedge \bigwedge_{i\in[k]} E_i(0)\right] dt\,.\end{aligned}$$

Now fix some $j \in [k]$ and let us assume that the values of all random variables except for p_j are fixed as well. Define

$$A_j = \max_{v\in G_j} \big(p(Y_v) - p(X_v \setminus \{j\})\big).$$

Notice that A_j is independent of p_j and, therefore, A_j is also fixed. This way, the expression $E_j(t)$ is equivalent to the expression $p_j \geq A_j + t$. If there is some $v \in [u-1]$ with $j \in S_v$ and $p(S_v) > p(S_u) + t$ then $\bigwedge_{i\in[k]} E_i(t) = \emptyset$, because p_j is the only variable not fixed. This case is not relevant for the integral above. So assume that for all subsets S_v with $v \in [u-1]$ and $j \in S_v$ it holds $p(S_v) \leq p(S_u) + t$. Then the expression $\bigwedge_{i=1}^{j-1} E_i(t) \wedge \bigwedge_{i\in[k]} E_i(0)$ is equivalent to $p_j \geq A'_j$, for some $A'_j \geq A_j$, since $E_j(0)$ corresponds to $p_j \geq A_j$. Consequently,

$$\begin{aligned}\mathbf{Pr}\left[E_j(t) \;\middle|\; \bigwedge_{i=1}^{j-1} E_i(t) \wedge \bigwedge_{i\in[k]} E_i(0)\right] &= \mathbf{Pr}\left[p_j \geq A_j + t \mid p_j \geq A'_j\right] \\ &\geq \mathbf{Pr}\left[p_j \geq A_j + t \mid p_j \geq A_j\right].\end{aligned}$$

Recall that T_j is the tail function for the random variable p_j. Hence,

$$\mathbf{Pr}\left[p_j \geq A_j + t \,|\, p_j \geq A_j\right] = \frac{T_j(A_j + t)}{T_j(A_j)}\quad.$$

For the exponential distribution with parameter α, it holds $\frac{T_j(A_j+t)}{T_j(A_j)} = T_j(t) = e^{-\alpha t}$. Here the first equality corresponds to the so-called Markovian or memoryless property of the exponential distribution. For other long-tailed distributions we need a slightly more complicated calculation. Recall, for all $i \in [n]$, $x \in \mathbb{R}_{\geq 0}$, we assume $\mathrm{slope}_{T_i}(x) = -[\ln(T_i(x)]' \leq \alpha_i$. This yields

$$\begin{aligned}\ln\left(\frac{T_j(x+t)}{T_j(x)}\right) &= \ln(T_j(x+t)) - \ln(T_j(x)) \\ &\geq (\ln(T_j(x)) - \alpha_j t) - \ln(T_j(x)) \;= -\alpha_j t\end{aligned}$$

so that $\mathbf{Pr}\left[p_j \geq A_j + t \,|\, p_j \geq A_j\right] = \frac{T_j(A_j+t)}{T_j(A_j)} \geq e^{-\alpha_j t}$, regardless of the outcome of A_j. Putting all together,

$$\mathbf{E}[\Delta_u \,|\, \Delta_u > 0] \geq \int_0^\infty \prod_{j\in[k]} \exp(-\alpha_j t) dt \geq \int_0^\infty \exp(\textstyle\sum_{j\in[n]} - \alpha_j t) dt = \dfrac{1}{\sum_{j\in[n]} \alpha_j},$$

for every $u \in \{2,\dots,m\}$. Moreover $\mathbf{E}[p([n])] = \sum_{i\in[n]} \mu_i$. Thus, analogous to the proof of Lemma 1.5, we are now able to bound the expected number of Pareto optimal sets by

$$\mathbf{E}[q] \;\leq\; 1 + \frac{\mathbf{E}[p([n])]}{\min_{u=2}^{m}(\mathbf{E}[\Delta_u \mid \Delta_u > 0])} \leq 1 + \left(\sum_{i\in[n]} \mu_i\right) \cdot \left(\sum_{i\in[n]} \alpha_i\right).$$

This completes the proof of Lemma 1.7. □

Example distributions

Let us illustrate the power of Lemma 1.7 by investigating $\mathbf{E}[q]$, the expected number of Pareto points, for some specific long-tailed probability distributions. In order to simplify the presentation of the results, let us assume that all coefficients $p_1,\dots,p_n$ follow the same distribution.

Corollary 1.8. *If $p_1,\dots,p_n$ are chosen according to the exponential distribution, then $\mathbf{E}[q] = O(n^2)$.*

This result is tight. In Section 1.3 we show a corresponding lower bound. Observe that the result for the exponential distribution does not depend on the parameter of this distribution as the mean μ is reciprocal of the slope α, regardless of the choice for the parameter of this distribution. This is slightly different in the case of the Pareto distribution. For the Pareto distribution with parameter $a > 1$, we choose $\alpha = a$ and $\mu = \frac{1}{a-1}$. This gives the following upper bound on the running time.

Corollary 1.9. *If $p_1,\dots,p_n$ are chosen according to the Pareto distribution with parameter $a > 1$ then $\mathbf{E}[q] = O(\frac{a}{a-1}\, n^2)$.*

Observe that the expectation of the Pareto distribution is not well-defined for parameter $a \leq 1$ so that our proof technique yields meaningful results only for $a > 1$.

1.2.3 General Distributions

In this section, we extend our result towards general, continuous distributions over $\mathbb{R}_{\geq 0}$ with finite mean and bounded density functions. The following Lemma shows that the expected number of Pareto points increases only linearly with the maximum expectation and the maximum density of the probability distributions over the coefficients $p_1,\dots,p_n$. In Section 1.4 we show how this result can be generalized towards discrete probability distributions.

Lemma 1.10. *For every $i \in [n]$, let coefficient p_i be a non-negative random variable with density function $f_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Suppose $\mu = \max_{i\in[n]}(\mathbf{E}[p_i])$ and $\phi = \max_{i\in[n]}\left(\sup_{x\in\mathbb{R}_{\geq 0}} f_i(x)\right)$. Let $S_1,\dots,S_m$ be an arbitrary but fixed sequence of subsets of $[n]$ and let q denote the number of Pareto optimal set over $S_1,\dots,S_m$. Then $\mathbf{E}[q] = O(\phi\mu n^4)$.*

Proof. Unfortunately, the analysis presented in the previous section fails for distributions with short tails. In particular, the idea to lower-bound the increase in profit at every Pareto optimal set does not work for short-tailed distributions. In fact, one can define a collection of short-tailed distributions for which the

expected increase in profit at some sets, if they become Pareto optimal, is arbitrarily small. The point is, however, that those sets are very unlikely to become Pareto optimal. This property needs to be exploited in the following analysis.

Let $T_i(a) = \int_a^\infty f_i(t)dt$ denote the tail function for the random variable p_i, $i \in [n]$. For each p_i we define an auxiliary random variable $x_i = T_i(p_i)$. Observe that the x_i's are uniformly distributed over $[0,1]$. Furthermore, we introduce a cascade of events on which we will condition. For $k \geq 0$, let X_k denote the event $\forall i \in [n] : x_i \geq 2^{-k}/n$. Observe that $X_{k-1} \subseteq X_k$, for every $k \geq 1$. By conditioning on this cascade of events, we obtain the following upper bound on the expected number of Pareto optimal sets:

$$\begin{aligned}\mathbf{E}[q] &= \mathbf{Pr}[X_0] \cdot \mathbf{E}[q \mid X_0] + \sum_{k=1}^{\infty} \mathbf{Pr}[X_k \wedge \neg X_{k-1}] \cdot \mathbf{E}[q \mid X_k \wedge \neg X_{k-1}] \\ &\leq \mathbf{Pr}[X_0] \cdot \mathbf{E}[q \mid X_0] + \sum_{k=1}^{\infty} \frac{\mathbf{E}[q \mid X_k \wedge \neg X_{k-1}]}{2^k} \quad , \end{aligned} \tag{1.4}$$

where the last inequality follows because

$$\mathbf{Pr}[X_k \wedge \neg X_{k-1}] < \mathbf{Pr}\left[\exists i \in [n] : \frac{1}{2^k n} \leq x_i < \frac{1}{2^{k-1}/n}\right] < n \cdot \max_{i \in [n]} \left\{\mathbf{Pr}\left[\frac{1}{2^k n} \leq x_i < \frac{2}{2^k n}\right]\right\} = \frac{n}{2^k n} = 2^{-k}.$$

Unfortunately, placing conditions in such a direct way does not yield longer tails but shorter tails as conditioning on X_k simply cuts the tail function at position $T_i^{-1}(2^{-k}/n)$. The following trick avoids this kind of unwanted effect by *masking* the short tails. We parameterize the random variable q with the solution set $\mathcal{S}$, that is, $q(\mathcal{S})$ denotes the number of Pareto optimal sets over the subsets in $\mathcal{S}$. For every $k \geq 0$ and every $\mathcal{S} \subseteq 2^{[n]}$ we define an auxiliary random variable $q_k(S)$ as follows.

$$q_k(S) \;=\; \begin{cases} q(S) & \text{if } X_k\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

We also write q_k for $q_k(\mathcal{S})$ if $\mathcal{S}$ is the solution set $\{S_1, \ldots, S_m\}$ to be analyzed. The variable $q_k(\mathcal{S})$ masks Pareto optimal sets in case of $\neg X_k$. The following lemma shows that this masking technique enables us to get rid of conditional probabilities so that our analysis for long tails can be applied to estimate the unconditioned q_k variables subsequently.

Lemma 1.11. *For every $k \geq 0$, $\mathbf{E}[q] \leq \sum_{k=0}^{\infty} 2^{-k+4}\, \mathbf{E}[q_k(\mathcal{S}_k)]$, for some $\mathcal{S}_k \subseteq 2^{[n]}$.*

Proof. First we rewrite Equation 1.4 in terms of the q_k variables, that is,

$$\mathbf{E}[q] \quad \leq \quad \mathbf{Pr}[X_0] \cdot \mathbf{E}[q_0 \mid X_0] + \sum_{k=1}^{\infty} \frac{\mathbf{E}[q_k \mid X_k \wedge \neg X_{k-1}]}{2^k} \quad . \tag{1.5}$$

Since $\mathbf{E}[q_0 \mid \neg X_0] = 0$ we have

$$\mathbf{Pr}[X_0] \cdot \mathbf{E}[q_0 \mid X_0] \;=\; \mathbf{Pr}[X_0] \frac{\mathbf{E}[q_0]}{\mathbf{Pr}[X_0]} \;=\; \mathbf{E}[q_0] \quad . \tag{1.6}$$

Next we upper bound $\mathbf{E}[q_k \mid X_k \wedge \neg X_{k-1}]$. For this purpose we partition the event $\neg X_{k-1}$ into n disjoint events $Z_1, \ldots, Z_n$ with

$$Z_j = \left[x_1, x_2, \ldots, x_{j-1} \geq \frac{1}{2^{k-1}n} \wedge x_j < \frac{1}{2^{k-1}n} \right].$$

Then, for $k \geq 1$,

$$\begin{aligned}
\mathbf{Pr}[X_k \,|\, \neg X_{k-1}] &= \sum_{j=1}^{n} \mathbf{Pr}[X_k \,|\, Z_j] \cdot \mathbf{Pr}\left[Z_j \,|\, \bigcup_{i\in[n]} Z_i \right] \\
&= \sum_{j=1}^{n} \frac{1}{2} \left(1 - \frac{1}{2^k n} \right)^{n-j} \mathbf{Pr}\left[Z_j \,|\, \bigcup_{i\in[n]} Z_i \right] \\
&\geq \frac{1}{2} \left(1 - \frac{1}{2^k n} \right)^{n-1} \sum_{j=1}^{n} \mathbf{Pr}\left[Z_j \,|\, \bigcup_{i\in[n]} Z_i \right] \\
&= \frac{1}{2} \left(1 - \frac{1}{2^k n} \right)^{n-1} \geq \frac{1}{4} \ .
\end{aligned}$$

Thus, we get

$$\mathbf{E}[q_k \,|\, X_k \wedge \neg X_{k-1}] \leq \frac{\mathbf{E}[q_k \,|\, \neg X_{k-1}]}{\mathbf{Pr}[X_k \,|\, \neg X_{k-1}]} \leq 4\,\mathbf{E}[q_k \,|\, \neg X_{k-1}]. \tag{1.7}$$

Finally, we need to get rid of the conditioning on $\neg X_{k-1}$. This condition states that at least one of the coefficients $p_1, \ldots, p_n$ has a very large value. Roughly speaking, only one variable will be affected by this condition. The idea is now to make use of the fact that every individual coefficient p_i can increase the number of Pareto points only by a factor of two.

For all $i \in [n]$, let Y_i denote the event $x_i < 2^{-(k-1)}/n$. As $\neg X_{k-1} = \bigcup_{i\in[n]} Y_i$,

$$\mathbf{E}[q_k \,|\, \neg X_{k-1}] = \mathbf{E}\left[q_k \,\middle|\, \bigcup_{i\in[n]} Y_i \right] \leq \sum_{j\in[n]} \mathbf{E}[q_k \,|\, Y_j] \cdot \mathbf{Pr}\left[Y_j \,\middle|\, \bigcup_{i\in[n]} Y_i \right] \ .$$

Observe that the last estimation would hold with equality if the events $Y_1, \ldots, Y_n$ were disjoint. As these events overlap, however, the right hand term slightly overestimates the left hand term. Now $\mathbf{Pr}[Y_j] = 2^{-k+1}/n$ and $\mathbf{Pr}\left[\bigcup_{i\in[n]} Y_i \right] = 1 - (1 - 2^{-k+1}/n)^n \geq 2^{-k}$, so that

$$\mathbf{Pr}\left[Y_j \,\middle|\, \bigcup_{i\in[n]} Y_i \right] = \frac{\mathbf{Pr}[Y_j]}{\mathbf{Pr}\left[\bigcup_{i\in[n]} Y_i \right]} \leq \frac{2^{-k+1}/n}{2^{-k}} \leq \frac{2}{n} \ .$$

As a consequence,

$$\begin{aligned}
\sum_{j\in[n]} \mathbf{E}[q_k \,|\, Y_j] \cdot \mathbf{Pr}\left[Y_j \,\middle|\, \bigcup_{i\in[n]} Y_i \right] &\leq \max_{j\in[n]} (\mathbf{E}[q_k \,|\, Y_j]) \sum_{j\in[n]} \mathbf{Pr}\left[Y_j \,\middle|\, \bigcup_{i\in[n]} Y_i \right] \\
&\leq 2 \max_{j\in[n]} (\mathbf{E}[q_k \,|\, Y_j]) \ .
\end{aligned}$$

Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be the solution set to be analyzed. For all $j \in [n]$, define $\mathcal{S}'_j = \{S_i \mid i \in [m], j \notin S_i\}$, and $\mathcal{S}''_j = \{S_i \setminus \{j\} \mid i \in [m], j \in S_i\}$. Observe that $q_k(\mathcal{S}) \leq q_k(\mathcal{S}'_j) + q_k(\mathcal{S}''_j)$ for all $j \in [n]$. As no subset in $\mathcal{S}'_j$ and $\mathcal{S}'_j$ contains element j, $\mathbf{E}[q_k \,|\, Y_j] \leq \mathbf{E}\left[q_k(\mathcal{S}'_j) + q_k(\mathcal{S}''_j)\right]$. Hence,

$$2 \max_{j \in [n]} (\mathbf{E}[q_k \,|\, Y_j]) \leq 2 \max_{j \in [n]} \mathbf{E}\left[q_k(\mathcal{S}'_j) + q_k(\mathcal{S}''_j)\right] \quad .$$

Let j denote the index that maximizes the last expressions. Define $\mathcal{S}' = \arg\max \left\{\mathbf{E}\left[q_k(\mathcal{S}'_j)\right], \mathbf{E}\left[q_k(\mathcal{S}''_j)\right]\right\}$. Then,

$$\mathbf{E}[q_k \,|\, \neg X_{k-1}] \;\leq\; 2\,(\mathbf{E}[q_k \,|\, Y_j]) \;\leq\; 4\mathbf{E}\left[q_k(\mathcal{S}')\right] \quad . \tag{1.8}$$

Using Equations (1.6), (1.7) and (1.8) to substitute Equation (1.5) yields the lemma. □

Next we apply our analysis for the long tailed distributions to the q_k variables.

Lemma 1.12. *For every $k \geq 0$ and $\mathcal{S} \subseteq 2^{[n]}$, $\mathbf{E}[q_k(\mathcal{S})] \leq \min\{n^3 2^k \phi\mu + n + 1, 2^n\}$.*

Proof. The bound $\mathbf{E}[q_k(\mathcal{S})] \leq 2^n$ holds trivially because there are at most 2^n different subsets over n elements. The bound $\mathbf{E}[q_k(\mathcal{S})] \leq n^3 2^k \phi\mu + n + 1$ can be shown with the help of Lemma 1.7. Define $B_{i,k} = T_i^{-1}(2^{-k}/n)$. Remember that q_k counts only Pareto points under X_k. Therefore, the behavior of the tail function for values larger that $B_{i,k}$ is irrelevant for q_k and we can modify the tail function for values larger than $B_{i,k}$ without affecting q_k. In fact, changing $T_i(x)$, for $x > B_{i,k}$, enables us to bound the slope of the tail function as needed for the application of Lemma 1.7.

Consider the following variants of our tail functions. We cut the tail functions T_i at position $B_{i,k}$ and replace the original, possibly short tails by long, exponential tails. For $i \in [n]$ and $k \geq 0$, define

$$T_{i,k}(t) \;=\; \begin{cases} T_i(t) & \text{if } t \leq B_{i,k}\,, \\ \exp(-\phi n(t - B_{i,k}))/(n2^k) & \text{if } t > B_{i,k}\,. \end{cases}$$

The slope of this tail function can be bounded as follows. For $t \leq B_{i,k}$,

$$\text{slope}_{T_{i,k}}(t) \;=\; [-\ln(T_i(t))]' \;=\; \frac{-[T_i(t)]'}{T_i(t)} \;\leq\; \phi n 2^k$$

because $-[T_i(t)]' = f_i(t) \leq \phi$ and $T_i(t) \geq 2^{-k}/n$ since $t \leq B_{i,k} = T^{-1}(2^{-k}/n)$. The same upper bound holds also for $t > B_{i,k}$ since, in this case,

$$\text{slope}_{T_{i,k}}(t) \;=\; \left[-\ln\left(\frac{\exp(-\phi n(t - B_{i,k}))}{n2^k}\right)\right]' \;=\; \phi n \;\leq\; \phi n 2^k \quad .$$

Furthermore, observe that the expected value of the coefficient p_i under the tail function $T_{i,k}$ is at most $\mu + (\phi n^2 2^k)^{-1}$ because the added exponential tail increases the original mean value μ by at most

$(\phi n)^{-1}(n2^k)^{-1}$. Consequently, applying Lemma 1.7 with $\alpha_k = \phi n 2^k$ and $\mu_k = \mu + (\phi n^2 2^k)^{-1}$ yields $\mathbf{E}[q_k(\mathcal{S})] \leq \alpha_k \mu_k n^2 + 1 = \phi\mu n^3 2^k + n + 1$. □

Now we can complete our calculation for the upper bound on $\mathbf{E}[q]$. Combining Lemma 1.11 and 1.12 yields

$$\begin{aligned}
\mathbf{E}[q] &\leq \sum_{k=0}^{\infty} 2^{-k+4} \min\{\phi\mu n^3 2^k + n + 1, 2^n\} \\
&\leq 16 \sum_{k=0}^{n} \left(\phi\mu n^3 + 2^{-k}(n+1)\right) + 16 \sum_{k=n+1}^{\infty} 2^{n-k} \\
&\leq 16(\phi\mu n^4 + 2n + 3) \ .
\end{aligned}$$

Finally observe that $\phi\mu \geq \frac{1}{2}$, for every non-negative continuous distribution. Thus $\mathbf{E}[q] = O(\phi\mu n^4)$, which completes the proof of Lemma 1.10. □

1.3 A Lower Bound for Non-Increasing Density Functions

In this section we prove a lower bound for the number of Pareto points over the solution set $\mathcal{S} = 2^{[n]}$. Notice, that $2^{[n]}$ is the solution set for the bicriteria version of the knapsack problem. Our bound holds for all continuous distributions with non-increasing density functions. Instead of the explicit subset ordering that we used to prove the upper bounds, we assume two objective functions, the weight function $w(S) = \sum_{i \in S} w_i$ and the profit function $p(S) = \sum_{i \in S} p_i$, defined for all $S \in \mathcal{S}$. Thus, this lower bound holds for the more specific and very common case where the two objective functions are linear. However, the theorem is restricted to min-max and max-min problems, that is, the directions of optimization for weight and profit must be opposite. For a similar bound for min-min and max-max problems one would need to resort to a different solution set, since the solution $S = \emptyset$ (for the min-min case) and $S = [n]$ (for the max-max case) would dominate all other solutions, which would result in a single Pareto point.

Theorem 1.13. *Suppose profits $p_1, \ldots, p_n$ are drawn independently at random according to a continuous probability distribution with non-increasing density function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Assume that the direction of optimization for weight and profit is opposite. Let q denote the number of Pareto points over solution set $2^{[n]}$. There is a vector of weights $w_1, \ldots, w_n$ for which $\mathbf{E}[q] = \Omega(n^2)$.*

Proof. First consider the min-max case, that is, the weight is to be minimized and the profit is to be maximized. If the density function is non-increasing, then the distribution function $F : \mathbb{R}_{\geq 0} \to [0,1]$ is concave as $F' = f$. Furthermore, $F(0) = 0$. Observe that such a function is sub-additive, that is, $F(a+b) \leq F(a) + F(b)$, for every $a, b \geq 0$. This is the crucial property that we need in the following analysis.

Set $w_i = 2^i$ for all $i \in [n]$. For every $j \in [n]$, define $P_j = \sum_{i=1}^{j} p_i$. Let $\mathcal{P}(j-1)$ denote the sequence of Pareto optimal sets over the elements $1, \ldots, j-1$, listed in increasing order of weights. Notice, that all sets in $\mathcal{P}(j-1)$ have weight less than 2^j. All sets containing element j, however, have weight at least 2^j and, therefore, these sets cannot dominate the sets in $\mathcal{P}(j-1)$. Hence, $\mathcal{P}(j)$ contains all sets from

$\mathcal{P}(j-1)$. Furthermore, those sets $S \in \mathcal{P}(j-1)$ with profit $p(S) \in (P_{j-1}-p_j, P_{j-1}]$ create new Pareto optimal sets in $\mathcal{P}(j)$ with profit $p(S)+p_j > P_{j-1}$.

For any given $\alpha > 0$, let X_α^j denote the number of Pareto optimal sets in $\mathcal{P}(j)$ with profit at least $P_j - \alpha$, not counting the last set in this sequence, which is $[j]$. By induction we show $\mathbf{E}\left[X_\alpha^j\right] \geq F(\alpha)j$. Clearly, $\mathbf{E}\left[X_\alpha^1\right] = F(\alpha)$. For $j > 1$, it holds

$$\begin{aligned}
\mathbf{E}\left[X_\alpha^j\right] &= \mathbf{Pr}[p_j \leq \alpha]\left(\mathbf{E}\left[X_{p_j}^{j-1}\right] + \mathbf{E}\left[X_{\alpha-p_j}^{j-1}\right] + 1\right) + \mathbf{Pr}[p_j > \alpha]\,\mathbf{E}\left[X_\alpha^{j-1}\right] \\
&\geq \mathbf{Pr}[p_j \leq \alpha]\,(F(p_j)(j-1) + F(\alpha-p_j)(j-1) + 1) + \mathbf{Pr}[p_j > \alpha]\,F(\alpha)(j-1) \\
&\geq \mathbf{Pr}[p_j \leq \alpha]\,(F(\alpha)(j-1)+1) + \mathbf{Pr}[p_j > \alpha]\,F(\alpha)(j-1) \\
&= F(\alpha)(j-1) + F(\alpha) = F(\alpha)\,j \ ,
\end{aligned}$$

where the last inequality follows from $F(a)+F(b) \geq F(a+b)$. Now let $Y_j = |\mathcal{P}(j)| - |\mathcal{P}(j-1)|$ denote the number of new Pareto optimal sets in $\mathcal{P}(j)$. Observe that $Y_j = X_{p_j}^{j-1} + 1$. The additive 1 is due to the fact that the set $[j-1]$ is not counted in $X_{p_j}^{j-1}$ but yields a new set in $\mathcal{P}(j)$. Since p_j and X_α^{j-1} are independent, we get $\mathbf{E}[Y_j] = \mathbf{E}\left[X_{p_j}^{j-1} + 1\right]$. Furthermore, the number of Pareto optimal sets in $\mathcal{P}(n)$ is $q = 1 + \sum_{j=1}^n Y_j$. The additional 1 is due to the empty set which is always Pareto optimal. Therefore,

$$\mathbf{E}[q] \ = 1 + \sum_{j=1}^n \mathbf{E}[Y_j] \ = \ 1 + \sum_{j=1}^n \mathbf{E}\left[X_{p_j}^{j-1} + 1\right] \geq \ 1 + \sum_{j=1}^n \mathbf{E}[F(p_j)(j-1)+1] \geq \ 1 + \sum_{j=1}^n \mathbf{E}[F(p_j)]\,j\,.$$

In order to evaluate $\mathbf{E}[F(p_j)]$, we need to examine the distribution function of the random variable $F(p_j)$. Let $\mathcal{F}(\cdot)$ denote this distribution function. Recall that p_j is a random variable with distribution function F. Hence,

$$\mathcal{F}(x) = \mathbf{Pr}[F(p_j) \leq x] = \mathbf{Pr}\left[p_j \leq F^{-1}(x)\right] = F(F^{-1}(x)) = x.$$

Thus $F(p_j)$ is uniformly distributed in $[0,1]$ so that $\mathbf{E}[F(p_j)] = \frac{1}{2}$. Hence, $\mathbf{E}[q] \geq 1 + \frac{1}{2}\sum_{j=1}^n j = \Omega(n^2)$. This concludes the proof for the min-max case.

Next we show that the number of Pareto points for the max-min and the max-min case are in fact equal, irrespectively of the problem instance. For every $S \in 2^{[n]}$, let $\bar{S} := [n] \setminus S$ be the complemental subset of S. For simplicity, let us assume that no two subsets of $[n]$ are equal in profit and weight simultaneously. Then a subset $S \subseteq [n]$ is Pareto optimal w.r.t. minimizing weight and maximizing profit if and only of

$$\begin{aligned}
& \nexists T \subseteq [n], T \neq S : w(T) \leq w(S) \text{ and } p(T) \geq p(S) \\
\iff & \nexists T \subseteq [n], T \neq S : w([n]) - w(T) \geq w([n]) - w(S) \text{ and } p([n]) - p(T) \leq p([n]) - p(S) \\
\iff & \nexists T \subseteq [n], T \neq S : w(\bar{T}) \geq w(\bar{S}) \text{ and } p(\bar{T}) \leq p(\bar{S}) \\
\iff & \nexists T \subseteq [n], T \neq \bar{S} : p(T) \leq p(\bar{S}) \text{ and } w(T) \geq w(\bar{S}).
\end{aligned}$$

The last line states that $\bar{S}$ is Pareto optimal w.r.t. maximizing weight and minimizing profits. Therefore, S is Pareto optimal for min-max optimization if and only if $\bar{S}$ is Pareto optimal for max-min optimization.

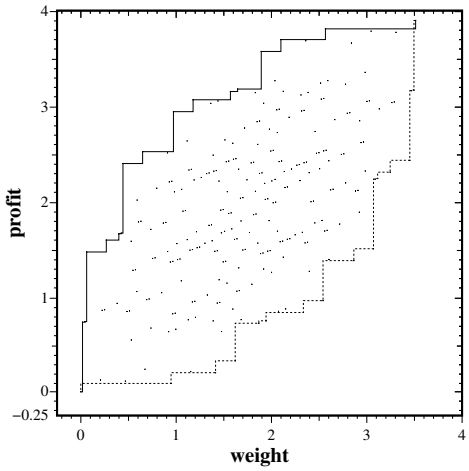

Figure 1.5: Each point represents a subset of 8 elements. Weight and profit of each element have been chosen uniformly at random from the interval $[0,1]$. The point cloud is symmetric under rotation by 180 degree. The two step functions for min-max and max-min optimization exhibit the same number of steps.

The symmetry can also be shown graphically. The situation is illustrated in Figure 1.5. The step function for the min-max case maps each weight value w to the maximum profit over all subsets with weight at most w. Each step correspond to a Pareto point. Accordingly, the step function for the max-min case maps each weight value w to the minimum profit over all subsets with weight at least w. The two step functions are symmetric under a 180 degree rotation and, hence, cover the same number of points. □

The theorem shows that for the bicriteria version of the knapsack problem our analysis of the expected number of Pareto points under the exponential distribution is tight. The same is true for all long-tailed distributions with finite mean and non-increasing density function. For the uniform distribution, however, lower and upper bound deviate by a factor $\Theta(n)$. Experimental results let us believe that the lower bound is tight and the asymptotic behavior for this distribution is $\Theta(n^2)$ as well.

1.4 Extending the Main Theorem to Discrete Distributions

In this section we generalize our main theorem towards discrete probability distributions, that is, we assume that profits are randomly drawn non-negative integers. In this case, we can prove a tradeoff ranging from polynomial to pseudo-polynomial running time, depending on the degree of randomness of the specified instances.

Theorem 1.14. *Let $\mathcal{S} \subseteq 2^{[n]}$ be an arbitrary collection of subsets of $[n]$. Let $w : \mathcal{S} \to \mathbb{R}$ be an arbitrary function and $p(S) = \sum_{i \in S} p_i$ a linear function over $\mathcal{S}$ to be optimized simultaneously. The direction of optimization (minimization or maximization) can be specified arbitrarily for each function. Let q denote the number of Pareto points over $\mathcal{S}$.*

a) For $i \in [n]$, let p_i be a discrete random variable with tail function $T_i : \mathbb{N}_0 \to [0,1]$. Let α be an appropriate positive term satisfying $T_i(t+1)/T_i(t) \geq e^{-\alpha}$ for all $i \in [n]$ and $t \in \mathbb{N}_0$. Suppose $\mu = \max_{i\in[n]}(\mathbf{E}[p_i])$. Then $\mathbf{E}[q] \leq \mu n(1-e^{-\alpha n})+1 \leq \mu\alpha n^2+1$.

b) For every $i \in [n]$, let p_i be a discrete random variable with probability function $f_i : \mathbb{N} \to [0,1]$, i.e., $f_i(t) = \mathbf{Pr}[p_i = t]$. Suppose $\pi = \max_{i\in[n]}(\sup_{x\in\mathbb{N}}(f_i(x)))$ and $\mu = \max_{i\in[n]}(\mathbf{E}[p_i])$. Then $\mathbf{E}[q] = O(\mu n^2(1-e^{-\pi n^2})) = O(\mu\pi n^4)$.

Part (a) of the theorem is the discrete counterpart of the bound for long-tailed distributions. The condition $[\ln(T(x)]' \geq -\alpha$ is replaced by $\frac{T(t+1)}{T(t)} \geq e^{-\alpha}$, or equivalently $\ln(T(t+1)) - \ln(T(t)) \geq -\alpha$, which bounds the decrease of the tail function in the same manner. For the sake of comparison to the continuous counterpart it makes sense to assume that the probability distributions "scale" with the number of elements. For example, consider the following discrete variant of the exponential distribution. Let $T_i(t) = e^{-\alpha t}$ ($t \geq 0$, $i \in [n]$) with $\alpha = \alpha(n)$ being a function in n, e.g., $\alpha(n) = \frac{1}{n}$. The mean μ of this distribution grows like $\frac{1}{\alpha(n)}$. Thus $\mathbf{E}[q] = O(n^2)$ under the discrete exponential distribution, regardless of the choice of α. In other words, we obtain the same bound as in the continuous case.

Part (b) of the theorem is the discrete counterpart of the bound for general distributions. The role of ϕ, the supremum of the probability distributions, is taken up by π, the supremum of the probability functions. In fact, the term $1-e^{-\pi n^2}$ in the upper bound translates into a pseudo-polynomial bound if the randomness in the specification goes to zero. For example, assume that for each element an adversary specifies an interval from which the profit of this item is drawn uniformly at random. Let M denote the maximum profit that can be drawn for any element and ℓ the minimum interval length over all elements. Set $\mu = M$ and $\pi = \frac{1}{\ell}$ so that $\mathbf{E}[q] = O(Mn^2(1-e^{-n^2/\ell}))$. If $\ell = \Theta(M)$ then this upper bound simplifies to $O(n^4)$ because $1-e^{-x} \leq x$, for all $x \in \mathbb{R}$. However, if $\ell = \Theta(1)$, then we are left with the pseudo-polynomial upper bound $\mathbf{E}[q] = O(Mn^2)$.

The proofs rely on the same methods that we used for continuous distributions. So we will point out only those parts of the proofs that need to be adapted. First, we consider long-tailed and then general discrete distributions.

1.4.1 Long-tailed Discrete Distributions

Let us assume that profits are chosen independently at random according to possibly different long-tailed distributions with finite mean. For $i \in [n]$, let profit p_i be a random variable with tail function $T_i : \mathbb{N}_0 \to [0,1]$, that is, for every $t \in \mathbb{N}_0$, $\mathbf{Pr}[p_i \geq t] = T_i(t)$.

Lemma 1.15. *Let α be an appropriate positive term satisfying $T_i(t+1)/T_i(t) \geq e^{-\alpha}$, for all $i \in [n]$ and $t \in \mathbb{N}_0$. Suppose $\mu = \max_{i\in[n]}(\mathbf{E}[p_i])$. Let $S_1,\ldots,S_m$ be an arbitrary but fixed sequence of subsets of $[n]$. Let q denote the number of Pareto optimal sets over $S_1,\ldots,S_m$. Then $\mathbf{E}[q] \leq \mu n(1-e^{-\alpha n})+1 \leq \mu\alpha n^2+1$.*

Proof. We can adapt the proof of Lemma 1.7 towards discrete long tailed distributions.

$$\begin{aligned}\mathbf{E}[\Delta_u \mid \Delta_u \geq 1] &= \mathbf{E}\left[\min_{v\in[u-1]}(p(X_v)-p(Y_v)) \;\middle|\; \min_{v\in[u-1]}(p(X_v)-p(Y_v)) \geq 1\right] \\ &= \sum_{t=1}^{\infty}\mathbf{Pr}[\forall v : p(X_v) \geq p(Y_v)+t \mid \forall v : p(X_v) \geq p(Y_v)+1] \,.\end{aligned}$$

We define the groups $G_1,\ldots,G_k$ as in the proof of Lemma 1.7. Let $E_j(t)$ denote the event $\forall v \in G_j : p(X_v) \geq p(Y_v)+t$. Then

$$\begin{aligned}\mathbf{E}[\Delta_u \mid \Delta_u \geq 1] &= \sum_{t=1}^{\infty}\mathbf{Pr}\left[\bigwedge_{i\in[k]}E_i(t) \;\middle|\; \bigwedge_{i\in[k]}E_i(1)\right] \\ &= \sum_{t=1}^{\infty}\prod_{j=1}^{k}\mathbf{Pr}\left[E_j(t) \;\middle|\; \bigwedge_{i=1}^{j-1}E_i(t)\wedge\bigwedge_{i\in[k]}E_i(1)\right] \\ &\geq \sum_{t=1}^{\infty}\prod_{j\in[k]}e^{-\alpha(t-1)} \geq \sum_{t=1}^{\infty}e^{-\alpha n(t-1)} = \frac{1}{1-e^{-\alpha n}}\,.\end{aligned}$$

We used the fact that $T_j(A_j+t)/T_j(A_j+1) \geq e^{-\alpha(t-1)}$ for all $j \in [k]$, $t \geq 1$ where A_j is defined as in the proof of Lemma 1.7. Notice, that the expected increase in profit at Pareto optimal sets is lower bounded by 1 which reflects the integrality of profits. Analogous to the proof of Lemma 1.5, it holds $p([n]) \geq \sum_{u=2}^{m}\Delta_u$ and

$$\begin{aligned}\mathbf{E}[p([n])] &\geq \sum_{u=2}^{m}\mathbf{E}[\Delta_u] \\ &= \sum_{u=2}^{m}\mathbf{Pr}[\Delta_u \geq 1]\cdot\mathbf{E}[\Delta_u \mid \Delta_u \geq 1] \\ &\geq \sum_{u=2}^{m}\mathbf{Pr}[\Delta_u \geq 1]\cdot\frac{1}{1-e^{-\alpha n}}\,.\end{aligned}$$

Now we can bound the number of Pareto optimal sets by

$$\mathbf{E}[q] = 1+\sum_{u=2}^{m}\mathbf{Pr}[\Delta_u \geq 1] \leq 1+\mathbf{E}[p([n])]\cdot(1-e^{-\alpha n}) \leq \mu n(1-e^{-\alpha n})+1.$$

Applying the inequality $1-e^{-x} \leq x$ yields $\mathbf{E}[q] \leq \mu\alpha n^2+1$. □

1.4.2 General Discrete Distributions

In this section, we analyze the number of Pareto optimal sets when profits are chosen according to general discrete probability distributions over $\mathbb{N}$ with finite mean. For every $i \in [n]$, we assume that p_i is a positive random variable with probability function $f_i : \mathbb{N} \to [0,1]$, i.e., $f_i(t) = \mathbf{Pr}[p_i = t]$.

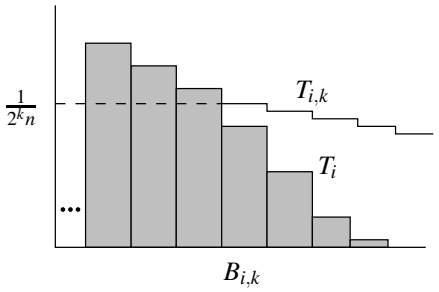

Figure 1.6: We cut the tail T_i at position $B_{i,k}$ and replace this part by a long tail $T_{i,k}$.

Lemma 1.16. *Suppose* $\pi = \max_{i\in[n]}(\sup_{x\in\mathbb{N}}(f_i(x)))$ *and* $\mu = \max_{i\in[n]}(\mathbf{E}[p_i])$. *Let* $S_1,\ldots,S_m$ *be an arbitrary but fixed sequence of subsets of* $[n]$. *Let* q *be the number of Pareto optimal sets over* $S_1,\ldots,S_m$. *Then* $\mathbf{E}[q] = O(\mu n^2(1-e^{-\pi n^2})) = O(\mu\pi n^4)$.

Proof. We adapt the proof of Lemma 1.10. Let $T_i : \mathbb{N} \to [0,1]$ be the tail function of p_i. To simplify the analysis, we introduce auxiliary random variables $x_1,\ldots,x_n$. These variables are drawn independently and uniformly over the interval $[0,1]$. Now we assume that the p_i's are generated from the x_i's by setting $p_i = \max\{j \in \mathbb{N} : T_i(j) \geq x_i\}$. In this way, $\mathbf{Pr}[p_i \geq t] = T_i(t)$, so that this is a proper way to generate the profits in order to obtain the same distribution as described in the lemma. Define auxiliary random variables $q_k(\mathcal{S})$ as before. Lemma 1.11 is valid for discrete probability distributions too. For the convenience of the reader we state it here again.

Lemma 1.17. *For every* $k \geq 0$, $\mathbf{E}[q] \leq \sum_{k=0}^{\infty} 2^{-k+4}\,\mathbf{E}[q_k(\mathcal{S}_k)]$, *for some* $\mathcal{S}_k \subseteq 2^{[n]}$.

Next we prove the discrete counterpart of Lemma 1.12.

Lemma 1.18. *For* $k \geq 0$, $\mathbf{E}[q_k] \leq \min\{\mu n(1-e^{-\pi 2^k n^2}) + n + 1, 2^n\}$.

Proof. The bound $\mathbf{E}[q_k] \leq 2^n$ holds trivially because there are at most 2^n different subsets over n elements. The other bound can be shown with the help of Lemma 1.15. For this purpose we need to bound the ratio of $T_{i,k}(t+1)/T_{i,k}(t)$ for all $t \in \mathbb{N}$. Define $B_{i,k} = \min\{j \in \mathbb{N} : T_i(j) < 2^{-k}/n\}$. Consider the following variants of our tail function. We cut the tail functions T_i at position $B_{i,k}$ and replace the original, possibly short tails by long, exponential tails. For $i \in [n]$ and $k \geq 0$, define

$$T_{i,k}(t) \;=\; \begin{cases} T_i(t) & \text{if } t < B_{i,k}, \\ \exp(-\pi n(t-B_{i,k}))/(n2^k) & \text{if } t \geq B_{i,k}. \end{cases}$$

Figure 1.6 illustrates the situation. For all $t \leq B_{i,k}-1$, $T_{i,k}(t+1) \geq 2^{-k}/n$ and $T_{i,k}(t) - T_{i,k}(t+1) \leq \pi$. In particular, $T_{i,k}(B_{i,k}-1) - T_{i,k}(B_{i,k}) \leq \pi$, because $T_{i,k}(B_{i,k}) \geq T_i(B_{i,k})$. Therefore $T_{i,k}(t) \leq T_{i,k}(t+1) + 2^k n \cdot T_{i,k}(t+1) \cdot \pi$ and

$$\frac{T_{i,k}(t+1)}{T_{i,k}(t)} \;\geq\; \frac{1}{1+2^k n\pi} \;\geq\; e^{-2^k n\pi}\;.$$

The same lower bound holds also for $t \geq B_{i,k}$ since, in this case,

$$\frac{T_{i,k}(t+1)}{T_{i,k}(t)} \;=\; e^{-n\pi} \geq e^{-2^k n\pi} \quad .$$

Furthermore, observe that the expected maximum profit of an element under the tail function $T_{i,k}$ is at most $\mu + 1/(2^k n(1-e^{-\pi n}))$, because the added exponential tail increases the original mean value μ by at most

$$\max_i \sum_{t=B_{i,k}}^{\infty} \frac{1}{2^k n} e^{-\pi n(t-B_{i,k})} \;=\; \frac{1}{2^k n} \cdot \frac{1}{(1-e^{-\pi n})} \quad .$$

Applying Lemma 1.15 with parameters $\mu_k = \mu + 1/(2^k n\,(1-e^{-\pi n}))$ and $\alpha_k = \pi n 2^k$ yields

$$\begin{aligned}
\mathbf{E}[q_k] &\leq \mu_k n(1-e^{-\alpha_k n}) + 1 \\
&= \mu n(1-e^{-\pi n^2 2^k}) + \frac{1-e^{-\pi n^2 2^k}}{2^k(1-e^{-\pi n})} + 1 \\
&\leq \mu n(1-e^{-\pi n^2 2^k}) + n + 1 \quad .
\end{aligned}$$

Thus, Lemma 1.18 is shown. □

Now we can complete our calculation for the upper bound on $\mathbf{E}[q]$. Combining Lemma 1.18 and 1.17 yields

$$\begin{aligned}
\mathbf{E}[q] &\leq \sum_{k=0}^{\infty} 2^{-k+4} \min\left\{\mu n(1-e^{-\pi n^2 2^k}) + n + 1, 2^n\right\} \\
&\leq 16\left(\sum_{k=0}^{n} \mu n \frac{(1-e^{-\pi n^2 2^k})}{2^k} + \frac{n+1}{2^k}\right) + 16 \sum_{k=n+1}^{\infty} 2^{n-k} \\
&\leq 16(\mu n^2(1-e^{-\pi n^2}) + 2n + 3) \quad .
\end{aligned}$$

Finally, we need to show $2n+3 = O(\mu n^2(1-e^{-\pi n^2}))$. As the domain of the considered distributions are the positive integers, $\mu \geq 1$ and $\pi\mu \geq \frac{1}{2}$ gives $(1-e^{-\pi n^2}) \geq (1-e^{-n^2/2\mu})$. For all $0 \leq x \leq 0.5$ and $n \geq 2$ it holds $x + e^{-n^3 x/2} \leq 1$. Substituting $x := 1/(\mu n)$ yields $1 \leq \mu n(1-e^{-n^2/2\mu})$. Thus Lemma 1.16 is shown. □

1.5 Algorithmic Applications

Our main theorem shows that the number of Pareto points for bicriteria optimization problems is polynomial on average, provided that one function is linear and exhibits sufficient randomness. However, even if there are only few Pareto points it can be hard to compute them. As an example, consider the bicriteria Traveling Salesperson problem where already the problem of finding any solution, i.e., the Hamiltonian cycle problem, is NP-hard. The same argument applies to all problems for which the single criterion version of the problem is hard to solve in the worst case. But even if the single criterion version of the

problem is in P and well studied, computing the Pareto points can be non-trivial. As an example, consider the minimum spanning tree problem with two linear objective functions to be minimized. To our knowledge, there is no efficient algorithm enumerating all Pareto points for this problem even though this problem has been treated in several publications. While Corley's algorithm [Cor85] is flawed as observed by others (e.g. [EG02]), Hamacher and Ruhe [HR94] give an approximation algorithm for the Pareto curve which produces also solutions that are not Pareto optimal. Ramos et al. [RASG98] provide an algorithm computing all Pareto optimal solutions, however, the running time is not polynomial in m and q, denoting the number of edges and the number of Pareto points, respectively.

We apply our main theorem to two problems; the knapsack problem and the constrained shortest path problem [Zie01]. For both problems exist efficient algorithms enumerating Pareto optimal solutions. As a result we obtain the first average-case analysis for these NP-hard problems. Since Chapter 2 is completely devoted to the knapsack problem we find it convenient to move the material concerning the knapsack problem together with experimental results to this chapter. In the following section we only address the constrained shortest path problem.

1.5.1 The Constrained Shortest Path Problem

Given a graph $G = (V,E)$ with source node s, target node t and a length function $l : E \to \mathbb{R}$, the shortest path problem ask for a shortest path in G from s to t. When we define an additional cost function $c : E \to \mathbb{R}$ on the set of edges, we obtain a bicriteria optimization problem. Usually, both functions are to be minimized. In the constrained shortest path problem we seek the shortest among all path with cost at most a given threshold B. This problem is NP-hard [GJ79] but admits a pseudo-polynomial time algorithm [Jok66, Law76] with running time $O(|E|B)$. Using the standard technique of rounding and scaling, this algorithm can be turned into an FPTAS [Has92]. For more recent literature on this problem we refer to the PhD-theses of Ziegelmann [Zie01] and Dumitrescu [Dum02]. Starting with the pioneering work of Hansen [Han80] the bicriteria shortest path problem has drawn much attention. A comprehensive survey of available literature on this topic has been compiled by Ehrgott and Gandibleux [EG02].

Next we describe a dynamic programming algorithm computing all Pareto points for the bicriteria shortest path problem. A variation of this algorithm was introduced in [CM85]. The algorithm is an extension of the Bellmann/Ford algorithm that solves the single criterion version of the shortest path problem. As all labeling approaches, it maintains a distance label for each vertex corresponding to the length of the shortest path found so far. For the bicriteria variant we maintain, for each vertex $v \in V$, a list $\mathcal{L}_v$ of Pareto optimal $s-v$ paths. The paths in each list are sorted according to the first objective. In the initialization phase each vertex is assigned the empty list, except the source s, whose list contains the zero length path. The operation $relax(u,v)$ extends list $\mathcal{L}_v$ by new Pareto optimal $s-v$ paths (with last edge (u,v)) as follows. Each path in $\mathcal{L}_u$ is extended by edge (u,v) and inserted into $\mathcal{L}_v$. Then paths dominated by other paths in $\mathcal{L}_v$ are removed from the list.

In fact, we maintain only the Pareto points instead of the corresponding paths in the lists $\mathcal{L}_v$. If the optimal path needs to be extracted at the end of the computation, we can maintain additionally a link

to the last vertex on the corresponding path. As the lists are sorted according to the first objective, the operation $relax(u,v)$ takes time that is linear in the length of the lists $\mathcal{L}_u$ and $\mathcal{L}_v$. The sequence of relax operations is implicitly given by the Bellmann/Ford algorithm.

Algorithm Bellman/Ford $G=(V,E)$
Initialize-Labels;
repeat $\|V\|-1$ times
forall $(u,v)\in E$ **do** relax(u,v);

Let $(R_1,\ldots,R_k)$, with $R_i \in E$ for all $i \in [k]$, denote the ordered sequence of relax operations performed by the (extended) Bellman/Ford algorithm. For any $u \in V$ and $i \in [k]$, let $\mathcal{P}_i(u)$ be the set of $s-u$ paths induced by the first i relax operation, i.e., all $s-u$ paths that appear as a subsequence in $R_1,\ldots,R_i$. In other words, $\mathcal{P}_i(u)$ contains exactly those $s-u$ paths that can potentially be discovered after the first i relax operations. Let $\mathcal{L}_u^i$ denote the computed list of Pareto optimal paths for vertex u after the ith iteration. The list $\mathcal{L}_u^i$ contains exactly the Pareto optimal paths over $\mathcal{P}_i(u)$. The overall running time T of the algorithm is

$$T = \sum_{i=1}^{k} |\mathcal{L}^{i-1}_{\text{source}(R_i)}| + |\mathcal{L}^{i-1}_{\text{target}(R_i)}|,$$

where source(R) and target(R) denote the source and target vertex of the relax operation R. Applying Theorem 1.2 on solution set $\mathcal{P}_i(u)$, we can upper bound $\mathbf{E}\left[|\mathcal{L}_u^i|\right]$ for all $i \in [k]$ and $u \in V$. Let U be an appropriate term upper bounding $\mathbf{E}\left[|\mathcal{L}_u^i|\right]$ for all $i \in [k]$ and $u \in V$. Then $\mathbf{E}[T] = O(nmU)$. Hence, Theorem 1.2 and 1.14 yield

Theorem 1.19. *Let $G=(V,E)$ be an arbitrary graph with $m=|E|$ and $n=|V|$ and two linear functions $l(\cdot)$ and $c(\cdot)$ to be minimized. The coefficients $c_1,\ldots,c_m$ of function $c(\cdot)$ are assumed to be adversarial, whereas the coefficients $l_1,\ldots,l_m$ are assumed to be random variables. Let T denote the running time of the Bellman/Ford algorithm with the extended relax operation.*

a) If $l_1,\ldots,l_n$ are chosen according to the uniform distribution over $[0,1]$, then $E[T] = O(nm^4)$.

b) For $i \in [n]$, let l_i be a random variable with tail function $T_i : \mathbb{R}_{\geq 0} \to [0,1]$. Define $\mu_i = \mathbf{E}[l_i]$ and let α_i be an appropriate positive real number satisfying $\text{slope}_{T_i}(x) \leq \alpha_i$ for every $x \geq 0$, $i \in [m]$. Let $\alpha = \max_{i\in[m]} \alpha_i$ and $\mu = \max_{i\in[m]} \mu_i$. Then

$$\mathbf{E}[T] \leq nm\left(\left(\sum_{i\in[m]} \mu_i\right)\cdot\left(\sum_{i\in[m]} \alpha_i\right)+1\right) \leq \alpha\mu nm^3 + mn.$$

c) For every $i \in [m]$, let l_i be a non-negative random variable with density function $f_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. Suppose $\mu = \max_{i\in[m]} (\mathbf{E}[l_i])$ and $\phi = \max_{i\in[m]} \left(\sup_{x\in\mathbb{R}_{\geq 0}} f_i(x)\right)$. Then $\mathbf{E}[q] = O(\phi\mu nm^5)$.

d) For $i \in [m]$, let l_i be a discrete random variable with tail function $T_i : \mathbb{N}_0 \to [0,1]$. Let α be an appropriate positive term satisfying $T_i(t+1)/T_i(t) \geq e^{-\alpha}$ for all $i \in [m]$ and $t \in \mathbb{N}_0$. Suppose $\mu = \max_{i\in[m]}(\mathbf{E}[l_i])$. Then $\mathbf{E}[T] \leq nm(\mu m(1-e^{-\alpha m})+1) \leq nm(\mu\alpha m^2+1)$.

e) For every $i \in [m]$, let l_i be a discrete random variable with probability function $f_i : \mathbb{N} \to [0,1]$, i.e., $f_i(t) = \mathbf{Pr}[l_i = t]$. Suppose $\pi = \max_{i\in[m]}(\sup_{x\in\mathbb{N}}(f_i(x)))$ and $\mu = \max_{i\in[m]}(\mathbf{E}[l_i])$. Then $\mathbf{E}[T] = O(\mu nm^3(1-e^{-\pi m^2})) = O(\mu\pi nm^5)$.

Remark 1.20. *Part (a), (b) and (c) of the theorem apply to continuous distributions whereas part (d) and (e) apply to discrete distributions. The theorem is valid as well for a deterministic length function $l(\cdot)$ and random edge costs $c_1, \ldots, c_m$.*

Corollary 1.3 immediately implies the following smoothed analysis result.

Corollary 1.21. *Let $G = (V,E)$ be an arbitrary graph with $m = |E|$ and $n = |V|$ and two linear functions $l(\cdot)$ and $c(\cdot)$ to be minimized. Assume we either perturb the coefficients of the cost or of the length function according to our smoothed analysis framework. For any fixed perturbation model f with non-negative domain the expected running time of the Bellman/Ford algorithm with the extended relax operation on the perturbed instance is $O(nm^5(1+\phi))$.*

There are other labeling approaches for the shortest path problem that have been extended to compute Pareto optimal paths. They are usually more efficient in practice. All these algorithms use the relax operation described above, but differ in the sequence of executed relax operations. In particular, the choice of the next relax operation usually depends on the current labeling which results not only from the previous relax operations but also from the length and cost functions. Therefore, the paths in $\mathcal{P}_i(u)$ depend on the random coefficients of the length function and we cannot apply our main theorem, as it requires a fixed solution set. In contrast, the sequence of relax operations in the (extended) Bellmann/Ford algorithm is fixed and so are the paths in $\mathcal{P}_i(u)$.

Chapter 2

Probabilistic Analysis of Algorithms for the Knapsack Problem

In this chapter we apply our main theorem to the knapsack problem leading to the first average-case analysis that proves a polynomial upper bound on the expected running time of an exact algorithm for the standard 0/1 knapsack problem. In particular, we consider the bicriteria version of the knapsack problem and describe a long-known algorithm by Nemhauser and Ullmann that can enumerate Pareto optimal knapsack fillings very efficiently. This way, the generality of the random input model provided by our main theorem transfers directly to the knapsack problem so that we obtain a smoothed complexity result as well. Our analysis confirms and explains practical studies showing that so-called *strongly correlated* instances are harder to solve than *weakly correlated* ones.

In the second part of this chapter, we study the average-case performance of knapsack core algorithms, the predominant algorithmic concept used in practice. We present the first theoretical result on the running time of a core algorithm that comes close to the results observed in experiments. In particular, we prove an upper bound of $O(n\,\mathrm{polylog}\,(n))$ on the expected running time of a core algorithm on instances with n items whose profits and weights are drawn independently, uniformly at random. Furthermore we extend our result to harder instances, where weights and profits are correlated.

Finally we present an experimental study for random knapsack instances. At first we investigate the number of Pareto points under various input distributions. Our experiments suggest that the theoretically proven upper bound of $O(n^3)$ for uniform instances and $O(n^4\phi)$ for general probability distributions is not tight. Instead we conjecture an upper bound of $O(n^2\phi)$ matching the lower bound from Section 1.3. Furthermore, we investigate the performance of different knapsack implementations on random instances. We combine four concepts (core, domination, loss and decomposition) that have been known since at least 20 years, but apparently have not been used together. The result is a very competitive code that outperforms the best known implementation *Combo* by orders of magnitude also for harder random knapsack instances.

2.1 Definition and Previous Work

The 0/1 knapsack problem is one of the most intensively studied combinatorial optimization problems having a wide range of applications in industry and financial management, e.g., cargo loading, cutting stock, budget control. For a comprehensive treatment we refer to the new book by Kellerer, Pferschy and Pisinger [KPP04] which is entirely devoted to the knapsack problem and its variations. The problem is defined as follows. A subset of n given items has to be packed in a knapsack of *capacity* b. Each item i has a *profit* p_i and a *weight* w_i. The problem is to select a subset of the items whose total weight does not exceed the capacity bound b and whose total profit is as large as possible. In terms of an integer linear program (ILP), the problem is to

$$\begin{aligned} \text{maximize} \quad & \sum_{i\in[n]} p_i x_i \\ \text{subject to} \quad & \sum_{i\in[n]} w_i x_i \leq b \\ \text{and} \quad & x_i \in \{0,1\}, \text{ for } i \in [n]\,. \end{aligned}$$

This problem is of special interest not only from a practical point of view but also for theoretical reasons as it can be seen as the simplest possible 0/1 linear program because the set of feasible solutions is described by a single constraint only. Starting with the pioneering work of Dantzig [Dan57], the problem has been studied extensively in practice (e.g. [MPT99, MPT00, Pis95, KPP04, MT90]) as well as in theory (e.g. [BZ80, BB94, MT97, HS74, Lue82, Lue98a, GMS84, DF89, BV03, BV04b]).

The knapsack problem is one of those optimization problems for which NP-hardness theory concludes that it is hard to solve in the worst case. In the book by Garey and Johnson [GJ79] the knapsack problem is listed under number MP9. The knapsack problem is NP-hard in the weak sense. There exist a pseudo-polynomial time algorithm with running time $O(nb)$ (see e.g. [KPP04]). The first FPTAS for the knapsack problem, published in 1975 by Ibarra and Kim [IK75], has subsequently been improved by Lawler [Law79], Magazine and Oguz[MO81] and Kellerer and Pferschy [KP99].

Despite the exponential worst-case running times of all known knapsack algorithms, several large scale instances can be solved to optimality very efficiently [MPT00, KPP04, Pis95]. In particular, randomly generated instances seem to be quite easy to solve. A natural and also the most investigated stochastic input model assumes that the weights and profits of all items are chosen independently uniformly at random from $[0,1]$. Although significant progress has been made in understanding the structure and complexity of these random knapsack instances (see e.g. [GMS84, Lue82, Lue98a, BB94, DF89][1]), until now there has been no proof that the knapsack problem can be solved in expected polynomial time under any reasonable stochastic input model. The best known result on the running time of an exact algorithm for the knapsack problem was shown by Goldberg and Marchetti-Spaccamela [GMS84]. They investigate *core algorithms*, an algorithmic concept suggested by Balas and Zemel [BZ80]. As a first step towards analyzing core algorithms, Lueker proved an upper bound on the expected gap between

[1] Related work deals with the hardness of random instances for knapsack cryptosystems [DP82, Fri86, IN96]. These cryptosystems, however, are based on the hardness of random subset sum problems which is not directly affected by our results.

the optimal integral and the optimal fractional solution [Lue82]. Based on this result, Goldberg and Marchetti-Spaccamela [GMS84] were able to prove structural properties of the core resulting in the following bound on the running time of a Las Vegas type core algorithm. For every fixed $k > 0$, with probability at least $1 - 1/k$, the running time of their algorithm does not exceed a specified upper bound that is polynomial in the number of items. However, the degree of this polynomial is quite large, the leading constant in the exponent is at least a large three digit number. Even more dramatically, the degree of the polynomial grows with the reciprocal of the failure probability like $\sqrt{k}\log(k)$. Observe that this kind of tail bounds does not allow to conclude any sub-exponential upper bound on the expected running time. An extension of this work to the multidimensional knapsack problem has been presented by Dyer and Frieze [DF89].

Our analysis is based on the Nemhauser/Ullmann algorithm which enumerates Pareto optimal knapsack fillings. By applying our main theorem from Chapter 1 we are able to prove a polynomial upper bound on the expected running time of the Nemhauser/Ullmann algorithm for random knapsack instances. We improve the result of Goldberg and Marchetti-Spaccamela in several other aspects as well. In particular, our random input model is not restricted to the uniform distribution but allows arbitrary probability distributions with bounded density and finite mean. Our analysis relies on the randomness of only one objective function (weight or profit). Hence, the other objective function is assumed to be adversarial. Furthermore, different coefficients of the weight function (or profit function) can follow different probability distributions, which implies a smoothed analysis result. In contrast to the algorithm analyzed by Goldberg and Marchetti-Spaccamela, the Nemhauser/Ullmann algorithm does not use the core concept. In fact, Section 2.5.4 shows that the Nemhauser/Ullmann algorithm is significantly slower in practice than the core algorithm by Goldberg and Marchetti-Spaccamela. However, the generality of our analysis allows us to bound the running time of a different core algorithm which is based on the technique of Nemhauser/Ullmann.

2.2 The Nemhauser/Ullmann Algorithm

In bicriteria version of the standard knapsack problem the weight is considered as a second objective function which is to be minimized. The set of feasible solutions consists of all 2^n subsets of n elements. In 1969, Nemhauser and Ullmann [NU69] introduce an elegant algorithm enumerating all Pareto points for this problem. Such an enumeration also solves the knapsack problem as an optimal solution for the corresponding knapsack problem with weight threshold b can be obtained by choosing the most profitable Pareto optimal knapsack filling with weight at most b. In the literature the Nemhauser/Ullmann algorithm is sometimes referred as "Dynamic programming with lists", e.g. in [KPP04].

The algorithm computes a list of all Pareto points in an iterative manner. For $i \in [n]$, let $\mathcal{L}_i$ be the list of all Pareto points over the solution set $\mathcal{S}_i = 2^{[i]}$, i.e., the solution set $\mathcal{S}_i$ consists of all subsets of items $1,\ldots,i$. Each Pareto point is a (*weight, profit*) pair. The Pareto points in $\mathcal{L}_i$ are assumed to be listed in increasing order of their weights. Clearly, $\mathcal{L}_1 = \langle (0,0),(w_1,p_1)\rangle$. The list $\mathcal{L}_{i+1}$ can be computed from $\mathcal{L}_i$ as follows: Create a new ordered list $\mathcal{L}_i'$ by duplicating $\mathcal{L}_i$ and adding (w_{i+1},p_{i+1}) to each point. Now we merge the two lists into $\mathcal{L}_{i+1}$ obeying the weight order of subsets. Finally, those subsets are

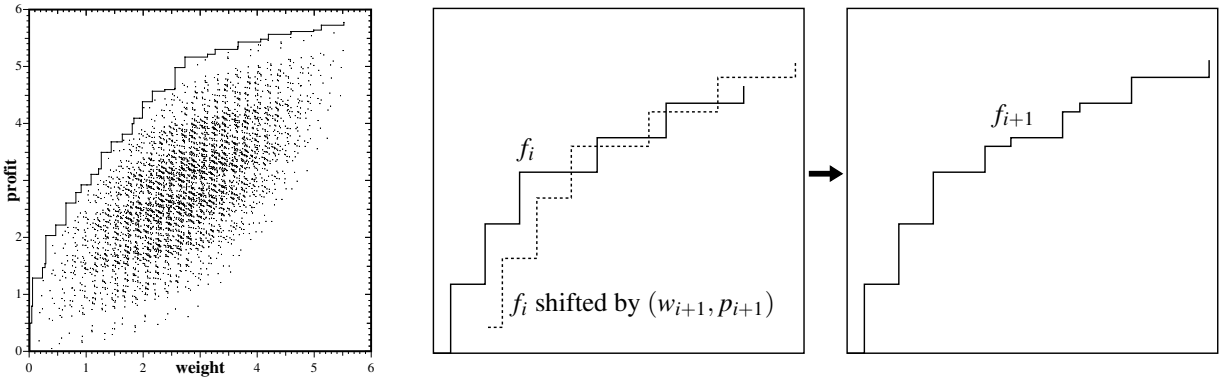

Figure 2.1: (a) Points correspond to subsets of 12 items. Weight and profit of each item have been chosen uniformly at random from $[0,1]$. Each step of function f corresponds to a Pareto point. (b) An iteration of the Nemhauser/Ullmann algorithm: a copy of step function f_{i-1} is shifted by (w_i, p_i). (c) Upper envelope gives step function f_i.

removed from the list that are dominated by other subsets in the list. The correctness of the algorithm is a consequence of the fact, that $\mathcal{L}_i$ contains all Pareto points over $\mathcal{S}_i$ and $\mathcal{L}'_i$ contains all Pareto points over $\mathcal{S}_{i+1} \setminus \mathcal{S}_i$. This way, $\mathcal{L}_i \cup \mathcal{L}'_i$ contains all Pareto points over $\mathcal{S}_{i+1}$. In order to extract the knapsack fillings corresponding to the computed Pareto points, we maintain a bit for each Pareto point in $\mathcal{L}_i$ indicating whether or not item i is contained in the corresponding knapsack filling. The knapsack filling can then reconstructed recursively using the standard technique from dynamic programming (see e.g. [KPP04]).

For the purpose of illustration and a better understanding, let us take a different view on this algorithm. For $i \in [n]$, let $f_i : \mathbb{R} \to \mathbb{R}$ be a mapping from weights to profits such that $f_i(t)$ is the maximum profit of a knapsack filling over items $1, \ldots, i$ with weight at most t. Observe that f_i is a non-decreasing step function changing its value at those weights that correspond to Pareto optimal knapsack fillings. In particular, the number of steps in f_i equals the number of Pareto points over $\mathcal{S}_i$. Figure 2.1(a) shows such a step function for a small instance, (b) shows a step function f_i and the copy of this step function which is generated in the algorithm described above, finally, (c) illustrates how the two step functions are merged: the graph of the resulting step function f_{i+1} is simply the upper envelope of the graph f_{i-1} and its shifted copy.

As the lists are sorted, $\mathcal{L}_i$ can be calculated from $\mathcal{L}_{i-1}$ in time that is linear in the length of $\mathcal{L}_{i-1}$, that is, linear in the number of Pareto points over $\mathcal{S}_{i-1}$. This yields the following lemma:

Lemma 2.1. *For every $i \in [n]$, let $q(i)$ denote an upper bound on the (expected) number of Pareto points over $\mathcal{S}_i$, and assume $q(i)$ is increasing. The Nemhauser/Ullmann algorithm computes an optimal knapsack filling in (expected) time*

$$O\left(\sum_{i=1}^{n} q(i-1)\right) = O(n \cdot q(n)).$$

1. group $i = 1,\ldots,l$	$w_i = p_i = 2^{i-1}$	$\lvert\mathcal{L}_i\rvert = 2^i$
2. group $i = 1,\ldots,k$	$w_{l+i} = \frac{i}{2(k+1)}$, $p_{l+i} = \frac{1}{2} + \frac{i}{2(k+1)}$	$\lvert\mathcal{L}_{l+i}\rvert = (i+1)2^l + \frac{i(i-1)}{2}$
3. group	$w_{k+l+1} = p_{k+l+1} = \frac{1}{2}$	$\lvert\mathcal{L}_{l+k+1}\rvert = 2^{l+1} + \frac{k(k+1)}{2}$

Table 2.1: Definition of knapsack instances $I(l,k)$. Items are divided into three groups containing l,k and one item respectively. Weight and profit of item i are denoted by w_i and p_i. The last column gives the number of Pareto points after each iteration of the Nemhauser/Ullmann algorithm.

The assumption on the monotonicity of $q(i)$ is required since the number of Pareto points can decrease when adding an item. In fact, there is a family of instances showing that $|\mathcal{L}_i|/|\mathcal{L}_{i-1}|$ can be arbitrarily small (see Section 2.2.1). We find it adequate to elaborate on this observation as the standard literature (e.g. [KPP04]) does not mention this fact. All bounds we prove, however, are monotonically increasing in n.

In the worst case, the number of Pareto points is 2^n. This case occurs when profits and weights are identical and no two subset have the same weight. In fact, this instance is also the worst case for the core algorithm of Goldberg and Marchetti-Spaccamela. In general, the running time of the Nemhauser/Ullmann algorithm can be bounded from above by $O(\sum_{i=1}^{n} 2^{i-1}) = O(2^n)$. However, if weights and profits are independent or only weakly correlated, experiments show that the number of Pareto points and, hence, the running time, is much smaller (see Section 2.5). Furthermore, if the input numbers are positive integers, then the running time of the algorithm can be bounded pseudo-polynomially as, in this case, step function f_i can have at most iP steps, with P denoting the maximum profit of an individual item. Notice the close connection to dynamic programming. Instead of explicitly evaluating the maximum profit of a knapsack filling at every integer weight, the Nemhauser/Ullmann algorithm keeps track only of those weights at which the maximum profit increases. Thus, it can be seen as a sparse dynamic programming approach.

2.2.1 Decreasing Number of Pareto Points

The following family of instances shows that the number of Pareto points can decrease when adding an item. In particular, $|\mathcal{L}_n|/|\mathcal{L}_{n-1}|$ can be arbitrary small when n goes to infinity. The instances are parameterized by two integers l and k satisfying $2^l \geq k$. Instance $I(l,k)$ has $n = l+k+1$ items, divided into groups, l items in the first group, k in the second group and one item in the last group. The weights and profits of each item are specified in Table 2.2.1. The last column of the table gives the number of Pareto points after each iteration of the Nemhauser/Ullman algorithm, provided that items are processed in the given order. Instead of giving a mathematical proof for these formulas we rather explain the basic idea of the construction. Items in the first group have a profit-to-weight ratio of 1 and their weight and profit grows exponentially. Hence, all subsets of those items are Pareto optimal. After the items of the first group have been processed by the Nemhauser/Ullmann algorithm, the set of Pareto points is

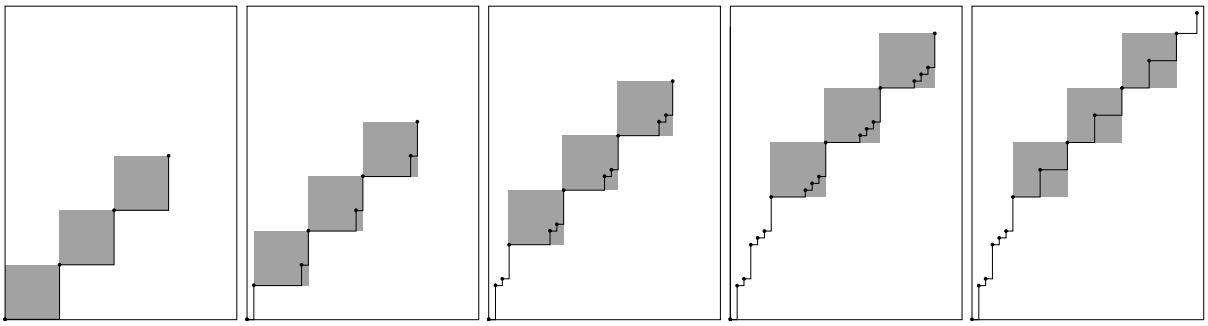

Figure 2.2: Pareto points for knapsack instance $I(2,3)$. The five pictures show $\mathcal{L}_2,\ldots,\mathcal{L}_6$ as step functions. In the last iteration the k small steps in each shaded square are replaced by one larger step reducing the number of Pareto points from 19 to 14 in our example.

$\{(m,m) \mid m = 0,1,\ldots,2^l-1\}$. The corresponding step function exhibits 2^l-1 equally sized steps which we call *large steps* (shaded region in Figure 2.2). Each of the items in the second group will create a small step on each large step. The last item destroys all these small steps. The k small steps on each large step are replaced by a new Pareto point. Using the formulas from Table 2.2.1, in the last iteration the number of Pareto points is reduced by factor

$$\frac{2^{l+1}+\frac{k(k+1)}{2}}{(k+1)2^l+\frac{k(k-1)}{2}},$$

which can become arbitrary large for appropriate values for k and l.

2.2.2 Smoothed/Average-case Analysis

In this section we use our main theorem from Chapter 1 to bound the number of Pareto optimal knapsack fillings enumerated by the Nemhauser/Ullmann algorithm. This way, we obtain a bound on the expected running time of the algorithm as well. As for the constrained shortest path problem, the generality of Theorem 1.2 transfers completely to the running time analysis of the Nemhauser/Ullmann algorithm. In particular, we are able to bound the expected running time under our smoothed analysis framework. Recall, that our main theorem requires only one linear objective function with random coefficients. For simplicity, we will state the following theorem using adversarial weights and random profits even though it hold for an opposite setting as well. Combining Lemma 2.1 and Theorems 1.2 and 1.14 we obtain

Theorem 2.2. *Suppose the weights are arbitrary positive numbers and profits $p_1\ldots,p_n$ are assumed to be independent random variables. Let T denote the running time of the Nemhauser/Ullmann algorithm.*

a) If $p_1,\ldots,p_n$ are chosen according to the uniform distribution over $[0,1]$, then $E[T] = O(n^4)$.

b) For $i \in [n]$, let p_i be a random variable with tail function $T_i : \mathbb{R}_{\geq 0} \to [0,1]$. Define $\mu_i = \mathbf{E}[p_i]$ and let α_i be an appropriate positive real number satisfying $\operatorname{slope}_{T_i}(x) \leq \alpha_i$ for every $x \geq 0$, $i \in [n]$. Let

$\alpha = \max_{i\in[n]} \alpha_i$ *and* $\mu = \max_{i\in[n]} \mu_i$. *Then*

$$\mathbf{E}[T] = O\left(n\left(\sum_{i\in[n]} \mu_i\right)\cdot\left(\sum_{i\in[n]} \alpha_i\right)\right) = O(\alpha\mu n^3).$$

c) *For every* $i \in [n]$, *let* p_i *be a non-negative random variable with density function* $f_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. *Suppose* $\mu = \max_{i\in[n]} (\mathbf{E}[p_i])$ *and* $\phi = \max_{i\in[n]} \left(\sup_{x\in\mathbb{R}_{\geq 0}} f_i(x)\right)$. *Then* $\mathbf{E}[T] = O(\phi\mu n^5)$.

d) *For* $i \in [n]$, *let* p_i *be a discrete random variable with tail function* $T_i : \mathbb{N}_0 \to [0,1]$. *Let* α *be an appropriate positive term satisfying* $T_i(t+1)/T_i(t) \geq e^{-\alpha}$ *for all* $i \in [n]$ *and* $t \in \mathbb{N}_0$. *Suppose* $\mu = \max_{i\in[n]} (\mathbf{E}[p_i])$. *Then* $\mathbf{E}[T] = O(\mu n^2(1-e^{-\alpha n})) = O(\mu\alpha n^3)$.

e) *For every* $i \in [n]$, *let* p_i *be a discrete random variable with probability function* $f_i : \mathbb{N} \to [0,1]$, *i.e.,* $f_i(t) = \mathbf{Pr}[p_i = t]$. *Suppose* $\pi = \max_{i\in[n]} (\sup_{x\in\mathbb{N}} (f_i(x)))$ *and* $\mu = \max_{i\in[n]} (\mathbf{E}[p_i])$. *Then* $\mathbf{E}[T] = O(\mu n^3(1-e^{-\pi n^2})) = O(\mu\pi n^5)$.

All bound hold for adversarial profits and random weights as well.

Part (a), (b) and (c) of the theorem apply to continuous probability distributions. They are implied by Theorem 1.2. Part (d) and (e) are the discrete counterparts of (b) and (c). They are a consequence of Theorem 1.14. In Section 2.5.2 we present an experimental study of the number of Pareto points for the random knapsack instances. The experiments indicate that the bound for the uniform distribution (part (a) of the theorem) is not tight. Instead we conjecture that the expected number of Pareto points is $\Theta(n^2)$. This would imply that the expected running time of the Nemhauser/Ullmann algorithm is $\Theta(n^3)$. Experiments on correlated instances let us believe that the bounds for general continuous and discrete probability distributions (part (c) and (e) of the theorem) can be improved by a factor of n^2. We argued already in Section 1.2 that μ and ϕ must have the same exponent in a consistent bound as scaling all coefficients (resp. the corresponding distributions) does not change the (expected) number of Pareto points but would change the product $\mu^a\phi^b$ for $a \neq b$. So we conjecture that $O(\phi\mu n^3)$ is a tight bound for part (c) of the theorem.

The following corollary is an immediate consequence of Lemma 2.1 and Corollary 1.3.

Corollary 2.3. *Assume we either perturb profits or weights according to our smoothed analysis framework. For any fixed perturbation model* f *with non-negative domain the expected running time of the Nemhauser/Ullmann algorithm on the perturbed instance is* $O(n^5(\phi+1))$.

These results enable us to study the effects of correlations between profits and weights. (This aspect has also been considered in several recent practical studies [MPT00, KPP04, Pis95].) We observe that if the adversary chooses profits equal to weights, then the unperturbed instance is completely correlated and exhibits 2^n Pareto points. Now perturbing the profits decreases the correlation. In particular, a small correlation corresponds to a small ϕ value and this in turn implies a small upper bound on the expected number of Pareto points. In other words, the complexity of the problem diminishes when decreasing the correlation between profits and weights.

2.3 Analysis of Knapsack Core Algorithms

Equipped with a very robust probabilistic analysis for the Nemhauser/Ullmann algorithm, we aim in this section at analyzing more advanced algorithmic techniques for the knapsack problem. Our focus lies on the analysis of *core algorithms*, the predominant algorithmic concept used in practice. Despite the well known hardness of the knapsack problem on worst-case instances, practical studies show that knapsack core algorithms can solve large scale instances very efficiently [Pis95, KPP04, MPT99]. For example, there are algorithms that exhibit almost linear running time on purely random inputs. For comparison, the running time of the Nemhauser/Ullmann algorithm for this class of instances is about cubic in the number of items. Core algorithms make use of the fact that the optimal integral solution is usually very similar to the optimal fractional solution in the sense that only a few items need to be exchanged in order to transform one into the other. Obtaining an optimal fractional solution is computationally very inexpensive. The idea of the core concept is to fix most variables to the values prescribed by the optimal fractional solution and to work with only a small number of free variables, called the core (items). The core problem itself is again a knapsack problem with a different capacity bound and a subset of the original items. Intuitively, the core contains those items for which it is hard to decide whether or not they are part of an optimal knapsack filling. We will apply the Nemhauser/Ullmann algorithm on the core problem and exploit the remaining randomness of the core items to upper bound the expected number of enumerated Pareto points. This leads to the first theoretical result on the expected running time of a core algorithm that comes close to the results observed in experiments. In particular, we will prove an upper bound of $O(n\,\mathrm{polylog}\,(n))$ on the expected running time on instances with n items whose profits and weights are drawn independently, uniformly at random. In addition, we investigate harder instances in which profits and weights are pairwise correlated. For this kind of instances, we can prove a tradeoff describing how the degree of correlation influences the running time.

2.3.1 Core Algorithms

Core algorithms start by computing an optimal solution for the *relaxed or fractional* knapsack problem. In this problem, the constraints $x_i \in \{0,1\}$ are replaced by $0 \leq x_i \leq 1$. An optimal solution to the fractional problem can be found by the following greedy algorithm [BZ80]. Starting with the empty knapsack, we add items one by one in order of non-increasing profit-to-weight ratio[2]. The algorithm stops when it encounters the first item that would exceed the capacity bound b. This item is called *break item* and the computed knapsack filling not including the *break item* is called *break solution*. Finally, we fill the remaining capacity with an appropriate fraction $f < 1$ of the break item. It has been shown that the break solution, and hence also the fractional solution, can be found in linear time by solving a weighted median problem [BZ80]. For a geometrical interpretation, let us map each item $\langle w_i, p_i \rangle$ to the point $(w_i, p_i) \in \mathbb{R}^2$. Then the greedy algorithm described above can be pictured as rotating a ray clockwise around the origin starting from the vertical axis. Items are added to the knapsack in the order they are swept by the ray.

[2] In previous studies, the *profit-to-weight ratio* of an item is also called its *density*. In this thesis, the term *density* solely refers to the density function describing the probability distributions of the profits.

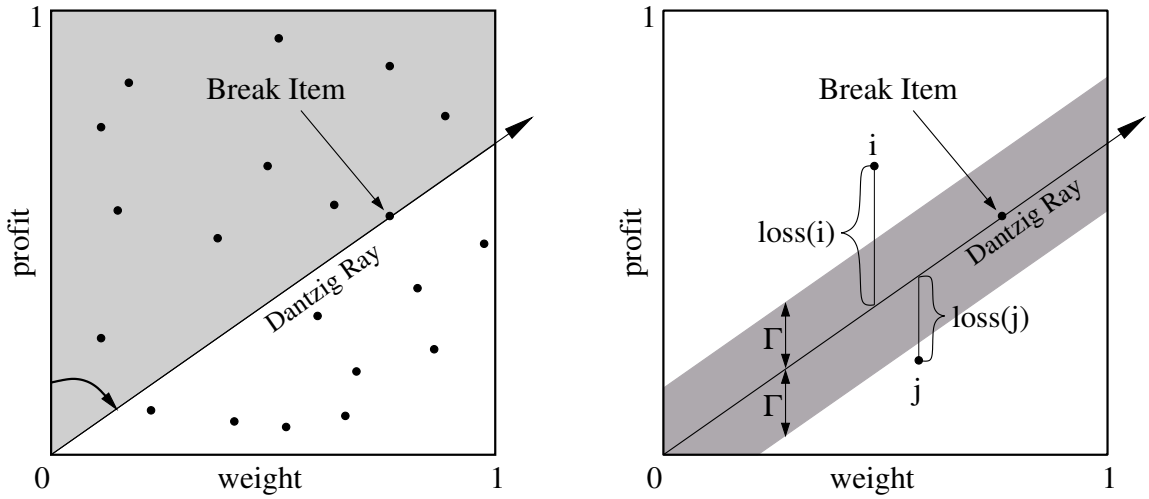

Figure 2.3: **a)** Finding the optimal fractional solution: The ray rotates clockwise and swept items are added to the knapsack. The break item is the first item that exceeds the capacity. It defines the Dantzig ray. **b)** The core stripe has vertical width 2Γ where Γ denotes the integrality gap. The loss of an item is the vertical distance to the Dantzig ray.

The first item exceeding the capacity bound is the break item. The ray defined by the break item is called the *Dantzig ray* (see Figure 2.3).

Based on the computed break solution, an appropriate subset of items is fixed, called the core. The core problem is to compute an optimal knapsack filling under the restrictions that items outside the core are fixed to the value prescribed by the break solution. Clearly, if the chosen core contains all items that need to be exchanged in order to transform the break solution to an optimal integer solution, then an optimal solution for the core problem is also optimal for the original problem. Observe that the core problem is again a knapsack problem defined on a subset of the items. Furthermore, the capacity is reduced by the weight of the items outside the core that are included in the break solution.

The core concept leaves much space for variations and heuristics, most notably the selection of the core items. A common approach selects items whose profit-to-weight ratio is closest to the profit-to-weight ratio of the break item [KPP04]. We follow the definition of Goldberg and Marchetti-Spaccamela [GMS84], who introduce the notion of the loss of items.

Let $\tilde{X} = (\tilde{x}_1, \ldots, \tilde{x}_n)$ be the binary solution vector of the break solution. For any feasible integer solution $X = (x_1, \ldots, x_n)$, define $ch(X) = \{i \in [n] : \tilde{x}_i \neq x_i\}$, i.e., $ch(X)$ is the set of items on which $\tilde{X}$ and X do not agree. When removing an item from the break solution, the freed capacity can be used to include other items which have profit-to-weight ratio at most that of the break item. A different profit-to-weight ratio of the exchanged items causes a loss in profit that can only be compensated by better utilizing the knapsack capacity. This loss can be quantified as follows:

Definition 2.4. *Let r denote the profit-to-weight ratio of the break item. Define the loss of item i to be $loss(i) = |p_i - rw_i|$, or equivalently, the vertical distance of item i to the Dantzig ray.*

Goldberg and Marchetti-Spaccamela[GMS84] proved that the difference in profit between any feasi-

ble integer solution $X = (x_1, \ldots, x_n)$ and the optimal fractional solution $\bar{X}$ can be expressed as

$$p(\bar{X}) - p(X) = r\left(c - \sum_{i\in[n]} x_i w_i\right) + \sum_{i\in ch(X)} \text{loss}(i) \ .$$

The first term on the right hand side corresponds to an unused capacity of the integer solution X whereas the second term is due to the accumulated loss of all items in $ch(X)$. Let X^* denote an arbitrary optimal integral solution. The integrality gap $\Gamma = p(\bar{X}) - p(X^*)$ gives an upper bound for the accumulated loss of all items in $ch(X^*)$, and therefore, also for the loss of each individual item in $ch(X^*)$.

Fact 2.5. *For any optimal integer solution X^* it holds $\sum_{i\in ch(X^*)} loss(i) \leq \Gamma$.*

Thus, items with loss larger than Γ must agree in $\tilde{X}$ and X^*. Consequently, we define the core to consist of all elements with loss at most Γ. This ensures that the optimal solution of the core problem is also an optimal solution for the original problem. Since we do not know Γ, we use a dynamic growing core. Starting with the break item, which has loss 0, we add items in order of increasing loss to the core and compute an optimal solution for each extended core. We stop when the next item has loss larger than the current profit gap, i.e., the gap in profit between $\bar{X}$ and the best integer solution found so far. Figure 2.3 illustrates the definitions of the core.

2.3.2 The Probabilistic Model

Adapting the conventions in previous analyses [Lue82, GMS84, KPP04], we assume the following probabilistic input model. We define a natural mapping of items to points in the plane, where an item with weight w and profit p corresponds to the point $(w, p) \in \mathbb{R}^2$. Given a region $\mathcal{R} \subset \mathbb{R}^2$, items resp. points are sampled independently and uniformly at random from $\mathcal{R}$. The number of sampled items n is a random variable following a Poisson distribution with parameter N. The reason for these assumptions is to simplify the analysis by avoiding disturbing but insignificant dependencies. In particular, the number of points in two disjoint regions of $\mathcal{R}$ are independent random variables. More precisely, for any region $R \subseteq \mathcal{R}$,

$$\mathbf{Pr}[\, R \text{ contains exactly } k \text{ points}] = \frac{e^{-\lambda}\lambda^k}{k!} \ \text{, with } \lambda = N\frac{\text{area}(R)}{\text{area}(\mathcal{R})} \ . \tag{2.1}$$

Except for Section 2.4, we assume that $\mathcal{R}$ is the unit square, that is, weights and profits are drawn independently, uniformly at random from $[0,1]$. In fact, we will restrict ourself to distributions where the weights are chosen uniformly at random from $[0,1]$. Thus, the expected accumulated weight of all items is $N/2$. For the capacity of the knapsack we assume $b = \beta N$, for some constant $\beta \in (0, \frac{1}{2})$ to be specified.

2.3.3 Properties of the Core

Under the probabilistic model introduced, Lueker [Lue82] bounds the expected integrality gap for uniformly random instances. In particular, he shows $\mathbf{E}[\Gamma] = O(\frac{\log^2 N}{N})$. Goldberg and Marchetti-Spaccamela [GMS84] observe that this analysis can easily be generalized to obtain exponential tail bounds on Γ.

Lemma 2.6. *There is a constant c_0 such that for every $\alpha \leq \log^4 N$,* $\mathbf{Pr}\left[\Gamma \geq c_0 \alpha \frac{\log^2 N}{N}\right] \leq 2^{-\alpha}$.

In other words, the integrality gap does not exceed $O(\frac{\log^2 N}{N})$, with high probability. Let us remark that the value of c_0 depends on β, the constant determining the knapsack capacity. The constraint $\alpha \leq \log^4 N$ satisfies our requirements but, in general, the tail bound can be extended to hold for every $\alpha \leq N^\kappa$, for every fixed $\kappa < \frac{1}{2}$.

Beside the upper bound of $O(\frac{\log^2 N}{N})$, Lueker also gave a lower bound of $1/n$ for the expected size of the integrality gap. An improved lower bound of $\Omega(\frac{\log^2 N}{N})$ has been claimed by Goldberg and Marchetti-Spaccamela ([GMS84]) but a complete proof was never published. They also show that the size of $ch(X^*)$ is roughly $\log(N)$. More precisely, they prove

$$\lim_{N\to\infty} \mathbf{Pr}\left[\frac{\log(N)}{\alpha \log\log(N)} \leq |ch(X^*)| \leq f(N)\log(N)\right] = 1 \tag{2.2}$$

for every monotonically increasing unbounded function $f(N)$ and every $\alpha > 2$. They use, however, a slightly different definition of $ch(X)$. Goldberg and Marchetti-Spaccamela define $ch(X)$ on basis of the optimal fractional solution whereas our definition is based on the break solution. The deviation is at most 1 due to the fractional value of the core item.

Similar to Goldberg and Marchetti-Spaccamela [GMS84], will use Lemma 2.6 to obtain a tail bound on the number of core items. Let us briefly sketch their analysis and explain why it fails to bound the expected running time. It is the basis for our modifications to the core algorithm.

Let X denote the number of *core items*, i.e., the number of items with vertical distance at most Γ from the Dantzig ray. Basically, the expected value of X corresponds to N times the area covered by the Γ-region around the Dantzig ray, that is, $\mathbf{E}[X] \approx 2\Gamma N$. (There are some slight dependencies that we neglect here. In fact, the Dantzig ray has some tendency to fall into dense regions.) If Γ is fixed then this number is sharply concentrated. Furthermore, because of Lemma 2.6 the random variable Γ is sharply concentrated around $O(\frac{\log^2 N}{N})$. Combining these results, it follows $X = O(\log^2 N)$, with high probability. Consequently, the number of subsets that can be generated by the core items is $2^X = N^{O(\log N)}$. Obviously, enumerating all these sets cannot be done in polynomial time. Therefore, Goldberg and Marchetti-Spaccamela use a filter mechanism that significantly reduces the number of subsets enumerated. This mechanism generates only those subsets with loss at most Γ. Hence, we will call it *loss filter*. For every fixed $\varepsilon > 0$, they show that with probability $1-\varepsilon$ the number of subsets generated is $2^{O(\sqrt{X})} = N^{O(1)}$. Unfortunately, the degree of this polynomial grows rapidly with the reciprocal of the failure probability ε. Roughly speaking, this is because the random variable X has moved to the exponent so that small deviations in X might cause large deviations in the running time. For this reason, the analysis of Goldberg and Marchetti-Spaccamela fails to bound the expected running time. Moreover, constant factors lost in the analysis of X and Γ go directly to the exponent of the polynomial running time bound.

Instead, we will replace the loss filter by the dominance filter used by the Nemhauser/Ullmann algorithm reducing the number of enumerated subsets from $2^{O(\sqrt{X})}$ to $NX^{O(1)}$. This way, we will be able to bound the expected running time by $N\,\mathrm{polylog}\,N$.

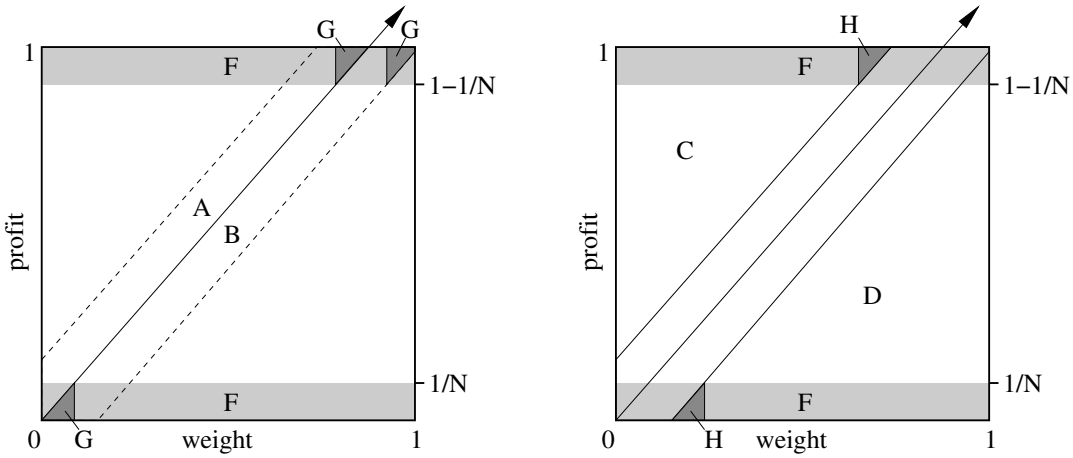

Figure 2.4: Left picture: The core regions A and B consist of all points with vertical distance at most d to the Dantzig ray. The static core consists of all items falling into $A \cup B$. Items in region G have insufficient variation in profit. Right picture: We add all items in region H to the core to ensure that the profit of the remaining items in C and D have sufficient variation in profit.

2.3.4 Algorithm FastCore 1: Filtering Dominated Solutions

The dynamically growing core described above introduces many dependencies which complicates the analysis. Therefore, we resort to a static core. Our first algorithm, called FastCore 1, has almost linear running time but fails in case the core size is too small. The core is chosen large enough to ensure that the algorithm succeeds with high probability. In the next section we remove the remaining failure probability without increasing the expected running significantly.

Let A and B denote the set of points above and below the Dantzig ray r with vertical distance at most $d = c_d N^{-1} \cdot \log^3 N$ (for a suitable constant c_d) from r, respectively. The core consists of all items falling into region $A \cup B$, that is, an item belongs to the core if its loss is at most d. Figure 2.4 illustrates these definitions. The algorithm proceeds as follows. First we compute an optimal fractional solution $\bar{X}$ (and thus the Dantzig ray) in linear time [BZ80]. Then we apply the Nemhauser/Ullmann algorithm to the core problem. Let X denote the solution found. If $p(\bar{X}) - p(X) \leq d$, then we output X, otherwise we output FAILURE.

Notice that $\Gamma = p(\bar{X}) - p(X^*) \leq p(\bar{X}) - p(X)$. If $p(\bar{X}) - p(X) \leq d$ then $\Gamma \leq d$. In this case, items outside the core have loss strictly larger than Γ and, by Fact 2.5, cannot be contained $ch(X^*)$, for any optimal solution X^*. Hence, X must be optimal. The algorithm fails if and only if the chosen height of the core stripes A and B is smaller than the integrality gap Γ, that is, if $\Gamma > d$. Notice, that in this case the computed solution X might nevertheless be optimal, however, a proof of optimality is missing. From Lemma 2.6 it follows that the failure probability is $\mathbf{Pr}[\Gamma > d] \leq 2^{-c_d \log N / c_0} = N^{-c_d/c_0}$.

In the following probabilistic analysis, we will show that the expected running time of our algorithm is $O(N \operatorname{polylog} N)$. This analysis is quite tricky and complicated because of vast dependencies between different random variables. The central ideas, however, are rather simple and elegant. Thus, before going

into the details, let us try to give some intuition about the main ideas behind the analysis. As the area covered by the core stripe $A \cup B$ is $O(N^{-1}\log^3 N)$, the number of core items is only $O(\log^3 N)$, with high probability. The running time bound (Theorem 2.2) of the Nemhauser/Ullmann algorithm depends polynomially on the number of core items and is linear in ϕ, i.e., the maximum density of the probability distributions for the profits of the core items. When calculating ϕ, we have to condition on the fact that these items have weights and profits that fall into the core stripe. The height of the core stripe is $\Theta(\frac{\log^3 N}{N})$. Thus, intuitively, the profit of a "typical" core item follows a uniform distribution over an interval of length $\Theta(\frac{\log^3 N}{N})$. More formally, we will show that the conditional probability distributions for the profits of almost all core items have density at most N. Hence, $\phi \leq N$. By Theorem 2.2, the running time can be upper-bounded by $O(N\,\text{polylog}\,N)$. Interestingly, the linear term in the upper bound on the running time is not due the number of core items but due to the density of their profit distributions.

Theorem 2.7. *For uniformly random instances, algorithm FastCore 1 computes an optimal solution with probability at least $1 - N^{-c_d/c_0}$. The expected running time of this algorithm is $O(N\,\text{polylog}\,N)$.*

Proof. The bound on the failure probability follows from previous discussions. It remains to prove the upper bound on the expected running time. In the following, we identify the ray with its slope, i.e., $r = p_\kappa/w_\kappa$ with κ denoting the index of the break item. Define

$$\begin{aligned} G = &\{(w,p) \in A \,|\, rw > 1 - \tfrac{1}{N}\} \cup \\ &\{(w,p) \in B \,|\, rw < \tfrac{1}{N}\} \cup \\ &\{(w,p) \in B \,|\, rw - d > 1 - \tfrac{1}{N}\} \ . \end{aligned}$$

Define $F = \{(x,y) \in \mathbb{R}^2 : x \in [0,1] \wedge y \in [0,\frac{1}{N}] \cup [1-\frac{1}{N},1]\}$. Figure 2.4 depicts these regions. The motivation for these definitions will become clear soon.

For a moment, let us assume that r and the number of items in A and B as well as their individual weights are fixed arbitrarily. Consider an item i with weight w_i in region A. The profit of this item corresponds to a point on the line segment $L_i = \{p \in [0,1] : (p,w_i) \in A\}$. Observe that the Dantzig ray depends on the weight w_i of this item but not on its profit. In particular, moving the point corresponding to item i arbitrarily on the line segment L_i does not affect the position of the Dantzig ray. Under these assumptions, the random variable p_i is chosen uniformly from the interval L_i. The same holds for the items in region B. Observe that the profit intervals for the items in $(A \cup B) \setminus G$ have length at least $1/N$, except for the break item that will be handled separately. As a consequence, the density of the profit distributions for these items is upper bounded by N. Hence, applying part (c) of Theorem 1.2 yields the following bound on the expected number of Pareto points over the core items[3]. For any given region $R \subseteq \mathbb{R}^2$, let X_R and q_R denote the number of items in R and the number of Pareto points over these items, respectively.

Lemma 2.8. *For sufficiently large N, there exists a constant c_1 such that*

$$\mathbf{E}\left[q_{(A\cup B)\setminus G} \,|\, X_{(A\cup B)\setminus G} = k\right] \leq c_1 N k^4 + 1 \ ,$$

[3] With "number of Pareto points over a set of items" we refer to the number of Pareto points over all subsets of those items.

for any $k \in \mathbb{N}$.

Every additional item can increase the number of Pareto points at most by a factor of two. For this reason, we can assume that the break item is covered by the bound in Lemma 2.8 as well. Furthermore, we can apply this fact to the items in G and obtain that the number of Pareto points over the core items is $q_{A\cup B} \leq 2^{X_G} \cdot q_{(A\cup B)\setminus G}$. The running time of the Nemhauser/Ullmann algorithm is roughly $\mathbf{E}[q_{A\cup B}]$ times the number of core items. In the following, we will (implicitly) show that $\mathbf{E}\left[q_{(A\cup B)\setminus G}\right] = N\operatorname{polylog} N$ and $\mathbf{E}\left[2^{X_G}\right] = O(1)$. Unfortunately, however, the random variables X_G and $q_{(A\cup B)\setminus G}$ are not completely independent as both of them depend on the position of the Dantzig ray. The major difficulty in the remaining analysis is to formally prove that this kind of dependence is insignificant.

We assume that the algorithm first computes all Pareto points over the items in region $(A\cup B)\setminus G$ adding the items in some order that is independent of the profits, e.g., in order of increasing weight. Afterwards, the items from region G are added. Let T denote the running time of this algorithm. Lemma 2.1 combined with Theorem 2.2 yields

$$\mathbf{E}[T \,|\, X_{A\cup B} = k \,\wedge\, X_G = g] \leq c_2[(N(k-g)^4+1)(k-g) + ((N(k-g)^4+1)2^g)] \ ,$$

for every $k \geq g \geq 0$ and a suitable constant c_2. Define $f(k,g) = c_2 N(k-g+7)^5 2^g$. Notice that f is defined in a way such that it is monotonically increasing in g, for every $0 \leq g \leq k$. Rewriting the above bound in terms of this function, we obtain the following slightly weaker bound.

$$\mathbf{E}[T \,|\, X_{A\cup B} = k \,\wedge\, X_G = g] \ \leq \ f(k,g) \ .$$

Applying first $G \subseteq A\cup B$ and then $G \subseteq (A\cup B)\cap F$ combined with the monotonicity property of f yields

$$\begin{aligned}\mathbf{E}[T] \ &\leq \ \sum_{k=0}^{\infty}\sum_{g=0}^{k} \mathbf{Pr}[X_{A\cup B} = k \,\wedge\, X_G = g]\, f(k,g) \\ &\leq \ \sum_{k=0}^{\infty}\sum_{g=0}^{k} \mathbf{Pr}\left[X_{A\cup B} = k \,\wedge\, X_{(A\cup B)\cap F} = g\right] f(k,g) \ .\end{aligned}$$

Next replacing $f(k,g)$ by $c_2 N(k-g+7)^5 2^g$ and rearranging the sums yields

$$\begin{aligned}\mathbf{E}[T] \ &\leq \ c_2 N \sum_{g=0}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\cap F} = g\right] 2^g \ \sum_{k=g}^{\infty} \mathbf{Pr}\left[X_{A\cup B} = k \,|\, X_{(A\cup B)\cap F} = g\right] (k-g+7)^5 \\ &= \ c_2 N \sum_{g=0}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\cap F} = g\right] 2^g \ \sum_{k=0}^{\infty} \mathbf{Pr}\left[X_{(A\cup B)\setminus F} = k \,|\, X_{(A\cup B)\cap F} = g\right] (k+7)^5 \ .\end{aligned}$$

Let us switch to a more compact notation and define $X = X_{(A\cup B)\setminus F}$ and $Y = X_{(A\cup B)\cap F}$. This way,

$$\mathbf{E}[T] \leq c_2 N \sum_{g=0}^{\infty} \mathbf{Pr}[Y = g]\, 2^g \sum_{k=0}^{\infty} \mathbf{Pr}[X = k \,|\, Y = g]\, (k+7)^5 \ . \tag{2.3}$$

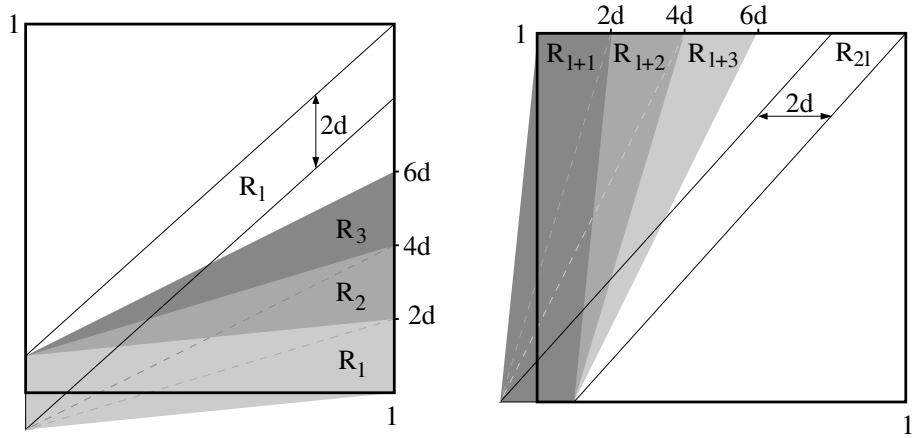

Figure 2.5: Left: Overlapping regions (parallelograms) of height $2d$ cover right lower triangle of the unit square. Right: Regions of width $2d$ cover left upper triangle of the unit square.

In the following, we upper-bound the two probability terms occurring in this bound one after the other. We start by proving $\sum_{k=0}^{\infty} \mathbf{Pr}[X = k \,|\, Y = g]\,(k+7)^5 = O(\text{polylog}\, N)$, for any choice of $g \geq 0$. Afterwards, we show that $\sum_{g=0}^{\infty} \mathbf{Pr}[Y = g]\, 2^g = O(1)$. Hence, $\mathbf{E}[T] = O(\text{polylog}\, N)$ so that the theorem follows.

First, let us study the term $\mathbf{Pr}[X = k \,|\, Y = g]$. Observe, the variables X and Y describe numbers of points in the disjoint regions $(A \cup B) \setminus F$ and $(A \cup B) \cap F$, respectively. If these regions would be fixed, then these two variables would be independent as points are generated by the Poisson process. Unfortunately, however, both regions are defined by the Dantzig ray r and this ray somehow depends on the points found in these regions. In fact, both of the considered regions are adjacent to r, and r has some tendency to fall into a dense region. Therefore, we have to take into account the dependency of the variables X and Y on the random variable r. We deal with this dependency by placing worst-case assumptions on r. In particular, we assume an adversary knowing all points in the unit square chooses the ray r to be any ray through the origin instead of assuming that r is the ray through the break item. The regions $(A \cup B) \setminus F$ and $(A \cup B) \cap F$ are now defined with respect to this adversarial ray r. Let $\mu = 2dN$. Observe that the value of $\mathbf{E}[X]$ is roughly equal to μ.

Lemma 2.9. *The probability, that a random knapsack instance admits a ray r such that $X \geq 2\alpha\mu$ is at most $N\left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu$, for every $\alpha \geq 1$.*

Proof. The idea is to consider only a few essential positions of the ray r and to sum up the probability over all these positions. For this purpose, we define a set of overlapping parallelograms R_i with the property that any given core stripe $A \cup B$ is completely covered by two of these parallelograms. The first $l = \lceil 1/(2d) \rceil$ parallelograms cover the right lower triangle of the unit square $\mathcal{U}$. Parallelogram R_i $(i = 1, \ldots, l)$ is a quadrangle with corner coordinates $(0,d), (0,-d), (1, i \cdot 2d - 2d)$ and $(1, i \cdot 2d)$. (See left picture in Figure 2.5.) Another l parallelograms cover the upper left triangle of the unit square $\mathcal{U}$. Parallelogram R_{l+i} $(i = 1, \ldots, l)$ is a quadrangle with corner coordinates $(-d,0), (d,0), (i \cdot 2d, 1)$ and $(i \cdot 2d - 2d, 1)$. (See right picture in Figure 2.5.) Every parallelogram covers an area of size $2d$. It is easy

to verify that this set of regions has the required property. Let X_i denote the number of items in region R_i. Then $\mathbf{E}[X_i] = \text{area}(\mathcal{U} \cap R_i) \cdot N < 2dN = \mu$. Using Chernoff bounds it holds for all $\alpha \geq 1$ and $i \in [2l]$:

$$\mathbf{Pr}[X_i \geq \alpha\mu] \leq \left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu .$$

Hence, the probability, that a random knapsack instance admits a ray r with $X \geq 2\alpha\mu$ is at most

$$\mathbf{Pr}[\exists i : X_i \geq \alpha\mu] \leq \sum_{i=1}^{2l} \mathbf{Pr}[X_i \geq \alpha\mu] \leq 2l \left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu \leq N \left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu ,$$

as $l = \left\lceil \frac{N}{2c_d \log^3 N} \right\rceil$. This completes the proof of Lemma 2.9. □

The above tail bound on X holds for any adversarial choice for the ray r that is made after the random experiment. Consequently, it also holds for the random Dantzig ray. Furthermore, it holds for any choice of the variable Y, too, as the dependence between the variables X and Y is only via the position of the Dantzig ray. As a consequence, for every $g \geq 0$,

$$\begin{aligned}
&\sum_{k=0}^{\infty} \mathbf{Pr}[X = k \,|\, Y = g]\,(k+7)^5 \\
&\qquad \leq \mathbf{Pr}[X \leq 4\mu \,|\, Y = g]\,(4\mu+7)^5 + \sum_{\alpha=2}^{\infty} \mathbf{Pr}[X \geq 2\alpha\mu \,|\, Y = g]\,(2(\alpha+1)\mu+7)^5 \\
&\qquad \leq (4\mu+7)^5 + \sum_{\alpha=2}^{\infty} N \left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu (2(\alpha+1)\mu+7)^5 .
\end{aligned}$$

Using $\mu = 2dN = 2c_d \log^3 N$ yields

$$\sum_{\alpha=2}^{\infty} N \left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^\mu (2(\alpha+1)\mu+7)^5 = \mu^5 \sum_{\alpha=2}^{\infty} N \left(\frac{e^{\alpha-1}}{\alpha^\alpha}\right)^{2c_d \log^3 N} \left(2\alpha+2+\frac{7}{\mu}\right)^5 .$$

For any fixed $c_d > 0$, this term is bounded by $O(\mu^5)$. Consequently, there exists a constant $c_3 > 0$, such that

$$\sum_{k=0}^{\infty} \mathbf{Pr}[X = k \,|\, Y = g]\,(k+7)^5 \;\leq\; c_3 \log^{15} N .$$

Applying this upper bound to Equation 2.3 gives

$$\mathbf{E}[T] \;\leq\; c_2 c_3 N \log^{15} N \cdot \sum_{g=0}^{\infty} \mathbf{Pr}[Y = g]\, 2^g . \tag{2.4}$$

We further simplify.

$$\sum_{g=0}^{\infty} \mathbf{Pr}[Y=g]\,2^g = \mathbf{E}\left[2^Y\right] = \mathbf{E}\left[2^{X_{(A\cup B)\cap F}}\right] \leq \mathbf{E}\left[2^{X_F}\right] . \tag{2.5}$$

The latter term can be bounded as follows.

Lemma 2.10. $\mathbf{E}\left[2^{X_F}\right] = e^2$.

Proof. The number of points in F follows the Poisson distribution with parameter 2, as $\text{area}(F) = 2/N$ (see Equation 2.1). Consequently,

$$\mathbf{E}\left[2^{X_F}\right] = \sum_{f\geq 0} \mathbf{Pr}[X_f = f]\,2^f = \sum_{f\geq 0} \frac{e^{-2}\cdot 2^f}{f!}\cdot 2^f = e^{-2}\sum_{f\geq 0}\frac{4^f}{f!} = e^{-2}e^4 = e^2 .$$

□

Finally, combining Lemma 2.10 with the Equations 2.4 and 2.5 yields

$$\mathbf{E}[T] \leq c_2c_3N\log^{15}N\cdot\mathbf{E}\left[2^{X_F}\right] \leq c_2c_3e^2N\log^{15}N .$$

Thus Theorem 2.7 is shown. □

2.3.5 Algorithm FastCore 2: Two Lists of Pareto Points

In order to obtain an algorithm that always computes an optimal solution, one can dynamically expand the core item by item until algorithm FastCore 1 is successful. This, however introduces many new dependencies. A somewhat extreme variant of this approach is to start with a core stripe that immediately gives a high success probability, say $1-N^{-3}$, and to use a single backup routine that computes the Pareto points over all items if the primary core algorithm fails. Theorem 2.2a shows that the expected running time of the Nemhauser/Ullmann algorithm under the uniform distribution is only $O(N^4)$. Thus, neglecting dependencies, this approach promises an expected running time of $N^{-3}O(N^4) = O(N)$ for the backup routine. Let us have a closer look at this approach. The running time is mainly determined by the number of enumerated Pareto points. Let q_{in}, q_{out} and q_{all} denote the number of Pareto points over the items in the core, over items outside the core and over all item, respectively. Let $\mathcal{E}$ denote the event that the core computation is successful. Then the expected number of enumerated subsets is $\mathbf{E}[q_{\text{in}}] + \mathbf{E}[q_{\text{all}}|\neg\mathcal{E}]\cdot\mathbf{Pr}[\neg\mathcal{E}]$. The difficulty lies in estimating $\mathbf{E}[q_{\text{all}}|\neg\mathcal{E}]$. Observe that the event $\neg\mathcal{E}$, by definition, is very unlikely. Thus the value of q_{all} might be extremely biased by $\neg\mathcal{E}$, and, hence, the running time of the Nemhauser/Ullmann algorithm conditioned on $\neg\mathcal{E}$ might be much larger than $O(N^4)$.

In order to bypass the difficulties caused by dependencies, we use a different approach utilizing two lists of Pareto points. First, we compute a list of Pareto points over the items in the core, just as in algorithm FastCore 1. If the core computation fails, we compute a second list containing all Pareto points over items outside the core. At this point, the key observation is that we do not need to compute the

complete set of Pareto points over all item. Instead we only need to find the Pareto point that is maximal with respect to the given capacity bound b. This, however, given the two sorted lists of Pareto points can be done in time $O(q_{\text{in}} + q_{\text{out}})$ by scanning the two sorted lists as described by Horowitz and Sahni in [HS74]. They use this technique to reduce the worst case running time of the Nemhauser/Ullmann algorithm from $O(2^n)$ to $O(2^{n/2})$. We use it to deal with the dependencies in our average-case analysis. This way, the expected number of Pareto optimal knapsack fillings generated by our algorithm can be estimated by $\mathbf{E}[q_{\text{in}}] + \mathbf{E}[q_{\text{out}}|\neg\mathcal{E}]$. We have shown $\mathbf{E}[q_{\text{in}}] = O(N \operatorname{polylog} N)$ in the proof of Theorem 2.7. The following analysis mainly deals with bounding $\mathbf{E}[q_{\text{out}}|\neg\mathcal{E}]$ and the time needed to compute the second list.

Theorem 2.11. *Algorithm FastCore 2 always computes an optimal solution. Its expected running time is $O(N \operatorname{polylog} N)$.*

Proof. We begin the proof by specifying some details of the algorithm that are missing in the description above. The algorithm uses a fixed core stripe with $d = 5c_0(\log N)^3/N$, where c_0 is the constant given in Lemma 2.6. Let us adopt the notation from the previous section for the regions defined by the core stripe as depicted in Figure 2.4. Additionally, let C and D denote the regions above and below the core stripe, respectively. For the purpose of a simple analysis, we add the region H (see Figure 2.4) to the core stripe. The analysis of the running time of algorithm FastCore 1 for the items inside the core is not affected by this change as we upper-bounded X_G by X_F and $F \supseteq G \cup H$. Including H into the core stripe ensures that all items in $C \cup D$ follow a distribution with density at most N.

Let T, $T_{A\cup B\cup H}$, and $T_{C\cup D}$ denote the running time of algorithm FastCore 2, algorithm FastCore 1 applied to the items in $A \cup B \cup H$, and the Nemhauser/Ullmann algorithm applied to the items in $C \cup D$, respectively. Furthermore, let $\mathcal{E}$ be the event that the core computation is successful, i.e., the integrality gap is at most d. We account the time for combining the two lists to the time needed for their computation. This way,

$$\mathbf{E}[T] = \mathbf{E}[T_{A\cup B\cup H}] + \mathbf{Pr}[\neg\mathcal{E}] \cdot \mathbf{E}[T_{C\cup D} \,|\, \neg\mathcal{E}]$$

The analysis of algorithm FastCore 1 yields $\mathbf{E}[T_{A\cup B\cup H}] = O(N \operatorname{polylog} N)$. In the following, we will show

$$\mathbf{E}[T_{C\cup D} \,|\, \neg\mathcal{E}] \;\leq\; \frac{N \operatorname{polylog} N}{\mathbf{Pr}[\neg\mathcal{E}]} \;\;,$$

which yields the theorem.

First, let us verify that every item in $C \cup D$ follows a distribution with density at most N. For a moment assume that the Dantzig ray is fixed. Suppose an adversary specifies the weight w_i of an item i in region C (or analogously in region D). Then the item can be moved up and down on the line segment $L_i = \{p \in [0,1] : (p, w_i) \in C\}$ without affecting the event $\neg\mathcal{E}$. The length of this line segment is at least $\frac{1}{N}$ as we added the region H to the core. Consequently, independent of the outcome of $\neg\mathcal{E}$, the profit of each item follows a uniform distribution with density at most N. By Theorem 1.2, for all $k \in \mathbb{N}$,

$$\mathbf{E}[T_{C\cup D} \,|\, \neg\mathcal{E} \wedge X_{C\cup D} = k] = O(k^5 N) \;\;.$$

Let us switch to a more compact notation and define $X = X_{C \cup D}$. It remains to show

$$\mathbf{E}\left[X^5 \,|\, \neg\mathcal{E}\right] = O\left(\frac{\operatorname{polylog} N}{\mathbf{Pr}\left[\neg\mathcal{E}\right]}\right) \quad .$$

Observe that applying Lemma 2.6 with $d = 5c_0(\log N)^3/N$ yields $\mathbf{Pr}\left[\neg\mathcal{E}\right] \leq 2^{-5\log N} = N^{-5}$. Hence, it suffices to show that $\mathbf{E}\left[X^5 \,|\, \neg\mathcal{E}\right] = O(\max\{N^5, \mathbf{Pr}\left[\neg\mathcal{E}\right]^{-1}\})$.

We use the fact that the random variable X is very sharply concentrated around its mean $\mathbf{E}[X] < N$ so that conditioning on $\neg\mathcal{E}$ does not significantly change the expected value of X. For every $\tau \geq 1$,

$$\begin{aligned}
\mathbf{E}\left[X^5 \mid \neg\mathcal{E}\right] &= \sum_{i=1}^{\infty} \mathbf{Pr}\left[X^5 \geq i \mid \neg\mathcal{E}\right] \\
&\leq \tau + \sum_{i=\tau}^{\infty} \mathbf{Pr}\left[X^5 \geq i \mid \neg\mathcal{E}\right] \\
&\leq \tau + \sum_{i=\tau}^{\infty} \frac{\mathbf{Pr}\left[X^5 \geq i\right]}{\mathbf{Pr}\left[\neg\mathcal{E}\right]} \quad .
\end{aligned}$$

As X follows a Poisson distribution with mean at most N, $\mathbf{Pr}\left[X^5 \geq (2\alpha N)^5\right] \leq \exp(-\frac{1}{2}\alpha N)$, for every $\alpha \geq 1$. Hence, for $i \geq (2N)^5$, $\mathbf{Pr}\left[X^5 \geq i\right] \leq \exp\left(-\frac{1}{4}i^{1/5}\right)$. Now setting $\tau = (2N)^5$ gives

$$\begin{aligned}
\mathbf{E}\left[X^5 \,|\, \neg\mathcal{E}\right] &\leq (2N)^5 + \mathbf{Pr}\left[\neg\mathcal{E}\right]^{-1} \sum_{i=\tau}^{\infty} \exp\left(-\tfrac{1}{4}i^{1/5}\right) \\
&= (2N)^5 + \mathbf{Pr}\left[\neg\mathcal{E}\right]^{-1} O(1) \, .
\end{aligned}$$

This completes the proof of Theorem 2.11. □

2.4 Harder Problems: δ-Correlated Instances

Several experimental studies [MPT00, MPT99, MT90, Pis95, PT98] do not only investigate uniformly random instances but also some other, harder classes of random inputs, e.g., so-called *weakly correlated instances*. We generalize this class and define δ-*correlated instances*, $0 < \delta \leq 1$, a parameterized version of weakly correlated instances (the latter correspond to δ-correlated instances with $\delta = 0.1$) as follows. All weights are randomly drawn from $[0,1]$. Profits are set to a random perturbation of the corresponding weight value, i.e., for all $i \in [n]$, $p_i := w_i + r_i$, where r_i is a random variable uniformly distributed in $[-\delta/2, \delta/2]$. In term of the probabilistic model introduced in Section 2.3.2, we choose points uniformly from the quadrangle defined by the points $(0,\delta/2), (0,-\delta/2), (1,1-\delta/2), (1,1+\delta/2)$ (see Figure 2.6). For the following analysis, we again assume that the value of n is chosen according to the Poisson distribution with parameter N, and $b = \beta N$, for some constant $\beta \in (0, \frac{1}{2})$.

2.4.1 Integrality Gap for δ-Correlated Instances

In this section we prove an upper bound on the integrality gap Γ for δ-correlated instances.

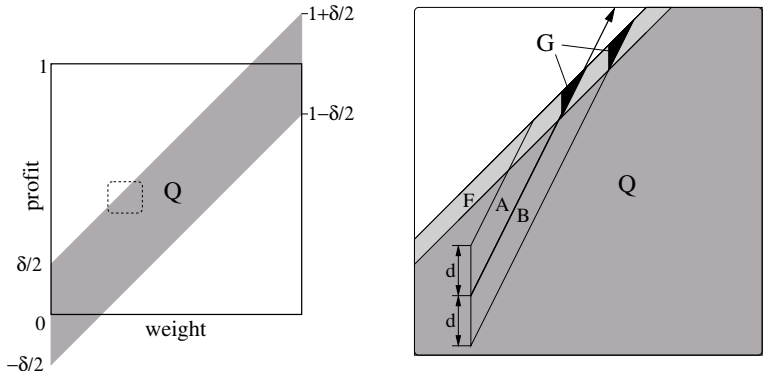

Figure 2.6: For δ-correlated instances items are uniformly sampled from region Q. The right figure (magnified rectangle from the left figure) shows an example for core stripes A and B with height d. Items in region $G \subseteq F$ have insufficient randomness.

Lemma 2.12. *There is a constant c_0 such that for every $1 \leq \alpha \leq \log^4 N$,* $\mathbf{Pr}\left[\Gamma \geq c_0 \alpha \frac{\delta}{N} \log^2 \frac{N}{\delta}\right] \leq 2^{-\alpha}$.

Proof. We adapt the proof from Lueker [Lue82] for uniform instances. Please refer to [Lue82] for details. In particular, we compare the solution of an approximation algorithm with the optimal fractional solution. The difference of the objective values is an upper bound on the integrality gap. The approximation works as follows. Define $k := \log_4(N/\delta)$ and $\varepsilon = O(k4^{-k})$. Assume that items are ordered according to non-decreasing profit-to-weight ratio. Starting with the empty knapsack we insert items in this order until the remaining knapsack capacity is about vk, where v is the expected weight of the next item (the average weight of points in a region swept by the density ray when slightly rotating it further). Let *cap* be the remaining capacity of the knapsack. We then repeatedly consider successive sets $S_1, S_2, \ldots$ of about $2k$ items trying to find a subset $S \subset S_i$ with weight in $[cap - 2\varepsilon, cap]$. The probability that we find such a subset is at least $\frac{1}{2}$ for every S_i. This can be shown with the help of the following lemma, which essentially shows that subsets of random numbers lay exponentially dense.

Lemma 2.13. *([Lue82]) Let f be a piecewise continuous density function with domain $[-a,a]$. Suppose f is bounded and has mean 0 and variance 1. Let x_k be a real sequence with $x_k = o(\sqrt{k})$. Suppose we draw $2k$ variables $X_1, \ldots, X_{2k}$ according to f. Then, for large enough k, the probability that there exists some k-item subset $S \subset [2k]$ with $\sum_{i \in S} X_i \in [x_k - \varepsilon, x_k + \varepsilon]$ is at least $\frac{1}{2}$, provided $\varepsilon = 7k4^{-k}$.*

The profit-gap between the approximate and the optimal fractional solution has two contributions: residual capacity and cumulated loss of packed items. When the approximation algorithm find a suitable subset S, the residual capacity of the knapsack is at most 2ε causing a loss in profit of at most $r2\varepsilon = O(\frac{r\delta}{N} \cdot \log \frac{N}{\delta})$, where $r = p_\kappa / w_\kappa$ is the slope of the Dantzig ray. With high probability the slope r is upper-bounded by a constant term depending on β. The cumulated loss can be estimated by $cap(r - r_l) \leq k(r_0 - r_l)$ where l denotes the number of iterations performed by the algorithm and r_0 and r_l are the slopes of the density ray before the first and after the last iteration, respectively. In each iteration the density ray advances $O(\delta k/N)$ with high probability. The accumulated loss of items in $S \subset S_l$ is $O(l\delta k^2/N)$. In each

iteration the success probability is at least $\frac{1}{2}$, therefore $\mathbf{Pr}[l > \alpha] \leq 2^{-\alpha}$, for all $\alpha \in \mathbb{N}$. □

2.4.2 Expected Running Time of the Core-Algorithm

We adapt algorithm FastCore 2 to the new situation by using a smaller core stripe, that is, we set $d = c_d \frac{\delta}{N} \log^3 \frac{N}{\delta}$ instead of $d = c_d N^{-1} \log^3 N$. This way, we obtain the following result.

Theorem 2.14. *The expected running time of FastCore 2 on δ-correlated instances is $O(\frac{N}{\delta}\, polylog\, \frac{N}{\delta})$.*

Proof. Let Q be the region from where we sample the items (see Figure 2.6). The area of Q has size δ. Compared to the uniform model, the concentration of items is larger by factor $1/\delta$. Choosing core height $d = c_d \frac{\delta}{N} \log^3 \frac{N}{\delta}$ we expect about $2c_d \log^3 \frac{N}{\delta}$ core items in contrast to $2c_d \log^3 N$ for the uniform case. Let A and B denote the core regions above and below the Dantzig ray, respectively. Consider the case when the slope of the Dantzig ray is larger than 1 (the other case is similar). Define regions F and G with $G \subset F$ (see Figure 2.6).

$$F = \{(x,y) \in Q : y - x \in [d - \delta/N, d]\} \ ,$$
$$G = \{(w,p) \in A \,|\, (w, rw) \in F\} \cup \{(w,p) \in B \,|\, (w, rw - d) \in F\}.$$

For items in $(A \cup B) \setminus G$ the maximum density of the profit distribution is N/δ. Therefore, for any $j \in \mathbb{N}$, $\mathbf{E}\left[q_{(A\cup B)\setminus G} \,|\, X_{(A\cup B)\setminus G} = j\right] \leq c_1 (N/\delta) j^4 + 1$. Items in G have larger densities, so we again pessimistically assume that each of these items doubles the number of Pareto optimal sets. We have chosen the size of F so that on average there is only one item in F. This way our analysis, which accounts also for items in G, applies here as well. □

2.5 Experiments

We implemented different algorithmic concepts for the knapsack problem, including the Nemhauser/Ullmann algorithm and different core algorithms. In this Section we present various experimental results.

In Section 2.5.2 we investigate the number of Pareto points for the knapsack problem under various random input distributions. An interesting question concerns the tightness of the upper bounds in Theorem 1.2. Recall that we proved a lower bound of $n^2/4$ for non-decreasing density functions using exponentially growing weights (Theorem 1.13). This leaves a gap of order n for the uniform distribution and order n^2 for general distributions. Even though the generality of the upper bounds (adversarial weights, arbitrary probability distributions, different distributions for different items) hardly allows a qualitative confirmation of the bounds, we found some evidence that the lower bound is tight, which in turn opens the chance of improving the upper bounds.

In Section 2.5.3 we study structural properties of random knapsack instances, e.g. the integrality gap and the core size. Finally, we compare the efficiency of different algorithmic concepts under various random input distributions. These results can be found in Section 2.5.4.

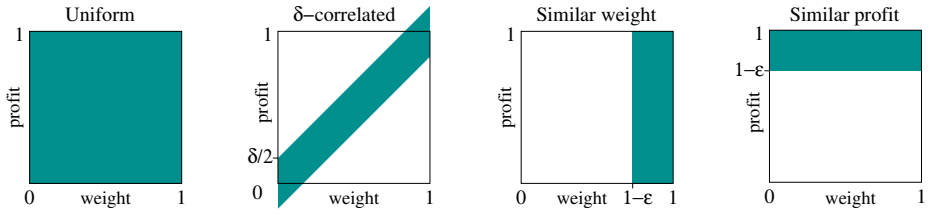

Figure 2.7: Test instances: Items are samples uniformly from the shaded area. To obtain integers, we scale each number by R and round to the next integer. **a)** Uniform instances **b)** δ-correlated Instances **c)** Instances with similar weight **d)** Instances with similar profit.

2.5.1 Experimental Setup

In order to avoid effects that are caused by the pseudo-polynomial complexity of the algorithms we used a large integer domain of $[0, 2^{30}-1]$ for the weights and profits of items. The implementation uses 64 bit integer arithmetic (c++ type *long long*). All programs have been compiled under SunOS 5.9 using the GNU compiler g++, version 3.2.1 with option "-O4". All experiments have been performed on a Sunfire© 15k with 176GB of main memory and 900MHz SparcIII+ CPUs (SPECint2000= 535 for one processor, according to www.specbench.org).

Random input distributions Following the definition in [KPP04], $X \leftarrow \mathcal{U}[l,r]$ means that the random variable X is chosen independently uniformly at random from the interval $[l,r]$. We investigate the following input distributions:

Uniform distribution: $w \leftarrow \mathcal{U}[0,1]$ and $p \leftarrow \mathcal{U}[0,1]$.

δ-correlated instances: $w \leftarrow \mathcal{U}[0,1]$ and $p \leftarrow w+r$ with $r \leftarrow \mathcal{U}[-\delta/2, \delta/2]$ for a given $0 < \delta \leq 1$.

Instances with similar weight: $w_i \leftarrow \mathcal{U}[1-\varepsilon, 1]$ and $w_i \leftarrow \mathcal{U}[0,1]$.

Instances with similar profit: $w_i \leftarrow \mathcal{U}[0,1]$ and $w_i \leftarrow \mathcal{U}[1-\varepsilon, 1]$.

Notice that δ-correlated instances are a parameterized version of *weakly correlated instances* used in former studies (e.g. [KPP04, MPT99, Pis95, MPT00]). The latter correspond to δ-correlated instances with $\delta = 1/10$. The corresponding density function of the profits is upper bounded by $\phi = 1/\delta$. In order to obtain integer weights and profits that can be processed by our algorithm, we scale all values by R and round to the next integer. In this study we used $R = 2^{30}-1$ for all experiments. These definitions are illustrated in Figure 2.7. Observe that for δ-correlated instances some items can have negative profits. This affects a fraction $\delta/8$ of all items on average. These items have no effects on the Pareto curve, nevertheless, they are processed by the Nemhauser/Ullmann algorithm. We consider these instances in our study as they are the basis of the analysis in Section 2.4. In contrast to the theoretical study, the capacity b of the knapsack is set to $\beta \cdot \sum_{i=1}^{n} w_i$, with parameter $\beta < 1$ specified in each figure. Furthermore, the number of items is not a random variable but fixed for each experiment. If not indicated otherwise, each data point represents the average over T trials, where T is a parameter specified in each figure.

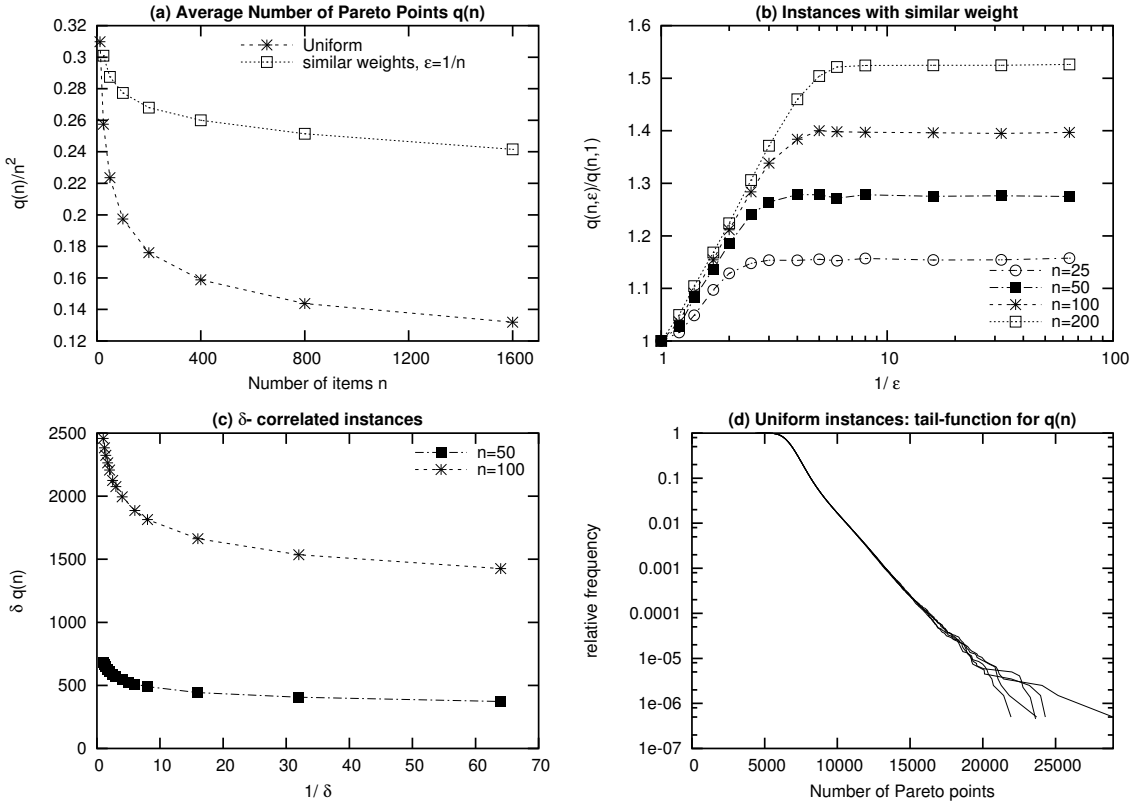

Figure 2.8: Average number of Pareto points. **a)** Uniform instances and instances with similar weight using $\varepsilon = 1/n$, $T \geq 2000$: the ratio $q(n)/n^2$ decreases in n indicating that $q(n) = O(n^2)$. **b)** Instances with similar weights, $T = 10^4$: $q(n,\varepsilon)/q(n,1)$ converges to $c \cdot \log(n)$ for $\varepsilon \to 0$ **c)** δ-correlated instances, $T = 10^4$: $\delta q(n)$ decreases in $1/\delta$ indicating that $q(n) = O(n^2/\delta)$. **d)** Uniform instances: tail functions for $q(n)$, graph shows the fraction of experiments with $q(n) > x$, n=200. We run 5 identical experiments to examine the random deviations.

2.5.2 Number of Pareto Points

In the following, $q(n)$ denotes the (average) number of Pareto points over n items computed by the Nemhauser/Ullmann algorithm. First we tested uniform knapsack instances (Figure 2.8a). We observe that the ratio $q(n)/n^2$ is decreasing in n suggesting that $\mathbf{E}[q(n)] = O(n^2)$. As the decrease might be caused by lower order terms, this bound might be tight. The next experiment for instances with similar weight gives some evidence for the contrary. Notice that by choosing $\varepsilon = 1$, we obtain uniform instances. Figure 2.8b shows the dependence of ε on average number of Pareto points for instances with similar weight. When decreasing ε then $q(n,\varepsilon)$ first increases about linearly but stays constant when ε falls below some value that depends on n. At this value, subsets of different cardinality are well separated in the Pareto curve, i.e., if $\mathcal{P}_i$ denote the set of Pareto optimal subsets with cardinality i then all knapsack fillings in $\mathcal{P}_i$ have less weight (and less profit) than any solution in $\mathcal{P}_{i+1}$. For $\varepsilon = 1/n$, subsets are obviously separated, but the effect occurs already for larger values of ε. Notice that a similar separation of subsets has been used in the proof for the lower bound of $\Omega(n^2)$ in Section 1.3. The ratio $q(n,\varepsilon)/q(n,1)$

apparently converges to $c \cdot \log(n)$ for $\varepsilon \to 0$ and some $c \in \mathbb{R}_{>0}$. In this case, the separation would cause a logarithmic increase of $q(n,\varepsilon)$ compared to uniform instances. As our data suggests that $\mathbf{E}[q(n,\varepsilon)] = O(n^2)$ for instances with similar weight using $\varepsilon = 1/n$ (see Figure 2.8a), the asymptotic behavior of $q(n)$ for uniform instances should be smaller than n^2 by at least a logarithmic factor, i.e., there is evidence that $\mathbf{E}[q(n)] = O(n^2/\log n)$. The experimental data, however, does not allow any conclusion whether or not $q(n,\varepsilon) = \Theta(n^2)$ for $\varepsilon = 1/n$. Notice that the results for instances with similar profit and instances with similar weight are the same. This is due to the fact that, for any individual knapsack instance, exchanging the weight and profit of each item has no influence on the Pareto curve of that instance, a fact which has been shown implicitly in the proof of Theorem 1.13.

Next we consider δ-correlated instances. As the density parameter for the profit distributions is $\phi = 1/\delta$, applying Theorem 2.2 yields a bound of $O(n^4/\delta)$. Figure 2.8a shows that $\delta \cdot q(n,\delta)$ decreases in $\phi = 1/\delta$. Since instances and δ-correlated instances are quite similar for large δ, the data proposes an asymptotic behavior of $q(n,\delta) = O(n^2/\delta)$. We conjecture that a separation of subsets with different cardinality would increase the average number of Pareto points by a logarithmic factor $\log(n/\delta)$ and that $O(n^2/\delta) = O(\phi n^2)$ is a tight bound for adversarial weights.

Finally we investigate the distribution of $q(n)$ on uniform instances. Theorem 2.2 bounds the expected number of Pareto points. Applying the Markov inequality yields only a weak tail bound: for all $\alpha \in \mathbb{R}_{>0}$, $\mathbf{Pr}[q(n) > \alpha \mathbf{E}[q(n)]] \leq 1/\alpha$. Figure 2.8d shows a much faster decay of the probability (notice the log scale). For example, the probability that $q(200)$ is larger than three times the average is about 10^{-5} instead of $1/3$. The experiments suggest that the tail decreases faster than a polynomial $1/\alpha^c$ but possibly slower than exponential $1/c^\alpha$.

In all our experiments (which we could not present here all) we have not found a distribution where $q(n)$ was larger than $n^2\phi$ on average, where ϕ denotes the maximum density over the profit distributions of all items. This gives us some hope that the upper bound can be improved.

2.5.3 Properties of Random Knapsack Instances

Our probabilistic analysis of the core algorithms in Sections 2.3.4 and 2.3.5 is based (among other results) on the upper bound for the integrality gap Γ. Lueker [Lue82] proved that $\mathbf{E}[\Gamma] = O(\frac{\log^2 n}{n})$, provided the weights and profits are drawn uniformly at random from $[0,1]$. Lueker also gave a lower bound of $\frac{1}{n}$ which was later improved to $\Omega(\frac{\log^2 n}{n})$ by Goldberg and Marchetti-Spaccamela [GMS84]. A complete proof of the last result, however, was never published. Our experiments are not sufficient to strongly support this lower bound. For an affirmative answer, the curve in Figure 2.9a would need to converge to a constant $c > 0$ (notice the log-scale for the number of items). Our experiments suggest that in this case the constant c would be at most $1/12$. The small constants involved also account for the moderate number of core items when using the core definition of Goldberg and Marchetti-Spaccamela. The average number of core items in our experiments was 20.2 for uniformly random instances with 1000 item and 67.5 for instances with 10^6 items.

According to Lemma 2.6, Γ is sharply concentrated. More precisely, there is a constant c_0 such that for every $\alpha \leq \log^4 N$, $\mathbf{Pr}\left[\Gamma \geq \alpha c_0 (\log n)^2/n\right] \leq 2^{-\alpha}$. Our experiments (see Figure 2.9b) show an even

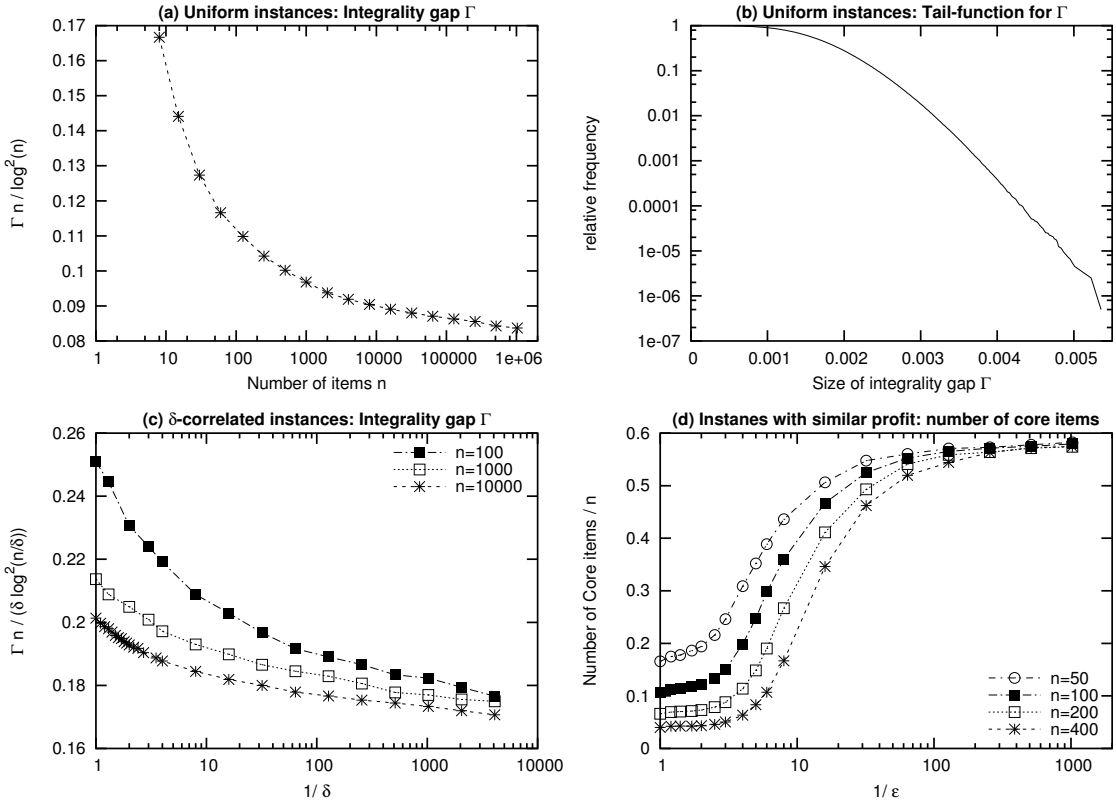

Figure 2.9: **a)** Average integrality gap Γ for uniform instances **b)** Tail function of Γ for uniform instances for $n = 10000$: graph $f(x)$ gives the fraction of experiments with $\Gamma > x$. **c)** Average integrality gap Γ for δ-correlated instances. **d)** Random instances with similar profit: average number of core items as a function of ε. We used $T = 10000$, $\beta = 0.4$ for all experiments.

more rapid decay, also in terms of absolute values: We performed 10^6 experiments using 10000 items. The average integrality gap was $A = 1.695 \cdot 10^{-3}$. The fraction of experiments in which Γ was larger than $2A$ was $4.5 \cdot 10^{-3}$. Only in three experiments Γ was larger than $3A$ and the largest Γ was about $3.16A$. In Section 2.4.1 we showed a similar bound for the integrality gap of δ-correlated instances: $\mathbf{E}[\Gamma] = O(\frac{\delta}{N}\log^2\frac{N}{\delta})$. Figure 2.9c comply with the theorem and, as in the uniform case, give a weak indication that the bound could be tight, as the curves show some tendency to converge to a constant $c > 0$.

Next we consider instances with similar profit. Instead of the integrality gap we investigated the number of core items. Recall that Goldberg and Marchetti-Spaccamela's definition of the core closely relate the two quantities. In particular, the core includes all items whose vertical distance to the Dantzig ray is at most Γ. Figure 2.9d shows that the core contains more than half of all items when ε becomes small. The reason is that items close to the Dantzig ray have a very similar weight. If the remaining capacity of the knapsack is not close to the weight (or a multiple) of these items, it is difficult to fill the knapsack close to its capacity. Therefore, items more distant to the Dantzig ray have to be used to fill the knapsack, but they exhibit a larger loss. As a consequence, the integrality gap is much larger on average

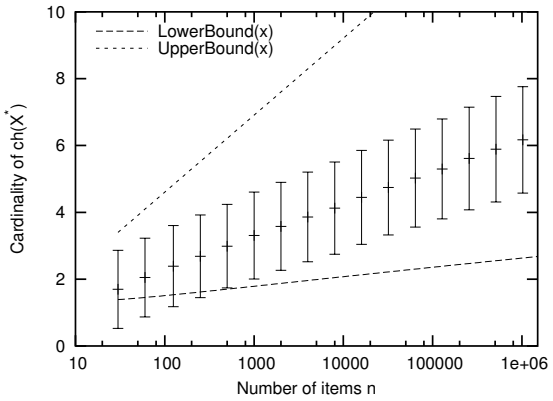

Figure 2.10: Uniform instances: Number of items in the optimal core solution, i.e., $|ch(X^*)|$. The error bars gives the standard deviation. We used $\beta = 0.4$, $T = 10000$.

and so is the number of core items.

Figure 2.10 shows the average number of items in the optimal core $ch(X^*)$, that is, the number of variables, in which the break solution and the optimal solution differ. Goldberg and Marchetti-Spaccamela prove that in the limit, $|ch(X^*)|$ is roughly between $\frac{\log(n)}{2\log\log(n)}$ and $\log(n)$ (see Equation 2.2). These bounds give good estimates for the average size of $ch(X^*)$ also for small values of n. The slowly increasing standard deviation supports the claim of Goldberg and Marchetti-Spaccamela. Observe the small constants. For random instances with 10^6 items, one only has to exchange about 6 items on average, to transform the break solution to the optimal solution.

Finally we investigate the influence of the knapsack parameter β on the integrality gap and the core size of uniform instances. Recall that in our experiments we used a knapsack capacity of $c = \beta \sum_{i=1}^{n} w_i$. Thus, for $\beta = 1$ all items fit into the knapsack. This is different in the probabilistic model introduced in Section 2.3.2, where we assumed $c = \beta N$ such that c does not depend on the random choice of the weights. For uniform instances, the expected weight of an item is $1/2$. Hence, for any $\beta > 1/2$, all items fit into the knapsack with probability going to 1 as $N \to \infty$ (see Lueker [Lue82]). We strongly believe, however, that the results for both models are very similar except for the factor 2 in the range for β. Observe, that for $\beta = 1/3$ the Dantzig ray has expected slope 1 when setting $c = \beta \sum_{i=1}^{n} w_i$.

As already mentioned, the constant c_0 in Lemma 2.6 and, hence, the hidden constant in the upper bound $\mathbf{E}[\Gamma] = O((\log n)^2/n)$ depends on the choice for β. The top two curves in Figure 2.11 show the average integrality gap and the average number of core items as a function of β for uniform instances with $n = 1000$ items. As β becomes small, the integrality gap Γ increases rapidly. The number of core items, however, decreases. This is different for large β. For $\beta > 1/3$, Γ as well as the number of core items decreases.

Recall, that the difference in profit between any feasible integer solution $X = (x_1, \ldots, x_n)$ and the

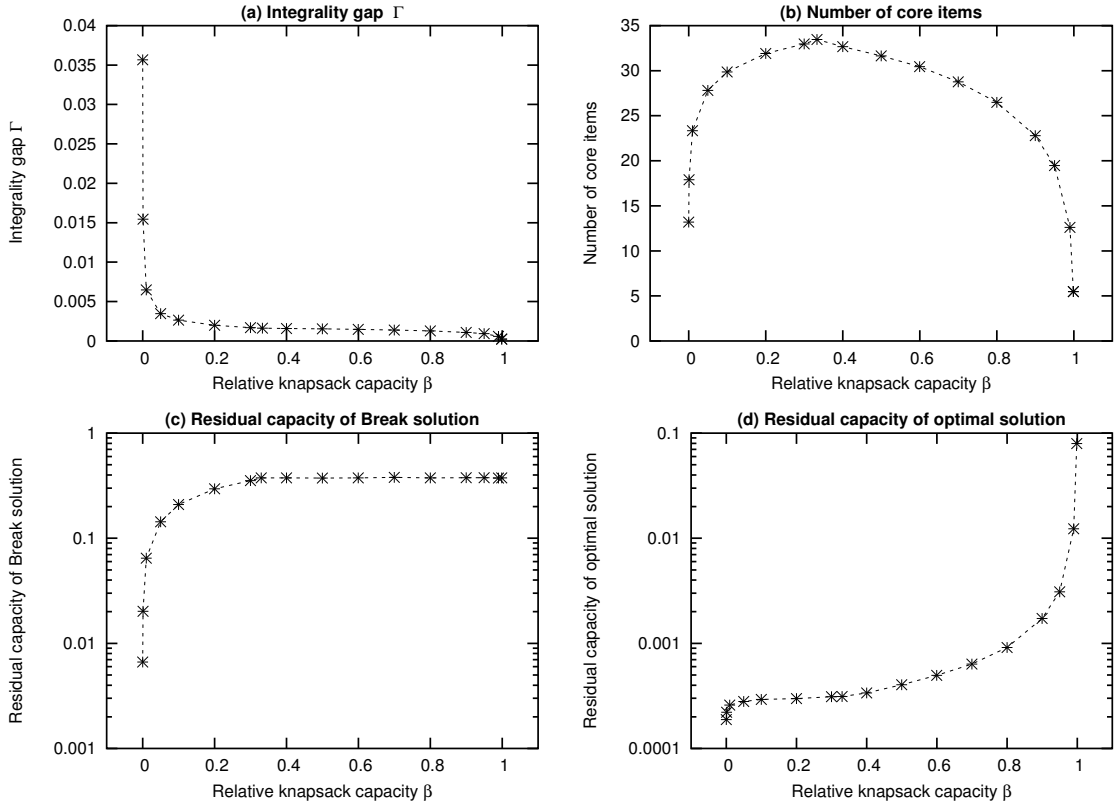

Figure 2.11: Structural properties of uniformly random knapsack instances as a function of the threshold parameter β: **a)** Integrality gap Γ **b)** Number of core items **c)** Residual capacity of the Break solution **d)** Residual capacity of the optimal integral solution. We used $n = 10000$, $T = 10000$ for all experiments.

optimal fractional solution $\bar{X}$ can be expressed as

$$p(\bar{X}) - p(X) = r\left(c - \sum_{i\in[n]} x_i w_i\right) + \sum_{i\in ch(X)} \text{loss}(i) \ .$$

Hence, the optimal solution is a tradeoff between minimizing the residual capacity and using, for this purpose, items with small loss, i.e., items with small vertical distance to the Dantzig ray. The residual capacity has a larger influence on the profit gap as r becomes large, that is, when β becomes small. Furthermore, due to the steep Dantzig ray, one can expect only few items with small loss, as the area of a fixed core region (intersected with the unit square) becomes smaller when increasing slope r of the Dantzig ray. Therefore, decreasing β causes an increase of r as well as an increasing loss of items which are needed to fill the knapsack close to its capacity. Both effects contribute to the increase of Γ. An opposite effect has the small weight of core items and the small residual capacity of the break solution. For small β (or large slope r) items in the core stripe have small weight. In particular, the break item has small weight. As the residual capacity is upper bounded by the weight of the break item, it is small as

well (see Figure 2.11c). Assume, the weight of core items as well as the residual capacity of the break solution fall into interval $[0,s]$, for some $s < 1$, instead of $[0,1]$. Then one expects that the subsets of the core items lie denser by factor $1/s$. In other words, filling the knapsack close to its capacity needs less core items. This observation is consistent with Figure 2.11d. Even though the number of core items decreases significantly as β becomes small, the residual capacity of the optimal integer solution decreases only by factor 1.6 in the range $\beta \in [10^{-4}, 1/3]$. Despite the large integrality gap for small values of β, the number of core items is small. The reason for this phenomenon is that a fixed core region of a steep Dantzig ray usually contains only few items.

For $\beta > 1/3$, the expected slope of the Dantzig ray is less than 1 and the expected residual capacity of the break solution is $3/8$. The decrease of the number of core items for increasing $\beta > 1/3$ is mainly due to the decreasing value of r. With smaller r, minimizing the residual capacity is less important and a smaller core is sufficient. Figure 2.11c confirms this observation by showing a rapid increase of the residual capacity of the computed optimal solution for large β. The running time of the Nemhauser/Ullmann algorithm is maximum for $\beta = 1/3$, which is consistent with the maximum for the number of core items. For very small or very large values of β the running time is significantly smaller.

2.5.4 Efficiency of Different Implementations

Besides the Nemhauser/Ullmann algorithm and the core concept of Goldberg and Marchetti-Spaccamela, we used two other ideas in our implementation, namely the loss filter, used in the theoretical study by Goldberg and Marchetti-Spaccamela, and the Horowitz and Sahni decomposition. Both concepts are known since at least 20 years. While the Horowitz and Sahni decomposition has found many practical applications in algorithms especially for the subset sum problem, the loss filter apparently has not been paid attention in the standard literature on the knapsack problem (e.g. [KPP04]). A quite natural combination of those ideas yields a very competitive algorithm that even outperforms the *Combo* code, the most successful implementation reported in scientific literature.

The Loss Filter

In Section 2.3.1 we already defined the loss of items to be the vertical distance to the Dantzig ray. Fact 2.5 states that the accumulated loss of all items that do not agree in the break solution $\tilde{X}$ and an arbitrary optimal solution X^* is bounded by the integrality gap Γ. We used this fact to provide a bound on the size of the core region that ensures a successful core computations. Goldberg and Marchetti-Spaccamela further exploit Fact 2.5. Observe that solving the core problem is equivalent to finding the set $ch(X^*)$ for some optimal solution X^*. Let C_A denote the set of core items included into the break solution. We formulate the core problem relative to the break solution, that is, adding items from C_A into the core solution corresponds to removing them from the break solution. Hence, the weight and profit of items in C_A are negated for the core problem. The capacity of this core problem is $b_B = b - w(\tilde{X})$, the residual capacity of the break solution. The loss of a core solution X is simply the accumulated loss of items in X, that is, $\text{loss}(X) = \sum_{i\in[n]} \text{loss}(i)x_i$. Due to Fact 2.5, solutions with loss larger than the integrality gap Γ can not be optimal and, furthermore, cannot be extended to an optimal solution by adding more core items

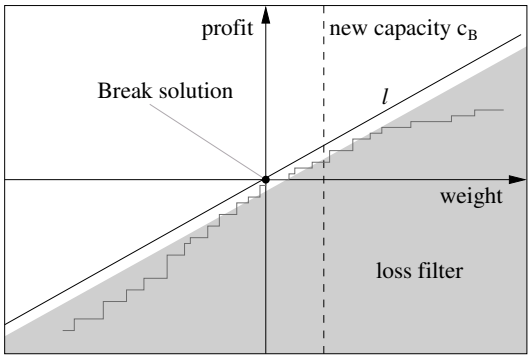

Figure 2.12: Pareto curve of the core problem as a step function. Core items included in the break solution are negated as adding them to the core solution corresponds to removing them from the break solution. As a result, the Pareto curve moves $|w(C_A)|$ to the left and $|p(C_A)|$ down, where C_A denotes the set of negated items. The slope of line l is the profit-to-weight ratio of the break item. The loss of a subset of core items is the vertical distance to l. Solutions in the shaded area (points whose vertical distance to l is larger than the current integrality gap Γ) are filtered out by the loss filter. Observe, that if a Pareto point (w, p) is filtered out then all solutions dominated by (w, p) are in the shaded area as well. The optimal core solution is the rightmost Pareto point left of the dashed line that represents the capacity b_B of the core problem.

as the loss of items is non-negative. So these solutions can be filtered out in the enumeration process. Instead of the integrality gap Γ which we do not know, we use upper bounds on Γ, e.g., the difference in profit between the optimal fractional solution and the best integral solution found so far.

The Horowitz and Sahni Decomposition

Horowitz and Sahni [HS74] present a nice variation of the Nemhauser/Ullmann algorithm using two lists. They exploit the observation that not all Pareto points need to be enumerated. Instead, one only has to find the most profitable among the feasible ones. Let $\mathcal{L}$ and $\mathcal{L}'$ denote the sorted lists of Pareto points over disjoint item sets I and I', respectively. Each Pareto point over item set $I \cup I'$ can be constructed by combining two points from $\mathcal{L}$ and $\mathcal{L}'$ (or unifying the corresponding Pareto optimal subsets of items). For all $p \in \mathcal{L}$, one has to find the most profitable $q \in \mathcal{L}'$ such that the combination of p and q is still feasible,i.e., their combined weight is not larger than the capacity b. Since the lists are sorted, the $p \in \mathcal{L}$ together with its most profitable combination can be found by a parallel scan of the two lists in opposite directions (for details, see [HS74]).

Our Implementation

We tested different implementations, all core algorithms that use a subset of the discussed concepts (domination concept, loss filter, Horowitz and Sahni decomposition). Furthermore, we added some

heuristics, which further improve the efficiency of our code significantly. In the following we give a more detailed description of our implementation.

First we compute the break solution $\tilde{X}$ in linear time. Then we use a dynamic core, adding items in order if increasing loss. The capacity of the core problem is $b - w(\tilde{X})$ and items included in $\tilde{X}$ become negative, that is, we negate the weight and profit of those items. The core problem is solved using the Nemhauser/Ullmann algorithm, except for the variant that uses only the loss filter, as analyzed in [GMS84]. The loss filter can easily be incorporated into the Nemhauser/Ullmann algorithm, as the loss of a Pareto point is determined by its weight and profit (see Figure 2.12). Pareto points whose loss is larger than the current integrality gap are deleted from the list. Observe that all solutions dominated by such a filtered Pareto point have an even larger loss. Hence, domination concept and loss filter are "compatible". When using two lists (Horowitz and Sahni decomposition) we alternately extend the lists with the next item and check for a new optimal solution in each such iteration.

Additional Heuristics

Instead of alternately extending the two lists we first extend only one list, which grows exponentially at the beginning. Later we use the second list. The idea is that the first list is based on items with small loss such that the loss filter has hardly an effect. The items used in the second list are expected to have larger loss such that the loss filter keeps the list relatively small and the extension of the list with new items is fast. We use lazy evaluation, i.e., we do not scan the two lists in each iteration to find new optimal solutions as we expect the first list to be significantly larger than the second. Knowing the length of the lists, we balance the work for extending a list and scanning both lists. Finally, we delete Pareto points from the list that cannot be extended by forthcoming items as their loss is too large. If the next item has loss l and the current integrality gap is Γ, then we delete Pareto points with loss larger that $\Gamma - l$. This heuristic is especially efficient at the end of the computation when the loss of forthcoming items is close to Γ.

Results

We investigate the influence of the different concepts (domination concept, loss filter, Horowitz and Sahni decomposition and the additional heuristics) on the running time of our algorithm, under different random input distributions. We compare our implementation to *Combo*, a knapsack solver due to Pisinger [MPT99]. *Combo* combines various advanced algorithmic techniques that have been independently developed to cope with specifically designed difficult test instances. For instance, it applies cardinality constraints generated from extended covers using Lagrangian relaxation, rudimentary divisibility tests and pairs dynamic programming states with items outside the core. Our implementation, however, is rather simple but proves to be very efficient in reducing the number of enumerated knapsack filling for a given core. In particular, we add items strictly in order of increasing loss to the core. This can lead to a very large core, as shown in Figure 2.9d. We observed that whenever *Combo* was superior to our implementation, the core size of *Combo* was usually significantly smaller. So we consider our implementation not as a competing knapsack solver but rather as a study that points out possible improvements.

Due to technical difficulties with measuring small amounts of processing time and huge differences in the running times we employ, besides the running time, another measure, called work. Each basic step takes a unit work. This way, an iteration of the Nemhauser/Ullmann algorithm with input list $\mathcal{L}$ costs $2|\mathcal{L}|$ work units. Finding an optimal solution by scanning 2 lists $\mathcal{L}_1$ and $\mathcal{L}_2$ in parallel (Horowitz and Sahni) costs $|\mathcal{L}_1|+|\mathcal{L}_2|$ work units.

Figure 2.13 shows the work and running time for different implementations. According to Theorem 2.2, we expect the work as well as the running time to increase quasi-linear in n and $1/\delta$ for uniform and δ-correlated instances. For an easier comparison, we divided work and running time by n and $1/\delta$, accordingly. Notice that almost all scales are logarithmic.

In contrast to the theoretical analyses of core algorithms in [GMS84] and Section 2.3 where the domination concept leads to a better running time bound than the loss filter, the latter is superior in all experiments. This remains true for large instances, even if the domination concept is complemented by the Horowitz/Sahni decomposition (2 lists of Pareto points). The *Combo* code is superior to all three variants. Interestingly, the combination of all three 3 concepts (PO+Loss+2) clearly outperform *Combo* for uniform and δ-correlated instances. In fact, it exhibits a sublinear running time for uniform instances (without linear time preprocessing). Using the described heuristics improves the running times further. The average running time of *Combo* and our fastest implementation differs by a factor of 190 for uniform instances with 1024000 items. For δ-correlated instances with $\delta = 1/1024$, the factor is about 700.

The limitations of our implementation is illustrated in Figure 2.13e, which shows the running time (with preprocessing) for instances with similar profit. Besides the disadvantage of having a large core (Figure 2.9d), the large integrality gap makes the loss filter less efficient.

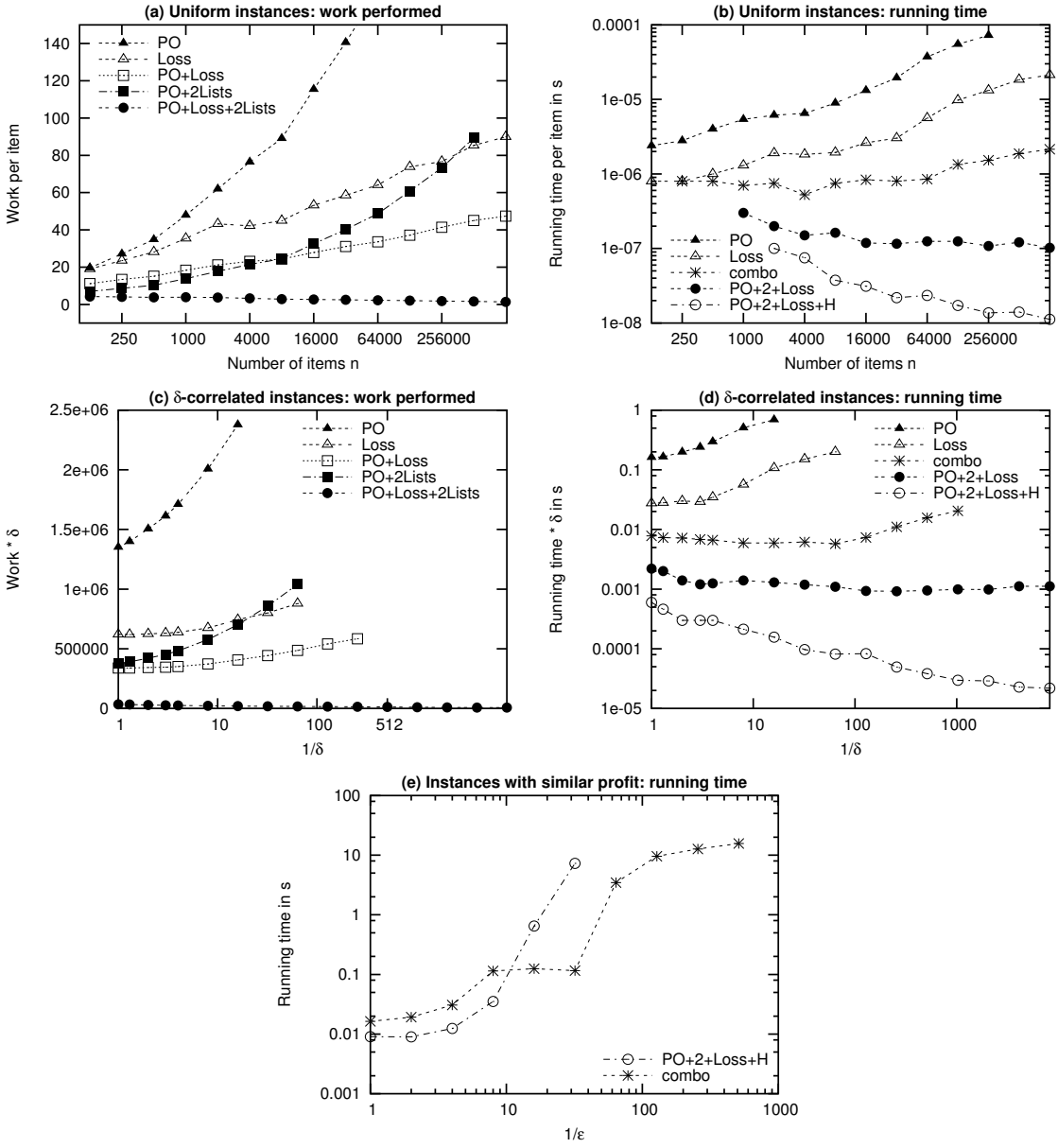

Figure 2.13: Efficiency of different implementations. The upper two figures show, for uniform instances, the dependence of the number of items on the work performed (left) and on the running time (right). The two figures in the middle show, for δ-correlated instances with $n = 10000$ items, the dependence of the parameter δ on the work performed (left) and the running time (right). The bottom figure shows the efficiency for instances with similar profit ($n = 10000$). Each curve represents an implementation that uses a subset of the features as indicated: PO = filtering dominated solutions, Loss = loss filter, 2 = two lists, H = additional heuristics, Combo = *Combo* code. For all experiments we used $T = 1000$ and $\beta = 0.4$. The given running times for uniform and δ-correlated instances do not include the linear time preprocessing (the generation of the random instance, finding the optimal fractional solution, partitioning the set of items according to the loss of items) as it dominates the running time for the easy instances.

Chapter 3

A General Framework for Constrained Binary Optimization Problems

Although our probabilistic analysis of the size of the Pareto curve is already quite general and has been successfully applied to the average-case analysis of two well-studied optimization problems, it exhibits two limiting factors that prevent an immediate application to an even broader class of problems. First, enumerating all Pareto optimal solutions is for most problems a hard task in itself. In order to prove a polynomial bound on the expected running time of such an enumeration algorithm, it has to be very efficient, that is, the running time bound should depend quasi linearly on the number of enumerated solutions. The second point concerns the restriction to only two objective functions, which might, however, be overcome by a more general analysis.

In order to avoid the complications caused by handling more than one objective, the standard method is to choose one criteria as the objective function and bounding the remaining criteria by appropriate thresholds (see Chapter 1.1). If the criteria are linear functions to be minimized, then the resulting constraints have the form $c^T x \leq t$, for some vector c. Hence, a multicriteria problem of the form

$$\begin{array}{rl} \text{optimize} & f(c_1^T x, \ldots, c_k^T x) \\ \text{subject to} & x \in \mathcal{S} \end{array}$$

is turned into a single criterion optimization problem of the form

$$\begin{array}{rl} \text{minimize} & c_1^T x \\ \text{subject to} & x \in \mathcal{S} \\ \text{and} & c_i^T x \leq t_i, \quad \text{for all } i \in \{2, \ldots, k\} \ , \end{array}$$

where $\mathcal{S}$ denotes the set of solutions of the underlying optimization problem. The set $\mathcal{S}$ is, of course, implicitly specified by an appropriate description of the problem instance. As before, we are interested in binary optimization problems, so $\mathcal{S}$ is a set of 0/1-vectors of length n. We call the resulting problem (constrained) binary optimization problem. In this chapter we aim at a general smoothed/average-case

analysis for this class of problems. Our average-case analysis for the number of Pareto points in Chapter 1 is solely based on the randomness in the objective functions. The underlying set of solutions $\mathcal{S}$ is assumed to be adversarial. Translated to constrained binary optimization problems, only the objective function and the new constraints should be stochastic. We will follow this idea by introducing a semi-random input model that allows to specify which constraints of the problem formulation are of stochastic nature and which are assumed to be adversarial. This leads to a framework for a general smoothed/average-case analysis for a large class of optimization problems which also takes into account the combinatorial structure of each individual problem. We will see that this framework allows a meaningful analysis for problems which are not constrained versions of a multicriteria optimization problem, like scheduling and the general assignment problem.

Based on the concept of polynomial average-case complexity we define polynomial smoothed complexity. We are able to characterize the smoothed complexity of (constraint) binary optimization problems in terms of their worst-case complexity. A constrained binary optimization problem has *polynomial smoothed complexity* if and only if it has pseudo-polynomial complexity in the stochastic expressions and polynomial complexity in the remaining input.

Our analysis is centered around structural properties of binary optimization problems, called *winner*, *loser*, and *feasibility gap*. We show, when the coefficients of the objective function are stochastic, then there usually exist a polynomial $n^{-\Omega(1)}$ gap between the best and the second best solution. If the coefficients of the constraints are stochastic, then the slack of the optimal solution with respect to this constraint has usually polynomial size $n^{-\Omega(1)}$ as well. Similar to the condition number for linear programming, these gaps describe the sensitivity of the optimal solution to slight perturbations of the input and can be used to bound the necessary accuracy as well as the complexity for solving an instance. We exploit the properties of these gaps in form of an adaptive rounding scheme increasing the accuracy of calculation until the optimal solution is found. The strength of our techniques is illustrated by applications to various NP-hard optimization problems from mathematical programming, network design, and scheduling for which we obtain the first algorithms with polynomial smoothed/average-case complexity. We can strengthen our results by allowing zero preserving perturbations that allow an application to problems beyond constrained versions of multicriteria problems.

3.1 Optimality and Precision

Many combinatorial optimization problems have an objective function or constraints specified in terms of real numbers representing natural quantities like time, weight, distance, or utility. This includes some well-studied optimization problems like, e.g., traveling salesperson, shortest path, minimum spanning tree as well as various scheduling and packing problems. When analyzing the complexity of algorithms for such problems, we usually assume that these numbers are integers or rational numbers with a finite length representation. The hope is that it suffices to measure and compute with some bounded precision in order to identify an optimal or close to optimal solution. In fact, if real numbers occur only in the objective function and if this objective function is well-behaved (e.g., a linear function) then calculating with reasonable approximations of the input numbers yields a feasible solution whose objective value is

at least close to the optimal objective value. More problematically, however, if the constraints are defined by real numbers, then calculating with rounded input numbers might miss all interesting solutions or might even produce infeasible solutions.

How can one solve optimization problems (efficiently) on a computer when not even the input numbers can be specified exactly? – In practice, optimization problems in which real numbers occur in the input are solved by simply rounding the real numbers more or less carefully. Fortunately, this approach seems to yield reasonable results. We seek for a theoretically founded explanation why this rounding approach usually works. Studying this issue under worst case assumptions does not make very much sense as, in the worst case, the smallest inaccuracy might lead to an infeasible or utterly sub-optimal solution. This question needs to be studied in a stochastic model. In the following probabilistic analysis, we will show that, under some reasonable and quite general stochastic assumptions, one can usually round real-valued input numbers after only a logarithmic number of bits without changing the optimal solution.

3.2 A Semi-Random Input Model for Discrete Optimization Problems

We consider optimization problems defined on a vector of n binary variables $x = (x_1, \ldots, x_n)$. The set of feasible solutions is described by the intersection of a ground set of solutions $\mathcal{S} \subseteq \{0,1\}^n$ and solutions that satisfy linear constraints of the form $w^T x \leq t$ or $w^T x \geq t$. The ground set $\mathcal{S}$ of solutions can be specified arbitrarily, for instance, by linear constraints. While the part that specifies $\mathcal{S}$ is adversarial, we assume that the coefficients in the additional linear constraints are random or randomly perturbed real numbers. The reason for distinguishing stochastic and adversarial part of the input is that we do not want that the randomization destroys the combinatorial structure of the underlying optimization problem. In principle, the objective function can be any function $f : \mathcal{S} \to \mathbb{R}$. Our model allows also a stochastic objective function. In this case, the objective function must be linear, that is, the objective is of the form *minimize* (or *maximize*) $c^T x$, where c is a vector of random and randomly perturbed real valued coefficients $c_1, \ldots, c_n$. In the following, we use the phrase *expression* as a generic term for the linear expressions $c^T x$ and $w^T x$ occurring in the objective function and the constraints, respectively. The number of stochastic expressions is denoted by $k \geq 1$ and the number of marked constraints by $k' \in \{k-1, k\}$, depending on whether or not the objective function is stochastic. For $k' \geq 1$ let $\mathcal{B}_j$ denote the set of solutions that satisfy the jth constraint, for all $j \in [k']$. The set of feasible solutions for a given problem instance is then $\mathcal{S} \cap \mathcal{B}_1 \cap \ldots \cap \mathcal{B}_{k'}$ The coefficients in the stochastic expressions are specified by independent continuous probability distributions with domain $\mathbb{R}$. Different coefficients might be drawn according to different distributions. The only restriction on these distributions is that their density function is piecewise continuous and bounded. Assuming bounded densities is necessary as otherwise worst-case instances could be approximated arbitrarily well by specifying distributions with very high density. For a given distribution, the supremum of its density function is called its *density parameter*. We will see that the maximum density parameter over all distributions plays an important role in our analysis. This parameter is denoted by ϕ. Intuitively, ϕ can be seen as a measure specifying of how close the instances might be to the worst case. A worst-case instance can be interpreted as a stochastic instance in which the

probability measure for each stochastic number is mapped to a single point. Thus, the larger ϕ, the closer we are to a worst-case analysis.

In our probabilistic analysis, we assume that the objective function defines a unique ranking among all solutions in $\{0,1\}^n$ according to the objective function. Observe, if the objective function is stochastic then the coefficients are continuous random variables so that the probability that there exist solutions with same objective value is 0. In other words, a unique ranking is given with probability 1. Recall that the objective function does not have to be linear if it is adversarial, but if it is linear, i.e., of the form $c^T x$, $c \in \mathbb{Q}^n$, then a unique ranking can always be enforced by encoding the lexicographical order among the solutions into the less significant bits of the objective function without changing the computational complexity of the underlying optimization problem by more than a polynomial factor. In fact, most of the algorithmic problems that we will study have algorithms that implicitly realize a unique ranking. In this case, one does not even need an explicit encoding. Given a unique ranking, we aim at finding the *winner*, i.e., the highest ranked solution in $\mathcal{S} \cap \mathcal{B}_1 \cap \ldots \cap \mathcal{B}_{k'}$. In the following, optimization problems satisfying all the conditions above are called *binary optimization problems with stochastic expressions* or, for short, *binary optimization problems*.

Let us illustrate our semi-random input model by an example. In the minimum spanning tree problem one seeks for a spanning tree in a given graph that has minimum weight. In the binary program formulation of this problem there is a variable x_e for each edge $e \in E$. Thus, n corresponds to the number of edges. A 0/1 solution x is feasible if the edges in the set $\{e \in E \,|\, x_e = 1\}$ form a spanning tree. Let $\mathcal{S}$ denote the set of all solutions satisfying this condition. The combinatorial structure described by $\mathcal{S}$ should not be touched by our randomization. It makes sense, however, to assume that the objective function is stochastic as its coefficients describe measured quantities. So we may assume that these coefficients are perturbed with uniform ϕ-perturbations, that is, each of these coefficients corresponds to the sum of an adversarial number from $[0,1]$ and an independent random number drawn uniformly from $[0,\phi^{-1}]$. In the constrained minimum spanning tree problem (see, e.g., [GR96]), edges do not only have weights but additionally each edge has a cost c_e. Now one seeks for the minimum weight spanning tree satisfying an additionally specified linear cost constraint $\sum_e c_e x \leq T$. This additional constraint corresponds to a subset $\mathcal{B}_1 \subseteq \{0,1\}^{|E|}$ so that now $\mathcal{B}_1 \cap \mathcal{S}$ is the set of feasible solutions. Due to the additional constraint, the problem becomes NP-hard. We will see, however, that there is an algorithm with "polynomial smoothed complexity" if either the objective function or the additional latency constraint is stochastic.

3.3 How Accurately Do We Need to Calculate?

More precisely, we ask how many bits of each stochastic input number do we need to reveal in order to determine the winner? – We say that the winner is *determined* after revealing some number of the bits, when there is only one possible candidate for the winner, regardless of the outcomes of the unrevealed bits. We will prove the following theorem.

Theorem 3.1. *Consider any instance of a binary optimization problem* Π. *Let* $n \geq 1$ *denote the number of binary variables and* $k \geq 1$ *the number of stochastic expressions.*

a) Suppose the expected absolute value $\mathbf{E}[|w|]$ *of every stochastic coefficient* w *is bounded from above by* $\mu > 0$*. Then the number of bits in front of the floating point of any stochastic coefficient* w *is bounded by* $O(\log(1+\mu nk))$*,* **whp**[1].

b) Let $\phi > 0$ *denote the maximum density parameter, that is, all density functions are upper-bounded by* ϕ*. Then the winner is uniquely determined when revealing* $O(\log(1+\phi nk))$ *bits after the binary point of each stochastic coefficient,* **whp**.

Remark 3.2. *One can always decrease* ϕ *or* μ *without changing the problem by scaling the appropriate stochastic expression (and the respective bound, where applicable) by some factor* $\gamma > 0$*. As the scaled distribution has mean* $\gamma\mu$ *and density parameter* ϕ/γ*, this kind of scaling does not change the overall number of bits that need to be revealed since* $\log(\mu nk) + \log(\phi nk) = \log(\gamma\mu nk) + \log(\frac{1}{\gamma}\phi nk)$*. It only transfers significant bits from positions before the binary points to a position after the binary point. One can eliminate one parameter by normalizing the distributions such that one parameter becomes 1.*

Let us explain the concepts and ideas behind the analysis for Theorem 3.1. Part (a)) of the theorem follows simply by applying the Markov inequality to the expected absolute values of the individual coefficients. The interesting part of the theorem is stated in b). In order to identify the winner one needs to *isolate* the winner from other feasible solutions having a worse objective value. Furthermore, one needs to *separate* the winner from those infeasible solutions that have a better objective value than the winner. Our analysis is based on a *generalized Isolating Lemma* – i.e., a generalization of the well-known Isolating Lemma by Mulmuley, Vazirani and Vazirani [MVV87] – and a novel *Separating Lemma*.

The Isolating Lemma was originally presented in an article about RNC algorithms for perfect matchings [MVV87]. It is known, however, that the lemma does not only apply to the matching problem but also to general binary optimization problems with a linear objective function. The lemma states that the optimal solution of a binary optimization problem is unique with probability at least $\frac{1}{2}$ when choosing the coefficients of the objective function independently, uniformly at random from the set $\{1,2,\ldots,2n\}$. This is a very surprising and counterintuitive result as there might be an exponential number of feasible solutions whose objective values fall all into a polynomially large set, namely the set $\{1,2,\ldots,2n^2\}$, so that one can expect that an exponential number of solutions are mapped to the same objective value. The reason why the winner nevertheless is isolated is that the objective values of different solutions are not independent but the solutions represent subsets over a ground set of only n random numbers. We adapt the Isolating Lemma towards our continuous setting and generalize it towards piecewise continuous probability distributions. In particular, different coefficients may follow different continuous probability distributions. Suppose only the objective function is stochastic, and the feasible region is fixed arbitrarily. Let ϕ denote the maximum density parameter over all coefficients in the objective functions. Define the *winner gap* to be the difference between the objective value of the winner and the second-best feasible solution, provided there are at least two feasible solutions. The generalized Isolating Lemma states that the winner gap is a continuous random variable whose density function is bounded from above by $2\phi n$, and this bound is tight. From this result one can immediately derive the following lower bound on the

[1] with high probability, with probability $1-(nk)^{-\alpha}$, for every fixed $\alpha > 0$

size of the winner gap. For every $\varepsilon \in [0,1]$, the winner gap is lower-bounded by $\frac{\varepsilon}{2\phi n}$ with probability at least $1-\varepsilon$. As a consequence, it suffices to reveal only $O(\log(\phi n))$ bits of each coefficient of the objective function in order to identify the winner, **whp**.

We accompany the Isolating Lemma with a novel *Separating Lemma*, enabling us to separate the winner from infeasible solutions with better objective value than the winner. For the time being, consider any binary optimization problem in which a single constraint is stochastic. The difficulty in checking the feasibility with respect to this constraint is that it might be likely that there are many solutions that are exponentially close to the constraint hyperplane. Nevertheless, we will see that the optimal solution can be identified by inspecting only a logarithmic number of bits per input number, **whp**. The reason is that we do not need to check the feasibility of all solutions but only of some particular solutions. The *losers* are those solutions that have a rank higher than the winner (i.e., they have a better objective value) but they are infeasible because of the considered constraint. The *loser gap* is defined to be the minimal amount by which a loser (except for the solution 0^n) exceeds the constraint threshold. The Separating Lemma shows that the supremum of the density function of the loser gap is at most ϕn^2. Hence, for every $\varepsilon > 0$, the loser gap is at least $\frac{\varepsilon}{\phi n^2}$ with probability at least $1-\varepsilon$. Let us try to give some intuition about this result. If there are only a few losers then one can imagine that neither of them comes very close to a random or randomly perturbed hyperplane. However, there might be an exponential number of losers. In this case, however, the winner has a relatively low rank as there is an exponential number of solutions better than the winner; but this is very unlikely if the constraint hyperplane is likely to come very close to the good solutions which correspond to the losers. Seeing it the other way around, if there are many losers then the hyperplane is likely to be relatively far away from the losers, which might intuitively explain the phenomenon described by the Separating Lemma. Besides the loser gap, we study the so-called *feasibility gap* corresponding to the slack of the optimal solution with respect to the stochastic constraint. Essentially, we prove that the density functions of loser and feasibility gaps have the same maximum supremum so that the density of the feasibility gap is lower-bounded by $\frac{\varepsilon}{\phi n^2}$ as well. In fact, our analysis for loser and feasibility gaps is heavily based on symmetry properties between them.

Let us remark that, when analyzing the winner gap, it is assumed a random objective function and a fixed feasible region. In contrast, when analyzing loser and feasibility gaps, it is assumed a random constraint or a set of random constraints instead of a random objective function. In other words, the random expressions defining the objective function and the constraints are assumed to be stochastically independent. In fact, if the feasible region and the objective function are correlated, then winner, loser, and feasibility can not be lower-bounded by a polynomial. The optimization variant of the subset-sum problem (i.e., knapsack with profits equal to weights) is a simple counterexample. Lueker [Lue98b] proved that random instances of the subset-sum problem have usually exponentially small gaps.

3.4 Analysis of the Gap Properties

In this section we will formally define winner, loser, and feasibility gaps and prove upper bounds on the density functions of these random variables. Recall that, for a well-behaved continuous random variable X, f_X denotes the corresponding density function. Before we turn to the winner, loser, and feasibility

gaps let us prove the following lemma which we will frequently apply in our analysis.

Lemma 3.3. *Let $X_1,\ldots,X_n$ and X denote well-behaved continuous random variables. Suppose X always takes a value equal to one of the values of the variables $X_1,\ldots,X_n$. Then for all $t\in\mathbb{R}$, $f_X(t)\leq\sum_{i\in[n]}f_{X_i}(t)$.*

Proof. Let $X_1,\ldots,X_n$ and X denote well-behaved continuous random variables. Suppose X always takes a value equal to one of the values of the variables $X_1,\ldots,X_n$. We have to prove, $f_X(t)\leq\sum_{i\in[n]}f_{X_i}(t)$, for all $t\in\mathbb{R}$. For every $T\subseteq\mathbb{R}$,

$$\mathbf{Pr}[X\in T]\ \leq\ \mathbf{Pr}[\exists i\in[n]:X_i\in T]\ \leq\ \sum_{i\in[n]}\mathbf{Pr}[X_i\in T]\ .$$

Hence, for every continuous point $t\in\mathbb{R}$,

$$f_X(t)\ =\ \lim_{\substack{\varepsilon\to0\\ \varepsilon>0}}\frac{\mathbf{Pr}[X\in[t,t+\varepsilon]]}{\varepsilon}\ \leq\ \lim_{\substack{\varepsilon\to0\\ \varepsilon>0}}\sum_{i\in[n]}\frac{\mathbf{Pr}[X_i\in[t,t+\varepsilon]]}{\varepsilon}\ =\ \sum_{i\in[n]}\lim_{\substack{\varepsilon\to0\\ \varepsilon>0}}\frac{\mathbf{Pr}[X_i\in[t,t+\varepsilon]]}{\varepsilon}\ =\ \sum_{i\in[n]}f_{X_i}(t)\ .$$

This concludes the proof of Lemma 3.3. □

3.4.1 The Winner Gap

We consider an instance of a discrete optimization problem whose solutions are described by n binary variables $x_1,\ldots,x_n$. The set of feasible solutions (ground set intersected with the constraints) is now denoted by $\mathcal{S}\subseteq\{0,1\}^n$. Fix some arbitrary set $\mathcal{S}$ with at least two solutions. The objective function is denoted by c^Tx. The numbers $c_i\in\mathbb{R}$, $i=1,\ldots,n$, are assumed to be stochastic, that is, they are treated as independent random variables following possibly different, well-behaved continuous probability distributions with bounded density. Without loss of generality, we consider a maximization problem. Let $x^*=\operatorname{argmax}\{c^Tx\,|\,x\in S\}$ denote the winner and $x^{**}=\operatorname{argmax}\{c^Tx\,|\,x\in S\setminus\{x^*\}\}$ the second best solution. The *winner gap* Δ is defined to be the difference between the objective values of a best and a second best solution, that is,

$$\Delta\ =\ c^Tx^*-c^Tx^{**}\ .$$

The random variable Δ is well-behaved, i.e., Δ admits a piecewise continuous density function. This can be seen as follows. The probability space of Δ is $(c_1\times\ldots\times c_n)\subseteq\mathbb{R}^n$. Each pair of solutions defines a hyperplane in $\mathbb{R}^n$, consisting of all points where the two solutions have the same objective function value. These hyperplanes partition $\mathbb{R}^n$ into a finite number of polyhedral cells. Fix any cell $C\subseteq\mathbb{R}^n$. In this cell, x^* and x^{**} are uniquely determined. In particular, $(\Delta|C)=c^Tx^*-c^Tx^{**}$. Thus, the random variable $\Delta|C$ is a linear functional of the well-behaved continuous variables $c_1,\ldots,c_n$. Thus the density of $\Delta|C$ corresponds to the convolution of piecewise-continuous variables, and hence, it is piecewise continuous, too. Consequently, $f_\Delta=\sum_C\mathbf{Pr}[C]\,f_{\Delta|C}$ is piecewise continuous as well, so that Δ is well-behaved. The same kind of argument applies to other gap variables that we will define later in this thesis.

Lemma 3.4 (Generalized Isolating Lemma). *Let ϕ_i denote the density parameter of c_i, $1\leq i\leq n$, and $\phi=\max_i\phi_i$. For every choice of the feasible region $\mathcal{S}$ and every choice of the probability distributions of $c_1,\ldots,c_n$, the density function of Δ is bounded from above by $2\sum_{i\in[n]}\phi_i\leq2\phi n$.*

Proof. At first we observe, if there is a variable x_i that takes the same value in all feasible solutions, then this variable does not affect the winner gap and it can be ignored. Thus, without loss of generality, for every $i \in [n]$, there are at least two feasible solutions whose vectors differ in the i-th bit, i.e., with respect to the i-th variable. Under this assumption, we can define the winner gap with respect to bit position $i \in [n]$ by

$$\Delta_i \quad = \quad c^T x^* - c^T y \;\mid\; x^* = \text{argmax}\{c^T x \,|\, x \in \mathcal{S}\},\; y = \text{argmax}\{c^T x \,|\, x \in \mathcal{S}, x_i \neq x_i^*\} \;. \tag{3.1}$$

In words, Δ_i is the difference between the objective value of the winner x^* and the value of a solution y that is best among those solutions that differ in the i-th bit from x^*, i.e., the best solution in $\{x \in \mathcal{S} \,|\, x_i \neq x_i^*\}$.

Clearly, the best solution, $x^* = (x_1^*, \ldots, x_n^*)$, and the second best solution, $x^{**} = (x_1^{**}, \ldots, x_n^{**})$, differ in at least one bit, that is, there exists $i \in [n]$ such that $x_i^* \neq x_i^{**}$. If the best and the second best solution differ in the i-th bit then $\Delta = \Delta_i$. Thus, Δ is guaranteed to take a value also taken by at least one of the variables $\Delta_1, \ldots, \Delta_n$. Observe that the random variables $\Delta_1, \ldots, \Delta_n$ are well-behaved continuous, but there might be various kinds of dependencies among these variables. In the following, we will prove an upper bound of $2\phi_i$ on the density function for the random variable Δ_i, for every $i \in [n]$. Combining this bound with Lemma 3.3 immediately yields an upper bound of $2\sum_{i\in[n]} \phi_i$ for the density function of Δ, and the lemma is shown.

Let us fix an index $i \in [n]$. It only remains to be shown that the density of Δ_i is bounded from above by $2\phi_i$. We partition $\mathcal{S}$, the set of feasible solutions, into two disjoint subsets $\mathcal{S}_0 = \{x \in \mathcal{S} \,|\, x_i = 0\}$ and $\mathcal{S}_1 = \{x \in \mathcal{S} \,|\, x_i = 1\}$. Now suppose all random variables $c_k, k \neq i$ are fixed arbitrarily. Obviously, under this assumption, the winner among the solutions in $\mathcal{S}_0$ and its objective value are fixed as the objective values of the solutions in $\mathcal{S}_0$ do not depend on c_i. (In fact, there can be many solutions with maximum objective value.) Although the objective values of the solutions in $\mathcal{S}_1$ are not fixed, the winner of $\mathcal{S}_1$ is determined as well because the unknown outcome of the random variable c_i does not affect the order among the solutions in $\mathcal{S}_1$. For $j \in \{0,1\}$, let $x^{(j)}$ denote a winner among the solutions in $\mathcal{S}_j$. We observe $\Delta_i = |c^T x^{(1)} - c^T x^{(0)}|$ because the solutions x^* and y as defined in Equation (3.1) cannot be contained in the same set $\mathcal{S}_j$, $j \in \{0,1\}$. Hence, Δ_i takes either the value of $c^T x^{(1)} - c^T x^{(0)}$ or the value of $c^T x^{(0)} - c^T x^{(1)}$. Observe that the random variable c_i appears as a simple additive term in both of these expressions, and the density of c_i is at most ϕ_i. Therefore, both expressions describe random variables with density at most ϕ_i as well. (Observe that this holds, regardless of whether we assume that the other variables are fixed or random numbers.) Consequently, Lemma 3.3 yields that the density of Δ_i is at most $2\phi_i$. This completes the proof of the Generalized Isolating Lemma. □

Next we show that the given bound in the Generalized Isolating Lemma is tight.

Lemma 3.5. *Let ϕ_i denote the density parameter of c_i, $1 \leq i \leq n$. For every choice of $\phi_1, \ldots, \phi_n$, there is a way to define $\mathcal{S}$ and the distributions of $c_1, \ldots, c_n$, such that the maximum of the density function of Δ is at least $2\sum_{i\in[n]} \phi_i$.*

Proof. For every choice of n and $\phi_1, \ldots, \phi_n$, we have to present an example of an optimization problem with a stochastic optimization function $c^T x$, $c \in \mathbb{R}^n$, and a feasible region $\mathcal{S}$ such that ϕ_i is the supremum

of the density function describing coefficient c_i, and the supremum of the density function of Δ is at least $2\sum_{i\in[n]}\phi_i$. Let us simply define $\mathcal{S}=\{0,1\}^n$. Then the optimal solution x^* satisfies $x_i^*=1\Leftrightarrow c_i>0$. Let $k=\operatorname{argmin}_i(|c_i|)$. The second best solution x^{**} satisfies $x_i^{**}=x_i^*$, for all $i\neq k$, and $x_k^{**}=1-x_k^*$. Thus, $\Delta=\min_i(|c_i|)$. Assume the density functions of $c_1,\ldots,c_n$ are continuous around 0. Then

$$f_\Delta(0) \;=\; \lim_{\substack{\varepsilon\to 0\\ \varepsilon>0}}\frac{\mathbf{Pr}\left[\Delta\in[0,\varepsilon]\right]}{\varepsilon} \;=\; \lim_{\substack{\varepsilon\to 0\\ \varepsilon>0}}\frac{\mathbf{Pr}\left[\exists i\in[n]:|c_i|\in[0,\varepsilon]\right]}{\varepsilon} \;=\; \sum_{i\in[n]}\lim_{\substack{\varepsilon\to 0\\ \varepsilon>0}}\frac{\mathbf{Pr}\left[|c_i|\in[0,\varepsilon]\right]}{\varepsilon}$$

The last equality is due to the fact that for any $S\subset\mathcal{S}$ with $|S|>1$,

$$\lim_{\substack{\varepsilon\to 0\\ \varepsilon>0}}\frac{\mathbf{Pr}\left[\bigwedge_{i\in S}|c_i|\in[0,\varepsilon]\right]}{\varepsilon}=0\ .$$

Hence,

$$\lim_{\substack{\varepsilon\to 0\\ \varepsilon>0}}\frac{\mathbf{Pr}\left[|c_i|\in[0,\varepsilon]\right]}{\varepsilon} \;=\; \lim_{\substack{\varepsilon\to 0\\ \varepsilon>0}}\frac{\mathbf{Pr}\left[c_i\in[0,\varepsilon]\right]}{\varepsilon}+\lim_{\substack{\varepsilon\to 0\\ \varepsilon>0}}\frac{\mathbf{Pr}\left[c_i\in[-\varepsilon,0]\right]}{\varepsilon} \;=\; 2f_{c_i}(0)\ .$$

Finally, if we assume that the density functions of the coefficients take their maximum at 0, then $\phi_i=f_{c_i}(0)$ so that the maximum density is (at least) $2f_\Delta(0)=2\sum_{i\in[n]}\phi_i$, which completes the proof. □

3.4.2 Loser and Feasibility Gaps for a Single Constraint

We consider an instance of an optimization problem over n binary variables. The objective function can be fixed arbitrarily; we rank all solutions (feasible and infeasible) according to their objective value in non-increasing order. Solutions with the same objective values are ranked in an arbitrary but fixed fashion. The feasible region is described by a subset $\mathcal{S}\subseteq\{0,1\}^n$ intersected with the half-space $\mathcal{B}$ described by an additional linear constraint. Without loss of generality, the constraint is of the form $w^Tx\leq t$. The set $\mathcal{S}$ and the threshold t are assumed to be fixed. The coefficients $w_1,\ldots,w_n$ correspond to independent random variables following possibly different, well-behaved continuous probability distributions with bounded density. The *winner*, denoted by x^*, is the solution in $\mathcal{S}\cap\mathcal{B}$ with highest rank. The *feasibility gap* is defined by

$$\Gamma \;=\; \begin{cases} t-w^Tx^* & \text{if } \mathcal{S}\cap\mathcal{B}\neq\emptyset, \text{ and}\\ \perp & \text{otherwise.}\end{cases}$$

In words, Γ corresponds to the slack of the winner with respect to the constraint $w^Tx\leq t$. Observe that x^* might be undefined as there is no feasible solution. In this case, the random variable Γ takes the value $\perp$ (*undefined*). The domain of Γ is $\mathbb{R}_{\geq 0}\cup\{\perp\}$. The density function f_Γ over $\mathbb{R}_{\geq 0}$ is well-behaved continuous. The function f_Γ does not necessarily integrate to 1 but only to $1-\mathbf{Pr}\left[\Gamma=\perp\right]$. In the following, when talking about the density of Γ, we solely refer to the function f_Γ over $\mathbb{R}_{\geq 0}$, that is, we ignore the probability of the event $\{\Gamma=\perp\}$ as it is of no relevance to us.

A solution in $\mathcal{S}$ is called a *loser* if it has a higher rank than x^*, that is, the losers are those solutions from $\mathcal{S}$ that have a better rank than the winner, but they are cut off by the constraint $w^Tx\leq t$. The set of losers is denoted by $\mathcal{L}$. If there is no winner, as there is no feasible solution, then we define $\mathcal{L}=\mathcal{S}$. The

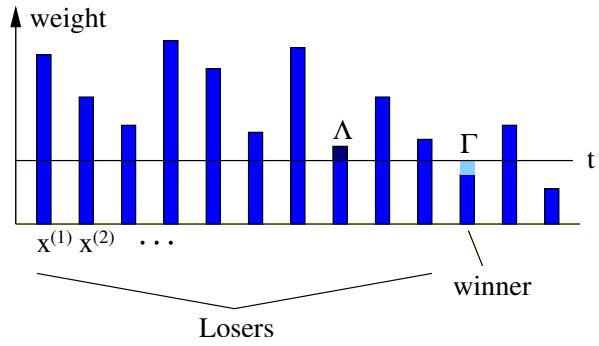

Figure 3.1: Loser and Feasibility gap: Solutions are listed according to the ranking. The height of the bars correspond to the weight $w^T x$. The winner is the first solution whose weight obeys the threshold t. The feasibility gap is the difference between threshold t and the weight of the winner. All solutions left of the winner are losers. The loser gap is the smallest amount by which any loser violates the threshold t.

loser gap is defined by

$$\Lambda = \begin{cases} \min\{w^T x - t \mid x \in \mathcal{L}\} & \text{if } \mathcal{L} \neq \emptyset, \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

In case $\mathcal{L} \neq \emptyset$, the loser that determines Λ, i.e., $\operatorname{argmin}_{x\in\mathcal{L}}\{w^T x\}$, is called *minimal loser*. As for the feasibility gap, when talking about the density of the loser gap, we solely refer to the function f_Λ over $\mathbb{R}_{\geq 0}$ and ignore the probability of the event $\{\Lambda = \bot\}$.

Our goal is to upper-bound the densities of Γ and Λ. Observe that the solution 0^n is different from all other solutions in $\mathcal{S}$ as its feasibility does not depend on the outcome of the random coefficients $w_1, \ldots, w_n$. Suppose $0^n \in \mathcal{S}$ and 0^n has the highest rank among all solutions in $\mathcal{S}$. Then one can enforce $\Gamma = 0$ by setting $t = 0$. Similarly, one can enforce $\Lambda \to 0$ for $t \to 0$. For this reason, we need to exclude the solution 0^n from our analysis. Assuming $0^n \notin \mathcal{S}$, the following theorem shows that both the loser and the feasibility gap are likely to have polynomial size.

Lemma 3.6 (Separating Lemma). *Let ϕ_i denote the density parameter of w_i, for all $i \in [n]$, and define $\phi = \max_{i\in[n]} \phi_i$. Suppose $0^n \notin \mathcal{S}$. Then the densities of Γ and Λ are upper bounded by $n\sum_{i=1}^n \phi_i \leq \phi n^2$.*

Proof. We will heavily use symmetry properties between the two gaps. At first, we will prove an upper bound of ϕn on the density of the loser gap under the assumption that the ranking satisfies a certain monotonicity property. Next, we will show that the suprema of the density functions for the loser and the feasibility gap are identical for worst-case choices of the threshold t. This way, the upper bound on the density of the loser gap holds for the feasibility gap as well. Then we will show that monotonicity assumption for the feasibility gap can be dropped at the cost of an extra factor n, thereby achieving an upper bound of ϕn^2 on the density of the feasibility gap. Finally, by applying the symmetry between loser and feasibility gap again, we obtain the same result for the loser gap.

A ranking of the solutions is called *monotone* if all pairs of solutions $x, y \in \mathcal{S}$, x having a higher rank than y, satisfy that there exists $i \in [n]$ with $x_i > y_i$. When considering the binary solution vector as subsets

of $[n]$, a ranking is *monotone* if each subset S is ranked higher than all its proper subsets $T \subset S$. This property is naturally satisfied for maximization problems having a linear objective function with positive coefficients, but also if all solutions in $\mathcal{S}$ have the same number of ones.

Lemma 3.7. *Suppose $0^n \notin \mathcal{S}$ and the ranking is monotone. Then f_Λ is bounded from above by $\sum_{i\in[n]} \phi_i$.*

Proof. Fix $t \in \mathbb{R}$ arbitrarily. As in the proof for the winner gap, we define n random variables $\Lambda_1, \ldots, \Lambda_n$ with maximum densities $\phi_1, \ldots, \phi_n$ such that at least one of them takes the value of Λ. For $i \in [n]$, define $\mathcal{S}_i = \{x \in \mathcal{S} \mid x_i = 1\}$ and $\bar{\mathcal{S}}_i = \mathcal{S} \setminus \mathcal{S}_i$. Let $\bar{x}^{(i)}$ denote the winner from $\bar{\mathcal{S}}_i$, i.e., the solution with highest rank in $\bar{\mathcal{S}}_i \cap \mathcal{B}$. Now let $\mathcal{L}_i$ denote the set of losers from $\mathcal{S}_i$ with respect to $\bar{x}^{(i)}$, that is, $\mathcal{L}_i = \{x \in \mathcal{S}_i \,|\, x \text{ has a higher rank than } \bar{x}^{(i)}\}$. If $\bar{x}^{(i)}$ does not exist then we set $\mathcal{L}_i = \mathcal{S}_i$. Now define the minimal loser of $\mathcal{L}_i$, $x^{(i)}_{\min} = \operatorname{argmin}\{w^T x \,|\, x \in \mathcal{L}_i\}$, and

$$\Lambda_i = \begin{cases} w^T x^{(i)}_{\min} - t & \text{if } \mathcal{L}_i \neq \emptyset, \text{ and} \\ \bot & \text{otherwise.} \end{cases}$$

Observe that $\mathcal{L}_i$ is not necessarily a subset of $\mathcal{L}$ as $\bar{x}^{(i)}$ can have a lower rank than x^*. In fact, $x^{(i)}_{\min}$ can be feasible so that Λ_i can take negative values. The reason for this "wasteful" definition is that it yields some kind of independence that we will exploit in the following arguments.

Claim A: The density of Λ_i is at most ϕ_i.
This claim can be seen as follows. The definitions above ensure $\mathcal{L}_i \subseteq \mathcal{S}_i$ while $\bar{x}^{(i)} \in \bar{\mathcal{S}}_i$. Suppose all variables $w_j, j \neq i$ are fixed arbitrarily. We prove that the density of Λ_i is bounded by ϕ_i under this assumption, and hence the same bound holds for randomly chosen $w_j, j \neq i$ as well. The winner $\bar{x}^{(i)}$ can be determined without knowing the outcome of w_i as $\bar{x}^{(i)} \in \bar{\mathcal{S}}_i$ and for all solutions in $\bar{\mathcal{S}}_i$ the i-th entry is zero. Observe that $\mathcal{L}_i$ is fixed as soon as $\bar{x}^{(i)}$ is fixed, and so is $x^{(i)}_{\min}$, as the i-th bit of all losers in $\mathcal{L}_i$ is one. Hence, w_i is not affected by the determination of $x^{(i)}_{\min}$. As the i-th bit of $x^{(i)}_{\min}$ is set to one, the random variable Λ_i can be rewritten as $\Lambda_i = w^T x^{(i)}_{\min} - t = \kappa + w_i$, where κ denotes a fixed quantity and w_i is a random variable with density at most ϕ_i. Consequently, the density of Λ_i is bounded from above by ϕ_i.

Claim B: If $\Lambda \neq \bot$, then there exists $i \in [n]$ such that Λ takes the value of Λ_i.
To show this claim, let us first assume that x^* exists and $\mathcal{L} \neq \emptyset$. Let $x_{\min} \in \mathcal{L}$ denote the *minimal loser*, i.e., $x_{\min} = \operatorname{argmin}\{w^T x \,|\, x \in \mathcal{L}\}$. By definition, $x_{\min}$ has a higher rank than x^*. Because of the monotonicity of the ranking, there exists $i \in [n]$ such that $x^* \in \bar{\mathcal{S}}_i$ and $x_{\min} \in \mathcal{S}_i$. From $x^* \in \bar{\mathcal{S}}_i$, we conclude $x^* = \bar{x}^{(i)}$. Consequently, $x_{\min} \in \mathcal{L} \cap \mathcal{S}_i = \mathcal{L}_i$ so that $x_{\min} = x^{(i)}_{\min}$. Hence, $\Lambda = \Lambda_i$. Now suppose x^* does not exist. Then $\mathcal{L} = \mathcal{S}$ and $\mathcal{L}_i = \mathcal{S}_i$, for all $i \in [n]$. Thus, there exists $i \in [n]$ with $x_{\min} = x^{(i)}_{\min}$ because $\mathcal{S} = \bigcup_{i\in[n]} \mathcal{S}_i$ as $0^n \notin \mathcal{S}$. Finally, if $\mathcal{L} = \emptyset$ then the claim follows immediately as $\Lambda = \bot$.

Now applying Lemma 3.3 to the Claims A and B immediately yields the lemma. □

The following lemma shows that upper bounds on the density function of the loser gap also hold for the feasibility gap and vice versa. For a given threshold t, let $R(t) \subseteq \mathbb{R}_{\geq 0}$ denote the set of points at which the distribution functions of $\Lambda(t)$ and $\Gamma(t)$ are differentiable. As $\Lambda(t)$ and $\Gamma(t)$ are well-behaved continuous, the points in $\mathbb{R} \setminus R(t)$ have measure 0 and, hence, can be neglected.

Lemma 3.8. *Suppose* $0^n \notin \mathcal{S}$. *Then* $\sup_{t\in\mathbb{R}} \sup_{s\in R(t)} f_{\Gamma(t)}(s) = \sup_{t\in\mathbb{R}} \sup_{s\in R(t)} f_{\Lambda(t)}(s)$.

Proof. We take an alternative view on the given optimization problem. We interpret the problem as a bicriteria problem. The feasible region is then the whole set $\mathcal{S}$. The first criteria is the rank which is to be minimized (high ranks are small numbers). The second criteria is the weight, defined by the linear function $w^T x$, which is to be minimized as well. A solution $x \in \mathcal{S}$ is called *Pareto-optimal* if there is no higher ranked solution $y \in \mathcal{S}$ with smaller (or equal) weight. For simplicity assume that no two solutions have the same weight. This assumption is justified as the probability that there are two solutions with same weight is 0.

Next we show that winners and minimal losers of the original optimization problem correspond to Pareto-optimal solutions of the bicriteria problem. First, let us observe that the winner x^* with respect to any given weight threshold t is a Pareto-optimal solution for the bicriteria problem because there is no other solution with a higher rank and weight at most $t \geq w^T x^*$. Moreover, for every Pareto-optimal solution x there is also a threshold t such that x is a winner, i.e., $t = w^T x$.

The same kind of characterization holds for minimal losers as well. Recall, for a given threshold t, the minimal loser is defined to be $x_{\min} = \operatorname{argmin}\{w^T x | x \in \mathcal{L}\}$. We claim that there is no other solution y that simultaneously achieves a higher rank and smaller weight than $x_{\min}$. This can be seen as follows. Suppose y is a solution with higher rank than $x_{\min}$. If $w^T y \leq t$ then $y \in \mathcal{B}$ and, hence, $x_{\min}$ would not be a loser. However, if $w^T y \in (t, w^T x_{\min})$ then y and $x_{\min}$ would both be losers, but y instead of $x_{\min}$ would be minimal. Furthermore, for every Pareto-optimal solution x there is also a threshold t such that x is a loser. This threshold can be obtained by setting $t \to w^T x$, $t < w^T x$.

Now let us describe winner and loser gap in terms of Pareto-optimal solutions. Let $\mathcal{P} \subseteq \mathcal{S}$ denote the set of Pareto-optimal solutions with respect to the fixed ranking and the random weight function $w^T x$. Then feasibility and loser gaps are characterized by

$$\begin{aligned} \Gamma(t) &= \min\{t - w^T x \mid x \in \mathcal{P}, w^T x \leq t\} \ , \\ \Lambda(t) &= \min\{w^T x - t \mid x \in \mathcal{P}, w^T x > t\} \ . \end{aligned}$$

For a better intuition, we can imagine that all Pareto-optimal solutions are mapped onto a horizontal line such that $x \in \mathcal{P}$ is mapped to the point $w^T x$. Then $\Gamma(t)$ is the distance from the point t on this line to the closest Pareto point *left* of t (i.e., less than or equal to t), and $\Lambda(t)$ is the distance from t to the closest Pareto point *strictly right* of t (i.e., larger than t). Now let f be a measure over $\mathbb{R}$ describing the density of the Pareto points on the line, that is, for every $t \in \mathbb{R}$,

$$f(t) = \lim_{\varepsilon\to 0} \frac{\mathbf{Pr}\left[\exists x \in \mathcal{P} : w^T x \in [t, t+\varepsilon]\right]}{\varepsilon}$$

Let us remark that f is not a probability density function as it does not integrate to one. It is a measure for the expected number of Pareto-optimal solutions over the line, that is, $\int_{-\infty}^{t} f(r)dr$ describes the expected number of Pareto-optimal solutions in the interval $[-\infty, t]$. In the following, we will first prove $\sup_t f_{\Lambda(t)}(0) = f(t) = \sup_t f_{\Gamma(t)}(0)$, and afterwards we will show, for every $s \in \mathbb{R}$, $\sup_t f_{\Lambda(t)}(s) \leq$

$\sup_t f_{\Lambda(t)}(0)$ as well as $\sup_s f_{\Gamma(t)}(s) \leq \sup_t f_{\Gamma(t)}(0)$. Combining these equations proves the theorem.

We start by showing $f_{\Lambda(t)}(0) = f(t) = f_{\Gamma(t)}(0)$, for any $t \in \mathbb{R}$. The distribution function of the loser gap $\Lambda(t)$ is defined by

$$F_{\Lambda(t)}(s) \;=\; \mathbf{Pr}\left[\Lambda(t) \leq s\right] \;=\; \mathbf{Pr}\left[\exists x \in \mathcal{P} : w^T x \in (t, t+s]\right] \;=\; \mathbf{Pr}\left[\exists x \in \mathcal{P} : w^T x \in [t, t+s]\right] \;.$$

Let us assume, without loss of generality, that the density $f_{\Lambda(t)}$ is defined to be the right derivative of the distribution $F_{\Lambda(t)}$. Then

$$f_{\Lambda(t)}(0) \;=\; F'_{\Lambda(t)}(0) \;=\; \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \frac{F_{\Lambda(t)}(\varepsilon) - F_{\Lambda(t)}(0)}{\varepsilon} \;=\; \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \frac{\mathbf{Pr}\left[\exists x \in \mathcal{P} : w^T x \in [t, t+\varepsilon]\right]}{\varepsilon} \;=\; f(t) \;,$$

where the third equation follows because $F_{\Lambda(t)}(0) = 0$. The feasibility gap behaves in a symmetric fashion. As it describes the distance to the closest Pareto point on the left instead of the right of t, we obtain

$$f_{\Gamma(t)}(0) \;=\; \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \frac{\mathbf{Pr}\left[\exists x \in \mathcal{P} : w^T x \in [t, t-\varepsilon]\right]}{\varepsilon} \;=\; f(t) \;.$$

Therefore, $f_{\Lambda(t)}(0) = f(t) = f_{\Gamma(t)}(0)$, for every $t \in \mathbb{R}$. As a consequence, $\sup_t f_{\Lambda(t)}(0) = \sup_t f(t) = \sup_t f_{\Gamma(t)}(0)$.

It remains to generalize this result towards other parameters of the density functions than 0. First, let us consider the loser gap. Λ corresponds to the distance between a given threshold t and the *closest* Pareto point that is (strictly) right of t. Consequently, the density of Λ at a given point on the line should not decrease when the threshold is moved towards this point. Formally, for every $s \in R(t)$,

$$\begin{aligned}
f_{\Lambda(t)}(s) &= \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \frac{F_{\Lambda(t)}(s+\varepsilon) - F_{\Lambda(t)}(s)}{\varepsilon} \\
&= \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \frac{\mathbf{Pr}\left[\exists x \in \mathcal{P} : w^T x \in [t, t+s+\varepsilon]\right] - \mathbf{Pr}\left[\exists x \in \mathcal{P} : w^T x \in [t, t+s]\right]}{\varepsilon} \\
&\leq \lim_{\substack{\varepsilon \to 0 \\ \varepsilon > 0}} \frac{\mathbf{Pr}\left[\exists x \in \mathcal{P} : w^T x \in [t+s, t+s+\varepsilon]\right]}{\varepsilon} \;=\; f_{\Lambda(t+s)}(0) \;.
\end{aligned}$$

Analogously, $f_{\Gamma(t)}(s) \leq f_{\Gamma(t+s)}(0)$. Thus, $\sup_t \sup_s f_{\Lambda(t)}(s) = \sup_t f_{\Lambda(t)}(0) \sup_t f_{\Gamma(t)}(0) = \sup_t \sup_s f_{\Gamma(t)}(s)$. This completes the proof of Lemma 3.8. □

Combining the two lemmas, we observe that the density of the feasibility gap $\Gamma(t)$ is at most $\sum_{i \in [n]} \phi_i$, provided that the ranking is monotone and $0^n \notin \mathcal{S}$. Next we extend this result towards general rankings by breaking the original problem in subproblems. We partition $\mathcal{S}$ into the sets $\mathcal{S}^{(k)} = \{x \in \mathcal{S} \mid \sum_i x_i = k\}$, for $1 \leq k \leq n$. Observe that each of these sets contains only solutions with the same number of ones, and hence, satisfies the monotonicity condition. Let $\Gamma^{(k)}(t)$ denote the feasibility gap over the set $\mathcal{S}^{(k)}$. By Lemma 3.7, the density of $\Gamma^{(k)}(t)$ is at most $\sum_{i \in [n]} \phi_i$, for every $t \in \mathbb{R}$. Furthermore, $\Gamma(t)$ takes the value of one of the variables $\Gamma^{(k)}(t)$, $1 \leq k \leq n$ because the winner of one of the subproblems is the winner

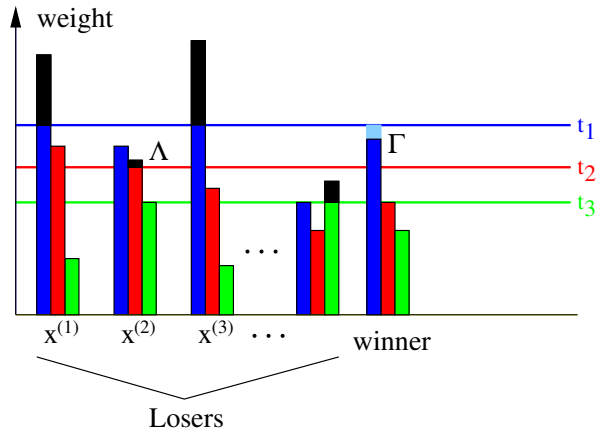

Figure 3.2: Loser and Feasibility gap for multiple constraints: Solutions are listed according to the ranking. Blue, red and green bars give the weight of each solution according to three different weight functions. The corresponding thresholds t_1, t_2 and t_3 are colored accordingly. The winner is the first solution obeying all thresholds. The feasibility gap is the smallest slack over the different weight functions of the winner. All solutions left of the winner are losers. The loser gap for an individual loser is the largest amount by which this loser violates any threshold (given in black). The loser gap Λ is the minimum of the individual loser gaps over all losers.

of the original problem. As a consequence of Lemma 3.3, the density of $\Gamma(t)$ is at most $n\sum_{i\in[n]}\phi_i$, for every continuous point $t \in \mathbb{R}$. Let us remark that such a kind of argument cannot directly be applied to the loser gap. By applying Lemma 3.8, however, the bound for the feasibility gap holds for the loser gap as well. Hence, Lemma 3.6 is shown. □

3.4.3 Loser and Feasibility Gap for Multiple Constraints

Assume there are $k \geq 2$ stochastic constraints. Without loss of generality, these constraints are of the form $w_j^T x \leq t_j$, for $j \in [k]$, and the sets of solutions satisfying these constraints are $\mathcal{B}_1, \ldots, \mathcal{B}_k$, respectively. We generalize the definition of feasibility and loser gap as follows. Given a set of solutions $\mathcal{S} \subseteq \{0,1\}^n$ and a ranking, the *winner* x^* is the highest ranked solution in $\mathcal{S} \cap \mathcal{B}_1 \cap \ldots \cap \mathcal{B}_k$. The *feasibility gap for multiple constraints* is the minimal slack of x^* over all stochastic constraints, that is, $\Gamma = \min_{j\in[k]}\{t_j - w_j^T x^*\}$, if x^* exists, and $\Delta = \perp$, otherwise.

The set of *losers* $\mathcal{L}$ consists of all solutions from $\mathcal{S}$ that have a rank higher than x^*. Observe that a loser only needs to be infeasible with respect to one of the k constraints. In particular, it is not true that the weight values of each loser are likely to be far away from the corresponding thresholds t_j, $j \in [k]$; not even if we consider only those constraints for which the respective loser is infeasible. Fortunately, however, we do not need such a property in the application of the loser gap. For every loser, one only needs a single constraint that renders the loser infeasible. Therefore, we define the *loser gap for k constraints* by

$$\Lambda = \begin{cases} \min_{x\in\mathcal{L}} \max_{j\in[k]}\{w_j^T x - t_j\} & \text{if } \mathcal{L} \neq \emptyset \text{ ,and} \\ \perp & \text{otherwise.} \end{cases}$$

Figure 3.2 illustrates this definition.

Lemma 3.9. *Let ϕ denote the maximum density parameter of all coefficients in the stochastic constraints. Suppose $0^n \notin \mathcal{S}$. Then $\mathbf{Pr}[\Gamma < \varepsilon] \leq \varepsilon k \phi n^2$ and $\mathbf{Pr}[\Lambda < \varepsilon] \leq \varepsilon k \phi n^2$, for all $\varepsilon \in \mathbb{R}_{\geq 0}$.*

Proof. First we show the bound for the feasibility gap. Let x^* denote the winner and suppose $\Gamma \leq \varepsilon$, for some $\varepsilon \in \mathbb{R}_{\geq 0}$. Then there exists $j \in [k]$ with $t_j - w_j^T x^* \leq \varepsilon$. Thus,

$$\mathbf{Pr}[\Gamma \leq \varepsilon] \leq \sum_{j \in [k]} \mathbf{Pr}\left[t_j - w_j^T x^* \leq \varepsilon\right] \quad .$$

For each individual $j \in [k]$, we can apply the Separating Lemma assuming that the set of feasible solutions with respect to all other constraints is fixed as the coefficients in this constraint are stochastically independent from the other constraints. This way, we obtain $\mathbf{Pr}[\Gamma \leq \varepsilon] \leq k \cdot \varepsilon \phi n^2$.

Next, we turn our attention to the loser gap. Unfortunately, we cannot generalize the bound on the loser gap from one to multiple constraints in the same way as we generalized the feasibility gap as the loser gap for multiple constraints does not correspond to the minimal loser gap over the individual constraints. Instead we will make use of the result for the feasibility gap established above. Assume $\Lambda \leq \varepsilon$, for some $\varepsilon \in \mathbb{R}_{\geq 0}$. Then there exists a loser x satisfying $\forall j \in [k] : w_j^T x - t_j \leq \varepsilon$. Let x_L denote the loser with this property that is ranked highest. Consider a relaxed variant Π' of the given optimization problem Π where the thresholds of all stochastic constraints are increased by ε, i.e., we have constraints $w_j^T x \leq t_j + \varepsilon,\ j \in [k]$. Observe that x_L is feasible in the relaxed problem Π' and, by the definition of x_L, no higher ranked solution is feasible. Thus, x_L is the winner of Π'. Since $t_j < w_j^T x_L \leq t_j + \varepsilon$ for some $j \in [k]$, the feasibility gap Γ' of the relaxed problem is smaller than ε. Hence, $\Lambda \leq \varepsilon$ implies $\Gamma' \leq \varepsilon$. Finally, applying the bound $\mathbf{Pr}[\Gamma' \leq \varepsilon] \leq \varepsilon k \phi n^2$ derived in the first part of the proof yields $\mathbf{Pr}[\Lambda \leq \varepsilon] \leq \varepsilon k \phi n^2$. □

3.4.4 Proof of Theorem 3.1

First we prove part (a) of the theorem. The probability that the absolute value of any of the kn stochastic numbers $w_{i,j}$, $(i \in [k], j \in [n])$, is larger than $\mu(nk)^{1+\alpha}$, for arbitrary $\alpha > 0$, is

$$\mathbf{Pr}\left[\exists (i,j) : |w_{i,j}| > \mu(nk)^{1+\alpha}\right] \leq \sum_{i,j} \mathbf{Pr}\left[|w_{i,j}| > \mu(kn)^{1+\alpha}\right] \leq \sum_{i,j} \frac{1}{(kn)^{1+\alpha}} = (nk)^{-\alpha}.$$

Next we prove part (b) of the theorem. First, suppose that the objective function is the only stochastic expression. We reveal b bits after the binary point of each coefficient c_i $(1 \leq i \leq n)$. Then we know the value of each c_i up to a absolute error of 2^{-b}. We will deal with this lack of precise information in terms of rounding, that is, we will think of rounding down all c_i to the next multiple of 2^{-b} causing a "rounding error" of less than 2^{-b} for each number.

Lemma 3.10. *Let ϕ denote the maximum density parameter over the coefficients $c_1, \ldots, c_n$ in the objective function. When revealing b bits after the binary point of each coefficient then the winner is uniquely determined with probability at least $1 - 2n^2\phi/2^b$.*

Proof. Let $\lfloor c \rfloor$ be the vector that is obtained by rounding each entry c_i of vector c down to the next multiple of 2^{-b}. Consider any two solutions $x, x' \in \mathcal{S}$. We have

$$|(c^T x - c^T x') - (\lfloor c \rfloor^T x - \lfloor c \rfloor^T x')| = |(c - \lfloor c \rfloor)^T (x - x')| < n2^{-b} \ ,$$

as $c_i - \lfloor c_i \rfloor < 2^{-b}$, for all $i \in [n]$. Hence, if the winner gap Δ (with respect to the exact coefficients $c_1, \ldots, c_n$) is at least $n2^{-b}$ then the rounding can not affect the optimality of the winner. In this case the winner is uniquely determined after revealing only b bits of each coefficient c_i. Let ϕ_Δ denote the supremum of the density of Δ. Since Δ is non-negative, $\mathbf{Pr}[\Delta < x] \leq x\phi_\Delta$, for all $x \in \mathbb{R}_{\geq 0}$. Using the generalized Isolating Lemma we obtain $\phi_\Delta \leq 2\phi n$. Setting $x = n2^{-b}$ yields $\mathbf{Pr}\left[\Delta < n2^{-b}\right] \leq \frac{2n^2\phi}{2^b}$. □

Next suppose only some of the constraints are stochastic and the objective function is adversarial. Let k' denote the number of stochastic constraints. Without loss of generality, the constraints are of the form $w_j^T x \leq t_j$, $j \in [k']$. Assume we reveal b bits after the binary point of each coefficient in the k' constraints.

Lemma 3.11. *Let ϕ denote the maximum density parameter over all coefficients in the stochastic constraints. When revealing b bits after the binary point of each coefficient, then the winner is uniquely determined with probability at least $1 - 2^{-b}k'n^3\phi$.*

Proof. As we round down, infeasible solutions might become feasible whereas feasible solutions stay feasible. (For constraints of the form $w_j^T x \geq t_j$ one would need to round up to the next multiple of 2^{-b}.) To ensure that the winner is uniquely determined it suffices to upper bound the maximum possible error in each constraint caused by the rounding of the coefficients. If this error is smaller than the loser gap then rounding cannot change the feasibility status of any loser, i.e., all infeasible solutions that have rank higher than the winner stay infeasible.

In order to apply the bound on the loser gap given in Lemma 3.9, let us first assume $0^n \notin \mathcal{S}$. The rounding error in each expression is at most $n2^{-b}$. The definition of the loser gap for multiple stochastic constraints states that for every loser x there is a constraint $j \in [k']$ such that $w_j^T x - t_j \geq \Gamma$. Therefore, if $\Gamma \geq n2^{-b}$ then every loser stays infeasible with respect to at least one constraint after rounding. Applying Lemma 3.9, the probability for this event is at least $1 - k'\phi n^3/2^b$.

The solution 0^n can influence the loser gap in two ways. At first, 0^n might belong to the set of losers and thus decrease the loser gap. However, rounding the coefficients w_i does not change the objective value of this solution. Thus, 0^n stays infeasible under rounding and the loser gap with respect to all other solutions in unaffected. At second, 0^n might be the winner, which would result in a different loser set when including 0^n to $\mathcal{S}$. In this case, however, 0^n has a higher rank than the previous winner so that the set of losers can only shrink. Therefore, the solution 0^n cannot decrease the loser gap and the probability that any loser becomes feasible can be estimated with the Separating Lemma as before. □

Now suppose some constraints and the objective function are stochastic. Let $k = k' + 1$ denote the number of stochastic expressions. The probability that winner and loser gap are sufficiently large, as described in the two lemmas above, is $1 - k'\phi n^3/2^b - 2\phi n^2/2^b \geq 1 - k\phi n^3/2^b$, for $n \geq 2$. This implies that the winner is uniquely determined when revealing $b = O(\log(k\phi n))$ bits, **whp**. This completes the proof of Theorem 3.1. □

3.5 Characterizing Polynomial Smoothed Complexity

Based on the gap properties, we aim at characterizing which discrete optimization problems have polynomial time algorithms under random perturbations. We can apply Theorem 3.1 to our smoothed analysis framework and bound the number of bits necessary to determine the winner. Taking into account that the domain of the initial adversarial choices for the stochastic coefficients is $[-1,1]$ or $[0,1]$, we observe that ϕ-perturbations yield coefficients with an expected absolute value of at most $\mu = O(1+\frac{1}{\phi})$. In order to simplify the notation, we restrict the parameter for the perturbation to the interesting case $\phi \geq 1$. This leads to the following result on the overall number of bits that need to be revealed per stochastic input number.

Corollary 3.12. *For any fixed perturbation model f, the winner is uniquely determined when revealing $O(\log(\phi nk))$ bits of each stochastic coefficient,* **whp**.

Based on our smoothed analysis framework we define *polynomial smoothed complexity*. Fix any binary optimization problem Π and any perturbation model f. Let I_N denote the set of all unperturbed instances of length N that the adversary may specify. The definition of the input length N needs some clarification as the coefficients in the stochastic expressions are assumed to be real numbers. We define that each of these numbers has a virtual length of one. (This way, we ensure $N \geq kn$.) The bits of the stochastic numbers can be accessed by asking an oracle in time $O(1)$ per bit. The bits after the binary point of each coefficient are revealed one by one from left to right. The deterministic part of the input does not contain real numbers and can be encoded in an arbitrary fashion. For an instance $I \in I_N$, let $I + f_\phi$ denote the random instance that is obtained by a ϕ-perturbation of I. We say that Π has *smoothed polynomial complexity* if and only if it admits an algorithm $\mathcal{A}$ whose running time T satisfies

$$\exists \alpha, \beta > 0 : \forall \phi \geq 1 : \forall N \in \mathbb{N} : \max_{I \in I_N} \mathbf{E}\left[\left(T(I+f_\phi)\right)^\alpha\right] \leq \beta \phi N \ .$$

This definition of polynomial smoothed complexity follows more or less the way how polynomial complexity is defined in average-case complexity theory, adding the requirement that the running time should be polynomially bounded not only in N but also in ϕ. It is not difficult to show that the assumption on the running time of $\mathcal{A}$ is equivalent to requiring that there exists a polynomial $P(N,\phi,\frac{1}{\varepsilon})$ such that for every $N \in \mathbb{N}, \phi \geq 1, \varepsilon \in [0,1]$, the probability that the running time of $\mathcal{A}$ exceeds $P(N,\phi,\frac{1}{\varepsilon})$ is at most ε. Observe that this does not imply that the expected running time is polynomially bounded. To enforce expected polynomial running time, the exponent α in the definition of polynomial smoothed complexity should have been placed outside instead of inside the expectation. The reason for not defining polynomial smoothed complexity based on the expected running time is that this is not a sufficiently robust notion. For example, an algorithm with expected polynomial running time on one machine model might have expected exponential running time on another machine model. In contrast, the above definition yields a notion of polynomial smoothed complexity that does not vary among classes of machines admitting polynomial time simulations among each other. Although polynomial smoothed complexity does not always imply polynomial bounds on the expected running time, we will show that several of our algorithmic results yield expected polynomial running time on a RAM.

We show that the smoothed complexity of a binary optimization problem Π can be characterized in terms of the worst-case complexity of Π. Theorem 3.1 shows that one usually needs to reveal only a logarithmic number of bits per real-valued input number. This suggests a connection between pseudo-polynomial worst-case running time and polynomial average-case complexity. For a binary optimization problem Π, let Π_u denote the corresponding optimization problem in which all numbers in the stochastic expression are assumed to be integers in unary representation instead of randomly chosen real-valued numbers. The following theorem holds for any fixed perturbation model f.

Theorem 3.13. *A binary optimization problem* Π *has polynomial smoothed complexity if and only if* $\Pi_u \in \mathsf{ZPP}$.

In other words, Π has polynomial smoothed complexity if it admits a (possibly randomized) algorithm whose (expected) running time for all instances is pseudo-polynomial in the stochastic constraints and polynomial in the remaining input. Notice that the expectation is over the randomization of the algorithm, not over the instances. This characterization immediately shows that strongly NP-hard optimization problems do not have polynomial smoothed complexity, unless $\mathsf{ZPP}=\mathsf{NP}$. This observation might not sound very surprising, as the hardness of strongly NP-hard problems does not rely on large or precisely specified input numbers. Observe, however, that the strong NP-hardness of a problem does not immediately rule out the possibility of a polynomial average-case complexity. For example, the TSP problem (on a complete graph) with edge lengths drawn uniformly at random from $[0,1]$ might have a polynomial average-case complexity. Our theorem, however, shows that it does not have a polynomial smoothed complexity, unless $\mathsf{ZPP} = \mathsf{NP}$. The more sophisticated part of the theorem is the other direction stating that every binary optimization problem admitting a pseudo-polynomial time algorithm has polynomial smoothed complexity. This result is based on the Generalized Isolating Lemma and the Separating Lemma. The idea is as follows. We design efficient verifiers checking whether a solution computed with a certain precision is actually the optimal solution of Π. The success probability of these verifiers is analyzed with the help of the gap properties. Using an adaptive rounding procedure, we increase the precision until we can certify that the computed solution is optimal. The overall running time of this meta-algorithm is polynomial if the algorithm computing solutions with bounded precision has pseudo-polynomial running time.

Proof. At first, we prove that the existence of a randomized pseudo-polynomial time algorithm for a binary optimization problem Π implies polynomial smoothed complexity for Π. We design an algorithm with polynomial smoothed complexity calling the pseudo-polynomial algorithm with higher and higher precision until the solution found is certified to be optimal. We describe verifiers, that, based on the first b bits after the binary point of each coefficient in the stochastic expressions, either certify optimality or reports FAILURE, stating that it has not sufficient information to ensure optimality. So the algorithm has access to the first b bits only, which corresponds to rounding down the numbers to the next multiple of 2^{-b}. Again we will interpret the lack of precise information as a rounding error. In the following, $\lfloor w \rfloor$ denotes the value of w rounded down to the next multiple of 2^{-b}.

Certifying optimality For a moment assume that only the objective function is stochastic. Without loss of generality, consider a maximization problem with objective function $c^T x$. At first, we compute an optimal solution x' for the problem with the rounded coefficients $\lfloor c_1 \rfloor, \ldots, \lfloor c_n \rfloor$. To check whether x' is optimal with respect to the original cost vector c, we generate another vector $\bar{c}$ of rounded coefficients. This time the rounding depends on the computed solution x'. For all i with $x'_i = 1$, we set $\bar{c}_i := \lfloor c_i \rfloor$ and for all i with $x'_i = 0$, we set $\bar{c}_i := \lceil c_i \rceil = \lfloor c_i \rfloor + 2^{-b}$. Observe that the function $\delta(x) = c^T x - \bar{c}^T x$ is maximal for $x = x'$. Next we compute an optimal solution x'' for the problem with the vector $\bar{c}$. If $x' = x''$ then x' simultaneously maximizes $\delta(x)$ and $\bar{c}^T x$. Consequently, it maximizes $\bar{c}^T x + \delta(x) = c^T x$ as well and, hence, x' corresponds to the true winner x^*. Thus, the algorithm outputs x' as a certified winner if $x' = x''$ and reports FAILURE otherwise. Observe, if the winner gap is large enough so that the winner is uniquely determined in the sense of Lemma 3.10, then the algorithm will always compute a certified winner. Hence, the probability that the algorithm is successful is at least $1 - 2n^2\phi/2^b$, corresponding to the bound given in Lemma 3.10. Observe that $\Gamma \geq n2^{-b}$ is a sufficient but not a necessary condition to certify the optimality of the winner with b revealed bits per coefficient. So the verifier might declare optimality of the computed solution even if $\Gamma < n2^{-b}$.

Certifying feasibility Now we show how to deal with stochastic constraints. Without loss of generality, we assume that all stochastic constraints are of the form $w_j^T x \leq t_j$, $1 \leq j \leq k'$. For constraints of the form $w^T x \geq t$, we would need to round up instead of rounding down. First we compute a certified winner, denoted by x', using the rounded coefficients in the stochastic constraints. If the objective function is not stochastic, then there is no need to certify the winner. As we round down all coefficients in the stochastic constraints we ensure that feasible solutions stay feasible. However, we have to detect infeasible solutions that become feasible due to the rounding and displace the true winner. Hence, we need to check whether x' is indeed feasible with respect to the original constraints. This would be trivial if the exact values of all constraint vectors $w_1, \ldots, w_k$ were available. However, we want to check the feasibility using only the knowledge of the first b bits after the binary point of each coefficient. Assume the solution x is infeasible with respect to the j-th constraint and becomes feasible due to rounding. Then $\lfloor w_j \rfloor^T x \leq t_j < w_j^T x$ and hence $t_j - \lfloor w_j \rfloor^T x < w_j^T x - \lfloor w_j \rfloor^T x \leq n2^{-b}$, i.e. the slack of x in the j-th constraint is less than $n2^{-b}$ with respect to the rounded vector $\lfloor w_j \rfloor$. Our verifier will use this property and classifies x' as possibly infeasible if it has slack less than $n2^{-b}$ for any of the k constraints with respect to the rounded coefficients. This way, we have two failure sources. At first, there might be a loser that becomes feasible because of rounding. As seen in the proof of Lemma 3.11, this can happen only if the loser gap is smaller than $n2^{-b}$. At second, the true winner can be rejected since its slack is less than $n2^{-b}$. This happens only if the feasibility gap is smaller than $n2^{-b}$. Applying Lemma 3.9 yields that the probability that one of these events happens is at most $2k'\phi n^3/2^b$.

Adaptive rounding procedure. Consider a binary optimization problem Π with n binary variables and k stochastic expressions. Assume there exists an algorithm $\mathcal{A}$ for the special variant of Π, where the domain of the coefficients in the stochastic expressions is restricted to $\mathbb{Z}$. Furthermore, assume that the worst case running time of $\mathcal{A}$ is bounded by some polynomial in W, n, k and N, where W denotes the

largest absolute value of any coefficient in the stochastic expressions. Recall that N specifies the size of the deterministic part of the input. We use the following adaptive rounding scheme. We start by revealing $b = \log(k\phi n^3)$ bits of each coefficient in the stochastic expressions. We obtain integer coefficients by scaling these numbers by 2^b. Now we use the algorithm $\mathcal{A}$ to obtain a solution and use the verifiers to test for optimality. In case of FAILURE we increment b by one and iterate until the verifiers conclude optimality of the computed solution.

To analyze the running time of $\mathcal{A}$ we need to estimate W, the largest absolute value of any integer in the stochastic expressions. It is the product of two factors. The first factor, $W_1 = 2^b$, is due to the scaling and depends on the number of revealed bits after the binary point of each coefficient. The second factor W_2 corresponds to the integer part of the largest absolute value of any coefficient. This way, $W = W_1 W_2$. We have to show that there exists a polynomial $P(N, \phi, \frac{1}{\varepsilon})$ such that for every $N \in \mathbb{N}, \phi > 0, \varepsilon \in [0,1]$: $\mathbf{Pr}\left[T > P(N,\phi,\frac{1}{\varepsilon})\right] \leq \varepsilon$. The running time of $\mathcal{A}$ is polynomial in W, n, k and N. As n, k and N are deterministic, it suffices to show such a polynomial bound for W. Let us first prove an upper bound on W_2. Let $r_{j,i}$ be the random variable that is added to the i-th coefficient in the j-th stochastic expression when perturbing the instance. Recall that the mean of the absolute values of the probability distribution that defines the perturbation model, is a constant, denoted E. It holds $W_2 \leq \max_{j\in[k],i\in[n]} |r_{j,i}| + 1$. Hence, for every $\alpha \geq 1$,

$$\mathbf{Pr}\left[\max_{j\in[k],i\in[n]} |r_{j,i}| > \alpha E\right] \leq \sum_{j\in[k],i\in[n]} \mathbf{Pr}\left[|r_{j,i}| > \alpha E\right] \leq \frac{nk}{\alpha} \ ,$$

where the last inequality holds by an application of the Markov Inequality. Hence, $\mathbf{Pr}[W_2 > \alpha E + 1] \leq nk/\alpha$. Setting $\varepsilon = 2nk/\alpha$, it holds $W_2 \leq \alpha E + 1 = \frac{2nkE}{\varepsilon} + 1$ with probability at least $1 - \varepsilon/2$. Next consider the term $W_1 = 2^b$. There are two reasons why the checkers can declare the computed solutions to be suboptimal or infeasible. First, the random instance happens to have a small winner gap or a small loser gap such that the winner is not uniquely determined. Using Theorem 3.10 and 3.11, the probability for this event is at most $k\phi n^3/2^b$. Secondly, the feasibility checker reports false negatives due to a small feasibility gap. This happens with probability at most $k'\phi n^3/2^b$. Allowing a failure probability of $\varepsilon/4$ for each event, we obtain $W_1 = 2^b = \frac{k\phi n^3 4}{\varepsilon}$. Therefore, with probability at least $1-\varepsilon$, none of these bad events occur and it holds $W = W_1 W_2 \leq \left(\frac{2nkE}{\varepsilon} + 1\right) \frac{k\phi n^3 4}{\varepsilon}$. As k and E are assumed to be constant and $n \leq N$, there exists a polynomial $P(N, \phi, \frac{1}{\varepsilon})$ such that for all $N \in \mathbb{N}, \phi > 0, \varepsilon \in [0,1]$: $\mathbf{Pr}\left[W > P(N,\phi,\frac{1}{\varepsilon})\right] \leq \varepsilon$.

From polynomial smoothed complexity to pseudo-polynomial running time Finally, we need to show that polynomial smoothed complexity of a binary optimization problem Π implies the existence of a randomized pseudo-polynomial algorithm for Π. Since we are aiming for a pseudo-polynomial time algorithm, we can assume that all numbers in the stochastic expressions are integers. Let M denote the largest absolute value of these numbers. The idea is to perturb all numbers only slightly such that the perturbation changes the value of each expression by at most $\frac{1}{2}$. To ensure that the set of feasible solutions is not changed by the perturbation, we relax all constraints by $\frac{1}{2}$, i.e., we replace $w^T x \leq t$ by $w^T x \leq t + \frac{1}{2}$ for all stochastic constraints. We then use an algorithm with polynomial smoothed complexity to compute an optimal solution x^* for the perturbed problem. By bounding the error due to

the random perturbation, x^* can be shown to be optimal for the original problem as well.

Let us describe the proof idea in more detail. Our smoothed analysis framework assumes that all numbers in the stochastic expressions fall into the interval $[-1,1]$ (or $[0,1]$) before they are perturbed. To adapt our problem to this framework, we first scale all input numbers in the stochastic expressions by M^{-1}. Consequently, we have to ensure that the perturbation changes the value of an expression by at most $1/(2M)$. In particular, we will allow only perturbations that change each individual number by at most $1/(2Mn)$. We call such a perturbation *proper*. For the uniform distribution, we could simply set $\phi = 2Mn$. However, we have to deal with arbitrary families of distributions, as defined in our smoothed analysis framework, and they do not necessarily have a finite domain. The idea is to choose ϕ large enough so that a random perturbation is proper with probability at least $1/2$. Recall that the perturbation model is described by the density function f with density parameter $\phi = 1$. For other values of ϕ, we scale f appropriately. By our assumptions on f, it holds $\int |t| f_\phi(t)dt = E/\phi$ for some fixed $E \in \mathbb{R}$. Let r be a random variable following f_ϕ. Setting $\phi = 4n^2kEM$ and applying the Markov inequality yields $\mathbf{Pr}\left[|t| > \frac{1}{2nM}\right] = \mathbf{Pr}\left[|t| > \frac{2nkE}{\phi}\right] \leq \frac{1}{2nk}$. Our perturbation draws kn of these random variables. The probability that the perturbation is proper, i.e., the probability that their absolute value is at most $\frac{1}{2nM}$, is $1/2$.

Consider any binary optimization problem Π with polynomial smoothed complexity. Polynomial smoothed complexity implies that the problem admits an algorithm $\mathcal{A}$ whose running time can be bounded polynomially in n and ϕ, with any constant probability. In particular, there exists a polynomial $P(n,\phi)$ such that the probability that the running time exceeds $P(n,\phi)$ is at most $\frac{1}{4}$. We use $\mathcal{A}$ as a subroutine in order to obtain a pseudo-polynomial algorithm. This algorithm works as follows. At first, it generates a perturbation and checks whether it is *proper*, If it is proper, then it runs $\mathcal{A}$ for at most $P(n,\phi)$ time steps. If $\mathcal{A}$ has not finished within this time bound, the algorithm returns FAILURE. Let Q be the event that the perturbation is proper. Observe that for any two events A and B it holds $\mathbf{Pr}[A \wedge B] \geq \mathbf{Pr}[A] + \mathbf{Pr}[B] - 1$. Therefore, the success probability of our algorithm is

$$\mathbf{Pr}[Q \wedge (T \leq P(n,\phi))] \;\geq\; \mathbf{Pr}[Q] - \mathbf{Pr}[T > P(n,\phi)] \;\geq\; \frac{1}{4} \;.$$

The running time of this algorithm is pseudo-polynomial because $\phi = O(Mn^2k)$. Hence, $\Pi_u \in \mathsf{ZPP}$. This completes the proof of Theorem 3.13. □

3.6 Algorithmic Applications

Let us illustrate the strength of Theorem 3.13 by giving some algorithmic applications to some well-known optimization problems and comparing these results with previous work on the probabilistic analysis of optimization problems. To our knowledge and besides our results for the knapsack problem (Chapter 2) and the constrained shortest path problem (Chapter 1), there are no NP-hard optimization problems that are previously known to have polynomial smoothed complexity.

Due to its simple structure, the knapsack problem has been the subject of extensive research for a long time. The multi-dimensional knapsack problem is a natural generalization with multiple packing

constraints instead of only one.

$$\begin{array}{lll} \text{maximize} & c^T x & \\ \text{subject to} & Ax \leq b, & A \in \mathbb{R}^{k \times n}_{\geq 0}, b \in \mathbb{R}^k \\ & x \in \{0,1\}^n, c \in \mathbb{R}^n_{\geq 0}. & \end{array}$$

Dyer and Frieze [DF89] generalized Lueker's result on the expected integrality gap to multiple constraints and extended the ideas of the core algorithm from Goldberg and Marchetti-Spaccamela [GMS84] to prove the following result. For any fixed $\varepsilon > 0$, there is an algorithm for the multi-dimensional knapsack problem whose running time does not exceed a polynomial $P(n)$ with probability at least $1 - \varepsilon$, provided the number of constraints k is constant and the coefficients in the constraints as well as in the objective function are chosen uniformly at random from $[0,1]$. Their analysis, however, does not yield polynomial average-case complexity as the dependence of the failure probability on the running time bound is not bounded by a polynomial. For a constant number of constraints, the multi-dimensional knapsack problem admits an algorithm that is pseudo-polynomial in the largest weight, i.e., the largest entry in the constraint matrix A [KPP04]. The algorithm is a straight-forward extension of the standard dynamic programming algorithm for the knapsack problem with a single constraint and uses a dynamic programming table of dimension $1 + k$ and size $n(nC)^k$, where C denotes the largest weight occurring in the instance. Hence, Theorem 3.13 implies

Corollary 3.14. *The multi-dimensional knapsack problem with a constant number of constraints has polynomial smoothed and, consequently, polynomial average-case complexity under random perturbations of all constraints.*

Observe that our result does not rely on a random objective function. By viewing the multi-dimensional knapsack problem as a multiobjective optimization problem, one can easily see that our result holds if at least k of the objectives are random, allowing one objective to be adversarial. If the number of constraints is unbounded, then the problem is strongly NP-hard [KPP04], so that, in general, these problems have no polynomial smoothed complexity.

Notice that the multi-dimensional knapsack problem is a special version of general 0/1-integer programming, which allows only non-negative entries in the constraint matrix and, hence, belongs to the class of packing problems. The dynamic programming algorithm for the multi-dimensional knapsack problem can be applied to general 0/1-integer programming as well by extending the table to negative weights. Let C denote the largest absolute value of any entry in the constraint matrix. Then the dynamic programming table has size $n(2nC+1)^k$, with indices running from $-Wn$ to $+Wn$ for the dimensions that correspond to constraints. The running time is $n(2nC+1)^k$, which is pseudo-polynomial for constant k. In fact, any integer linear program with a constant number of constraints has a pseudo-polynomial time algorithm [Sch86]. This yields the following corollary.

Corollary 3.15. *General 0/1-integer programming with a constant number of constraints has polynomial smoothed and, consequently, polynomial average-case complexity under random perturbations of all constraints.*

It is important to interpret this corollary in the right way. It does not prove that all problems that can be formulated as a 0/1-integer program have polynomial smoothed complexity. In the formulation of these problems variables usually appear in more than one constraint. Furthermore, the constraint matrix is usually sparse, containing many zero entries. Perturbing the constraints independently, as our smoothed analysis framework assumes, destroys the structure of the problem. So instead of averaging over "similar" instances of the problem, we would be averaging over instances of some other structure that might be easier to solve on average. We will see in Section 3.9 how to strengthen our analysis by allowing zero entries in the constraints that are not perturbed. This way we are able to extend our analysis to problems with sparse constraint vectors.

Scheduling Problems

The problem of scheduling to minimize the weighted number of tardy jobs is defined by n jobs each of which has a processing time p_i, a due date d_i, and a penalty c_i, which has to be paid if job i has not been finished at due date d_i. The jobs shall be scheduled on a single machine such that the sum of the penalties is minimized. In terms of n binary variables $x_1, \ldots, x_n$, the objective is to minimize $c^T x$ where $x_i = 1$ if job i is not finished in time. Observe that the problem is essentially solved once these binary variables are determined as we can assume without loss of generality that an optimal schedule executes the selected jobs in the order of non-decreasing due dates. (All tardy jobs are scheduled after all non-tardy jobs. The non-tardy jobs can be rearranged into non-decreasing due-date order without making any of the jobs tardy. This can be shown by observing that swapping two neighboring non-tardy jobs does not change the tardiness of the jobs. By a repeated application of this swapping operation, any optimal schedule can be transformed into an optimal schedule with non-decreasing due date order.) Notice that the feasible region is completely specified by the processing times and the due dates. The objective, however, is a linear function $c^T x$.

Using dynamic programming, the problem can be solved in time $O(n^2 C)$, where C denotes the largest penalty for any job [Par95]. Hence, the problem has polynomial smoothed complexity for stochastic penalties.

Corollary 3.16. *The problem of scheduling to minimize the weighted number of tardy jobs has polynomial smoothed and, consequently, polynomial average-case complexity under random perturbations of the penalties.*

3.7 Multi-Criteria Optimization Problems

We introduced the semi-random input model to avoid that the randomization destroys the underlying combinatorial structure of the problem at hand. So it is quite natural to interpret the stochastic constraints as different criteria (or objective functions) over the solution space of the underlying combinatorial structure. As already mentioned in Section 1.1, if there are multiple criteria to be optimized simultaneously then usually one of them is declared to be the objective function, and the others are formulated in form of

a constraint with an appropriate threshold. To show that such a problem has polynomial smoothed complexity, Theorem 3.13 suggests to find an algorithm which is polynomial in the size of the deterministic part and pseudo-polynomial in the coefficients of the stochastic expressions. By applying elementary coding theory, this problem can be reduced to the problem of finding a pseudo-polynomial time algorithm for the exact version of the single criterion problem, that is, for a given k, one seeks a solution with objective value exactly k instead of the maximum (minimum) possible objective value. For example, the exact version of the shortest path problems asks for a path of length exactly k.

The underlying idea of this reduction is as follows. Let W denote the size of the largest integer domain over the stochastic expressions. The inequality of a single constraint can be reduced to the disjunction of at most W equalities. Furthermore, the problem of finding a solution with minimum (maximum) objective value can be reduced to at most W tests for the existence of a solution with objective function value exactly i. Hence, we can solve the original problem by solving W^k problems with linear equations. Furthermore, every system of linear equations over binary variables with integer coefficients can be transformed into a single linear equation with the same set of solutions [Sal75]. The encoding of several equations into a single one increases the sizes of the coefficients, but the increase is bounded by a polynomial if the number of equations is fixed. So we can use the pseudo-polynomial algorithm for the exact version of the single criterion problem to solve the problem where all criteria are combined into one. By the above arguments, such an algorithm has pseudo-polynomial running time as well. Combining this observation with Theorem 3.13 yields the following result.

Corollary 3.17. *Let Π be a (single objective) binary optimization problem. Suppose the exact version of Π admits an algorithm with pseudo-polynomial running time. Assume we deal with multicriteria variants of Π by choosing one criterion as the objective function and bounding the remaining criteria by arbitrary thresholds. Then any multicriteria variant of Π with a constant number of criteria has polynomial smoothed complexity, provided that all criteria are stochastic.*

Examples for problems that allow pseudo-polynomial time algorithms for the exact version of the single-criteria problem are shortest path, spanning tree, and matching [PY00, BP98, MVV87]. Hence, their multicriteria variants have polynomial smoothed complexity. A similar approach was used in [PY00] to derive approximation schemes for multiobjective optimizations problems. One does not always need to assume that all criteria are of stochastic nature. For example, the bicriteria variant of the shortest path problem, i.e., the constrained shortest path problem, has an algorithm whose running time is pseudo-polynomial with respect to the objective function and another algorithm that is pseudo-polynomial with respect to the additional constraint. Applying Theorem 3.13 directly to these algorithms yields polynomial smoothed complexity for the constrained shortest path problem even when either only the objective function or the additional constraint is stochastic. This observation complies with our findings in Section 1.5.1, which bound the complexity of computing the Pareto curve for the bicriteria version of the shortest path problem. This result too relies on the randomness of only one of the criteria.

3.8 Expected Polynomial Running Time

The main advantage of polynomial smoothed/average-case complexity is its robustness under different machine models. Besides, it allows a nice characterization of binary optimization problems under random inputs in terms of the problems' worst-case complexity. However, the guaranty on the running time provided by polynomial smoothed/average-case complexity is weaker than the guaranty that the expected running time is polynomial. Making additional assumptions, we can use our analysis to conclude expected polynomial running time for certain binary optimization problems.

We use again our adaptive rounding scheme, which increases the precision of the stochastic input numbers until the computed solution is certified to be optimal. We assume that the running time of the last iteration of this meta-algorithm dominates the cumulative running time of all previous iterations. In fact, it suffices if all previous iterations have cumulative running time at most a factor n^l larger than the running time of the last iteration, for some constant $l \in \mathbb{R}$. In order to obtain expected polynomial running time under random perturbations, one needs an algorithm with "pseudo-linear" instead of pseudo-polynomial running time. The only parameter in the running time that depends on the random perturbations is W, the largest absolute value of any (rounded and scaled) coefficient appearing in a stochastic expression.

We use the notation from the proof of Theorem 3.13. We divided W into two parts $W = W_1 W_2$, where W_1 is due to the scaling by factor $2^b = W_1$ and depends on b, the number of revealed bits after the binary point of each coefficient. The second factor W_2 corresponds to the integer part of the largest absolute value of any coefficient. In the proof of Theorem 3.13 we have shown the following bounds on the random variables W_1 and W_2.

$$\begin{aligned} \mathbf{Pr}\left[W_1 > 2^b\right] &\leq 2k\phi n^3/2^b, \quad \text{for any } b \in \mathbb{N}, \text{ and} \\ \mathbf{Pr}\left[W_2 > \alpha E + 1\right] &\leq nk/\alpha\,, \qquad \text{for any } \alpha \in \mathbb{R}_{>0}. \end{aligned}$$

These bounds allow 2^b as well as α to grow linearly with the reciprocal of the failure probability, thus, $W \approx 2^b \alpha E$ can grow quadratically. Hence, these bounds do not allow to conclude a polynomial expected running time. Therefore, we will restrict the choice for the perturbation model by allowing only probability distributions whose tail function exhibits an exponential decay. More precisely, we assume that if X is drawn according to perturbation model f then there exists some $E \in \mathbb{R}_{\geq 0}$, such that $\mathbf{Pr}\left[|X| > \alpha E\right] \leq 2^{-\alpha}$, for every $\alpha \geq 2$. For example, the Gaussian and the exponential distribution have this property as well as all distributions with finite domain.

In the following analysis we exploit the fact that it is very unlikely that any of the coefficients in the stochastic expressions is much larger that E. Define event $\mathcal{E}$ by $(W_2 \leq n\log(nk)E + 1) \wedge (W_1 \leq 2^n)$. As $\mathbf{Pr}\left[W_2 > nlog(nk)E\right] \leq kn\mathbf{Pr}\left[X > n\log(nk)E\right] \leq 2^{-n}$ and $\mathbf{Pr}\left[W_1 > 2^n\right] \leq 2k\phi n^3/2^n$ it holds $\mathbf{Pr}\left[\neg\mathcal{E}\right] \leq O(k\phi n^3/2^n)$. Assume that we have a pseudo-linear running time bound of $T \leq N^l W$ for some constant $l \in \mathbb{R}$. In case of $\neg\mathcal{E}$, we use a brute force enumeration of all 2^n possible $0/1$-vectors which takes time $2^n P(N)$ for some polynomial $P(N)$. Additionally, we contribute the time for the unsuccessful adaptive rounding scheme, which is at most $N^l 2^n(n\log(nk)E + 1)$, to the enumeration process. Hence, the total

running time is $O(2^n P(N))$ for some other polynomial $P(N)$. Next define random variable

$$W_1' = \begin{cases} W_1 & \text{in case of } \mathcal{E}, \\ 0 & \text{otherwise .} \end{cases}$$

The expectation of W_1' is

$$\mathbf{E}\left[W_1'\right] = \sum_{i=0}^{n} \mathbf{Pr}\left[W_1' = 2^i\right] 2^i \leq \sum_{i=0}^{n} \mathbf{Pr}\left[W_1 \geq 2^i\right] 2^i \leq \sum_{i=0}^{n} \frac{4\phi k n^3}{2^i} 2^i = O(\phi k n^4) \ .$$

Let T denote the running time of our algorithm. As $\mathbf{E}[W_1'] = \mathbf{Pr}[\mathcal{E}]\,\mathbf{E}[W_1' \mid \mathcal{E}]$,

$$\begin{aligned} \mathbf{E}[T] &\leq \mathbf{Pr}[\mathcal{E}] \cdot \mathbf{E}\left[N^l W_1 W_2 \mid \mathcal{E}\right] + \mathbf{Pr}[\neg\mathcal{E}] \cdot 2^n P(N) \\ &\leq N^l (n\log(nk)E + 1) \cdot \mathbf{Pr}[\mathcal{E}]\,\mathbf{E}\left[W_1' \mid \mathcal{E}\right] + \mathbf{Pr}[\neg\mathcal{E}] \cdot 2^n P(N) \\ &\leq N^l (n\log(nk)E + 1)\,\mathbf{E}\left[W_1'\right] + O(k\phi n^3/2^n) \cdot 2^n P(N) \\ &= O(N^l (n\log(nk)E + 1)\, 4\phi k n^4 + k\phi n^3 P(N)) \ . \end{aligned}$$

Corollary 3.18. *Assume, the perturbation model f satisfies $\mathbf{Pr}[|X| > \alpha E] \leq 2^{-\alpha}$, for some $E \in \mathbb{R}_{\geq 0}$ and all $\alpha \geq 2$. If a binary optimization problem has pseudo-polynomial running time $O(poly(N)W)$, then this problem allows an algorithm with expected polynomial running time.*

Remark 3.19. *The requirement of a pseudo-linear running time is a strong restriction for binary optimization problems with $k > 1$ stochastic constraints, as the standard approaches, like dynamic programming, usually exhibit a running time of $poly(n)W^k$. For example, in case of dynamic programming, every additional constraint will add another dimension to the dynamic programming table.*

Such pseudo-linear algorithms exist on a uniform RAM, e.g., for the knapsack problem, the problem of scheduling to minimize weighted tardiness or the constrained shortest path problem. Hence, assuming the uniform RAM model, all these problems admit algorithms with expected polynomial running time under random perturbations.

3.9 Zero-Preserving Perturbations

One criticism of the smoothed analysis of the Simplex algorithm is that the additive perturbations destroy the zero-structure of an optimization problem, as it replaces zeros with small values. See also the discussion in [ST01]. The same criticism applies to the zero-structure in binary programs. It turns out, however, that our probabilistic analysis in Section 3.4 is robust enough to deal with zero-preserving perturbations. In particular, we can extend our semi-random input model introduced in Section 3.2 by allowing the coefficients in the stochastic expressions to be fixed to zero instead of being a random variable. In the model of smoothed analysis, this corresponds to strengthen the adversary by avoiding the perturbation of these zero-coefficients.

For the expression $w^T x$, let Z be the set of indices i with w_i fixed to zero. Let $\mathcal{S}$ be the set of solutions. We call two solutions $x_1, x_2 \in \mathcal{S}$ equivalent, if they differ only in variables from Z. This way, Z defines equivalence classes on $\mathcal{S}$ with respect to the expression $w^T x$. Obviously, $w^T x$ evaluates to the same value for solutions in the same equivalence class.

In the following we show that the Separation Lemma is unaffected by zero preserving perturbations. Assume we fix $n-l$ coefficients to zero and, without loss of generality, these are the last coefficients, i.e., $Z = \{l+1, l+2, \ldots, n\}$. Hence, $w = (w_1, \ldots, w_l, 0, \ldots, 0)$. Recall that solutions are $0/1$ vectors of length n. Solutions that agree in the first l positions are equivalent. Observe that only the highest ranked solution in each equivalence class is relevant for the loser and feasibility gap. For the purpose of the analysis, we virtually remove all other solutions. For convenience, let us call the resulting solution set $\mathcal{S}$. In order to apply the Separating Lemma, we define a reduced problem with l variables corresponding to the first l variables of the original problem. The reduced solution set $\mathcal{S}'$ is obtained accordingly by cutting the last $n-l$ entries of each solution in $\mathcal{S}$. Notice that $x \in \mathcal{S}$ and the corresponding $x' \in \mathcal{S}'$ have the same weight $w^T x$. The rank of a $x' \in \mathcal{S}'$ is the rank of the corresponding solution $x \in \mathcal{S}$. This way, solution x is a loser (winner) if and only if the corresponding x' is a loser (winner). Hence, the feasibility gap as well as the loser gap are identical in the original and the reduced problem.

A similar argument can be used to show that the Generalized Isolating Lemma stays valid with respect to equivalence classes, that is, the winner gap is defined to be the difference in profit between the best and the second best equivalence class. However, it might be very likely that there are many optimal solutions, as the winning equivalence class might have many solutions. Notice that this effects the procedure that verifies optimality for stochastic objective functions, described in the proof of Theorem 3.13. In particular, the two solutions x' and x'', which are optimal with respect to the rounded cost vectors $\lfloor c \rfloor$ and $\bar{c}$, only have to be in the same equivalence class to certify optimality, which can be checked easily.

Algorithmic Application

Using zero-preserving perturbations, we can apply our analysis to the General Assignment Problem (GAP), which is defined as follows.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{k} \sum_{j=1}^{n} p_{ij} x_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{n} w_{ij} x_{ij} \leq c_i, \quad i = 1, \ldots, k \\
& \sum_{i=1}^{k} x_{ij} = 1, \qquad j = 1, \ldots, n \\
& x_{ij} \in \{0,1\}, \qquad \text{for all } i \in [k], j \in [n] \quad .
\end{aligned}
$$

Intuitively, the goal is to pack n items into k bins with capacities $c_1, \ldots, c_k$. Packing item $j \in [n]$ into bin $i \in [k]$ occupies w_{ij} units of the capacity of bin i and yields a profit of p_{ij}. Even for two bins, the GAP-problem does not allow an FPTAS, unless $\mathsf{P} = \mathsf{NP}$ [MT90]. In fact, the problem is strongly NP-hard for an unbounded number of bins. If however, the number of bins is a constant, then GAP can be solved in

pseudo-polynomial time using standard dynamic programming as follows. Let $T(j, W_1, \ldots, W_k)$ denote the maximum profit over solutions that have available items $\{1, \ldots, j\}$ and whose weight of all items in bin i is exactly W_i, for all $i \in [k]$. All non-trivial entries can be computed by the following recursion.

$$T(j, W_1, \ldots, W_k) = \max_{i \in [k]} \{T(j-1, V_1, \ldots, V_k) + p_{ij} \mid V_i = W_i - w_{ij}, V_l = W_l, \text{ for all } l \neq i\}$$

Let W denote the largest weight, i.e., $W = \max_{j \in [n]} \max_{i \in [k]} w_{ij}$. Then the size of the dynamic programming table is at most $n(nW)^k$. Hence, the running time is at most $kn(nW)^k$.

How does the GAP problem fit into our framework? We have a vector of kn binary variables $(x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, x_{22}, \ldots, x_{2n}, \ldots, x_{k1}, x_{k2}, \ldots x_{kn})$ and the corresponding vector of profits. The weight constraint for bin $l \in [k]$ uses the weight vector with all entries w_{ij}, $i \neq l$, set to zero. These entries are declared to be fixed to zero, so they are not touched by the perturbation. This way, each w_{ij} appears in just one constraint and a perturbation of the profit and the constraint vectors has the same effect as perturbing each p_{ij} and w_{ij} individually. This yields

Corollary 3.20. *The General Assignment Problem with a constant number of constraints/bins has polynomial smoothed complexity if weights and profits are randomly perturbed.*

The discussion above reveals a limitation of our analysis. For this purpose, consider the multiple knapsack problem, a special case of the GAP problem, where the weight of an item is the same for all bins, i.e., $w_{ij} = w_j$, for all $i \in [k]$. Using the program formulation of the GAP problem, the weights w_j reoccur in the different constraints for the different knapsacks. The perturbation of the constraints, however, have to be independent such that the perturbed instance is, in general, no multiple knapsack instance any more. We observe that for a sensible application of our analysis, each variable that describes the problem instance, must appear at most once in the stochastic constraints of the linear program formulation. Or, seeing it another way, the coefficients in the stochastic constraints must allow independent perturbations without destroying the structure of the problem.

3.10 Other Aspects

Smoothed complexity and approximation schemes

If a binary optimization problem Π has polynomial smoothed complexity when perturbing only the coefficients in the objective function, then Π also admits an absolute fully polynomial randomized approximation scheme. The idea is to perturb the instance only slightly, resulting in a very similar optimal objective function value. The set of feasible solutions is not affected by the perturbation. In order to bound the running time, we can exploit the positive influence of the added randomness. Let us give some more details.

Given a (worst-case) instance of Π, the coefficients in the objective function are normalized such that the largest absolute value is 1. Subsequently, the normalized coefficients are perturbed using the uniform perturbation model with parameter ϕ. As each coefficient changes by at most $1/\phi$, the difference in the objective values of any two solutions can change by at most n/ϕ by this perturbation. Hence, the optimal

solution of the perturbed instance is, with respect to the unperturbed objective function, at most n/ϕ away from the optimum. In order to obtain polynomial expected running time of our approximation scheme we repeatedly generate perturbations until we find one that allows a small running time. In particular, according to the definition of polynomial smoothed complexity, we can fix some polynomial bound B such that the running time of the algorithm is smaller than B with probability at least $1/2$. We run the algorithm for at most B time steps. If the algorithm has not finished, we generate a new perturbation and try again. As the perturbations are independent, one needs 2 iterations on average until a solution is found. Hence, the expected running time is $2B$.

The approximation scheme only allows to bound the absolute error. If however, the optimum of the normalized instance can be lower bounded by some polynomial in $1/n$, then we obtain a randomized FPTAS.

The opposite direction works as well. Suppose Π is a binary optimization problem admitting a fully polynomial approximation scheme. It is well known that such an approximation scheme can be transformed into a pseudo-polynomial algorithm, cf. [GJ79]. The running time of this algorithm is pseudo-polynomial only with respect to the coefficients in the objective function. Thus, Theorem 3.13 shows that Π has polynomial smoothed complexity when perturbing only the objective function.

This shows that for the considered class of problems, there exists a correlation between problems that allow an FPTAS and problems with polynomial smoothed complexity for stochastic linear objective functions.

Observe that these results rely solely on the analysis of the winner gap. The results based on the loser and feasibility gaps yield polynomial smoothed complexity for problems for which no fully polynomial approximation scheme is known like, e.g., constrained spanning tree, or even for problems that cannot be approximated within any polynomial-time computable factor like, e.g., 0/1 programming with a constant number of constraints. In fact, the problems to which these gaps apply are exactly those problems admitting a bicriteria approximation scheme. In other words, our results show that the impact of randomly perturbing the objective function and/or the constraints on the computational complexity of a problem is comparable with the impact of relaxing the same set of expressions. The advantage of smoothed analysis against (bicriteria) approximation schemes, however, is that it yields optimal and feasible instead of almost optimal and almost feasible solutions.

Euclidean optimization problems. On a first view, the Euclidean variants of TSP and Steiner tree might look like interesting candidates for problems with polynomial smoothed complexity. Karp [Kar77] in a seminal work on the probabilistic analysis of algorithms studied the TSP problem in a model in which n points are drawn uniformly and independently from the unit square. A natural extension of this model would be to assume that first an adversary chooses points in the unit cube or ball and then one applies a multi-dimensional Gaussian perturbation. We claim, however, that neither Euclidean TSP nor Steiner tree do have polynomial smoothed complexity, since the perturbation does not change the set of feasible solutions. The change in the objective function due to the perturbation depends at most linearly on the magnitude of the perturbation. Since the optimal objective value is $\Omega(1)$, we could obtain a fully polynomial randomized approximation scheme for worst-case instances, which is widely believed not to

exist. Thus, one needs to give up either the requirement that the running time is polynomial in $1/\varepsilon$ or the requirement that the running time is polynomial in ϕ. Under the first relaxation the problem has been solved by Arora [Aro96]. The question whether there exist polynomial time algorithms for Euclidean TSP or Steiner tree under perturbations with a fixed density parameter ϕ or in the uniform input model of Karp [Kar77] is open.

Relationship to condition numbers In order to obtain a finer analysis of algorithms than that provided by worst-case complexity, one should find a way of distinguishing hard problem instances from easy ones. A natural approach is to find a quantity indicating the difficulty of solving a problem instance. In Numerical Analysis and Operations Research it is common to bound the running time of an algorithm in terms of a *condition number* of its input. The condition number is typically defined to be the sensitivity of the solution for a problem instance to slight perturbations of the input. For example, Renegar [Ren94, Ren95a, Ren95b] presents a variant of the primal interior point method and describes its running time as a function of the condition number. Remarkably, his running time bound depends only logarithmically on the condition number. Dunagan, Spielman, and Teng [DST03] study this condition number in the smoothed analysis framework. Assuming Gaussian ϕ-perturbations, the condition number can be bounded by a function that is polynomial in ϕ. Thus, the running time of Renegar's algorithm depends only logarithmically on the density parameter ϕ. In contrast, the running time bound of the Simplex algorithm presented by Spielman and Teng in [ST01] is polynomial in ϕ.

In [ST03], Spielman and Teng propose to extend the condition number towards discrete optimization problems in order to assist the smoothed analysis of such problems. As a natural definition for the condition number of a discrete function they suggest *the reciprocal of the minimum distance of an input to one on which the function has a different value*. In fact, the minimum of winner, loser, and feasibility gap is a lower bound on the amount by which the coefficients of a binary optimization problem need to be altered so that the winner, i.e., the solution taking the optimal value, changes. Let us define the reciprocal of this minimum to be the *condition number for binary optimization problems*. This allows us to summarize our analysis in an alternative way. Our probabilistic analysis in Section 3.4 shows that the condition number is bounded polynomially in the density parameter ϕ. Furthermore, in Section 3.5, we proved that a problem with pseudo-polynomial worst-case complexity admits an algorithm whose running time is bounded polynomially in the condition number. Combining these results, we obtained algorithms whose smoothed complexity depends in a polynomial fashion on the density parameter ϕ. Let us remark that this kind of dependence is best possible for NP-hard optimization problems, unless there is a subexponential time algorithm for NP-complete problems. In particular, a running time bound logarithmic in ϕ would yield a randomized algorithm with polynomial worst-case complexity.

Conclusion and Open Problems

The Pareto Curve for Bicriteria Problems

In this thesis we presented polynomial upper bounds on the expected number of Pareto points for binary optimization problems with two objective functions. Our analysis applies to arbitrary solution sets $\mathcal{S} \subseteq \{0,1\}^n$. We require one of the objectives to be a linear function with random coefficients. These coefficients are assumed to be stochastically independent, but can follow arbitrary continuous non-negative probability distributions with finite mean and density. An extension to discrete probability distributions proves a trade-off ranging from polynomial to pseudo-polynomial bounds, depending on the degree of randomness in the coefficients of the linear objective function.

We applied this result to the knapsack problem and the constrained shortest path problem. Both problems allow an efficient enumeration of all Pareto points such that the polynomial bound on the expected number of Pareto points implies a polynomial bound on the running time of these algorithms. We presented an experimental study for the knapsack problem, which suggests that the presented upper bounds are not tight. We also presented a lower bound for knapsack instances with adversarial weights and random profits.

There are various open questions that propose themselves.

- Can the upper bounds be improved? We believe that a tight bound for general distributions is a factor n^2 smaller than the bound proven. Our analysis relies on the bound for long-tailed distributions, Part (b) of Theorem 1.2, which is provably tight (Theorem 1.13). For long-tailed distributions we exploited the Markovian property and showed that, for every solution, the expected increase of the objective function value can be lower bounded by a polynomial in $O(1/n)$, in case this solution becomes Pareto optimal. This is not true for distributions with finite domain. So we had to rely on a cascade of conditions and apply the result for long-tailed distributions, where we lost a factor n^2 in the analysis. For a significant improvement of the bound for general distributions one would probably need a different approach.

- What happens to the bounds if we allow negative coefficients, that is, if we extend the domain of the probability distributions to $\mathbb{R}$ (instead of $\mathbb{R}_{\geq 0}$)? Our proofs are based on the Pareto optimality for explicit subset orderings. If such an ordering $S_1, \ldots, S_m$ is monotone, i.e., the subsets of a solution $S_i \in \mathcal{S} \subseteq 2^{[n]}$ precede S_i in this ordering, then the result for long-tailed distributions and thus for general distributions stays valid. If the subset ordering is not monotone, then we can

partition the set of solutions $\mathcal{S}$ into the sets $\mathcal{S}^{(k)} = \{S \in \mathcal{S} \mid |S| = k\}$, for $1 \leq k \leq n$. Each solution set $\mathcal{S}^{(k)}$ satisfies the monotonicity condition, as all subsets in $\mathcal{S}^{(k)}$ have he same cardinality. Thus, we can apply our main theorem for each solution set $\mathcal{S}^{(k)}$. Let q_k denote the number of Pareto points over $\mathcal{S}^{(k)}$. Then the number of Pareto points over $\mathcal{S}$ is at most $\sum_{k\in[n]} q_k$. Therefore, we obtain bounds that are a factor n larger than the bounds proven for non-negative distributions. Can this additional factor n be avoided?

- Can one prove a concentration result for the number q of Pareto points? Our analysis bounds the expected number of Pareto points. By applying the Markov Inequality we obtain a weak tail bound of $\mathbf{Pr}[q > \alpha\mathbf{E}[q]] \leq 1/\alpha$. Our experiments suggest that q is unlikely to deviate much from its mean. The fraction of experiments where q was at least twice as large as the average was only $5 \cdot 10^{-4}$, and in only 4 out of one million randomly generated instances, q was larger than three times the average.

- Can the bound on the expected number of Pareto optimal solutions be generalized to problems with more than two objectives? We conjecture that for problems with k objective functions the expected number of Pareto points is upper bounded by $O(n^{f(k)}\phi\mu)$, for some function $f : \mathbb{N} \to \mathbb{R}_{>0}$, provided that $k-1$ objectives are linear functions with random or randomly perturbed coefficients. As in our analysis, the parameters ϕ and μ describe the maximum density and the maximum expectation over the different probability distributions of the coefficients.

  Notice that there is a strong connection to the gap properties in Chapter 3, since the optimal solution and the minimal loser are Pareto optimal solutions. Our analysis of the loser and feasibility gap covers multiple constraints, which corresponds to multiple objective functions in the multiobjective variant of the problem. However, a bound on the size of the loser and feasibility gap does not directly imply a bound on the expected number of Pareto points for multiple objectives. The reason is that the Pareto points might be concentrated on a very small region in the objective space and the location of this region might be random. In this case no Pareto point from this region is likely to come very close to any of the hyperplanes corresponding to the deterministic constraints, which would comply to the proven gap properties. A first approach to tackle this problem might be to prove that the Pareto points are well dispersed.

- Is the expected number of Pareto points still polynomially bounded when allowing integer variables with bounded domain instead of 0-1 variables? Possibly, a similar bound can be shown assuming the following monotonicity property. The set of solutions is denoted by $\mathcal{S} \subseteq \{0,\dots k\}^n$. A sequence $x^{(1)},\dots,x^{(m)}$ of solutions is called monotone, if $\forall u,v \in [m], v < u : \exists\, i \in [n] : x_i^{(v)} < x_i^{(u)}$.

- For which binary optimization problems can we efficiently enumerate all Pareto points? We described such enumeration algorithms for the knapsack problem and the constrained shortest path problem. To our knowledge, this problem is unsolved for the constraint version of the minimum spanning tree problem. An important property is the connectedness of Pareto optimal solutions. Two solutions are connected if they can be transformed into each other by changing only two variables. Thus, two spanning trees are connected if they can be transformed into each other by

removing and subsequently adding one edge. If all Pareto optimal solutions are connected, then this set of solutions can usually be determined without considering dominated solutions by a simple local search. For the constrained minimum spanning tree problem, Pareto optimal solutions are not necessarily connected [EK97].

Knapsack Core Algorithms

The second chapter of this thesis we study advanced algorithmic concepts for the knapsack problem. We investigated the expected running time a core algorithm on random knapsack instances, where weight and profit of each item are chosen uniformly at random. We presented an upper bound of $O(n \operatorname{polylog} n)$ on the expected running time, which comes close to the results observed in experimental studies. An extension of our analysis to δ-correlated instances yields a running time bound of $O(\frac{n}{\delta} \operatorname{polylog} \frac{n}{\delta})$. Finally we presented a new core algorithm which combines several ideas for reducing the number of enumerated knapsack fillings. An experimental study shows that for uniform and δ-correlated instances, our implementation clearly outperforms the *Combo* core, a knapsack solver due to Pisinger.

Constrained Binary Optimization Problems

We presented a probabilistic analysis for binary optimization problems, where the solutions are defined by a deterministic ground set of solutions and a constant number of additional linear constraints with random coefficients. Furthermore, our model allows a random objective function which is also assumed to be linear. We proved that such a binary optimization problem has polynomial smoothed complexity if and only if the problem allows an algorithm with running time that is pseudo-polynomial in the coefficients of the stochastic coefficients and polynomial in the size of the remaining input. This result is based on a generalized version of the Isolating Lemma and a novel Separating Lemma. The Generalized Isolating Lemma bounds the density of the winner gap, a random variable describing the difference in the objective value between the best and the second best solution. It applies to a random objective function. The Separating Lemma bounds the density of the loser and the feasibility gap, two random variables that describe the slack of the optimal solution with respect to a linear stochastic constraint and the minimal amount by which any loser violates the linear constraint, respectively. Losers are solutions from the deterministic ground set that have a better objective function than the optimal solution but become infeasible due to the stochastic constraint. We defined corresponding gaps for multiple constraints and used them in an adaptive rounding scheme which calls the pseudo-polynomial time algorithm with higher and higher precision of the stochastic coefficients, until the optimality of the computed solution can be certified. We extended our results to zero preserving perturbations and proved polynomial smoothed complexity for various NP-hard optimization problems. There are some interesting open problems.

- Can the Generalized Isolating Lemma and the Separation Lemma be generalized to allow integer variables with a bounded domain instead of $\{0,1\}$? First considerations show that the winner gap allows such an extension, even if we allow different domains for different variables. The individual domains can be chosen arbitrarily, the only important parameter is the size l of the

domain, whose reciprocal $1/l$ has a linear influence on the expected size of the gap. This result implies a polynomial smoothed complexity for problems with the following properties:

  - The objective function is linear.
  - There exists an algorithm with running time which is pseudo-polynomial with respect to the coefficients of the objective function and polynomial with respect to the size of the remaining input.
  - The domain of the variables is bounded by a polynomial in n.

  A generalization for the loser and feasibility gap seems to be much more challenging and is still open.

- Are there problems with polynomial average case complexity but no polynomial smoothed complexity? In this case the dependence of ϕ on the running time should be super-polynomial.

- For binary optimization problems we have shown equivalence between pseudo-polynomial running time and polynomial smoothed complexity. Is this statement true for other problems? We believe that scheduling problems are promising candidates. We prove polynomial smoothed complexity for the problem of minimizing the weighted number of tardy jobs on a single machine. For this problem, however, the order of jobs in an optimal schedule is already determined when fixing the set of tardy/non-tardy jobs and hence it can be described by a 0/1 program. This is different for the problem of minimizing the total tardiness where our framework cannot be applied.

- Another challenging topic is how to prove that a problem has no polynomial smoothed complexity. For problems where only the objective function is perturbed and the objective function is "well behaved" we can resort to the approximation theory, which provides a rich knowledge for many problems. However, if we also allow the perturbation to change the set of feasible solutions, then we probably need new approaches.

Zusammenfassung

Wir untersuchen mit mathematischen Methoden die Effizienz verschiedener Algorithmen für schwierige Optimierungsprobleme auf zufälligen Eingaben. Im Zentrum steht die Analyse struktureller Eigenschaften zufälliger Probleminstanzen, die im Weiteren benutzt werden, um Laufzeitschranken für verschiedenste Optimierungsprobleme zu beweisen.

Im ersten Teil dieser Arbeit betrachten wir binäre Optimierungsprobleme mit zwei Zielfunktionen. Die Variablen sind also beschränkt auf die Werte 0 und 1. Für Probleme mit mehreren Zielfunktionen greift das Standardkonzept der Optimalität einer Lösung nicht mehr. Durch Ausnutzung der verbleibenden Dominanz zwischen den Lösungen erhält man das Konzept der Pareto-Optimalität. Eine Lösung ist somit Pareto-optimal, wenn es keine andere Lösung gibt, die bezüglich mindestens einer Zielfunktion besser und in keiner Zielfunktion schlechter ist. Im ungünstigsten Fall ist die Anzahl der Pareto-optimalen Lösungen exponentiell in der Anzahl der Variablen. Wir können zeigen, dass die erwartete Anzahl der Pareto-optimalen Lösungen polynomiell beschränkt ist, wenn mindestens eine Zielfunktion linear ist und ihre Koeffizienten genügend Zufall enthalten. Es werden keinerlei Annahmen über die zweite Zielfunktion gemacht. Unsere Analyse deckt allgemeine Wahrscheinlichkeitsverteilungen mit endlichem Erwartungswert ab und ermöglicht sogar, dass die Koeffizienten der linearen Zielfunktion unterschiedlichen Wahrscheinlichkeitsverteilungen folgen. Wir zeigen beispielsweise eine obere Schranke von $O(n^4\mu\phi)$, wobei n die Anzahl der Variablen, μ den maximalen Erwartungswert der Zufallsverteilungen für die verschiedenen Koeffizienten, und ϕ deren maximale Dichte bezeichnet.

Wir wenden dieses Ergebnis auf zwei Optimierungsprobleme an: das Kürzeste-Wege-Problem mit Nebenbedingungen (constrained shortest path problem) und das Rucksackproblem (knapsack problem). Beim Rucksackproblem betrachten wir das Gewicht und den Profit einer Rucksackfüllung als die beiden Zielfunktionen. Für beide Probleme können die Pareto-optimalen Lösungen effizient aufgezählt werden, so dass die obere Schranke für die Anzahl der Pareto-optimalen Lösungen eine polynomielle erwartete Laufzeit für diese Algorithmen impliziert. Wir zeigen beispielsweise eine Laufzeitschranke $O(n^4)$ für uniform zufällige Rucksackinstanzen, wobei n die Anzahl der Variablen bzw. der Gegenstände angibt.

Im zweiten Teil untersuchen wir die Laufzeit von *Core*-Algorithmen für das Rucksackproblem. *Core*-Algorithmen profitieren von der Tatsache, dass sich die optimale Lösung nur in wenigen Variablen von der optimalen Lösung des relaxierten Problems unterscheidet. Durch Fixieren eines Großteils der Variablen erhält man ein kleineres Rucksackproblem, das *Core*-Problem, welches für uniform zufällige Rucksackinstanzen üblicherweise lediglich polylogarithmische Größe hat. Das *Core*-Problem wird mit den Methoden aus dem ersten Teil der Arbeit analysiert. Als Ergebnis zeigen wir eine erwartete Laufzeit

von $O(n \operatorname{polylog} n)$ für uniform zufällige Rucksackinstanzen. Zusätzlich erweitern wir unsere Analyse auf δ-korrelierte Instanzen, eine parametrisierte Klasse von Eingabeverteilungen, die mit kleiner werdendem δ von bekannten Algorithmen immer schwieriger zu lösen ist.

Eine experimentelle Studie runded das Rucksackkapitel ab. Zum einen untersuchen wir strukturelle Eigenschaften von zufälligen Rucksackinstanzen, die in unseren theoretischen Analysen eine wichtige Rolle spielen, beispielsweise die Anzahl der Pareto-optimalen Rucksackfüllungen oder das *Integrality-gap*. Die Experimente geben ein Indiz dafür, dass die oben genannte Schranke für die Anzahl der Pareto-optimalen Lösungen nicht scharf ist. Stattdessen vermuten wir, dass $O(n^2\mu\phi)$ eine scharfe obere Schranke ist. Weiterhin vergleichen wir die Effizienz verschiedener *Core*-Algorithmen. Durch Kombination von verschiedenen Konzepten, die u.a. in früheren theoretischen Analysen von stochastischen Rucksackproblemen vorgestellt wurden, erhalten wir einen sehr effizienten Code. Als Referenz verwendeten wir den *Combo*-Code, der in anderen experimentellen Studien das beste Laufzeitverhalten zeigte. Für uniforme und δ-korrelierte Instanzen dominiert unsere Implementierung den *Combo*-Code deutlich. Für die schwierigste getestete Klasse von Instanzen zeigte sich unser Code im Durchschnitt 700 mal schneller.

Im letzten Teil der Arbeit wird ein semi-zufälliges Eingabemodell für binäre Optimierungsprobleme vorgestellt. Es ermöglicht eine probabilistische Analyse für eine große Klasse von Optimierungsproblemen und berücksichtigt gleichzeitig die zugrunde liegende kombinatorische Struktur der einzelnen Probleme. Die Menge der zulässigen Lösungen ist gegeben als Schnitt von einer beliebig spezifizierbaren Grundmenge von Lösungen und den zulässigen Lösungen von linearen Ungleichungen mit stochastischen Koeffizienten. Zudem erlaubt das Modell lineare Zielfunktionen mit zufälligen Koeffizienten. Unsere Analyse basiert auf strukturellen Eigenschaften, die wir *Winner*, *Loser* und *Feasibility-Gap* nennen. Sie beschreiben die Sensitivität der optimalen Lösung bezüglich kleiner Veränderungen des stochastischen Teils der Eingabe. Wir zeigen, dass diese *Gaps* normalerweise mindestens eine Größe aufweisen, die polynomiell in $1/(nk\phi)$ ist, wobei n die Anzahl der Variablen, k die Anzahl der stochastischen Ungleichungen und ϕ die maximale Dichte aller Wahrscheinlichkeitsverteilungen für die Koeffizienten der Ungleichungen bzw. der Zielfunktion bezeichnet. Somit kann man die Koeffizienten des stochastischen Teils der Eingabe auf logarithmisch viele Bits runden, ohne die optimale Lösung zu verändern. Diese Erkenntnisse werden in einem adaptiven Rundungsschema benutzt, welches die Genauigkeit der Koeffizienten solange erhöht, bis die Optimalität der berechneten Lösung verifiziert werden kann. Wir definieren *Polynomial Smoothed-Complexity*, angelehnt an *Polynomial Average-Case Complexity*, eine Klassifizierung für die Komplexität von Problemen unter zufälligen Eingaben. Für die von uns betrachtete Klasse von Optimierungsproblemen können wir eine genaue Charakterisierung der Probleme geben, die in diese Komplexitätsklasse fallen. Ein binäres Optimierungsproblem hat *Polynomial Smoothed-Complexity* genau dann, wenn es einen Algorithmus für dieses Problem gibt, dessen Laufzeit im schlechtesten Fall pseudopolynomiell im stochastischen Teil und polynomiell im restlichen Teil der Eingabe ist. Eine interessante Erweiterung unserer Analyse sind Null-Perturbationen, die eine sinnvolle Anwendung unserer Ergebnisse auf weitere Probleme ermöglichen.

Bibliography

[Aro96] S. Arora. Polynomial time approximation schemes for euclidean TSP and other geometric problems. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 2–11, 1996.

[BB94] K. H. Borgwardt and J. Brzank. Average saving effects in enumerative methods for solving knapsack problems. *Journal of Complexity*, 10:129–141, 1994.

[BBM03] C. Banderier, R. Beier, and K. Mehlhorn. Smoothed analysis of three combinatorial problems. In *Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS-2003)*, volume 2747, pages 198–207, Bratislava, Slovak Republic, 2003.

[BLMS+03] L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, G. Schäfer, and T. Vredeveld. Average case and smoothed competitive analysis of the multi-level feedback algorithm. In *Proceedings of the Forty-Fourth Annual IEEE Symposium on Foundations of Computer Science*, pages 462–471, 2003.

[BP98] F. Barahona and W. R. Pulleyblank. Exact arborescences, matchings and cycles. *Discrete Applied Mathematics*, 16:617–630, 1998.

[BV03] R. Beier and B. Vöcking. Random knapsack in expected polynomial time. In *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC-2003)*, pages 232–241, San Diego, USA, 2003.

[BV04a] R. Beier and B. Vöcking. An experimental study of random knapsack problems. In *Proc. 12th Annual European Symp. on Algorithms (ESA-04)*, volume 3221 of *Lecture Notes in Computer Science*, pages 616–627. Springer, 2004.

[BV04b] R. Beier and B. Vöcking. Probabilistic analysis of knapsack core algorithms. In *Proc. 15th Annual Symposium on Discrete Algorithms (SODA-2004)*, pages 461–470, New Orleans, USA, 2004.

[BV04c] R. Beier and B. Vöcking. Random knapsack in expected polynomial time. *Journal of Computer and System Sciences*, 28(3):306–329, 2004.

[BV04d] R. Beier and B. Vöcking. Typical properties of winners and losers in discrete optimization. In *Proc. 36th Annual ACM Symposium on Theory of Computing (STOC-2004)*, pages 343–352, Chicago, USA, 2004. Association of Computing Machinery, ACM.

[BZ80] E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28:1130–1154, 1980.

[CM85] H.W. Corley and I.D. Moon. Shortest paths in networks with vector weights. *Journal of Optimization Theory and Applications*, 46(1):79–86, 1985.

[Cor85] H.W. Corley. Efficient spanning trees. *Journal of Optimization Theory and Applications*, 45(3):481–485, 1985.

[Dan57] G. B. Dantzig. Discrete variable extremum problems. *Operations Research*, 5:266–277, 1957.

[DF89] M.E. Dyer and A. M. Frieze. Probabilistic analysis of the multidimensional knapsack problem. *Maths. of Operations Research*, 14(1):162–176, 1989.

[DP82] G. D'Atri and C. Puech. Probabilistic analysis of the subset-sum problem. *Discrete Applied Mathematics*, 4:329–334, 1982.

[DST03] J. Dunagan, D.A. Spielman, and S.-H. Teng. Smoothed analysis of interior-point algorithms: Condition number. *Math. Programming*, 2003.

[Dum02] I. Dumitrescu. *Constrained Path and Cycle Problems*. PhD thesis, Department of Mathematics and Statistics, University of Melbourne, 2002.

[EG02] M. Ehrgott and X. Gandibleux. *Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys*, volume 52 of *International Series in Operations Research and Management Science*. Kluwer Academic Publishers, Boston, 2002.

[Ehr00] M. Ehrgott. *Multicriteria optimization*. Springer Verlag, 2000.

[EK97] M. Ehrgott and K. Klamroth. Connectedness of efficient solutions in multiple criteria combinatorial optimization. *European Journal of Operational Research*, 97:159–166, 1997.

[EP92] V.A. Emelichev and V.A. Perepelitsa. On cardinality of the set of alternatives in discrete many-criterion problems. *Discrete Mathematics and Applications*, 2(5):461 – 471, 1992.

[Fri86] A. M. Frieze. On the lagarias-odlyzko algorithm for the subset-sum problem. *SIAM J. Comput.*, 15(2):536–539, 1986.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman and Company, New York, 1979.

[GMS84] A. Goldberg and A. Marchetti-Spaccamela. On finding the exact solution to a zero-one knapsack problem. In *Proceedings of the 16th ACM Symposium on Theory of Computing (STOC)*, pages 359–368, 1984.

[GR96] M. X. Goemans and R. Ravi. The constrained minimum spanning tree problem. In *Fifth Scandinavian Workshop on Algorithm Theory*, LNCS 1097, pages 66–75. Springer, 1996.

[Han80] P. Hansen. Bicriterion path problems. In *Multiple Criteria Decision Making: Theory and Applications*, Lectures Notes in Economics and Mathematical Systems 177, pages 109–127. Springer, 1980.

[Has92] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.

[HR94] H.W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52:209–230, 1994.

[HS74] E. Horowitz and S. Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM*, 21(2):277–292, April 1974.

[IK75] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subsets problems. *Journal of the ACM*, 22(4):463–468, 1075.

[IN96] R. Impagliazzo and M. Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.

[Jok66] H. Joksch. The shortest route problem with constraints. *Journal of Mathematical Analysis and Application*, 14:191–197, 1966.

[Kar77] R. M. Karp. Probabilistic analysis of partitioning algorithms for the traveling salesman problem. *Oper. Res*, 2:209–224, 1977.

[KP99] Hans Kellerer and Ulrich Pferschy. A new fully polynomial approximation scheme for the knapsack problem. *Journal of Combinatorial Optimization*, 3:59–71, 1999.

[KPP04] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[Law76] E. L. Lawler. *Combinatorial optimization: networks and matroids*. Holt, Reinhart, and Winston, 1976.

[Law79] E.L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339–356, 1979.

[Lue82] G. S. Lueker. On the average difference between the solutions to linear and integer knapsack problems. *Applied Probability - Computer Science, the Interface, Birkhäuser*, 1:489–504, 1982.

[Lue98a] G. S. Lueker. Average-case analysis of off-line and on-line knapsack problems. *Journal of Algorithms*, 19:277–305, 1998.

[Lue98b] G. S. Lueker. Exponentially small bounds on the expected optimum of the partition and subset sum problems. *Random Structures and Algorithms*, 12:51–62, 1998.

[MO81] M.J. Magazine and O. Oguz. A fully polynomial approximation algorithm for the 0-1 knapsack problem. *European Journal of Operational Research*, 8:270–273, 1981.

[MPT99] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45(3):414–424, 1999.

[MPT00] S. Martello, D. Pisinger, and P. Toth. New trends in exact algorithms for the 0-1 knapsack problem. *European Journal of Operational Research*, 123:325–332, 2000.

[MT90] S. Martello and P. Toth. *Knapsack Problems – Algorithms and Computer Implementations*. Wiley, 1990.

[MT97] S. Martello and P. Toth. Upper bounds and algorithms for hard 0-1 knapsack problems. *Operations Research*, 45(5):768–778, 1997.

[MVV87] K. Mulmuley, U. Vazirani, and V.V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[NU69] G. Nemhauser and Z. Ullmann. Discrete dynamic programming and capital allocation. *Management Science*, 15(9):494–505, 1969.

[Par95] R. Gary Parker. *Deterministic scheduling theory*. Chapman & Hall, 1995.

[Pis95] D. Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, DIKU, University of Copenhagen, 1995.

[PT98] D. Pisinger and P. Toth. *Knapsack Problems*. Kluwer Academic Publishers, 1998.

[PY00] C. H. Papadimitriou and M. Yannakakis. The complexity of tradeoffs and optimal access of web sources. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 86–92, 2000.

[RASG98] R.M. Ramos, S. Alonso, J. Sicilia, and C. González. The problem of the optimal biobjective spanning tree. *European Journal of Operational Research*, 111:617–628, 1998.

[Ren94] J. Renegar. Some perturbation theory for linear programming. *Math. Programming*, 65(1, Series A):73–91, 1994.

[Ren95a] J. Renegar. Incorporating condition measures into the complexity of linear programming. *SIAM J. Optimization*, 5(3):506–524, 1995.

[Ren95b] J. Renegar. Linear programming, complexity theory and elementary functional analysis. *Math. Programming*, 70(3, Series A):279–351, 1995.

[Sal75] H. M. Salkin. *Integer Programming*. Addison-Wesley, 1975.

[Sch86] A. Schrijver. *Theory of linear and integer programming*. Wiley, 1986.

[SP91] I.V. Sergienko and V.A. Perepelitsa. Finding the set of alternatives in discrete multicriterion problems. *Cybernetics*, 27(3):673 – 683, 1991.

[SS04] G. Schäfer and N. Sivadasan. Topology matters: Smoothed competitiveness of metrical task systems. In *Conference Proceedings of the 21st International Symposium on Theoretical Aspects of Computer Science*, pages 489–500, 2004.

[ST01] D. A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, pages 296–305, 2001.

[ST03] D. A. Spielman and Shang-Hua Teng. Smoothed analysis: Motivation and discrete models. In *Proceedings of WADS*, 2003.

[WN67] H.M. Weingartner and D.N. Ness. Methods for the solution of the multi-dimensional 0/1 knapsack problem. *Operations Research*, 15(1):83–103, 1967.

[Zie01] M. Ziegelmann. *Constraint Shortest Paths and Related Problems*. PhD thesis, Universität des Saarlandes, 2001.

CURRICULUM VITAE

PERSONAL DETAILS:

Name	René Beier
Date of birth	03.03.1974
Place of birth	Mühlhausen (Thuringia), Germany
Nationality	German
Children	Moritz Leon Beier (birth date 11/08/2003)

EDUCATION:

01/2001 – present	**Max-Planck-Institut für Informatik, Saarbrücken, Germany** Ph. D. student of computer science; advisor: Prof. Dr. Berthold Vöcking (Dortmund)
10/1993 – 10/2000	**Universität des Saarlandes, Saarbrücken, Germany** Student of computer science and biomedical engineering
04/1999 – 08/2000	Master's Thesis under supervision of Prof. Dr. Jop Frederik Sibeyn; Title of thesis: *Eine Heuristik für das Gossiping Problem*
10/2000	Diplom ($\approx$ Master's degree) in computer science
10/1997 – 06/1998	**University of Edinburgh, Great Britain** Visiting student
09/1988 – 07/1992	**Specialized School for Mathematics, Natural Sciences, and Engineering, Erfurt, Germany**
06/1992	Abitur ($\approx$ high-school diploma)
09/1985 – 07/1988	**Polytechnic Secondary School, Mühlhausen, Germany**

WORK AND TEACHING EXPERIENCE:

3/2001 – 9/2001 **Max-Planck-Institut für Informatik, Saarbrücken, Germany**
Teaching assistant for the lecture *Optimization* held by Prof. Dr. Dr.-Ing. E. h. Kurt Mehlhorn and Priv. Doz. Dr. E. Ramos

11/1998 – 10/2000 **Fraunhofer Institute for Biomedical Engineering (IBMT) in St. Ingbert**
Student assistant in the *Computer Aided Simulations* group headed by Dipl.-Ing. Peter K. Weber

07/1998 – 9/1998 **The Fraunhofer-IBMT Technology Center Hialeah (FTeCH), Florida, USA**
Student assistant in the *Ultrasonic applications* group headed by Dr. rer. nat. Rainer M. Schmitt

05/1995 – 08/1997 **Fraunhofer Institute for Biomedical Engineering (IBMT) in St. Ingbert**
Student assistant in the *Computer Aided Simulations* group headed by Dipl.-Ing. Peter K. Weber

8/1992 – 10/1993 **Alternative civilian service**
Nursing home in Mühlhausen, Germany

SCHOLARSHIP:

01/2001 – 12/2003 Member of the graduate studies program "Quality Guarantees for Computer Systems" at the Dept. of Computer Science, Universität des Saarlandes, Saarbrücken, Germany, funded by DFG (German research foundation).

LANGUAGE SKILLS:

English reading, writing, speaking

Russian high school level

STAYS ABROAD:

10/1997 – 06/1998 **University of Edinburgh, Great Britain**
Visiting student

07/1998 – 10/1998 **The Fraunhofer-IBMT Technology Center Hialeah (FTeCH), Florida, USA**
Student assistant in the *Ultrasonic applications* group headed by Dr. rer. nat. Rainer M. Schmitt

Saarbrücken, 26. September 2004