Combinatorial Optimization under Ellipsoidal Uncertainty

Dissertation

zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

Der Fakultät für Mathematik
der Technischen Universität Dortmund
vorgelegt von

Anna Ilyina

im März 2017

Dissertation

Combinatorial Optimization under Ellipsoidal Uncertainty

Fakultät für Mathematik

Technische Universität Dortmund

Erstgutachter: Prof. Dr. Christoph Buchheim

Zweitgutachter: Prof. Dr. Petra Mutzel

Tag der mündichten Prüfung: 28. Juni 2017

Abstract

We study combinatorial problems with ellipsoidal uncertainty in the objective function concerning their theoretical and practical solvability. Ellipsoidal uncertainty is a natural model when the coefficients are normally distributed random variables. Robust versions of typical combinatorial problems can be very hard to solve compared to their linear versions.

Complexity and approaches differ fundamentally depending on whether *uncorrelated* or *correlated* uncertainty occurs. We distinguish between these two cases and consider first the unconstrained binary optimization under uncorrelated ellipsoidal uncertainty. For this we develop an algorithm which computes an optimal solution by merely sorting the variables and, correspondingly, has a running time of $\mathrm{O}(n \log n)$. The algorithm is based on the *diminishing returns-property*, which is characteristic for submodular functions. We introduce a new and a more general *p-norm-uncertainty* and show that with only slight modifications the sorting algorithm can be easily applied. We also extend the algorithm to general integer variables, which in this case only leads to a pseudo-polynomial time.

The next step to the general case is investigation of problems with arbitrary combinatorial sets $X \subseteq \{0,1\}^n$ under uncorrelated ellipsoidal uncertainty. For this case we embed the $\mathrm{O}(n \log n)$-algorithm for the unconstrained binary problems into a *Lagrangean decomposition approach*. The approach separates the objective function from the combinatorial structure applying Lagrangean relaxation to some artificial connecting constraints. This creates two subproblems, one of which is the linear version of the combinatorial problem and the other one is just the unconstrained binary uncorrelated problem, which can be solved using the $\mathrm{O}(n \log n)$-algorithm. The solutions of the subproblems are used to obtain primal and dual bounds which are used in a branch and bound-approach. The approach shows an excellent performance in practice.

In the correlated case already the unconstrained binary problem turns out to be *strongly $\mathcal{NP}$-hard*. Here we also define a branch and bound-approach, now with lower bounds determined by underestimation of the given ellipsoid with certainly defined axis-parallel ellipsoids. We use this idea to extend the decomposition approach to general combinatorial problems under correlated uncertainty. In contrast to the uncorrelated case the uncertain subproblem of the decomposition is here strongly $\mathcal{NP}$-hard in itself. We solve it approximately using the developed underestimators which are determined in a preprocessing step. The approach offers room for improvement concerning in the primal extent a faster computation of the underestimators, which is done by solving semidefinite programs.

Zusammenfassung

Wir untersuchen kombinatorische Probleme unter Annahme ellipsoidaler Unsicherheit in der Zielfunktion auf ihre theoretische und praktische Lösbarkeit. Ellipsoidale Unsicherheit ist ein natürliches Modell wenn die Koeffizienten normalverteilte Zufallsvariablen sind. Die robuste Variante der typischen kombinatorischen Probleme kann dabei im Vergleich zu ihren linearen Varianten sehr schwer sein.

Die Komplexität und die Ansätze unterscheiden sich grundlegend, je nach dem ob es sich um die *unkorrelierte* oder *korrelierte* Unsicherheit handelt. Zwischen diesen zwei Fällen differenzieren wir auch und betrachten zunächst den Fall der unbeschränkten binären Optimierung unter unkorrelierter Unsicherheit. Dazu entwickeln wir einen Algorithmus, in dem im Wesentlichen das Sortieren der Variablen die optimale Lösung liefert und dessen Laufzeit entsprechend $\mathrm{O}(n \log n)$ ist. Dieser basiert auf der charakteristischen Eigenschaft der submodularen Funktionen, nämlich der der *reduzierenden Erträge*. Der Algorithmus lässt sich auf die *p-norm-Unsicherheit* erweitern, die wir einführen, und auch auf allgemeine ganzzahlige Variablen, im letzten Fall allerdings nur mit pseudo-polynomieller Laufzeit.

Der nächste Schritt zum allgemeinen Problem ist die Untersuchung der Probleme mit beliebigen kombinatorischen Mengen $X \subseteq \{0,1\}^n$ unter unkorrelierter Unsicherheit. Hierzu bauen wir den $\mathrm{O}(n \log n)$-Algorithmus für den unbeschränkten Fall in einen *Zerlegungsansatz* ein. Dieser trennt die Zielfunktion von den Nebenbedingungen mittels Lagrange-Relaxierung, angewendet auf künstliche Verbindungsvariablen. Es entstehen zwei Teilprobleme, von denen eins die lineare Variante des kombinatorischen Problems ist und das andere gerade das unbeschränkte binäre unkorrelierte Problem, was mit dem $\mathrm{O}(n \log n)$-Algorithmus lösbar ist. Die Lösungen der Teilprobleme benutzen wir zur Aufstellung dualer und primaler Schranken, die in ein Branch and Bound-Verfahren eingebaut werden. In der Praxis zeigt der Ansatz eine hervorragende Leistung.

Im korrelierten Fall offenbart sich schon das unbeschränkte binäre Problem als *stark $\mathcal{NP}$-schwer*. Wir definieren auch dazu ein Branch and Bound-Verfahren, in dem untere Schranken durch Unterschätzung des Ellipsoids mit achsen-parallelen Ellipsoiden ermittelt werden. Diese Idee benutzen wir um den Dekompositionsansatz nun auf allgemeine kombinatorische Probleme unter korrelierter Unsicherheit zu erweitern. Im Vergleich zu dem unkorrelierten Fall ist das unsichere Teilproblem in der Zerlegung schon an sich stark $\mathcal{NP}$-schwer. Dieses lösen wir approximativ mittels der entwickelten Unterschätzer, die wir in einem Preprocessing Schritt bestimmen. Der Ansatz zeigt Verbesserungspotenzial, was in erster Linie eine schnellere Bestimmung der Unterschätzer betrifft, die durch Lösung semidefiniter Programme geschieht.

Partial Publications and Collaboration Partners

All research introduced in this dissertation has been elaborated under the supervision of Christoph Buchheim (TU Dortmund). Partial results on the combinatorial $O(n \log n)$ algorithm in Chapter 3 and the results on the Lagrangean decomposition approach in Chapter 4, as well as the complexity result in Chapter 5 were developed together with Frank Baumann (TU Dortmund) and published in [8] and [9]. The theory on the labeling approach for the robust Shortest Path-problem in Chapter 4 was worked out in a productive cooperation with Luis Miguel Torres (Escuela Politécnica Nacional, Quito). The result on the robust multicriteria optimization was worked out in collaboration with Fritz Böckler (TU Dortmund). Some ideas on the robust Shortest Path-problem were investigated in a cooperation with André Chassein (TU Kaiserslautern).

Acknowledgements

My sincere thanks go to all my colleagues and friends for the great time at the university and for encouraging me throughout this experience. I am very thankful to my supervisor Christoph Buchheim who provided me an opportunity to join their team and guided me in all the time of research. Finally, I would like to thank my family for always being there for me.

In loving memory of Dmitriy Semenov.

Contents

Introduction

Making decisions is a component of life everyone is permanently exposed to. Often it is a difficult process, as we want to take the *best* decision, i.e. the one with the largest benefit. At the end of the day we want to be in a better position than if we would have decided differently.

Sometimes decision making can be abstracted and formalized, such that a mathematical optimization problem is to be solved. But even then it can be hard to make a choice as there may exist many different alternatives and the decision must fulfill certain requirements. In many real life-situations, however, we must decide even without having a full knowledge of the facts, such that the outcome can only be speculated about. Obviously, the decision-making process can get a lot harder when it is subject to *uncertainty* which may come from diverse directions. Now we not only have to decide, but perhaps also to take some *risk*, as uncertainty is linked to taking risks.

For a motivating example consider the underground coal mining process. In order to break through the rock, explosives are used in a primary phase. Thereby the coal layer should not be damaged. The coal miner specialist is facing the decision, in which sections of the surface to put the explosives in order to minimize the thickness of the remaining foreign material layers, to minimize in the end the expense of the subsequent digging. This decision is obviously essentially complicated by the uncertainty about the strength of the material above and below the coal layer, the regional geological conditions, topology, climate, ground water conditions and other environment factors. In addition, the actual coal layer does not run in a straight plane or in predictable directions in general. On the other hand different spots clearly correlate concerning the coal placement and this information should not be abstained from.

Decision making under uncertainty turns out to be a big challenge from mathematical and economical perspectives. The mankind tends to protect itself from any risk and to ensure guarantees, to become *robust* against uncertainty. Whatever *certainty* under *uncertain conditions* means, is the subject of *robust optimization*.

In the robust optimization community many ideas have been proposed, when a solution to an uncertain optimization problem is considered to be robust. The probably most obvious criterion for that is to consider the *worst-case scenario* of

a solution. The line of reasoning here is to know the performance in the worst case, such that it only can improve if a different scenario occurs.

But what is the worst-case scenario? The prerequisite to define it is to know what can happen, i.e. to be able to define the *set of all possible scenarios* or the *uncertainty set*. In robust optimization the information about what can happen is assumed to be available and the approaches to the arising optimization problem strongly depend on the shape of the uncertainty set. Among many imaginable shapes of $\mathcal{U}$ a frequent case is that the scenarios are normally distributed, which leads to the so-called *ellipsoidal uncertainty*. This special uncertainty afflicts all optimization problems addressed in this thesis.

In *combinatorial optimization*, where we want to find an optimal object from a finite set of certainly defined objects, i.e. from a set of solutions satisfying certain combinatorial constraints, it is often assumed that only the objective function coefficients are uncertain. That is, we study in this work combinatorial optimization problems under ellipsoidal uncertainty in the objective function.

Over the last few decades several results on ellipsoidal uncertainty concerning complexity and approaches to significant special cases have been published. Some authors inspected the situation where the objective function coefficients do not correlate, i.e. the *uncorrelated ellipsoidal uncertainty case*. The research of Bertsimas and Sim [15] and Nikolova [60] reveals that in the uncorrelated case optimization over matroids is easy. They establish a connection of the problem to bicriteria optimization. But this connection does not induce polynomial solvability of the famous Shortest Path-problem considered under uncorrelated ellipsoidal uncertainty. To this problem Chen et al. [25] apply a different approach: They develop a labeling algorithm, which though neither leads to a proven polynomiality, but contributes to a faster practical solvability of the problem.

Among special cases we sometimes find still interesting yet more special cases. On series-parallel graphs the robust Shortest Path-problem with uncorrelated ellipsoidal uncertainty is easy: Chassein et al. [24] derived tractability of the problem using the connection to bicriteria optimization established in [15] and [60].

In contrast to the uncorrelated case, the general correlated case got less attention. It was immediately classified by Bertsimas and Sim [15] as hard and was since then not extensively considered in the literature. In fact, the general case is even strongly $\mathcal{NP}$-hard as we will prove in the thesis. The common approach to this particularly difficult problem remains the reformulation to a mixed-integer second-order-cone-program as proposed by Atamtürk and Narayanan [4].

But what makes the general problem so hard? Is it only the correlation or is it the interaction of combinatorial structure with a non-linear objective function? In this thesis we want to break down the general combinatorial optimization problem under ellipsoidal uncertainty and inspect its parts looking for the base of the difficulty. To this aim we one by one relax particular potential difficulties and

consider first the corresponding special cases. We use the gained knowledge about the partial problems to reconnect the difficulties and to generalize the problem again. One major intention of this thesis is to develop a method to exactly solve combinatorial problems under general ellipsoidal uncertainty.

On the way there we intend to specify the theoretical complexity of the studied problems. For that reason we mostly consider combinatorial problems that are easy to solve if no uncertainty has to be taken into account: We want to observe if the complexity status changes with appearance of uncertainty. Moreover, we aim to detect structures in relevant special cases to assign existing or to develop new algorithms for these, which might later on constitute building blocks for more general cases. We initiate this research aiming to exactly solve in an alternative way the general case of combinatorial optimization under ellipsoidal uncertainty.

Contributions

We devise a new fast combinatorial algorithm to solve the unconstrained binary problem under uncorrelated ellipsoidal uncertainty. This algorithm is developed from the geometrical illustration of the diminishing returns-property of submodular functions. We adopt this algorithm for general integer variables and show the resulting pseudo-polynomial time complexity in this case.

We also introduce the concept of p-norm-uncertainty, not known in the literature so far, and show that under minimal adjustments it can be solved by the same combinatorial algorithm, maintaining the time complexity of $\mathrm{O}(n \log n)$.

Moreover, the combinatorial algorithm for the unconstrained uncorrelated case is used in a Lagrangean decomposition approach for general constrained binary problems with uncorrelated ellipsoidal uncertainty. The decomposition approach disconnects the combinatorial constraints from the mean-risk objective function, such that one of the arising subproblems is exactly the unconstrained binary problem with uncorrelated ellipsoidal uncertainty, which can be solved by our combinatorial algorithm. The decomposition approach shows useful theoretical properties, such as the transferability of homogenous inequalities from the uncertainty set to the Lagrangean multipliers. Also a big advantage is the oracle-based idea of the algorithm. In fact we only assume that the underlying combinatorial problems are defined by a linear optimization oracle, such that it is sufficient to provide an appropriate combinatorial algorithm for the underlying deterministic problem. In accordance with the theory the approach also shows a very good performance on both applications of the uncorrelated case, namely the robust Shortest Path-problem and the robust Knapsack-problem.

The *robust Shortest Path-problem with uncorrelated ellipsoidal uncertainty* is still unclassified from the complexity theoretical perspective. We analyze a labeling

approach specified for this problem and consider the geometry of the dominance conditions on the node-labels. We identify a tight bound on the number of the non-dominated paths over a given node.

The general case of ellipsoidal uncertainty is shown to be strongly $\mathcal{NP}$-hard. We create a novel method to solve the general unconstrained case, which is based on a problem-specific underestimation of the covariance matrix by a linear term and thus reduction to the uncorrelated case, which we can solve quickly. In this way lower bounds are determined and we can use them in a branch and bound-routine.

Finally, incorporating of the underestimator-approach into the Lagrangean decomposition we obtain a new algorithm to solve the strongly $\mathcal{NP}$-hard general case of ellipsoidal uncertainty in combinatorial optimization.

Outline

The thesis is divided into three parts, where the second and third parts provide our results contributed to the topic and the first part subsumes the knowledge about combinatorial optimization under ellipsoidal uncertainty in the community.

In the first chapter we repeat the very basic concepts of combinatorial optimization and give an introduction into complexity theory. Here complexity classes and pseudo-polynomial time as well as matroids and submodular functions are of particular importance, as we will refer to these throughout the thesis. Moreover, we introduce multicriteria optimization and, in particular, the concept of non-dominance, because some further observations are based on its concepts.

The second chapter is more specialized on our topic and first of all gives an introduction into the field of robust optimization and motivates its ideas and approaches. We introduce our notation for robust combinatorial problems and motivate the concept of ellipsoidal uncertainty. Here we go into more detail, present the common reformulations and a comprehensive overview to methodology. Complexity results and references are given at the corresponding positions. The second chapter also reflects the structure of the remaining thesis: Here we distinguish between the uncorrelated and the general cases of ellipsoidal uncertainty, as well as between constrained and unconstrained cases.

Part II is exclusively dedicated to the uncorrelated case and is subdivided into the constrained and unconstrained case.

In Chapter 3 we study the unconstrained uncorrelated case and provide a combinatorial algorithm to solve this special case of ellipsoidal uncertainty. We also provide some extensions of the algorithm, in particular, to integer variables and to a generalization of uncorrelated ellipsoidal uncertainty, the p-norm-uncertainty.

In Chapter 4 we move over to general combinatorial structures and consider

first the robust Shortest Path-problem in a context of a labeling approach. An exact approach for the general constrained uncorrelated case based on Lagrangean decomposition follows as well as its experimental evaluation. With that the consideration of the uncorrelated case is closed and we turn our attention to the correlated case in Part III.

Part III devises the same structure as Part II, i.e. a division into unconstrained and constrained cases, but concerns general and not necessarily axis-parallel ellipsoids.

First of all in Chapter 5 the unconstrained case is classified as strongly $\mathcal{NP}$-hard. Motivated by this result we design a branch and bound-approach based on uncorrelated underestimators and discuss it experimentally.

In Chapter 6 most results of this thesis come together in form of a general approach to combinatorial optimization under ellipsoidal uncertainty. It is the adjusted Lagrangean decomposition approach from Chapter 4, where we underestimate the correlated term like in Chapter 5 and solve the arising uncorrelated unconstrained problem like in Chapter 3.

In the concluding section we finally summarize our results and findings and point out interesting directions we yet did not enter on the way to our results or those arising from the recent considerations and summarize open questions.

Part I

Basics

The subject of this thesis – combinatorial optimization under ellipsoidal uncertainty – is an active research area in optimization, and its understanding requires certain specific prior knowledge. This part will serve to break down the topic by discussing its integral components and to show any background necessary for understanding the approaches, concepts and relationships developed in the following chapters. We will provide a basis of our research, while describing fundamental structures and contexts. This includes important topics concerning combinatorial optimization, theoretical complexity and ellipsoidal uncertainty. The particular aspects are mostly treated with respect to the actual usage in our results, such that the level of detail varies depending on relevancy of the subject. Still, a considerable degree of prior knowledge on mathematical optimization is expected from the reader.

With this part we define a starting point for our research on ellipsoidal uncertainty in combinatorial optimization, in particular, while manifesting the state of knowledge resulting from the literature.

Chapter 1

Preliminaries and Mathematical Background

This chapter is a collection of definitions and concepts, which are used in the thesis. In addition to the basic combinatorial problems we work on, such as the Shortest Path-, the Minimum Spanning Tree- or the Knapsack-problem, a conception of certain structures, such as matroids and submodular functions, is required. Here we provide information about these and further concepts. Moreover, some approaches described in the thesis are based on the ideas of multicriteria optimization, which we quickly discuss in Section 1.3. A significant part is dedicated to complexity theory, as a basic knowledge on it is required to understand the time complexity of the introduced algorithms, as well as to classify the investigated problems from the complexity theoretical point of view. In particular, we highlight the concepts of the complexity classes and of problems admitting algorithms with pseudo-polynomial running time and FPTAS.

1.1 Combinatorial Optimization

Problems of combinatorial optimization consist of finding the best object from a *finite* set of objects. However, the actual number of feasible objects may grow exponentially in the description of the objects, such that enumeration is not worth considering. Usually the set of objects has a certain structure, a property which combines the elements to feasible solutions and turns the optimization over this set into a certain combinatorial optimization problem.

Consider for example the following fundamental combinatorial problem:

KNAPSACK PROBLEM [47]
Given n items with positive integer weights w_i and positive integer profits $p_i, i = 1, \ldots, n$, and a capacity value W,
select a subset of these items that maximizes the total profit without exceeding the weight limit.

Here, the objects are subsets of items that comply with the capacity constraint. The set of subsets is the power set of the items, such that the number of feasible solutions may grow exponentially in the number of items.

A significant part of combinatorial optimization problems is defined on *graphs*. This is a structure which allows describing connections between items and is used to define most basic and important objects in combinatorial optimization. The definitions in the next section follow the reference [30].

1.1.1 Graphs and Graph Problems

A *(directed) graph* G is a pair (V, E), consisting of a non-empty finite set V (*nodes*) and a set $E \subseteq V \times V$ of (ordered) pairs *((directed) edges)*. Two nodes v, w are *adjacent*, or *neighbours*, if there is an edge $e = (v, w) \in E$ or $e = (w, v) \in E$. Then v and w are also called *end nodes* of (v, w). A graph is called *complete*, if all vertices of G are pairwise adjacent. A *subgraph* (V', E') of G is a graph with $V' \subseteq V$ and $E' \subseteq E$. The *underlying undirected graph* of a directed graph G is formed from G by replacing each directed edge in E by an undirected edge and elimination of any resulting double edges. A *path between v_1 and v_k* (or a $v_1 - v_k$-path) is a graph $P = (V, E)$ of the form

$$V = \{v_1, \ldots, v_k\}, \; E = \{(v_1, v_2), \ldots, (v_{k-1}, v_k)\},$$

where $v_1, \ldots, v_k$ are all distinct. In this case the node v_1 is called the *initial-* or *source node* and the node v_k the *end-* or *destination node* of the path P. Occasionally we refer to P as a sequence of nodes or a sequence of edges. A graph G is called *connected* if there exists a path between every two nodes in G. A *cycle* is a graph $C = (V, E)$ of the form

$$V = \{v_1, \ldots, v_k, v_1\}, \; E = \{(v_1, v_2), \ldots, (v_{k-1}, v_k), (v_k, v_1)\},$$

where $v_1, \ldots, v_k$ are all distinct. If G contains no cycles, it is called a *forest*. If G is additionally connected, then it is called a *tree*. A subgraph $T = (V', E')$ of $G = (V, E)$ is called a *spanning tree of G* if it is a tree and $V' = V$. A graph $G = (V, E)$ is called *bipartite*, if V admits a partition into two subsets, such that every edge $e \in E$ has its end nodes in different subsets. A subgraph $M = (V', E')$ of $G = (V, E)$ is called a *matching in G* if no two edges in E' have common nodes.

Often edges and/or nodes of a graph are associated with costs and many fundamental combinatorial optimization problems consist of finding a subgraph with

certain properties which minimizes the total costs. This is the case in the following three well-studied problems:

MINIMUM SPANNING TREE-PROBLEM Given a graph $G = (V, E)$ and a cost function $c : E \to \mathbb{R}_+$, find a spanning tree in G which minimizes the total costs.

SHORTEST PATH-PROBLEM Given a graph $G = (V, E)$, nodes $s, t \in V$, and a cost function $c : E \to \mathbb{R}_+$, find an $s - t$-path in G with minimal costs.

ASSIGNMENT PROBLEM Given a bipartite graph $G = (V_1 \cup V_2, E)$ and a cost function $c : E \to \mathbb{R}_+$, find a matching in G with minimal costs.

In the three problems the costs are *linear*, i.e. the goal is to optimize the value $c^\top x$, if x is the indicator vector of the corresponding subset of E. These important problems are extensively studied and also non-linear variants of these problems are considered in the literature. The basis for our research is, however, combinatorial problems with linear objective function. We will later consider these problems under ellipsoidal uncertainty, then the objective functions become non-linear.

1.1.2 Matroids and Submodular Functions

Often certain properties can be found in graphs and other structures, which can make easier the optimization. For example, some combinatorial problems can be formulated as optimization problems over *matroids*. A matroid is a structure which generalizes the concept of *linear independency* of a set of vectors:

Definition 1.1. Let E be a finite set and 2^E its power set. A tuple (E, I) is called a *matroid over* E, if $I \subseteq 2^E$ and the following three properties hold:

(M1) $\emptyset \in I$;

(M2) if $X \subseteq Y \in I$, then $X \in I$;

(M3) for all $X, Y \in I$ with $|X| = |Y| - 1$ there exists an element $j \in Y \setminus X$ such that $X \cup \{j\} \in I$.

The elements of I are called *independent sets*.

Matroids play an important role in combinatorial optimization and have been studied widely. There are many equivalent characterizations of matroids.

A classic example of a matroid is the *graphic matroid*. Its independent sets are the forests in a given graph, such that the Minimum Spanning Tree-problem is an optimization problem based on the graphic matroid.

One reason for a big interest in this structure is that the so-called *greedy-algorithm* yields an optimal solution when applied on matroids [62].

GREEDY ALGORITHM FOR MATROIDS (maximization version)
Given: Matroid (E, I), $E = \{1, \ldots, n\}, c : E \to \mathbb{R}^n$.

(1) Sort the elements of E in decreasing order by their weights c_i; $X := \emptyset$.

(2) For each $i = 1, \ldots, n$ do:
If $X \cup \{i\} \in I$ and $c_i \geq 0$, then $X := X \cup \{i\}$.

(3) Return X.

In combinatorial optimization the feasible solutions are usually subsets of a given ground set. Here we can sometimes also find helpful properties of the objective function, such as that of a *submodular set function*.

Definition 1.2. Let E be a finite set. A set function $f : 2^E \to \mathbb{R}$ is called *submodular*, if for every $X, Y \subseteq E$ with $X \subseteq Y$ and every $i \in E \setminus Y$ the property

$$f(X \cup \{i\}) - f(X) \geq f(Y \cup \{i\}) - f(Y)$$

holds.

The property in Definition 1.2 is called the *diminishing returns-property*.

Submodularity can also be characterized in different ways. There exists a connection between submodular set functions and matroids. Submodularity generalizes the so-called *rank function* of a matroid, which can be used to give an equivalent characterization of a matroid (see [39] and [62] for details). We hold down that detecting such structures in a problem formulation can be useful for optimization. In particular, minimization of a submodular set function over an unconstrained binary set $\{0, 1\}^n$ can be done efficiently [39].

1.2 Complexity Theory

One of the essential characteristics of a problem is its solvability. We address problems which are still not investigated extensively. Since our aim is to find algorithms for these problems, we could narrow down the search if we knew that in principal there cannot exist algorithms with certain performance guaranties.

On the other hand the presented approaches have to be analyzed with respect to their contribution: When is an algorithm for a certain problem a good one?

In this section we aim to introduce some basic concepts related to complexity theory and to agree on terminology and notation we are going to use in this

context. Here, the presentation of Sections 1.2.1 and 1.2.2 is based to a large extent on [48], of Section 1.2.3 on [40] and [48], of Section 1.2.4 on [5] and of Section 1.2.5 on [40] and [68].

1.2.1 Fundamentals

The complexity of a problem or an algorithm characterizes it with respect to computational *resources*, such as time and space, required to solve it or, respectively, to run it until it possibly terminates with the right output.

To formalize this definition we need to be robust against such factors like structural characteristics of the *computing machine* or the *encoding scheme*.

Using an encoding scheme we can describe problem instances or any other objects in a string of characters over an alphabet:

Definition 1.3. An *alphabet* is a finite set with at least two elements, not containing the special symbol $\sqcup$ (which is used for blanks). The set of all finite strings whose symbols are elements of an alphabet A is denoted by A^*. A *language* over A is a subset of A^*. The elements of a language are often called *words*. If $x \in A^n$, we write $size(x) := n$ for the *length* of the string.

Referring to the *instances of a problem*, which can be seen as certain words of a language, we refer to the *input size* as to the length of the input, i.e. the number of digits needed to present the instance. An encoding system using the alphabet $A = \{0, 1\}$, is called *binary encoding* and the components of every string (digits) are called *bits*. We assume a fixed efficient encoding of the input as a binary string. An important observation is, that for different natural *efficient* encoding schemes the input size does not differ significantly.

To also get rid of the dependency on the computing system, while describing complexity, we will make use of the *deterministic Turing machine*. It is a simple theoretical computational model, which is able to perform a sequence of simple instructions working on a string (see [48] for a formal definition). The concept might appear very restrictive. However, it can compute any function and run any algorithm in polynomial time, if these are computable in polynomial time on any other computational model, due to *Church's thesis*. Another assumption we make is thus, that the costs of an algorithm on two different computational models regardless of the input size will differ no more than by a multiplicative constant [5].

The overall number of elementary steps determines the efficiency of an implemented algorithm. Obviously this number depends on the input size of the instance. To quantify the behavior of costs (time) depending on the input size, the running time of an algorithm or the time required to solve a problem is written as a function of the input size $f(n)$. Here, the following aspects can be observed. Firstly, even the

same-sized instances may cause very different runtime. Thus it is convenient to consider the *worst-case instance* of each size, to provide a reliable upper bound. Secondly, the subtleties of implementation, initialization time and computational model features are to abstract from. Finally, the *growth* rate of the running time depending on the input size is of interest.

Among others, these three aspects are dealt with by using the so-called O-*notation* or *Landau-notation*, named after the German mathematician Edmund Landau who spread its usage. It is used to describe the growth rate or the *order of* a function (where the letter choice comes from), i.e. indicates its dominating term. If it takes, for example, $5n^3 + n \log n + 7$ steps on a deterministic Turing machine for an algorithm to run on the worst instance of size n, we say, the algorithm has complexity $\mathrm{O}(n^3)$, or has *order of* n^3.

Using the Landau-notation we can formalize the term *polynomial time*. We say, an algorithm or a Turing machine is of *polynomial time* if its running time is upper-bounded by a polynomial in the input size, i.e. there exists a constant k, such that the algorithm complexity is $\mathrm{O}(n^k)$, where n denotes the size of the input.

For some purposes different approaches might be interesting. Occasionally we use the notation $\Omega(f(n))$ for the *best case complexity*, i.e. a lower bound on the runtime, and $\Theta(f(n))$ for the *tight bound*, i.e. to express that both $\mathrm{O}(f(n))$ and $\Omega(f(n))$ are valid.

Using the O-notation one can define *complexity classes*. If a problem can be solved within time $\mathrm{O}(f(n))$, it belongs to the *class* $\mathrm{O}(f(n))$.

1.2.2 Complexity Classes $\mathcal{P}$ and $\mathcal{NP}$, $\mathcal{NP}$-Hard Problems

Even though the problems of interest here are *optimization problems*, i.e. problems of finding among *feasible solutions* the *best* solution with respect to given costs, to characterize their complexity we need to track back to the so-called *decision problems*, which the important classes $\mathcal{P}$ and $\mathcal{NP}$ consist of and which complexity theory is based on.

Definition 1.4. A Turing machine or an algorithm *decides* a language L, if for a given input string it can determine in finite time, if the string is a word of the language. If there exists a Turing machine, which can decide a language L in polynomial time, the language is said to be *decidable in polynomial time*.

A *decision problem* is a pair $P = (X, Y)$, where X is a language decidable in polynomial time, and $Y \subseteq X$. The elements of X are called *instances* of P, the elements of Y are *yes-instances*, those of $X \setminus Y$ are *no-instances*.
An *algorithm for a decision problem* (X, Y) is an algorithm which can decide for every $x \in X$, if $x \in Y$.

If additionally the language of the yes-instances is decidable in polynomial time, the problem is said to be in the *class* $\mathcal{P}$:

Definition 1.5. The class of all decision problems for which there is a polynomial-time algorithm is denoted by $\mathcal{P}$.

Thus, to show that a given problem belongs to the class $\mathcal{P}$, it is sufficient to specify a polynomial-time algorithm for it. For a problem to be in the *class* $\mathcal{NP}$ it is not required that the language of the yes-instances is decidable in polynomial time. It is merely required that for each yes-instance there is a *certificate*, which can be encoded in polynomial time and which enables to decide in polynomial time if a given instance is a yes-instance:

Definition 1.6. A decision problem $P = (X, Y)$ belongs to the *class* $\mathcal{NP}$ if there is a polynomial p and a decision problem $P' = (X', Y')$ in $\mathcal{P}$, where

$$X' := \{x\#c \mid x \in X, c \in \{0,1\}^{\lfloor p(size(x))\rfloor}\},$$

such that

$$Y = \{y \in X \mid \text{ there exists a string } c \in \{0,1\}^{\lfloor p(size(x))\rfloor} \text{ with } y\#c \in Y'\}.$$

If $y\#c \in Y'$, the string c is called *certificate* for y.

It is easy to conclude that $\mathcal{P} \subseteq \mathcal{NP}$, but whether the inclusion is strict, is still the most important open question in complexity theory: For many decision problems in $\mathcal{NP}$ no polynomial time algorithm is known. Sometimes it can just be proven that a given problem is not easier than others.

Definition 1.7. Let $P_1 = (X_1, Y_1)$ and $P_2 = (X_2, Y_2)$ be decision problems. We say that P_1 *polynomially transforms* to P_2 if there is a function $f : X_1 \to X_2$ computable in polynomial time such that $f(x) \in Y_2$ for all $x \in Y_1$ and $f(x) \in X_2 \setminus Y_2$ for all $x \in X_1 \setminus Y_1$. Such a transformation is called *Karp-reduction* and we write $P_1 \leq_K P_2$.

This concept can be used to define $\mathcal{NP}$*-complete* problems:

Definition 1.8. A decision problem $P \in \mathcal{NP}$ is called $\mathcal{NP}$*-complete* if all other problems in $\mathcal{NP}$ polynomially transform to P.

Obviously, polynomial transformation is *transitive* and to show that a decision problem P from the class $\mathcal{NP}$ is $\mathcal{NP}$-complete, it is sufficient to polynomially transform one $\mathcal{NP}$-complete problem to P.

However, we are mostly interested in a different kind of problems.

Definition 1.9. An $\mathcal{NP}$ *optimization problem* is a quadruple

$$P = (X, (S_x)_{x \in X}, c, \text{goal}),$$

where

- X is a language over $\{0,1\}$ decidable in polynomial time. The elements of X are called *instances* of P;
- S_x is a nonempty subset of finite strings for each instance $x \in X$, called *feasible solutions* of x; these are polynomially encodable in the input size and it is decidable in polynomial time, if a given string is a feasible solution to a given instance;
- $c : \{(x,y) \mid x \in X, y \in S_x\} \to \mathbb{Q}$ is a function computable in polynomial time;
- goal $\in \{\max, \min\}$.

We write $OPT(x) := \text{goal}\{c(x,y) \mid y \in S_x\}$. An *optimal solution* of x is a feasible solution $y \in S_x$ with $c(x,y) = OPT(x)$. An *algorithm for an optimization problem* computes a feasible solution for each instance $x \in X$. It is called *exact algorithm*, if it computes an optimal solution of the given problem. The set of all $\mathcal{NP}$ optimization problems is called *class* $\mathcal{NPO}$.

To rank such problems with respect to their complexity and to establish relations between them to obtain complexity bounds, an extension of polynomial transformation is required.

Definition 1.10. [5] Let P be a decision- or an optimization problem. An *oracle* for problem P is an abstract device which provides an answer, if the given instance is a yes-instance, or, respectively, returns an optimal solution of P, for any instance of P.

Definition 1.11. Let P_1 and P_2 be decision or optimization problems. We say that P_1 *polynomially reduces* to P_2 if there exists an algorithm for P_1, that calls an oracle for P_2 and has polynomial runtime provided that each oracle call is counted as one step. Such reduction is called *Turing-reduction* and we write $P_1 \leq_T P_2$.

Obviously, polynomial transformation is a special case of polynomial reduction.

Now $\mathcal{NP}$*-hard* problems can be defined:

Definition 1.12. An optimization or a decision problem P is called $\mathcal{NP}$*-hard* if all problems in $\mathcal{NP}$ polynomially reduce to P.

Polynomial reduction is *transitive* and to prove that a problem P is $\mathcal{NP}$-hard, it is sufficient to polynomially reduce to P one $\mathcal{NP}$-complete or $\mathcal{NP}$-hard problem:

Lemma 1.13. *If $P_1 \leq_T P_2$ and P_1 is $\mathcal{NP}$-hard, then P_2 is $\mathcal{NP}$-hard.*

Sometimes we call $\mathcal{NP}$-hard problems *intractable*, which is equivalent to *not having a polynomial time algorithm*, unless $\mathcal{P} = \mathcal{NP}$. *Tractable* in turn means polynomially solvable, i.e. having a fast or efficient algorithm.

To give an upper bound on complexity, we may sometimes say *$\mathcal{NP}$-easy*, to indicate that a problem is *at most* as hard as some $\mathcal{NP}$-complete problem.

1.2.3 Weakly and Strongly $\mathcal{NP}$-Hard Problems

Unless $\mathcal{P} = \mathcal{NP}$, no $\mathcal{NP}$-hard problem can be solved in polynomial time. However, especially among numerical problems, i.e. problems whose instances can be represented by lists of integers (for example costs of elements), there exist problems which are hard to solve only if the input numbers are big. These problems can possibly be solved in polynomial time if the numerical value of the input is polynomially bounded, even while being $\mathcal{NP}$-hard.

Definition 1.14. Let P be a decision or an optimization problem. We denote by $largest(x)$ the numerical value of the largest number in the input instance x. An algorithm for P is called *pseudo-polynomial* if there is a polynomial $q : \mathbb{R}^2 \to \mathbb{R}$, such that the running time is bounded by $q\,(size(x), largest(x))$.

This definition is based on the difference between the numerical value of an integer and its length of encoding: For a numerical value w of an integer $\log(w)$ bits are sufficient to encode it, i.e. w grows exponentially in the length $\log(w)$ of the encoding. But for a pseudo-polynomial algorithm we allow that w appears next to $size(x)$ as an argument in the running time function, "hiding exponentiality" due to the relationship $\text{largest}(x) = \text{O}(\exp(size(x))$. In other words, a pseudo-polynomial algorithm is polynomial in the numerical value of the input, but might be exponential in the length of the input. Based on this difference, the following distinction is made:

Definition 1.15. A decision- or an optimization problem P is called *strongly $\mathcal{NP}$-hard*, if there is a polynomial p such that the subset of instances of P with $\text{largest}(x) \leq p(\text{size}(x))$ is still $\mathcal{NP}$-hard.

As an immediate consequence the following observation can be made:

Lemma 1.16. *Unless $\mathcal{P} = \mathcal{NP}$, no strongly $\mathcal{NP}$-hard problem can be solved by a pseudo-polynomial algorithm.*

Accordingly, we call a problem *weakly $\mathcal{NP}$-hard*, if it can be solved by a pseudo-polynomial algorithm.

Corollary 1.17. *A problem P can be shown to be strongly $\mathcal{NP}$-hard by giving a polynomial reduction from a strongly $\mathcal{NP}$-hard problem to an instance x of P, such that largest$(x) \leq p(size(x))$, for some polynomial p. A problem can be shown to be at most weakly $\mathcal{NP}$-hard by giving a pseudo-polynomial algorithm for this problem.*

To round off this section we give some important examples of strongly- and weakly $\mathcal{NP}$-hard problems used throughout the thesis.

Example 1.18. The following *Subset Sum-problem* is weakly $\mathcal{NP}$-hard, due to the polynomial transformation from the *Partition-problem* and existence of a pseudo-polynomial-time algorithm [40].

SUBSET SUM PROBLEM
Instance: A finite set A of positive integers and a positive integer B.
Question: Is there a subset $A' \subseteq A$ such that the sum of the elements in A' is exactly B?

□

Example 1.19. The following so called *Max Cut-problem* is strongly $\mathcal{NP}$-hard.

MAXIMUM CUT
Instance: Undirected graph $G = (V, E)$ with edge cost $c(e) \in \mathbb{Z}_+$ for all $e \in E$.
Goal: Find a partition of nodes into two subsets S and $V \setminus S$, such that the sum of the costs of the edges with one node in each subset is maximized.

□

Example 1.20. The following problem *Integer Linear Programming* is strongly $\mathcal{NP}$-hard [40].

INTEGER LINEAR PROGRAMMING
For given $m \in \mathbb{N}, c \in \mathbb{Z}^n, A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}$, find an optimal solution of

$$\begin{aligned} \min \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b \\ & x \in \mathbb{Z}^n. \end{aligned}$$

□

Example 1.21. The *Knapsack-problem* (see Section 1.1) is weakly $\mathcal{NP}$-hard, it can be solved in pseudo-polynomial time by dynamic programming [47]. □

1.2.4 Approximability

Whenever dealing with $\mathcal{NP}$-hard problems, where it is unlikely to find an efficient algorithm, it seems useful for practical demands to have an algorithm which possibly does not provide an optimal solution, but a feasible solution close to the optimal and in short time. We call such algorithms *approximation algorithms* and the corresponding feasible solution *approximate solution*.

Of course, such algorithms differ in quality of the provided solution, or the *performance ratio*:

Definition 1.22. Given an optimization problem P, for any instance x of P and for any feasible solution y of x, the *performance ratio* of y with respect to x is defined as

$$R(x,y) = \begin{cases} \frac{v(x,y)}{OPT(x)} & \text{if } goal = \min \\ \\ \frac{OPT(x)}{v(x,y)} & \text{if } goal = \max, \end{cases}$$

where $OPT(x)$ denotes the optimal value of the instance and $v(x,y)$ the value of the solution y.

Approximation algorithms can be classified by their performance ratio:

Definition 1.23. Given an optimization problem P and an approximation algorithm A for P, we say that A is an *r-approximate algorithm* for P if, given any input instance x of P, the performance ratio of the approximate solution is bounded by r.

These algorithms are also referred to as *constant factor approximation algorithms*. Now, taking this definition into account, $\mathcal{NP}$-hard problems can be classified by the question how closely they can be approximated in polynomial time:

Definition 1.24. An $\mathcal{NPO}$ problem P is *r-approximable* if there exists a polynomial-time r-approximate algorithm for P, for some $r \geq 1$. The *class APX* consists of all r-approximable optimization problems, for any r.

Note that APX stands for *approximable*. Often an r-approximation for some constant r is not enough for practical use and we are willing to find an approximate solution with a better performance ratio, even though at expense of the running time.

Definition 1.25. For an $\mathcal{NPO}$ problem P an algorithm A is said to be a *polynomial-time approximation scheme (PTAS)* if, for any instance x of P and any rational value $r > 1$, A with input (x,r) returns an r-approximate solution in time polynomial in the input size of x. The *class PTAS* consists of all $\mathcal{NPO}$ problems that admit a polynomial-time approximation scheme.

Note that the running time of a PTAS may still depend on the performance ratio (which would be natural) and may grow super-polynomially in $\frac{1}{r-1}$. If this situation is still unsatisfactory in terms of running time, one might be interested in the following:

Definition 1.26. For an $\mathcal{NPO}$ problem P an algorithm A is said to be a *fully polynomial-time approximation scheme (FPTAS)* if, for any instance x of P and any rational value $r > 1$, the algorithm A with input (x, r) returns an r-approximate solution in time polynomial both in the input size of x and in $\frac{1}{r-1}$. The *class FPTAS* consists of all $\mathcal{NPO}$ problems that admit a fully polynomial-time approximation scheme.

Note that we are talking about approximation *schemes* because there is an algorithm A_{r-1} for any precision $r - 1$.

To admit an FPTAS is the strongest possible polynomial-time approximation result for an $\mathcal{NPO}$ problem. For some $\mathcal{NPO}$ problems it can be shown that they are not r-approximable, unless $\mathcal{P} = \mathcal{NP}$, i.e. the class APX is strictly contained in the class $\mathcal{NPO}$. Under the same assumption one can find problems in APX that do not admit a PTAS or an FPTAS, such that the following relationship holds:

Lemma 1.27. *If $\mathcal{P} \neq \mathcal{NP}$, then $\mathcal{P} \subset FPTAS \subset PTAS \subset APX \subset \mathcal{NP}$.*

Assuming $\mathcal{NP} \neq \mathcal{P}$, certain problems are truly harder to approximate than others. If a problem is at least as hard to approximate as any other problem in APX, it is called *APX-hard* and it does not admit a PTAS, unless $\mathcal{P} = \mathcal{NP}$. APX-hard problems can also be defined in analogy to Definition 5.1. However, a stronger reducibility concept is required, since problems that are only polynomially reducible to each other often have different approximability properties. Roughly speaking, it is not enough to map instances of one problem to instances of another problem. Additionally, good solutions should be mapped to good solutions. To prove that a given problem is APX-hard, a so-called *approximation preserving reduction* from another APX-hard problem is to be performed (see [5] for a formal definition). An example of an approximation-preserving reduction is the so-called *PTAS-reduction*.

1.2.5 FPTAS and Pseudo-Polynomial Algorithms

We have seen that some problems are easier to solve than others in a certain sense, unless $\mathcal{P} = \mathcal{NP}$. Problems of the class FPTAS are easy to approximate; weakly $\mathcal{NP}$-hard problems are easy to solve if we bound the numerical input value. An obvious question is if there is a relationship between FPTAS and pseudo-polynomial algorithms. It is known that most of the FPTAS are derived

from pseudo-polynomial algorithms, such that the property of weak $\mathcal{NP}$-hardness seems to be relevant for the existence of an FPTAS. Garey and Johnson [40] establish the following relationship:

Theorem 1.28. *Let P be an optimization problem such that all solution values are positive integers. If the optimal value of an instance x of P is bounded by a polynomial in the input size and the largest numerical value of the input, then the existence of an FPTAS for P implies the existence of a pseudo-polynomial algorithm for P.*

We point out the important contraposition of this theorem.

Corollary 1.29. *Under the assumptions of Theorem 1.28, no strongly $\mathcal{NP}$-hard problem can be solved by an FPTAS.*

Inversely, a relevant question is, if we can always derive an FPTAS from a pseudo-polynomial algorithm. In [68] the authors claim that this is not true and give several examples.

Example 1.30. The following problem, which is referred to as *Two-Dimensional Knapsack*, can be solved in pseudo-polynomial time and does not have an FPTAS.

TWO-DIMENSIONAL KNAPSACK
Instance: n items with positive integer weights w_i and positive integer volumes $v_i, i = 1, \ldots, n$, weight capacity W and volume capacity V.
Goal: Maximize the number of items without exceeding the weight and volume limits.

□

1.3 Multicriteria Optimization

Multicriteria or *multiobjective optimization* deals with the optimization of *more than one objective* simultaneously and is often referred to as *vector optimization*. This is a challenge arising in a wide range of situations, especially in interdisciplinary applications. A *multicriteria optimization problem* can be formulated as

$$\begin{array}{ll} \min & (f_1(x), \ldots, f_k(x)) \\ \text{s.t.} & x \in X \ , \end{array} \tag{1.1}$$

with $k \geq 2$ being the number of objectives. If $k = 2$, Problem (1.1) is called *bicriteria optimization problem*.

The non-trivial case of multiobjective optimization is the case when a solution that is optimal for all k objectives does not exist, as the objectives can be conflicting. Usually, minimization of the production costs does not walk along with the increase

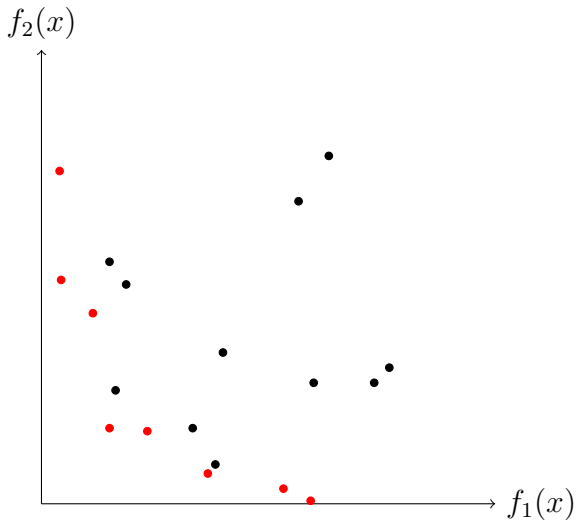

Figure 1.1: Projection of the feasible set X onto the span of the objectives $f_1(x)$ and $f_2(x)$. Pareto-frontier (red).

in quality, and maximizing a tank capacity contradicts the aim for a minimum weight. Criteria may though be totally unrelated, for example, when a solution has to comply with the requirements of engineering, logistics, economics and environment at the same time.

For this reason, it is not the original objective in multicriteria optimization to find one common optimal solution, which most likely does not exist. Instead a *set* of so-called *non-dominated* solutions is considered.

Definition 1.31. A feasible solution $x \in X$ is said to *(Pareto-) dominate* a feasible solution $y \in X$, if, assuming minimization,

(1) $f_i(x) \leq f_i(y)$ for all $i = 1, \ldots, k$ and

(2) $f_j(x) < f_j(y)$ for at least one $j \in \{1, \ldots, k\}$.

Definition 1.32. A feasible solution $x \in X$ is *(Pareto-) efficient* or *(Pareto-) optimal*, if there does not exist another solution that dominates it. The corresponding vector $f(x)$ is called a *non-dominated point*.

The set of all Pareto-efficient solutions constitutes the so-called *Pareto-frontier* and by abuse of notation is often referred to as the set of non-dominated points in the image domain (see Figure 1.1). The concept of *efficiency* is in most cases related to the decision space X, while the concept of *dominance* refers to the criterion space $f(X)$. The name goes back to Vilfredo Pareto, who introduced and extensively used this concept in his contributions to the field of microeconomics [63].

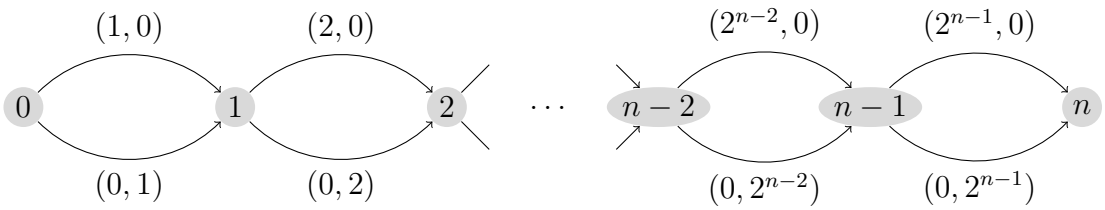

Figure 1.2: A Shortest Path-instance with an exponential number of efficient solutions.

If no additional subjective preference information is given, the Pareto-efficient solutions are considered equally good. Hereby, the biggest problem is that there might exist exponentially many efficient solutions:

Example 1.33. Consider the Shortest Path-problem on the graph in Figure 1.2. Here, all feasible solutions (all $0-n$-paths) are Pareto-efficient: Sorting them in ascending order by the values of the first function, we get the set of non-dominated points in the image domain

$$\{(0, 2^n - 1), (1, 2^n - 2), (2, 2^n - 3), \ldots, (2^n - 3, 2), (2^n - 2, 1), (2^n - 1, 0)\},$$

which is of exponential size. □

In this regard, a large variety of goals, philosophies and methods were developed in multicriteria optimization. To *solve* a multicriteria optimization problem can mean

- to find *all Pareto-optimal* solutions,
- to find a *representative set* of Pareto-optimal solutions, or
- to find one *most preferred* solution due to some certain subjective preferences criteria.

In the last point we can distinguish between *upfront* and *a posteriori* preferences. The a posteriori methods are mostly based on estimations of a human decision maker, who can assess which solution is a good compromise in a given application. In the upfront methods, Problem (1.1) is often converted into a single-objective problem. A famous approach here is, for example, the *weighted sum method* [56].

An overview of fundamental concepts, methods and results on multicriteria optimization is given in [34].

Chapter 2

Robust Optimization under Ellipsoidal Uncertainty

Now we discuss the actual topic of this thesis, namely robust combinatorial optimization under ellipsoidal uncertainty, in a greater detail. To this aim we will in Section 2.1 motivate and demarcate the idea of robustness and give an overview to the existing approaches, with the focus on strict robustness, as it is the central paradigm in this thesis. In this context we introduce different uncertainty sets and outline the corresponding complexity results.

As ellipsoidal uncertainty is our basic topic, we go into more detail and dedicate the Sections 2.2 to 2.4 to this uncertainty set. There we describe in detail the existing solution approaches and summarize the complexity results known in the literature. We not only introduce basic definitions but also discuss and prove some fundamental statements, such that we can build on this prior knowledge in the following chapters.

2.1 Robust Optimization

An abstract optimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in X, \end{array} \tag{2.1}$$

with the objective function f and the feasible set X, becomes a real-world situation as soon as concrete data is given, for example, a specific network with the costs or duration of proceeding through every edge. But in fact, the data is hardly ever known precisely, but subject to uncertainty caused by various matters. It can have a social, ecological or financial character, like uncertainty about the exact soil conditions while planning new power lines and related costs. It can be caused

by future and unpredictable events, like political decisions or natural disasters with associated loss of capacities. Disposal over only inexact or estimated data or limited access to information, as well as a simple measuring errors, also may produce uncertainty.

In the following we assume to be given an *uncertain optimization problem* of the form

$$\begin{array}{ll} \min & f(x, c) \\ \text{s.t.} & x \in X_c, \end{array} \tag{2.2}$$

where the objective function and the feasible set now in some way depend on a random variable c. For a fixed c Problem (2.2) becomes a *certain* Problem of the form (2.1), which we sometimes refer to as the *nominal* or *deterministic problem*.

In order to solve an optimization problem, we aim to find a feasible solution having the best objective value. Now with uncertain data, feasibility and/or optimality of a solution might be affected. Sometimes even small perturbations may lead to a complete uselessness of a chosen solution, while the decision-making process does not allow any changes of the adopted solution after the data reveals itself. What is the chosen solution worth, if its optimality or even feasibility are not guaranteed any more? The concept of *optimality* or even *solution* must be redefined when dealing with uncertainty. Uncertainty must be taken into account already in the problem definition.

The concept of *scenario* is a common and natural way to structure uncertainty. Usually it is at least possible to overview all the relevant scenarios, how the data may behave, and to put these together in a so-called *uncertainty set* $\mathcal{U}$. While redefining feasibility and optimality, it makes sense to ask for *feasibility in every possible scenario* and for certain *guarantee of quality*. This is a common thought in all existing *robustness criteria*. Many such criteria were proposed in a short time.

In the following we describe different approaches to define the so-called *robust counterpart* of Problem (2.2) and summarize the most important forms of $\mathcal{U}$.

2.1.1 Strict Robustness

The *strict robustness* was first mentioned by Soyster [70] and was extensively studied since then [10, 35, 13, 15]. The initial idea to call a solution *robust* against uncertainty is to require its feasibility and possibly good quality in every possible scenario (see Figure 2.1).

Such a perfect solution though may hardly exist. The general way out in robust optimization is to accept some quality loss against certain immunity to parameter changes. The idea of strict robustness is to consider the *worst-case* scenario, to be sure that the value of the solution would not get any worse in any case. But the worst-case scenario might vary from solution to solution, such that it might not be

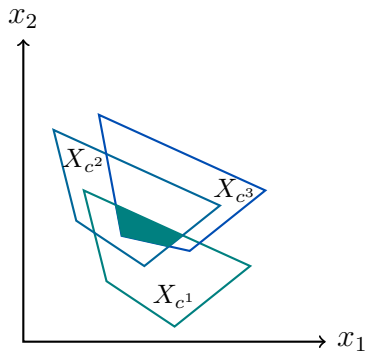

Figure 2.1: Strict Robustness: The feasible set of a strictly robust problem is the intersection of the feasible sets in every scenario.

enough to consider one bad scenario. Strict robustness considers for every solution its value in *its* worst-case scenario and compares solutions regarding these values, i.e. it solves the problem

$$\begin{aligned} \min \quad & \max_{c \in \mathcal{U}} f(x, c) \\ \text{s.t.} \quad & x \in X_c \text{ for all } c \in \mathcal{U}. \end{aligned} \tag{2.3}$$

This criterion is occasionally called *min-max*-robustness or *absolute* robustness. We minimize the maximum value a solution can reach among all possible scenarios from the uncertainty set $\mathcal{U}$. Problem (2.3) defines the so-called *robust counterpart* of the uncertain Problem (2.2) in the case of strict robustness.

Here, the difference to stochastic programming, which is another approach to deal with data uncertainty, has to be emphasized: We do not consider or estimate any probability distributions of the scenarios to get a *good solution on average*, but we want to be free of risk in *every* reasonable scenario, no matter how likely or unlikely it is to occur.

It must be said that this great pessimism also can be seen as a limit of this approach. Sometimes really improbable bad cases are strongly overestimated, such that the provided solution is very conservative and can be far away from the optimum in the realized scenario. This trade-off between risk-aversion and optimality is often referred to as *price of robustness* [14].

A related and less conservative approach to define an optimal solution under uncertainty is *min-max-regret* [2]. It aims to minimize the *regret* one might feel when the data reveals itself and the chosen solution cannot be changed any more, i.e. the difference between the value of the optimal solution in the realized scenario and the value of the chosen solution. The corresponding robust formulation is

$$\begin{aligned} \min \quad & \max_{c \in \mathcal{U}} f(x, c) - f(x_c^*, c) \\ \text{s.t.} \quad & x \in X_c \text{ for all } c \in \mathcal{U}, \end{aligned} \tag{2.4}$$

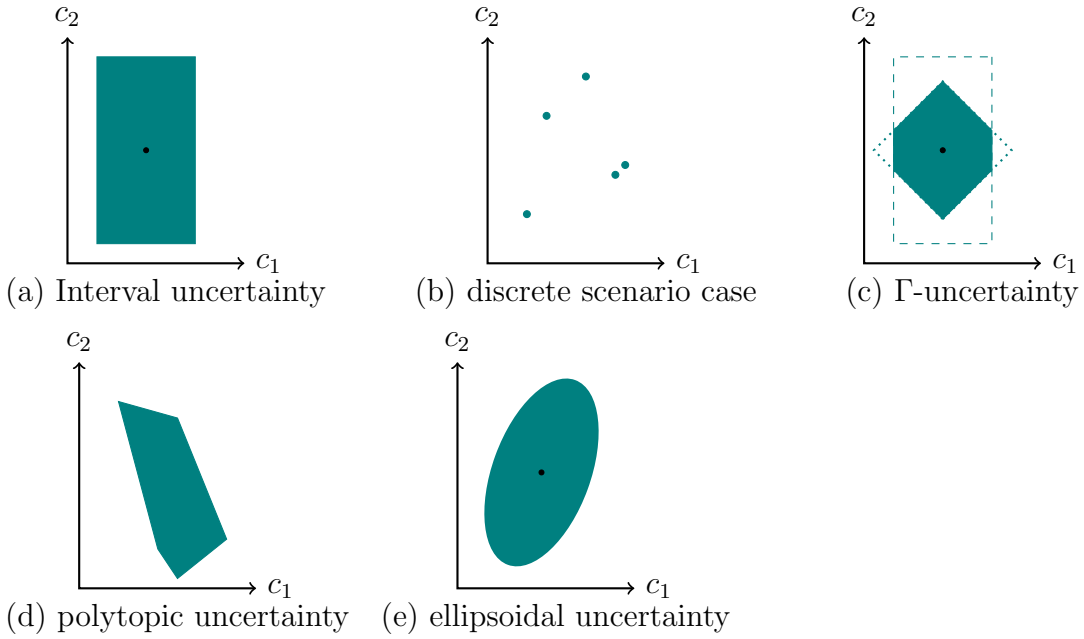

Figure 2.2: Uncertainty sets.

where x_c^* denotes the optimal solution in scenario c.

In general, Models (2.3) and (2.4) yield different solutions. The usage of one or another criterion depends on their appropriateness in a given application. The min-max-regret criterion is preferred in situations, where the importance of a comparison between the performances of different actions or actions of the opponent actors is crucial. This can be the case, for example, in the investment management, where the *opportunity costs* play a major role.

2.1.2 Uncertainty Sets

The complexity of and approaches to the robust counterparts (2.3) differ fundamentally depending on the definition of the uncertainty set $\mathcal{U}$. Clearly, in different situations the possible scenarios may constitute differently shaped sets. But the main criterion to consider different forms of $\mathcal{U}$ is the tractability of the resulting counterparts.

2.1.2.1 Boxes

One natural way to describe the set of possible scenarios is to define a lower and an upper bound on every uncertain coefficient, assuming that the costs may vary within the corresponding intervals. This leads to a box form of the uncertainty set (see Figure 2.2(a)) and the uncertainty type is called *interval uncertainty* [2].

From the practical point of view this case is rather attractive but theoretically uninteresting. The coefficients are completely unrelated to each other and the worst case of every solution is the same, namely the scenario with all coefficients being on their upper bounds. In this case the robust counterpart preserves the complexity of the certain Problem (2.1):

Theorem 2.1. *[2] The robust counterpart of* (2.2) *with interval uncertainty is solved by solving the nominal problem* (2.1) *with the worst-case scenario.*

On the other hand, the optimal solution is over-conservative, given the fact that the case with all coefficients on their upper bounds is very unlikely to happen, which makes the user accept an unnecessarily bad value. These two facts make it worth to consider different forms of uncertainty.

2.1.2.2 Finite Sets

In some situations the set of possible scenarios is explicitly given. Such situations are called *discrete-scenario case* (see Figure 2.2(b)). Here, the challenge is to reduce the cardinality of $\mathcal{U}$ as far as possible, due to the following.

Theorem 2.2. *[50] In the discrete scenario case the problem*

$$\begin{array}{ll} \min & \max\limits_{c \in \mathcal{U}} c^\top x \\ s.t. & x \in X \end{array} \tag{2.5}$$

is $\mathcal{NP}$-hard, even for $|\mathcal{U}| = 2$ and some X where linear optimization is easy.

Here and in further considerations we assume a linear objective function of the nominal problem, i.e. $f(x, c) = c^\top x$.

This result originates from [50] for the Shortest Path-problem, the Minimum Spanning Tree-problem, Minimum Cost Assignment and Resource Scheduling and was shown in [6] for the unconstrained case $X = \{0, 1\}^n$.

Nevertheless, for a constant number of scenarios many combinatorial problems can be solved in pseudo-polynomial time. But if the number of scenarios is part of the input, these problems become strongly $\mathcal{NP}$-hard (see [2] for an overview).

2.1.2.3 Trimmed Boxes

Another uncertainty set is proposed in [13] and allows to adjust the level of conservatism in contrast to the interval uncertainty. The authors suggest that it is very unlikely that all coefficients differ from the expected scenario at the

same time and restrict the number of changing coefficients to at most $\Gamma \geq 0$. The uncertainty set can be written as

$$\mathcal{U} = \{c \in [l_1, u_1] \times \cdots \times [l_n, u_n] \mid \#\{I \subseteq \{1, \dots n\} \mid |u_n - c_{0n}| > 0\} \leq \Gamma\},$$

where c_0 denotes the vector of the coefficients in the expected scenario, and is called Γ-*uncertainty*.

The parameter Γ allows to control the value of conservatism, while the corresponding robust counterpart remains computationally tractable.

Theorem 2.3. *[13] The robust counterpart* (2.5) *with* Γ*-uncertainty is reduced to solving* $n+1$ *corresponding nominal problems* (2.1) *if* n *is the number of variables.*

Another variant of Γ-uncertainty is to restrict the *total deviation* from the expected scenario [18], i.e.

$$\mathcal{U} = \left\{ c \in [l_1, u_1] \times \cdots \times [l_n, u_n] \mid \sum_{i=1}^{n} |c_i - \frac{l_i + u_i}{2}| \leq \Gamma \right\},$$

(see Figure 2.2(c)). Also here Problem (2.5) is reduced to solving $n + 1$ nominal problems, which can be shown in analogy to [13].

2.1.2.4 Polytopes

Interval and Γ-uncertainty are special cases of *polytopic uncertainty*. Here, the uncertainty set is a general polytope:

$$\mathcal{U} = \{c \in \mathbb{R}^n \mid Tc \leq s\},$$

with $T \in \mathbb{R}^{r \times n}$ and $s \in \mathbb{R}^r$ [18] (see Figure 2.2(d)).

Since $\mathcal{U} = \text{conv}(v_1, \dots, v_k)$ for some suitable k due to the Weyl-Minkowski Theorem [58], the following result applies.

Theorem 2.4. *The robust Problem* (2.5) *with polytopic uncertainty is at least as hard as with discrete uncertainty and a fixed number of scenarios, even for some* X *where linear optimization is easy.*

Since for a fixed finite number k of points their convex hull, which is a polytope, can be constructed in linear time, the discrete scenario case for a fixed number of scenarios, which can be hard due to Theorem 2.2, reduces to Problem (2.5) with polytopic uncertainty.

We hold down that the polytopic uncertainty is in general hard to treat, while some easy special cases, such as interval and Γ-uncertainty, exist.

2.1.2.5 Ellipsoids

In the focus of this thesis is *ellipsoidal uncertainty*, where $\mathcal{U}$ is given by an ellipsoid in $\mathbb{R}^n$:

$$\mathcal{U} = \left\{ c \in \mathbb{R}^n | \sqrt{(c - c_0)^\top A^{-1} (c - c_0)} \leq r \right\},$$

with $c_0 \in \mathbb{R}^n$, $r > 0$ and $A \in \mathcal{S}^n_{++}$ (see Figure 2.2(e)). Here $\mathcal{S}^n_{++}$ denotes the set of all positive definite matrices of order n.

Here the costs are assumed to be caused by some normal distribution with mean c_0 and a covariance matrix A, which relates to many applications (see Section 2.2 for motivation of this notation). This form of uncertainty is less conservative than interval uncertainty since the very unlikely extreme points of $\mathcal{U}$, where the worst case is taken in the interval case, is explicitly excluded here. Moreover, it allows to model correlations between the coefficients, both positive and negative. Finally, it is practically and theoretically interesting, as it incorporates both tractable and intractable special cases, as well as some still not specified cases.

2.1.3 Different Approaches

Strict robustness is very conservative and in many cases intractable. Also, in many applications different conditions apply, such that in the last two decades alternative robustness approaches have been developed. These define different criteria of robust optimality. Here, we shortly mention some of these approaches.

2.1.3.1 Recoverable Robustness

The concept of *recoverable robustness* was formalized in [53]. The idea is to find a solution which may become infeasible or bad when the data becomes certain, but which can then be *recovered* to a feasible or a better solution by means of one of the given *recovery algorithms* (see Figure 2.3).

Recoverable robustness hence distinguishes two phases: A *planning phase* and a *recovery phase*. A set $\mathcal{A}$ of *admissible recovery algorithms* is given beforehand and specifies the *recovery possibilities*. In the *planning phase* a solution x and an algorithm $A \in \mathcal{A}$ are determined. A specifies *how* x can be recovered for every scenario. In the *recovery phase* (when a scenario is realized), the algorithm A is used to turn the solution x into a feasible solution for the realized scenario. The *recovery robust problem* can be formulated as

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & A(x, c) \in X_c \text{ for all } c \in \mathcal{U} \\ & (x, A) \in X \times \mathcal{A}, \end{array} \tag{2.6}$$

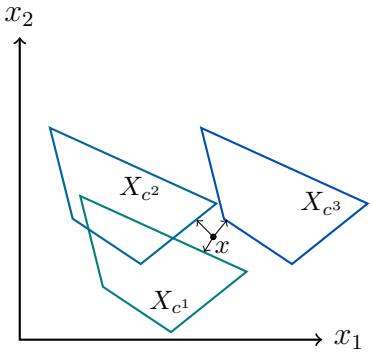

Figure 2.3: Recoverable robustness.

where X_c denotes the set of feasible solutions in the scenario c.

Note that the solution of Problem (2.6) is a *tupel* (x, A). If no recovery is possible for any solution x by means of the given set of algorithms, Problem (2.6) has value ∞.

Obviously, the set of recovery algorithms should be limited due to some reasonable criteria, which mostly concern two main aspects:

- the recovered solution must not be too far from the recoverable solution x with respect to a certain measure of distance;
- the computational effort of the algorithm A to turn the solution x to a feasible solution should be restricted, preferably in terms of the computational effort to compute the solution x in the planning phase.

If $\mathcal{A}$ consists of exactly one algorithm A with $A(x, c) = x$ for all $c \in \mathcal{U}$, then we are in the case of strict robustness.

Since its first formalization, recoverable robustness has been applied in the context of railways, shunting, timetabling and delay management and many other fields [26, 27]. Clearly, it strongly depends on the underlying problem and on the set $\mathcal{A}$ whether Problem (2.6) is tractable or not.

2.1.3.2 Adjustable or Two-Stage Robustness

In *adjustable* or *two-stage-robustness* [11] certain variables are allowed to be determined *after* the realization of a scenario. The set of variables is decomposed into

- the *here-and-now* variables $x \in \mathbb{R}^{n_1}$, which have to be determined in the *first stage*, i.e. *before* the realization and

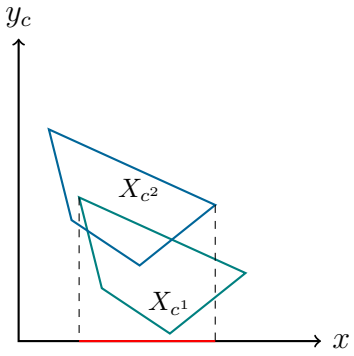

Figure 2.4: Adjustable robustness.

- the *wait-and-see* variables $y_c \in \mathbb{R}^{n_2}$, which have to be determined in the *second stage*, i.e. *after* the realization of a scenario.

The goal is to find an *adjustable solution* $x \in \mathbb{R}^{n_1}$, i.e. such that for every $c \in \mathcal{U}$ there exists a $y_c \in \mathbb{R}^{n_2}$ such that (x, y_c) is feasible for c. The *adjustable robust problem* is

$$\begin{array}{lll} \min\limits_{x \in \mathbb{R}^{n_1}} & \max\limits_{c \in \mathcal{U}} & \min \quad f(x, y_c, c) \\ & & \text{s.t.} \quad (x, y_c) \in X_c \\ & & \qquad y_c \in \mathbb{R}^{n_2} \end{array} \tag{2.7}$$

(see Figure 2.4). It turns out that even for the uncertainty sets that are tractable in the strictly robust case, Problem (2.7) cannot be solved efficiently in general, such that the research in this field is mostly concentrated on approximation of the adjustable robust problems or considering further restrictions and special cases [11].

2.1.3.3 K-Adaptability

Due to the computationally challenging treatment of the adjustable robust problem, a modified approach has been proposed in [43], which provides an approximation of Problem (2.7). The idea of *K-adaptability* is to *preselect* K of the second-stage decisions in the first stage and to choose among these the best in the second stage, i.e. after a scenario has been realized. The problem to solve is

$$\begin{array}{ll} \min & \max\limits_{c \in \mathcal{U}} \min\limits_{i=1,\dots,k} f(x, y^i, c) \\ \text{s.t.} & \forall\, c \in \mathcal{U}\ \exists\, i \in \{1, \dots, k\} \text{ such that } (x, y^i) \in X_c \\ & x \in \mathbb{R}^{n_1} \\ & y^i \in \mathbb{R}^{n_2}, i = 1, \dots, k. \end{array} \tag{2.8}$$

A survey on computational complexity and a reformulation to a mixed-integer linear program is given in [43].

2.1.3.4 Min-Max-Min-Robustness

A special case of K-adaptability and an extension of the strict robustness has been recently proposed in [52]. Like in the strict robustness, the worst case is minimized. But now we are not limited to only one solution, but determine *here and now* exactly k *solutions*, among which the best suitable can be chosen in the second stage. The robust counterpart can be formulated as

$$\begin{array}{ll} \min & \max\limits_{c\in\mathcal{U}} \min\limits_{i=1,\dots,k} f(x^i, c) \\ \text{s.t.} & x^1, \dots, x^k \in X. \end{array} \tag{2.9}$$

With $n_1 = 0$ and only uncertainty in the objective function this can be seen as a special case of K-adaptability. On the other hand, for $k = 1$ Problem (2.9) is equivalent to the strictly robust problem (2.3).

Problem (2.9) and its complexity were extensively analyzed in [19] and [52]. It turns out that for convex uncertainty sets and for $k \geq n$ under additional mild conditions the robust problem preserves the complexity of the nominal problem. However, it is $\mathcal{NP}$-hard for any fixed number $k < n$, even for $X = \{0, 1\}^n$. For discrete uncertainty Problem (2.9) is $\mathcal{NP}$-hard, which was explicitly shown for a number of combinatorial problems.

2.1.3.5 Light Robustness

Light Robustness [36] aims to be a more flexible and less expensive alternative to strict robustness. In order to obtain less conservative solutions than in the common strict robustness approach, i.e. solutions that are not too bad for the expected scenario c_0, a bound ρ on the distance to the optimal solution in the expected scenario is given ensuring certain nominal quality. This in general may lead to an infeasible task. To restore feasibility, slack variables γ are introduced which allow violation of the feasibility requirement, but which are minimized in terms of an auxiliary objective function. In other words, the slack variables penalize infeasibility. The corresponding problem reads

$$\begin{array}{ll} \min & \|\gamma\| \\ \text{s.t.} & f(x, c_0) \leq f^*_{c_0} + \rho \\ & F(x, c_0) \leq 0 \\ & F(x, c) \leq \gamma \text{ for all } c \in \mathcal{U} \\ & \gamma \geq 0, \end{array} \tag{2.10}$$

where the condition $F(x, c_0) \leq 0$ ensures feasibility in the expected scenario c_0. This concept was introduced in [36] for linear programming and interval uncer-

tainty. It was generalized in [66] to general optimization problems and arbitrary uncertainty sets and studied concerning its theoretical complexity. The authors show that in the case of polytopic uncertainty the *lightly-robust counterpart* of a linear program remains linear and in the case of ellipsoidal uncertainty a quadratic program has to be solved.

2.2 Ellipsoidal Uncertainty

We now address the central topic of this thesis, i.e. ellipsoidal uncertainty in combinatorial problems. Before stating our main results, we note down the state of knowledge in this area. For the proofs of the statements we will refer to the literature given within the text. Thus, in the following we will first present the model of a combinatorial problem under the assumption of uncertain data, more precisely, uncertain objective function, in the case of ellipsoidal uncertainty and then reformulate the problem in the way we will investigate it in this work, motivating this form in various ways. We will analyze the theoretical complexity of the problem as well as some of its special cases, as far as it is known in the literature, and expose the existing handling methods. Here the presence or absence of correlations between the uncertain objective function coefficients implies an essential difference in the complexity and methodology.

In particular, in the uncorrelated case we expand on the robust Shortest Path-problem, with its complexity status being a challenge in the complexity research.

2.2.1 Motivation and Modeling

The following formulation of the *robust combinatorial problem*

$$\begin{array}{ll} \min & \max\limits_{c\in\mathcal{U}} c^\top x \\ \text{s.t.} & x \in X\ , \end{array} \tag{2.11}$$

with $X \subseteq \{0,1\}^n$ describing the set of feasible solutions having some combinatorial structure, leaves open the question about the possible *scenarios* $c \in \mathbb{R}^n$. But we assume this information to be given and structured by specifying the form of the *uncertainty set* $\mathcal{U}$ which describes the set of possible scenarios. Among all various forms of $\mathcal{U}$ which are motivated by the tractability of the resulting problem (2.11), the requirements of the application, available information, preferences of the user and many other factors, we concentrate on *ellipsoidal uncertainty*. Here, $\mathcal{U}$ is given by an *ellipsoid* in $\mathbb{R}^n$:

$$\mathcal{U} = \left\{ c \in \mathbb{R}^n | \sqrt{(c-c_0)^\top A^{-1} (c-c_0)} \leq r \right\},$$

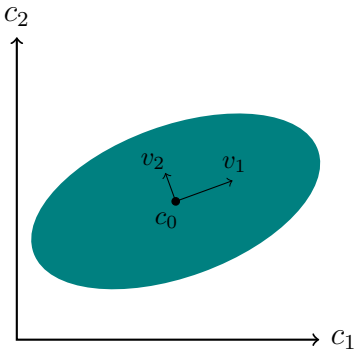

Figure 2.5: Geometry of an ellipse.

see Section 2.1.2.5. The vector $c_0 \in \mathbb{R}^n$ represents the centroid of the ellipsoid or the *mean* of the possible costs, and $A \in \mathcal{S}^n_{++}$ is a positive definite matrix containing the following information: The eigenvectors v_i of A (or A^{-1}) represent the orientations of the principal axes of the ellipsoid. The length of a half-axis in the direction of the eigenvector v_i is $\sqrt{\lambda_i}$, with λ_i being the eigenvalue of A belonging to the eigenvector v_i. In particular, the largest eigenvalue corresponds to the eigenvector pointing to the direction of the largest extent of the ellipsoid [38].

We refer to A as the *covariance matrix,* defining the deviation of costs from the mean value and their correlations among each other. The parameter $r > 0$ is used to scale the ellipsoid. Its increasing by a factor p is equivalent to decreasing of every entry of A by a factor p^2.

2.2.1.1 Optimality Conditions and Closed Formula

We start by stating Problem (2.11) with ellipsoidal uncertainty in a more manageable form. The objective function of the robust Problem (2.11) consists of maximizing a linear function over a convex compact set (an ellipsoid). Thus, we can consider the Karush-Kuhn-Tucker optimality conditions for the inner maximization problem. Note that a Slater-point is given by c_0 [6, 61].

For a non-trivial case of $x \neq 0$, consider the objective function

$$\max_{c \in \mathcal{U}} c^\top x = -\min_{c \in \mathcal{U}} -c^\top x \tag{2.12}$$

of (2.11) as an optimization problem in c. We find a Lagrangean multiplier $\lambda^* \geq 0$ satisfying the following necessary and sufficient conditions on the optimal solution c^* for every fixed x:

$$(c^* - c_0)A^{-1}(c^* - c_0) - r^2 \leq 0 \qquad \text{(Primal feasibility)}$$

$$-x + 2\lambda^* A^{-1}(c^* - c_0) = 0 \qquad \text{(Stationarity)}$$

$$\lambda^* \left((c^* - c_0)^\top A^{-1}(c^* - c_0) - r^2\right) = 0 \qquad \text{(Complementary slackness)}$$

Obviously, $\lambda^* > 0$, since otherwise the stationarity would imply $x = 0$. Thus, due to non-singularity of A, we can rewrite the stationarity condition to

$$c^* - c_0 = \frac{1}{2\lambda^*} Ax. \tag{2.13}$$

Complementary slackness and $\lambda^* > 0$ also yield

$$(c^* - c_0)^\top A^{-1}(c^* - c_0) - r^2 = 0, \tag{2.14}$$

where we can insert the equation (2.13) and get

$$\begin{aligned} &\left(\frac{1}{2\lambda^*} Ax\right)^\top A^{-1} \left(\frac{1}{2\lambda^*} Ax\right) - r^2 = 0 \\ \iff &\lambda^* = \frac{1}{2r}\sqrt{x^\top Ax}. \end{aligned}$$

Inserting λ^* into (2.13) we get a closed formula for the optimal solution

$$c^* = c_0 + r\frac{Ax}{\sqrt{x^\top Ax}}$$

of (2.12) and the formulation

$$c_0^\top x + r\sqrt{x^\top Ax}$$

of the objective function of (2.11). Hence, Problem (2.11) can be written in the form

$$\begin{array}{ll} \min & c_0^\top x + r\sqrt{x^\top Ax} \\ \text{s.t.} & x \in X\ . \end{array} \tag{MR}$$

In the trivial case $x = 0$ the reformulation of the objective function is obviously valid, too.

Problem (MR) is often referred to as the *mean-risk-problem* [60] and is mainly applied and studied in the realm of finance and investment management. The objective is a convex combination of the *mean* $c_0^\top x$ and the *standard deviation* $\sqrt{x^\top Ax}$ of a solution $x \in X$ with respect to a stochastic cost vector $c \in \mathbb{R}^n$, given by an independent distribution. More precisely, the entries of A contain the information about the correlations of the cost-coefficients between each other, with the variance values given by the diagonal of A, and the entries of c_0 contain the expected

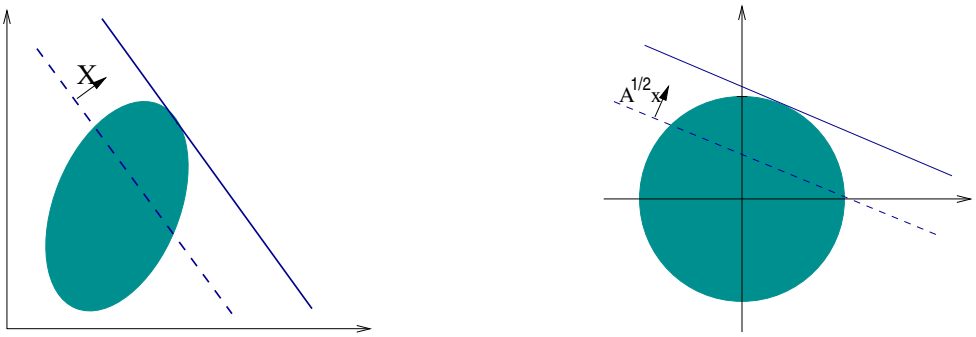

Figure 2.6: Reduction of linear optimization over an ellipsoid to linear optimization over a unit ball.

values of the coefficients. Moreover, the constant r characterizes the level of *risk aversion* of the user. Being a convex combination of the mean and the risk part, the model takes its name.

A small value of $r > 0$ corresponds to an ellipsoid with a small volume. In this case the costs may only vary in a small range around the mean value. That is, the user is optimistic and willing to accept risk. However, if the parameter r is big, the corresponding ellipsoid is large, containing and covering more scenarios. By considering these additional and usually more unrealistic scenarios the user wants to be secured in more cases – although they may appear very unlikely. Choosing a huge value of r thus characterizes a risk-averse user and leads to a conservative and more pessimistic solution. Hence, parametrizing the size of the ellipsoidal uncertainty set, the user is able to control the trade-off between robustness and performance.

Note that the reformulation of the objective function in (MR) can also be reproduced in a more elementary way. In particular, the objective function

$$\max_{c \in \mathcal{U}} c^\top x$$

is a linear maximization problem over an ellipsoid for every fixed x (see Figure 2.6). Since A is positive definite, there exists a regular matrix $A^{\frac{1}{2}}$ such that $A = A^{\frac{1}{2}}(A^{\frac{1}{2}})^\top$. By means of the substitution

$$z = (A^{-1})^{\frac{1}{2}}(c - c_0) \Longleftrightarrow c = A^{\frac{1}{2}} z + c_0$$

for all $z \in \mathbb{R}^n$, this task can be reduced to a linear maximization problem over a ball [17] (see Figure 2.6), which can be solved straightforward since the optimal

solution is given by the normalized vector of the objective function itself:

$$\begin{array}{rl} \max & c^\top x \\ s.t. & (c-c_0)^\top A^{-1}(c-c_0) \leq r^2 \end{array} \quad = \quad \begin{array}{rl} c_0^\top x + \max & (A^{\frac{1}{2}}x)^\top z \\ s.t. & \|z\| \leq r \end{array}$$

$$= c_0^\top x + (A^{\frac{1}{2}}x)^\top \frac{A^{\frac{1}{2}}x}{\|A^{\frac{1}{2}}x\|} r$$

$$= c_0^\top x + r\sqrt{x^\top A x}.$$

With this we arrive at the same formulation of Problem (2.11).

2.2.1.2 Value-At-Risk Model

An interesting fact is that we obtain the same formulation (MR) of the robust problem having originally an apparently different objective. In the financial sector [21] as well as in industrial- and trading companies [57, 45] the so-called *value-at-risk* is applied as a standard measure for quantification of different risks [64].

Using the standard value-at-risk model [17]

$$\begin{array}{ll} \min & t \\ \text{s.t.} & Pr(c^\top x \leq t) \geq \lambda \\ & x \in X, \end{array} \tag{2.15}$$

we assume that the costs $c \in \mathbb{R}^n$ in a given application follow some *normal distribution* with mean $c_0 \in \mathbb{R}^n$ and a covariance matrix $A \in \mathcal{S}^n_{++}$. The aim is to minimize the budget t (or total costs), such that the *probability* of not exceeding it reaches at least a given confidence level $\lambda \in [0.5, 1]$.

Example 2.5 (The Businessman-Problem [60])**.** A businessman needs to find a fastest track to the airport in the light of uncertain traffic conditions, to catch a flight. He thus needs to allocate sufficient time t to ensure on-time arrival with the confidence of 98%. The time needed for every street can be modeled by the stochastic cost vector c and the probability of not exceeding the allocated time t is then given by $Pr(c^\top x \leq t)$. With the goal of minimizing the time budget the situation is exactly described by Problem (2.15). In this example X is the set of all possible routes and $\lambda = 0.98$.

In this case we deal with the so-called *reliable (robust) Shortest Path-problem* [25], which asks for the fastest track that ensures certain guarantee regarding the degree of risk. □

The value-at-risk optimization is also known as optimization under probabilistic constraint or *chance constraint*, as the inequality in Problem (2.15) is called [59].

Assuming normally distributed costs, i.e. $c \sim N(c_0, A)$, a fixed feasible solution x also implies a normally distributed random variable $c^\top x$ with mean $c_0^\top x$ and covariance $x^\top Ax$. That means that the random variable

$$Z = \frac{c^\top x - c_0^\top x}{x^\top Ax} \sim N(0,1)$$

is standard normally-distributed [65] and that $Pr(Z \leq z) = \Phi(z)$, for $\Phi(\cdot)$ being a cumulative distribution function of a standard normal random variable $N(0,1)$. Hence, the chance-constraint can be transformed as follows [60]:

$$\begin{aligned} Pr(c^\top x \leq t) \geq \lambda &\iff Pr(\frac{c^\top x - c_0^\top x}{\sqrt{x^\top Ax}} \leq \frac{t - c_0^\top x}{\sqrt{x^\top Ax}}) \geq \lambda \\ &\iff \Phi(\frac{t - c_0^\top x}{\sqrt{x^\top Ax}}) \geq \lambda \\ &\iff \frac{t - c_0^\top x}{\sqrt{x^\top Ax}} \geq \Phi^{-1}(\lambda) \\ &\iff t \geq c_0^\top x + \Phi^{-1}(\lambda)\sqrt{x^\top Ax}. \end{aligned}$$

Thus in Problem (2.15) we can minimize the function $c_0^\top x + \Phi^{-1}(\lambda)\sqrt{x^\top Ax}$ over x instead of minimizing t and the value-at-risk model reduces to the mean-risk model, but with a special risk-coefficient $r = \Phi^{-1}(\lambda)$, which establishes the connection between the volume of the uncertainty ellipsoid and the risk attitude of the user.

Note the conceptual analogy between the two reformulations: Starting with the robust min-max problem an arbitrary ellipsoid is transformed into a unit ball. Starting with the chance-constraint we transform an arbitrary normally distributed random variable to a standard normally distributed random variable.

In the following, if not stated explicitly, we assume $r = 1$, as we can include it into the covariance matrix anyway. Consequently, it is sufficient to deal with the following formulation of the robust combinatorial problem with ellipsoidal uncertainty in the remainder of this thesis:

$$\begin{array}{ll} \min & c_0^\top x + \sqrt{x^\top Ax} \\ \text{s.t.} & x \in X\ . \end{array} \tag{2.16}$$

2.2.2 Correlated and Uncorrelated Case

When considering ellipsoidal uncertainty in robust optimization, an important special case can be distinguished from the general case, namely the assumption of a *diagonal* covariance matrix $A = Diag(a)$, $a \in \mathbb{R}^n_+$. Technically, A is then no longer a covariance matrix, but a *variance* matrix, modeling the situation without correlations between unique variable weights. In geometric terms this means that

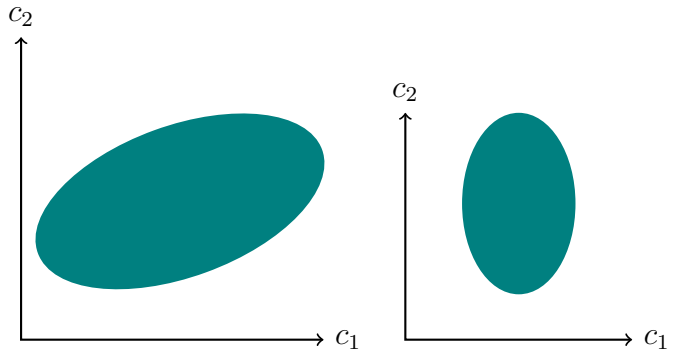

Figure 2.7: General and axis-parallel ellipsoids.

the ellipsoid is parallel to the axis (see Figure 2.7). Due to the binarity of x, Problem (2.16) then reduces to

$$\begin{array}{ll} \min & c_0^\top x + \sqrt{a^\top x} \\ \text{s.t.} & x \in X \ . \end{array} \tag{2.17}$$

Both cases (the general case and the *uncorrelated* case) are differently and widely applied.

Example 2.6 (The Correlated Businessman-Problem)**.** To the situation of Example 2.5, which, as we have seen, can be formulated in form (2.16), we add the information that the flight is taking place on Monday morning. We may suppose the traffic on certain roads to be correlated to the traffic on some other roads. Besides from a similar occupation of adjacent roads, a familiar situation is a similar occupation of "parallel" streets during the morning or evening rush hours, i.e. the streets leading to or out of the metropolitan centers. This tendency of the traffic to deviate from its expected value Monday mornings and Friday afternoons in a similar way on different roads (to covariate) can be expressed in covariances. □

The uncorrelated case, however, seems to be a significant restriction. Nevertheless, the huge variety of applications underpins the importance of this special case for users. The probably most widespread application is *portfolio selection* given a limited budget and expecting uncertain outcomes.

Example 2.7 (The Risk-Averse Capital Budgeting Problem [9])**.** An objective of an investor choosing a set of investments is to maximize their expected return. But he also has to take into account the *risk* associated with investments. The investment decisions are represented by binary variables $x_i, i = 1, \ldots, n$, with associated expected return values c_{0i} and variance values a_i. The variance values

model the risk that the realized return deviates from the expected return. The costs of investments are given by $w_i > 0$ and the available budget is $b > 0$. This leads to the following non-linear integer program

$$\begin{array}{rl} \max & c_0^\top x - \sqrt{a^\top x} \\ \text{s.t.} & w^\top x \leq b \\ & x \in \{0,1\}^n, \end{array} \tag{2.18}$$

which can easily be transformed to a minimization problem of the form (2.17). Here the underlying deterministic problem is the classical Knapsack-problem (see Section 1.1). □

The portfolio theory goes back to Harry Markowitz [55] and covers a big range of applications primarily on capital markets and in investment management, but can also be found in very different areas, as we can see in the following example.

Example 2.8 (The Skydiver-Problem)**.** The objective of a skydiver planning the next season is to maximize experience and fun while attending different skydiving events. For the next year a set $I = \{1, \ldots, n\}$ of events is announced. The exact experience and fun-outcomes for every event are not known, but can be characterized by an expected fun-value c_0 (structuring the known aspects such as aircraft and invited load organizers) and a variance vector a (summing up the uncertain factors such as weather and the skills level of other attending skydivers).

Moreover, the choice of the skydiver has to comply with the following restrictions:

1. The number of vacation days of the skydiver is limited to D and the event i lasts d_i working days.

2. The overall cost b_i for every event i (including the travel expenses, accommodation, registration fee and costs for the jump tickets) should not exceed the limited budget B for all events taken together.

3. Due to overlapping, the skydiver should attend to at most one event from certain subsets $M_j \subseteq I, j \in \{1, \ldots, k\}$, of events.

This leads to the following Multi-Knapsack-problem with uncorrelated ellipsoidal uncertainty

$$\begin{array}{rl} \max & c_0^\top x - \sqrt{a^\top x} \\ \text{s.t.} & d^\top x \leq D \\ & b^\top x \leq B \\ & \sum_{i \in M_j} x_i \leq 1 \text{ for all } j \in \{1, \ldots, k\} \\ & x \in \{0,1\}^n. \end{array}$$

□

Being still relevant in practice, the uncorrelated case differs fundamentally in its structural properties and theoretical complexity from the general case (with A being an arbitrary positive definite matrix). The objective function of (2.16) is a sum of a linear function in x and a norm induced by A [46], which means it is convex on $\mathbb{R}^n$. On contrary, the reformulation to (2.17) due to the binarity of x makes the objective function of (2.17) concave.

While the general case is known to be $\mathcal{NP}$-hard even with no additional constraints (with $X = \{0,1\}^n$), there are relevant non-trivial sets $X \subseteq \{0,1\}^n$ leading to a polynomially solvable problem in the uncorrelated ellipsoidal uncertainty case [60]. Moreover, there are some non-classified cases with unknown complexity-status, like the highly relevant case of the *robust Shortest Path-problem*, which we will also address later.

In this work we will also distinguish between the general and the uncorrelated cases. In the following we will give an overview over known facts from the literature on both of these problems.

2.3 General Case of Ellipsoidal Uncertainty

In this section we consider the general Problem (2.16) with an arbitrary positive definite matrix $A \in \mathcal{S}^n_{++}$ and various combinatorial sets $X \subseteq \{0,1\}^n$. We will classify the problem from the complexity-theoretical perspective and show some possible approaches discussed in the literature.

2.3.1 Complexity

As figured out in the previous section, there are numerous applications of Problem (2.16) with correlations. Being highly relevant in practice this problem turns out to be hard to solve theoretically according to the following theorem:

Theorem 2.9. *Problem* (2.16) *is $\mathcal{NP}$-hard, even for $X = \{0,1\}^n$.*

Proof. The unrestricted binary Problem (2.11) with $\mathcal{U}$ consisting of two scenarios is $\mathcal{NP}$-hard due to the reduction from the $\mathcal{NP}$-hard Subset Sum-problem [9]. Bertsimas and Sim [15] have introduced the following polynomial transformation of the objective $\max\{c_1^\top x, c_2^\top x\}$ of the two scenarios-problem to a special objective

of (2.16):

$$\begin{aligned}\max\{c_1^\top x, c_2^\top x\} &= \max\left\{\frac{c_1^\top x + c_2^\top x}{2} + \frac{c_1^\top x - c_2^\top x}{2}, \frac{c_1^\top x + c_2^\top x}{2} - \frac{c_1^\top x - c_2^\top x}{2}\right\} \\ &= \frac{c_1^\top x + c_2^\top x}{2} + \max\left\{\frac{c_1^\top x - c_2^\top x}{2}, -\frac{c_1^\top x - c_2^\top x}{2}\right\} \\ &= \frac{c_1^\top x + c_2^\top x}{2} + \left|\frac{c_1^\top x - c_2^\top x}{2}\right| \\ &= \frac{1}{2}(c_1 + c_2)^\top x + \frac{1}{2}\sqrt{x^\top (c_1 - c_2)(c_1 - c_2)^\top x}.\end{aligned}$$

Thus, solving Problem (2.16) with the mean $\frac{1}{2}(c_1+c_2)^\top$ and the covariance matrix $\frac{1}{4}(c_1 - c_2)(c_1 - c_2)^\top$ yields a solution of the $\mathcal{NP}$-hard two scenarios-problem.

However, the covariance matrix such defined is not positive definite but only positive semidefinite, if $n \geq 2$. As we required positive definiteness in our definition of the ellipsoidal uncertainty set, we extend this proof now for explicitly positive definite matrices.

We scale c_1 and c_2 to c_1' and c_2' such that $c_1', c_2' \in \mathbb{Z}^n$. Let $\epsilon \in (0, 1)$ and consider

$$A' = \frac{1}{4}(c_1' - c_2')(c_1' - c_2')^\top + \frac{\epsilon^2}{n} I \in \mathcal{S}_{++}^n,$$

with I being the identity matrix.

We show that the minimizer of

$$f'(x) = \frac{1}{2}(c_1' - c_2')^\top x + \frac{1}{2}\sqrt{x^\top \left((c_1' - c_2')(c_1' - c_2')^\top + \frac{4\epsilon^2}{n} I\right) x},$$

which is a well-defined instance of (2.16), is a minimizer of

$$f(x) = \frac{1}{2}(c_1' - c_2')^\top x + \frac{1}{2}\sqrt{x^\top (c_1' - c_2')(c_1' - c_2')^\top x},$$

which in turn is a minimizer of $\max\{c_1^\top x, c_2^\top x\}$, as we have seen above (scaling obviously does not affect the optimal solution in this case).

For all $x \in \{0, 1\}^n$ the inequalities

$$f(x) \leq f'(x) \leq f(x) + \frac{\epsilon}{\sqrt{n}}\sqrt{x^\top x} \leq f(x) + \epsilon < f(x) + 1 \tag{2.19}$$

hold, due to binarity of x and concavity of the square root. Then for a minimizer x^* of f' and an arbitrary $x \in \{0, 1\}^n$ follows

$$f(x) \overset{(2.19)}{>} f'(x) - 1 \geq f'(x^*) - 1 \overset{(2.19)}{\geq} f(x^*) - 1. \tag{2.20}$$

Due to scaling and the special form of the square root argument, f is integer-valued on $\{0, 1\}^n$, such that the optimality of x^* for f follows with (2.20). □

Since the transformation only acts on the objective function, arbitrary binary sets X which lead to an $\mathcal{NP}$-hard problem while combined with the objective $\max\{c_1^\top x, c_2^\top x\}$, also lead to an $\mathcal{NP}$-hard problem while combined with the objective $c_0^\top x + \sqrt{x^\top A x}$. This applies in particular to the robust Shortest Path-problem, the robust Minimum Spanning Tree-problem, the robust Assignment-problem and many other classical combinatorial problems being polynomial in the standard version [50]:

Corollary 2.10. *The Shortest Path-problem under ellipsoidal uncertainty is $\mathcal{NP}$-hard.*

Corollary 2.11. *The Minimum Spanning Tree-problem under ellipsoidal uncertainty is $\mathcal{NP}$-hard.*

Corollary 2.12. *The Assignment-problem under ellipsoidal uncertainty is $\mathcal{NP}$-hard.*

This follows from the fact that the corresponding problems are $\mathcal{NP}$-hard for two scenarios by [50].

Note that Theorem 2.9 does not provide any information about strong $\mathcal{NP}$-hardness of Problem (2.16), since the proof is based on the reduction of a weakly $\mathcal{NP}$-hard problem (see Chapter 1) [40].

We will show later that Problem (2.16) is in fact strongly $\mathcal{NP}$-hard, even with $X = \{0,1\}^n$ (see Section 4.1). Therefore, assuming $\mathcal{P} \neq \mathcal{NP}$, we cannot even hope to have a pseudo-polynomial algorithm for this problem.

However, this still does not exclude the existence of approximation algorithms for Problem (2.16). A non-approximability result was shown though for the robust Shortest Path-problem by Chassein et al. [24]. They use the fact that the problem Independent Set on Degree Three Graphs is APX-hard [12] and give a PTAS reduction of it to the Quadratic Shortest Path-problem (QSP). This proves the APX-hardness of the problem QSP. This fact is then used to show the non-approximability by a PTAS of the robust Shortest Path-problem on series-parallel graphs. More generally, they show that an APX-hard problem with objective function $f > 0$ implies that a problem over the same feasible set and with objective function $\sqrt{f}$ is APX-hard as well.

In particular, this applies for the *Minimum Quadratic Assignment-problem,* which is not even in APX [5]:

Corollary 2.13. *The Minimum Assignment-problem with correlated ellipsoidal uncertainty is not approximable.*

In the next section we show some approaches potentially appropriate or effectively used in the literature to handle the general Problem (2.16).

2.3.2 Second Order Cone Programming Formulation

We consider a different formulation of Problem (2.16) to suggest a spectrum of existing algorithms to solve it. Due to the non-linearity of the set $\mathcal{U}$ there is no straight-forward mixed-binary linear formulation of Problem (2.16) in the general ellipsoidal uncertainty case [8]. However, the problem can be formulated as a *mixed-binary second-order cone program* (SOCP), which is, of course, more general than a mixed-binary linear program [17].

Modeling the objective function by a *second-order cone constraint,* we get

$$\begin{aligned}
&\min \quad c_0^\top x + \sqrt{x^\top A x} \\
&\text{s.t.} \quad x \in X \subseteq \{0,1\}^n \\
\\
\iff &\min \quad c_0^\top x + \sqrt{x^\top A^{\frac{1}{2}} (A^{\frac{1}{2}})^\top x} \\
&\text{s.t.} \quad x \in X \subseteq \{0,1\}^n \\
\\
&\min \quad c_0^\top x + \sqrt{y^\top y} \\
\iff &\text{s.t.} \quad y = (A^{\frac{1}{2}})^\top x \\
&\qquad x \in X \subseteq \{0,1\}^n, y \in \mathbb{R}^n \\
\\
&\min \quad c_0^\top x + z \\
\iff &\text{s.t.} \quad \|y\|_2 \leq z \\
&\qquad y = (A^{\frac{1}{2}})^\top x \\
&\qquad x \in X \subseteq \{0,1\}^n, y \in \mathbb{R}^n, z \in \mathbb{R}_+ \\
\\
&\min \quad c_0^\top x + z \\
\iff &\text{s.t.} \quad (y,z)^\top \in \mathcal{K}_{n+1} \\
&\qquad y = (A^{\frac{1}{2}})^\top x \\
&\qquad x \in X \subseteq \{0,1\}^n, y \in \mathbb{R}^n, z \in \mathbb{R}_+,
\end{aligned}$$

with $\mathcal{K}_{n+1} = \{x \in \mathbb{R}^{n+1} \mid \|(x_1, ..., x_n)^\top\|_2 \leq x_{n+1}\}$ being a *second-order cone.*

That implies already a common approach to the robust Problem (2.16), namely the use of algorithms for general mixed-integer SOCPs [17, 54].

Continuous SOCPs are computationally tractable and can be solved in polynomial time (up to arbitrary precision), for example by interior point methods [3]. These provide a lower bound on Problem (2.16), which can be used in a branch and bound-approach. However, the general integer linear programming is a special case of the general mixed-binary SOCPs. The mixed-binary domain causes $\mathcal{NP}$-hardness for general mixed-integer SOCPs (which also follows from Theorem 2.9). Accordingly, the algorithms for general mixed-integer SOCPs are only partially satisfying, especially since relevant polynomial special cases of Problem (2.16)

exist. Nevertheless, this property already provides us a tool to solve Problem (2.16), which is in particular useful for an experimental evaluation of our algorithms later on.

2.4 Uncorrelated case

We have seen that the robust combinatorial Problem (2.16) with ellipsoidal uncertainty is hard to deal with in general. It is thus natural to study first the special case of the uncorrelated ellipsoidal uncertainty, i.e. Problem (2.17), in more detail. Moreover, there are numerous applications, some of which we have seen in the previous sections, as well as many open theoretical questions referring to it. To begin with, in this section we present the results on this problem known in the literature.

2.4.1 Unconstrained Case: Tractability

As in the correlated case in Chapter 2.2 we start our studies about the uncorrelated ellipsoidal uncertainty with the probably easiest case in combinatorial optimization – the unrestricted case, i.e. with $X = \{0,1\}^n$:

$$\begin{array}{ll} \min & c_0^\top x + \sqrt{a^\top x} \\ \text{s.t.} & x \in \{0,1\}^n. \end{array} \tag{2.21}$$

First we agree within this case on the assumption $c_0 < 0$, as we can directly fix all the variables i with $c_{0i} \geq 0$ and $a_i > 0$ to 0: It can easily be verified that assigning a value 1 to a variable i with $c_{0i} \geq 0$ and $a_i > 0$ will increase the value of the objective function, which cannot lead to an optimal solution.

Theorem 2.14. *Problem* (2.21) *is solvable in polynomial time.*

Proof. The objective function f of Problem (2.21) is *submodular*, since for $x, y \in \{0,1\}^n$ with $x \leq y$ and e_i being the i-th unit vector, the following calculation holds:

$$\begin{aligned} f(x+e_i) - f(x) &= c_0^\top (x+e_i) + \sqrt{a^\top (x+e_i)} - c_0^\top x - \sqrt{a^\top x} \\ &= c_{0i} + \sqrt{a^\top x + a_i} - \sqrt{a^\top x} \\ &\geq c_{0i} + \sqrt{a^\top y + a_i} - \sqrt{a^\top y} \\ &= f(y+e_i) - f(y). \end{aligned}$$

The inequality holds due to the concavity of the square root function. Since unconstrained minimization of submodular functions over a binary domain is known to be polynomial [39], the assertion of the theorem follows. □

With that the unrestricted uncorrelated case is apparently dealt with, but we have to have in mind that the running time $O(n^5)$ of the best known polynomial algorithm for general submodular functions makes those hardly applicable in practice [39].

Thus, to solve Problem (2.21) in an adequate time, we will present a new approach, exploiting the specificity of the objective function, in Section 3.1. For now we keep in mind the theoretical efficiency of the unrestricted case and turn our attention to the general set $X \subseteq \{0,1\}^n$.

2.4.2 Constrained Case: Connection to Bicriteria Optimization

For many classical combinatorial sets $X \subset \{0,1\}^n$ it is unknown if Problem (2.17) can be solved in polynomial time. Since we may consider ellipsoids consisting of exactly one scenario, this can obviously only be expected if the underlying linear combinatorial Problem

$$\begin{array}{ll} \min & c^\top x \\ \text{s.t.} & x \in X \end{array} \tag{2.22}$$

is polynomially solvable. On the other hand, to the best of our knowledge, no tractable linear combinatorial problem has been found still with the corresponding robust version (2.17) being $\mathcal{NP}$-hard. In other words, the open question is, if every easy combinatorial linear problem remains easy, when considered under uncorrelated ellipsoidal uncertainty.

2.4.2.1 Bertsimas and Sim-Approach

A general result of Bertsimas and Sim [15] states that for general sets X, solving the robust Problem (2.17) can be reduced to solving a number of underlying linear optimization problems over X.
Consider the set $W = \left\{ a^\top x \mid x \in \{0,1\}^n \right\}$ and the function

$$g(w) = \begin{cases} \frac{1}{2\sqrt{w}} & \text{if } w \in W \setminus \{0\} \\ \\ \frac{1}{\sqrt{a_{min}}} & \text{if } w = 0, \end{cases}$$

where $a_{min} = \min_{\{i|a_i>0\}} a_i$, as well as the optimization problem

$$\begin{array}{rll} Z(w) = \min & (c_0 + g(w)a)^\top x + \sqrt{w} - wg(w) \\ \text{s.t.} & x \in X, \end{array} \tag{2.23}$$

for $w \in W$. The idea is to choose from all solutions $x \in X$, implied by every $w \in W$, the one belonging to the $w \in W$ optimizing

$$\begin{array}{ll} \min & Z(w) \\ \text{s.t.} & w \in W. \end{array} \tag{2.24}$$

So, effectively we go through every possible value of $a^\top x$ (which are finitely but not necessarily polynomially many) and search for such a feasible solution $x_w \in X$, that

1. yields the value w, i.e. $a^\top x_w = w$ (we will see in the proof of the next theorem, that if an x_w^* optimal for a $Z(w)$ does not fulfill $a^\top x^* = w$, then this w is not optimal for (2.24)) and
2. minimizes the original objective function (note, that if 1 holds, the value of the objective function of (2.23) is equal to the value of the objective function of (2.17)).

This approach admits the following theorem.

Theorem 2.15. *[15] Let w be an optimal solution of* (2.24) *and $x_w^* \in X$ an optimal solution of Problem* (2.23) *corresponding to w. Then x_w^* is optimal for* (2.17).

Proof. It is enough to show that the optimal value of (2.17) coincides with the optimal value of (2.24).

To this end let first x^* be an optimal solution of (2.17) and $w^* = a^\top x^* \in W$. Then

$$\begin{aligned} \min_{x \in X} c_0^\top x + \sqrt{a^\top x} &= c_0^\top x^* + \sqrt{a^\top x^*} \\ &= c_0^\top x^* + \sqrt{w^*} \\ &= c_0^\top x^* + \sqrt{w^*} + g(w^*)a^\top x^* - g(w^*)w^* \\ &\geq \min c_0^\top x + \sqrt{w^*} + g(w^*)a^\top x - g(w^*)w^* \\ &= Z(w^*) \\ &\geq \min_{w \in W} Z(w). \end{aligned}$$

Now, for arbitrary $w \in W$, let x_w be an optimal solution of the corresponding Problem (2.23). In the case $w \neq 0$, the inequality

$$\frac{a^\top x_w}{2\sqrt{w}} + \frac{w}{2} \geq \sqrt{a^\top x_w},$$

which is always true since

$$\begin{aligned} &\frac{a^\top x_w}{2\sqrt{w}} + \frac{w}{2} \geq \sqrt{a^\top x_w} \\ \Longleftrightarrow &a^\top x_w + w \geq 2\sqrt{w}\sqrt{a^\top x_w} \\ \Longleftrightarrow &(\sqrt{a^\top x_w} + \sqrt{w})^2 \geq 0, \end{aligned}$$

yields

$$\begin{aligned} Z(w) &= \min_{x\in X} c_0^\top x + \frac{1}{2\sqrt{w}} a^\top x + \frac{1}{2}\sqrt{w} \\ &= c_0^\top x_w + \frac{1}{2\sqrt{w}} a^\top x_w + \frac{1}{2}\sqrt{w} \\ &\geq c_0^\top x_w + \sqrt{a^\top x_w} \\ &\geq \min_{x\in X} c_0^\top x + \sqrt{a^\top x}. \end{aligned}$$

In the case $w = 0$, we obtain

$$\frac{1}{\sqrt{a_{min}}} a^\top x_w \geq \sum_{i=1}^{n} \sqrt{a_i}(x_w)_i = \sum_{i=1}^{n} \sqrt{a_i (x_w)_i} \geq \sqrt{a^\top x_w}$$

due to the concavity of the square root function, and, hence, deduce

$$Z(w) = c_0^\top x_w + \frac{1}{\sqrt{a_{min}}} a^\top x_w \geq c_0^\top x_w + \sqrt{a^\top x_w} \geq \min_{x\in X} c_0^\top x + \sqrt{a^\top x}.$$

□

Based on the enumeration of all possible values under the square-root (all possible elements of W), this approach immediately implies the polynomial solvability of Problem (2.17) for some specific assignments of c_0 and a, for example:

Corollary 2.16. *Let $a_i = s$ for all $i = 1, \ldots, n$, for some $s > 0$. Then Problem* (2.17) *can be solved by solving $n + 1$ linear problems over X.*

This result does not involve the specific choice of the set $X \subseteq \{0, 1\}^n$. But a further corollary of Theorem 2.15 becomes important for particular structures of the feasible set.

Corollary 2.17. *There exists a $\lambda \in \left[g(e^\top a), g(0)\right]$ such that every optimal solution of the problem*

$$\begin{array}{ll} \min & c_0^\top x + \lambda a^\top x \\ s.t. & x \in X \end{array} \tag{2.25}$$

is an optimal solution of Problem (2.17).

We point out the obvious connection between robust optimization with uncorrelated ellipsoidal uncertainty and *parametric optimization*, which we describe more closely in the next section.

At this stage we can recognize that this implication of Theorem 2.15 is of particular importance for matroids.

Corollary 2.18. *If* (2.17) *is an optimization problem over a matroid, then it reduces to solving an at most quadratic number of underlying linear problems.*

Proof. For matroids, the optimal solutions with respect to linear costs depend only on the order of the elements with respect to the costs [33]. When passing through all possible values of λ monotonously, the order of every pair of elements with respect to the cost function $c_0^\top x + \lambda a^\top x$ can change only once, yielding that at most $\binom{n}{2}+1$ different orderings of elements are possible. The interval borders of λ leading to the same ordering can be computed easily by solving the equations

$$(c_0 + \lambda a)_i = (c_0 + \lambda a)_j$$

for every pair (i, j) of variables. □

Corollary 2.18 yields a complexity bound for Problem (2.17) with some important sets $X \subseteq \{0,1\}^n$.

Corollary 2.19. *The robust Minimum Spanning Tree-problem* (2.17) *is solvable in time* $O(n^2 \log^2(n))$.

Corollary 2.20. *Problem* (2.21) *is solvable in time* $O(n^2)$.

The first claim rests on the upper bound of $\mathrm{O}(n \log n)$ on the time complexity of *Kruskal's* algorithm [51]. The second claim uses that a linear instance of Problem (2.21) can be solved in linear time.

2.4.2.2 Nikolova-Approach

We observe an obvious trade-off between the parts $c_0^\top x$ and $\sqrt{a^\top x}$ in the objective function of Problem (2.17), which in particular becomes evident in the theory of Bertsimas and Sim [15].

This feature is getting even clearer and more essential in the approach of Nikolova [60]. Here, the author mainly focuses on finding general approximation algorithms for Problem (2.17), but by unfolding an important connection to *bicriteria* and *parametric optimization* also provides as a by-product an exact algorithm to solve this problem. This connection forms the base of further research in this area [24, 23, 67].

In *bicriteria optimization* we aim to optimize two independent and sometimes conflicting objectives – lets say $c_0^\top x$ and $a^\top x$ – over the same feasible set, searching for all solutions with the property that the costs of one of the objectives cannot be decreased without increasing the costs of the other objective, see Section 1.3. Lets take a look on the projection

$$Proj_{c_0,a}(X) = \left\{ \begin{pmatrix} c_0^\top x \\ a^\top x \end{pmatrix} \middle| x \in X \right\} \subseteq \mathbb{R}^2$$

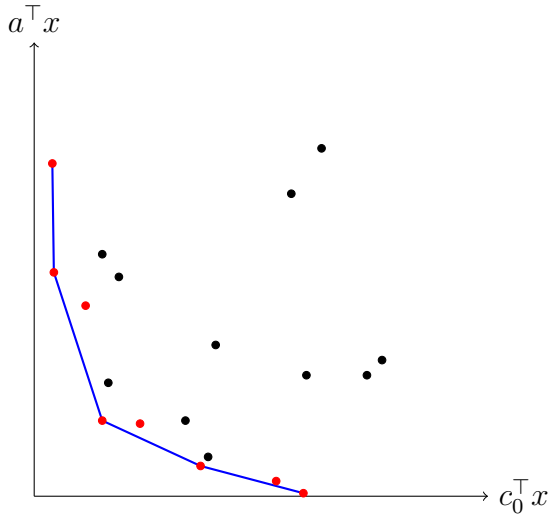

Figure 2.8: Projection of the feasible set X onto the span of the objectives $c_0^\top x$ and $a^\top x$. Efficient solutions (red), extreme efficient solutions (connected by blue lines).

of the feasible set X onto the span of both objectives $c_0^\top x$ and $a^\top x$ (see Figure 2.8). Among these so-called *efficient solutions*, constituting the *Pareto-frontier*

$$pf_{c_0,a}(X) = \left\{ \begin{pmatrix} c_0^\top x \\ a^\top x \end{pmatrix} \middle| \begin{array}{r} x \in X, \nexists y \in X : c_0^\top y < c_0^\top x, a^\top y \leq a^\top x \\ \text{or } c_0^\top y \leq c_0^\top x, a^\top y < a^\top x \end{array} \right\},$$

we find some points of particular importance – the *extreme efficient solutions*– solutions corresponding to the extreme points of $\text{conv}(pf_{c_0,a}(X))$ (see Figure (2.8)).

In *parametric optimization* we aim to solve Problem (2.25) for every value $\lambda \in [0, \infty)$. Here, of particular importance are the so-called *break points* – the values of λ in which the optimal solution of Problem (2.25) changes.

It is easy to see that every extreme point of $\text{conv}(pf_{c_0,a}(X))$ corresponds to an optimal solution of the parametric Problem (2.25) for some values of λ [42, 22]. Conversely, every value of $\lambda \in [0, \infty)$ obviously leads to an optimal solution located in an extreme point of $\text{conv}(pf_{c_0,a}(X))$. Thus, finding the break points of the parametric Problem (2.25) corresponds exactly to finding the extreme efficient solutions of the bicriteria problem.

There is an important connection between *robust optimization under uncorrelated ellipsoidal uncertainty* and both *bicriteria* and *parametric optimization*:

Theorem 2.21. *Every optimal solution of the robust Problem* (2.17) *is an extreme efficient solution of the corresponding bicriteria problem, i.e. an extreme point of the set conv*$(pf_{c_0,a}(X))$.

Proof. [18] Clearly the smallest combination of the mean and the variance is located at one of the extreme points of the boundary of the Pareto-frontier due to the monotony of the objective function of Problem (2.17) in its mean and variance parts and due to concavity of the square-root.

More precisely, let $x^* \in X$ be an optimal solution of the robust Problem (2.17). Consider

$$D_{c_0,a}(X) = \left\{ z \in \mathbb{R}^2 \mid \exists y \in conv(Proj_{c_0,a}(X)) : z \geq y \right\},$$

the so-called *dominant* of $\text{conv}(Proj_{c_0,a}(X))$, i.e. the set of points that are coordinate-wise bigger or equal to the points in $\text{conv}(Proj_{c_0,a}(X))$. Note that the extreme points of $D_{c_0,a}(X)$ coincide with the extreme points of $\text{conv}(pf_{c_0,a}(X))$.

We have

$$\begin{pmatrix} c_0^\top x \\ a^\top x \end{pmatrix} \in Proj_{c_0,a}(X) \subseteq D_{c_0,a}(X),$$

and for some $x_1, x_2 \in X$ with $\begin{pmatrix} c_0^\top x_1 \\ a^\top x_1 \end{pmatrix}$ and $\begin{pmatrix} c_0^\top x_2 \\ a^\top x_2 \end{pmatrix}$ being extreme points of $D_{c_0,a}(X)$ we can write

$$\begin{pmatrix} c_0^\top x^* \\ a^\top x^* \end{pmatrix} = \lambda \begin{pmatrix} c_0^\top x_1 \\ a^\top x_1 \end{pmatrix} + (1-\lambda) \begin{pmatrix} c_0^\top x_2 \\ a^\top x_2 \end{pmatrix} + \mu_1 e_1 + \mu_2 e_2,$$

for some $\lambda \in [0,1]$ and $\mu_1, \mu_2 \geq 0$. Hence,

$$\begin{aligned} f(x^*) &= c_0^\top x^* + \sqrt{a^\top x^*} \\ &= \lambda c_0^\top x_1 + (1-\lambda) c_0^\top x_2 + \mu_1 + \sqrt{\lambda a^\top x_1 + (1-\lambda) a^\top x_2 + \mu_2} \\ &\geq \lambda c_0^\top x_1 + (1-\lambda) c_0^\top x_2 + \lambda \sqrt{a^\top x_1} + (1-\lambda) \sqrt{a^\top x_2} \\ &= \lambda f(x_1) + (1-\lambda) f(x_2). \end{aligned}$$

This implies $f(x_1) \leq f(x^*)$ or $f(x_2) \leq f(x^*)$. So, x_1 or x_2 are optimal for Problem (2.17) as well. □

With these considerations the same result to Corollary 2.17 can be observed. Let x^* be an optimal solution of Problem (2.17). Due to Theorem 2.21, $\left(c_0^\top x^*, a^\top x^*\right)^\top$ is then an extreme point of $D_{c_0,a}(X)$. As $\left(c_0^\top x^*, a^\top x^*\right)^\top$ is an extreme point of a convex set, there exists a $\lambda^* > 0$ such that $\left(c_0^\top x^*, a^\top x^*\right)^\top$ is a unique optimal solution of the problem

$$\begin{array}{ll} \min & z_1 + \lambda^* z_2 \\ \text{s.t.} & z \in D_{c_0,a}(X) \end{array} \tag{2.26}$$

and thus of the problem

$$\begin{array}{ll} \min & z_1 + \lambda^* z_2 \\ \text{s.t.} & z \in Proj_{c_0,a}(X)\ , \end{array} \tag{2.27}$$

since $Proj_{c_0,a}(X) \subseteq D_{c_0,a}(X)$ and $\left(c_0^\top x^*, a^\top x^*\right)^\top \in Proj_{c_0,a}(X)$. Going back to the inverse image of $Proj_{c_0,a}(X)$ we conclude, due to the uniqueness of the solution, that every optimal solution of the parametric Problem (2.25) has the same values $c_0^\top x^*$ and $a^\top x^*$, i.e. the same value $c_0^\top x^* + \sqrt{a^\top x^*}$ in Problem (2.17) and thus, every optimal solution of (2.25) is also optimal for (2.17), since x^* was optimal for (2.17). This implies that a $\lambda \in (0, \infty)$ exists such that every optimal solution of the parametric problem (2.25) is an optimal solution of Problem (2.17) [18].

With the equivalence between finding the extreme efficient solutions and finding the break points, Corollary 2.17 and Theorem 2.21 state about the same – Problem (2.17) can be solved by computing and comparing of all the extreme points of the Pareto-frontier, as well as by computing and comparing of one optimal solution implied by every pair of consecutive break points.

Especially for *matroids,* due to the definition of a break point, the implication of polynomial solvability of Problem (2.17) mentioned in Corollary 2.18 is straightforward and based on the same argument. Defined as the values of λ where the optimal solution changes, the break points in this case are exactly the values where the ordering of the costs changes. Thus there are at most quadratically many break points.

Computing all the break points in the case of matroids is very easy, as we have seen in the previous section, due to the specificity of the problem. But in general, it is not always clear for which values of λ the optimal solution of (2.25) changes. In particular, the number of these values might not be polynomial, as the following Lemma shows.

Lemma 2.22. *The number of break points in the parametric Shortest Path-problem* is $n^{O(\log n)}$ [22].

As the number of break points in the parametric problem corresponding to Problem (2.17) defines the complexity of the suggested exact algorithm of Nikolova [60], it is called *parametric complexity* of Problem (2.17), which we will occasionally refer to.

The connection to the extreme efficient solutions yields an opportunity to enumerate the break points efficiently (with respect to the number of the break points and provided an efficient oracle for the underlying linear problem) and thus to solve Problem (2.17) exactly [60].

The extreme efficient solutions can be enumerated as follows. Every extreme efficient solution minimizes a linear function $c_0^\top x + \lambda a^\top x$ over the set X for some value λ. It is easy to find the two extreme efficient solutions minimizing $c_0^\top x$ and $a^\top x$ and the corresponding projections B and A, respectively (see Figure (2.9)). The slope of the connection line between A and B induces the new optimization direction, i.e. a new linear objective function $c_0^\top x + \lambda a^\top x$ for some λ. Optimizing it over the feasible set X yields a new extreme efficient solution and the corresponding

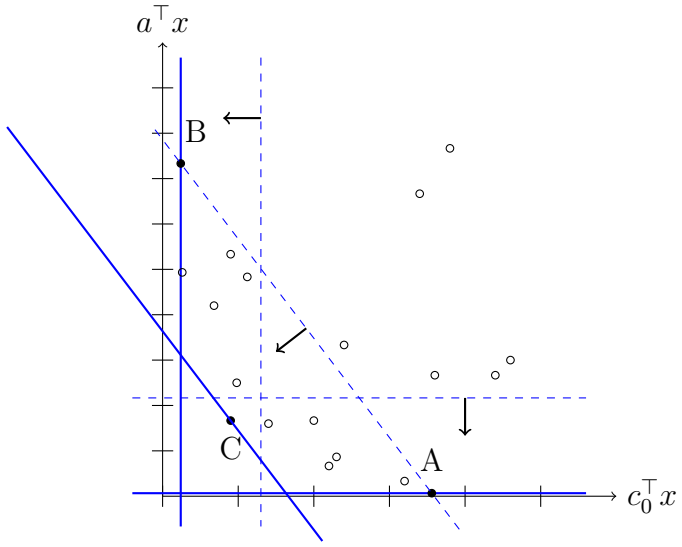

Figure 2.9: Enumerating extreme efficient solutions.

projection C (see Figure (2.9)). The procedure is continued considering every two neighboring extreme points found so far. If an optimal solution for some λ corresponds to an extreme point which is already found, then there are no more extreme points between the corresponding endpoints. The procedure takes $2k$ oracle calls to the deterministic problem if k is the number of extreme points [60].

2.4.2.3 FPTAS and Pseudo-Polynomial Algorithm

As there are still combinatorial problems under uncorrelated ellipsoidal uncertainty which are unclassified from the complexity-theoretical point of view, efficient algorithms to solve them might not exist. We sketch an approach how to handle these problems anyway, presented by Nikolova [60]. It is an FPTAS based on the connection between the considered problems and bicriteria optimization outlined in Section 2.4.2.2 and it requires the existence of a linear oracle for the corresponding deterministic problem.

We consider again the projection of the feasible set of Problem (2.17) onto the mean-variance span. We recall that the optimal solution of (2.17) is one of the extreme points of the non-dominated set. The corresponding projections $Proj_{c_0,a}(L_\lambda)$ of the level sets

$$L_\lambda := \left\{ x \in \mathbb{R}^n \mid c_0^\top x + \sqrt{a^\top x} = \lambda \right\}$$

to values $\lambda \in \mathbb{R}$ of the objective function are parabolas, described by

$$a^\top x = (c^\top x)^2 + \lambda^2 - 2\lambda c^\top x.$$

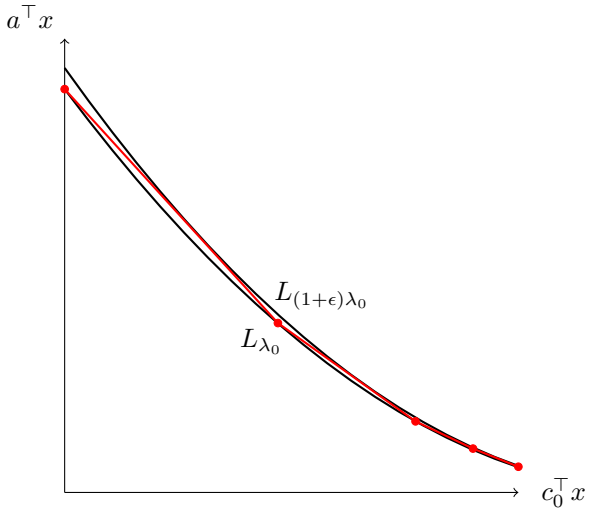

Figure 2.10: Inscribing a polygon between the two level sets L_{λ_0} and $L_{(1+\epsilon)\lambda_0}$ of $c_0^\top x + \sqrt{a^\top x}$ in the mean-variance-span.

Starting from some lower bound $f_{lb} =: \lambda_0$ on Problem (2.17), consider two level sets L_{λ_0} and $L_{(1+\epsilon)\lambda_0}$ for a given precision $\epsilon \in (0, 1)$. The *approximate nonlinear separation oracle* provided in [60] returns a feasible solution $x' \in X$ with $f(x') \leq (1+\epsilon)\lambda_0$ or determines that in the entire feasible set there is no solution with the function value better than λ_0, i.e. that $f(x) > \lambda_0$ for all $x \in X$.

To this aim it first inscribes a polygon between the two level sets (parabolas) such that the extreme points of it are on the level set L_{λ_0} and its sides are tangent to the level set $L_{(1+\epsilon)\lambda_0}$ (see Figure 2.10). The sides of the polygon induce linear functions, which are optimized over the feasible set using the linear oracle for the corresponding deterministic problem. If none of the sides (or rather the corresponding objectives) yields a feasible solution $x' \in X$ such that $f(x') \leq (1+\epsilon)\lambda_0$, the oracle returns the answer that $f(x) > \lambda_0$ for all $x \in X$ and the procedure goes on considering level sets to values $(1+\epsilon)\lambda_0$ and $(1+\epsilon)^2\lambda_0$ etc.

To ensure that the number of the oracle calls to the linear algorithm for the deterministic problem is small (in particular, it is polynomial in $\frac{1}{\epsilon}$ and independent on the problem size) for every level set, i.e. that the number of segments approximating each level set is small, Nikolova [60] proves that the segments are sufficiently long. For the exact characterization of the segment-endpoints and a more formal definition of the oracle and details see [60].

The existence of an FPTAS for all combinatorial problems under uncorrelated ellipsoidal uncertainty provided an efficient linear oracle is a very strong result.

Unfortunately, the requirements of Theorem 1.28 are not valid for Problem (2.17),

in particular, the output of Problem (2.17) is not integrally-valued. That means, at this point we cannot deduce weak $\mathcal{NP}$-hardness of Problem (2.17) from existence of an FPTAS.

Nevertheless, a stronger complexity bound follows from the approach presented in [15] and outlined in Section 2.4.2.1:

Corollary 2.23. *Problem* (2.17) *is at most weakly $\mathcal{NP}$-hard, if the underlying deterministic problem is tractable.*

Proof. Bertsimas and Sim [15] provide an exact approach with the running time $\mathrm{O}(|W|n^p)$, where $|W|$ is the cardinality of the set

$$W = \left\{ a^\top x \mid x \in \{0,1\}^n \right\}$$

and $\mathrm{O}(n^p)$ the polynomial upper bound on the running time of the linear oracle for the corresponding deterministic problem (2.22). Due to the obvious bound

$$|W| \leq a_{max} n + 1$$

a conservative upper bound on the overall running time of the approach is $\mathrm{O}((a_{max}n+1)n^p) = \mathrm{O}(a_{max}n^{p+1})$, which is polynomial if we assume $a_{max} \leq n^q$ for a fixed $q \in \mathbb{N}$. Due to Definition 1.25 the approach is then pseudo-polynomial and Problem (2.17) is at most weakly $\mathcal{NP}$-hard. □

2.4.3 Robust Multicriteria Optimization

The reduction of the robust Problem (2.17) to a bicriteria problem, as shown in Section 2.4.2, makes it possible to naturally generalize this approach for *multi-objective* combinatorial problems under uncorrelated ellipsoidal uncertainty. So we now take a look at the problem

$$\begin{array}{ll} \min & \left(f_1(x), \ldots, f_k(x), \max\limits_{c \in \mathcal{U}^1} c^\top x, \ldots, \max\limits_{c \in \mathcal{U}^l} c^\top x \right)^\top \\ \text{s.t.} & x \in X \,, \end{array}$$

with certain functions $f_1(x), \ldots, f_k(x)$ und robust functions

$$\max_{c \in \mathcal{U}^1} c^\top x, \ldots, \max_{c \in \mathcal{U}^l} c^\top x.$$

If $\mathcal{U}_1, \ldots, \mathcal{U}_l$ are axis-parallel ellipsoids, the problem obviously reduces to

$$\begin{array}{ll} \min & \left(f_1(x), \ldots, f_k(x), (c_0^1)^\top x + \sqrt{(a^1)^\top x}, \ldots, (c_0^l)^\top x + \sqrt{(a^l)^\top x} \right)^\top \\ \text{s.t.} & x \in X \,. \end{array} \tag{2.28}$$

Example 2.24. Within the enforcing energy revolution in Europe and especially in Germany new power supply grids have to be identified to transport electricity over long distances [1]. The power paths (vectors $x \in X \subseteq \{0,1\}^n$, with X modeling a set of paths) are evaluated regarding several objectives:

1. A minimal distance to residential areas should be maintained, so that the edges of the underlying graph are weighted by their compliance;
2. bundling on highways is highly desirable;
3. nature-, ground- and resources policies should be complied with.

These three objectives can be structured to functions f^1, f^2, and f^3. However, aside from the given highways networks as well as the nature- and ground constitution, the long-term distance to the residential areas is subject to uncertainty: The populated areas are growing or shrinking and as the minimal distance by the time of construction should still be maintained, the expansion risk should be taken into account.

The now-distance can be quantified to the mean vector $c_0 \in \mathbb{R}^n$. It is natural to structure the mentioned risk by means of a variance vector $a \in \mathbb{R}^n_+$, which can be defined, for example, in proportion to density of the settlements. The task then reduces to the following problem:

$$\begin{array}{ll} \min & \left(c_0^\top x + \sqrt{a^\top x}, f_2(x), f_3(x)\right)^\top \\ \text{s.t.} & x \in X\ . \end{array}$$

□

Theorem 2.21 implies that Problem (2.28) can be solved in the following three steps:

ALGORITHM FOR THE ROBUST MULTIOBJECTIVE PROBLEM (2.28)

(1) Find the Pareto set P_m of the multiobjective problem

$$\begin{array}{ll} \min & \left(f_1(x), \ldots, f_k(x), (c_0^1)^\top x, (a^1)^\top x, \ldots, (c_0^l)^\top x, (a^l)^\top x\right)^\top \\ \text{s.t.} & x \in X \end{array} \tag{2.29}$$

using the methods of multiobjective optimization;

(2) for every $x \in P_m$ compute the corresponding objective values of (2.28);

(3) obtain the Pareto set P_r of (2.28) by eliminating the solutions that are now dominated by others in P_m, if such exist.

Corollary 2.25. *The Pareto set P^* of (2.28) is P_r.*

Proof. Since P_m is a Pareto set of (2.29), for every $y \in X \setminus P_m$ there is an $x \in P_m$ which satisfies

$$\begin{aligned} f^1(x) &\leq f^1(y) \\ &\vdots \\ f^k(x) &\leq f^k(y) \\ (c_0^1)^\top x &\leq (c_0^1)^\top y \\ (a^1)^\top x &\leq (a^1)^\top y \\ &\vdots \\ (c_0^l)^\top x &\leq (c_0^l)^\top y \\ (a^l)^\top x &\leq (a^l)^\top y. \end{aligned}$$

This implies that for every $y \in X \setminus P_m$ there is an $x \in P_m$ which satisfies

$$\begin{aligned} f^1(x) &\leq f^1(y) \\ &\vdots \\ f^k(x) &\leq f^k(y) \\ (c_0^1)^\top x + \sqrt{(a^1)^\top x} &\leq (c_0^1)^\top y + \sqrt{(a^1)^\top y} \\ &\vdots \\ (c_0^l)^\top x + \sqrt{(a^l)^\top x} &\leq (c_0^l)^\top y + \sqrt{(a^l)^\top y}, \end{aligned}$$

such that $P^* \subseteq P_m$ holds. After the second step the equality $P^* = P_r$ is established. □

Obviously, P^* might be a proper subset of P_m:

Example 2.26. Consider two elements $x, y \in P_m$ with

$$\begin{pmatrix} f^1(x) \\ f^2(x) \\ c_0^\top x \\ a^\top x \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 4 \end{pmatrix} \text{ and } \begin{pmatrix} f^1(y) \\ f^2(y) \\ c_0^\top y \\ a^\top y \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 4 \\ 1 \end{pmatrix}.$$

We obtain

$$\begin{pmatrix} f^1(x) \\ f^2(x) \\ c_0^\top x + \sqrt{a^\top x} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \text{ and } \begin{pmatrix} f^1(y) \\ f^2(y) \\ c_0^\top y + \sqrt{a^\top y} \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix},$$

i.e. x dominates y with respect to Problem (2.28). □

Part II

Uncorrelated Case

In robust optimization with ellipsoidal uncertainty sets, usually defined by a mean $c_0 \in \mathbb{R}^n$ and a covariance matrix $A \in \mathcal{S}^n_{++}$, the theoretically most interesting special case is defined by means of a *diagonal* matrix A – the case where we assume to not have any covariances between the costs of the variables. This case leads to Model (2.17), which we focus on in this section. Although, due to diagonality, the problem is considerably simplified compared to the general model, the difficulty of a trade-off between the linear and the square-root term remains in the objective function. The objective function keeps being non-separable on account of the square-root operator and therefore still not trivial. A need for investigation arises also due to the fact that the complexity status of Problem (2.17) for a wide range of underlying combinatorial sets $X \subseteq \{0,1\}^n$ (including the important robust Shortest Path-problem) is still unknown. Exact solution methods, even when not polynomial in theory, are therefore in demand. For matroids, where the polynomial solvability has been shown in the uncorrelated case, the algorithms found so far lead to the runtime of $\mathrm{O}(l^2)$, if l is the time needed for one oracle call to a corresponding deterministic algorithm. This is still too long for a practical use. For the set $X = \{0,1\}^n$ we mentioned two approaches in Chapter 2, both with polynomial running time. Firstly, solving Problem (2.21), we minimize a submodular function over an unrestricted binary domain, which is possible in time $\mathrm{O}(n^5)$. Secondly, solving Problem (2.21), we minimize a mean-risk objective function over a matroid, which in this case leads to a running time of $\mathrm{O}(n^2)$. Still for a practical use faster algorithms are required, even in these special cases.

In this part we introduce our results of the research on the uncorrelated case. We will pay a particular attention to two special subsets of $X \subseteq \{0,1\}^n$: In Chapter 3 we will take a closer look at the set $X = \{0,1\}^n$ and aim at an improvement of the runtime of the known algorithms for general submodular functions as well as of the runtime of the approach for matroids given in [60]. We will also give a glimpse on the case of integer variables combined with ellipsoidal uncertainty, for which we provide a pseudo-polynomial algorithm.

In Chapter 4 the set X of all $s-t$-paths in a graph is in the center of attention. This set in combination with uncorrelated ellipsoidal uncertainty is currently very popular, since in the light of a high degree of relevance there is still no polynomial algorithm for the corresponding robust Shortest Path-problem. We will also describe an exact algorithm for general robust combinatorial problems, which is in essence suitable for any type of uncertainty combined with any type of constraints. It is a branch and bound-algorithm that uses lower bounds obtained from Lagrangean decomposition.

Chapter 3

Unconstrained Uncorrelated Case

Dealing with uncorrelated ellipsoidal uncertainty in combinatorial optimization, i.e. Problem (2.17), where we have seen the importance of the exact form of the underlying combinatorial set $X \subseteq \{0,1\}^n$ for the problem complexity, it appears natural to consider first the unrestricted case $X = \{0,1\}^n$.

To imagine some real world situation, which can be modeled by this apparently very special case, one can think again about portfolio selection (like in Example 2.7), but without any budget- or other restrictions. Aiming to maximize the return of stocks and if the budget allows to invest in all the given n items, we still do not want to make a loss of our invested funds. This is though possible due to the uncertain economical developments and the associated variances of the expected return values.

So we focus on the Problem (2.21), which we already know to be polynomially solvable due to submodularity of the objective function (see Section 2.4).

In this section we devise a faster algorithm to solve this problem. This is an important step in our research in the area of robust optimization, since the unrestricted binary Problem (2.21) appears as a subproblem in the decomposition approach for Problem (2.17) with arbitrary combinatorial sets $X \subseteq \{0,1\}^n$ and later also with arbitrary A, which we describe in Chapters 4 and 6. We discuss its properties and generalizability.

3.1 Combinatorial $O(n \log n)$-Algorithm

We deduce now the new combinatorial algorithm to solve Problem (2.21) efficiently in time $\mathrm{O}(n \log n)$ and prove its correctness and runtime. These results have been published in our paper [8] and in our technical report [9]. After that we will analyze the properties of the algorithm and apply it to general p-norms.

3.1.1 The Algorithm

The unrestricted binary Problem (2.21) with uncorrelated ellipsoidal uncertainty only deals with the question for every variable if it is worthwhile for the objective function to set the variable to 1 or not, since there is no further constraint on solutions to be satisfied. If the cost contribution of a variable to the value of the objective function is negative, then the variable should be set to 1, if it is non-negative, the variable should be set to 0. Besides from the cost coefficient c_{0i} which influences explicitly the value of the objective function, this decision depends also on the effect of the other variables on the impact of the coefficient a_i, which varies due to concavity of the square-root. Submodularity of the objective function, documented in Section 2.4, reveals that setting a variable to 1 has a decreasing effect on the contributions of other variables. We analyze in more detail the contribution of a variable to the objective function value depending on the values of other variables.

To this purpose, we consider two solutions of Problem (2.21) which differ in exactly one variable i. The difference between the corresponding objective values is

$$\Delta_i f(J) = c_{0i} + \sqrt{\sum_{j \in J} a_j + a_i} - \sqrt{\sum_{j \in J} a_j}, \tag{3.1}$$

where J denotes the set of variables which are 1 in both solutions. This value is also known as the *discrete derivative* of variable i [71]. It describes the *contribution* of setting variable i to 1 and clearly depends on the set J or, more precisely, on the value $\sum_{j \in J} a_j$. We abstract this dependency by replacing the quantity $\sum_{j \in J} a_j$ through a continuous variable $z > 0$ and define for each variable i its *contribution function* by

$$C_i(z) = c_{0i} + \sqrt{z + a_i} - \sqrt{z}$$

(see Figure 3.1). The contribution functions C_i are strictly decreasing and therefore have at most one root each. The root r_i of C_i indicates the value which $\sum_{j \in J} a_j$ must reach such that setting variable i to 1 becomes profitable. We use the fact that setting a variable to 1 never has a negative effect on the contributions of other variables, due to submodularity of the objective function of (2.21). Our basic idea for the construction of an optimal solution of (2.21) is that, due to the definition of r_i, a variable i cannot be 1 in the optimal solution while another variable having a smaller root is 0. This is due to the fact that, if a variable i has the value 1 in the optimal solution, its contribution to the solution value is necessarily negative due to the unrestricted minimization sense. In other words, the value $r_i > 0$, necessary for the contribution of this variable to be negative, is reached by the sum of the a-coefficients of the other variables with the value 1 in this solution. But that means that the contribution of the variables having a smaller root than r_i and which are assumed to be 0 in the optimal solution would

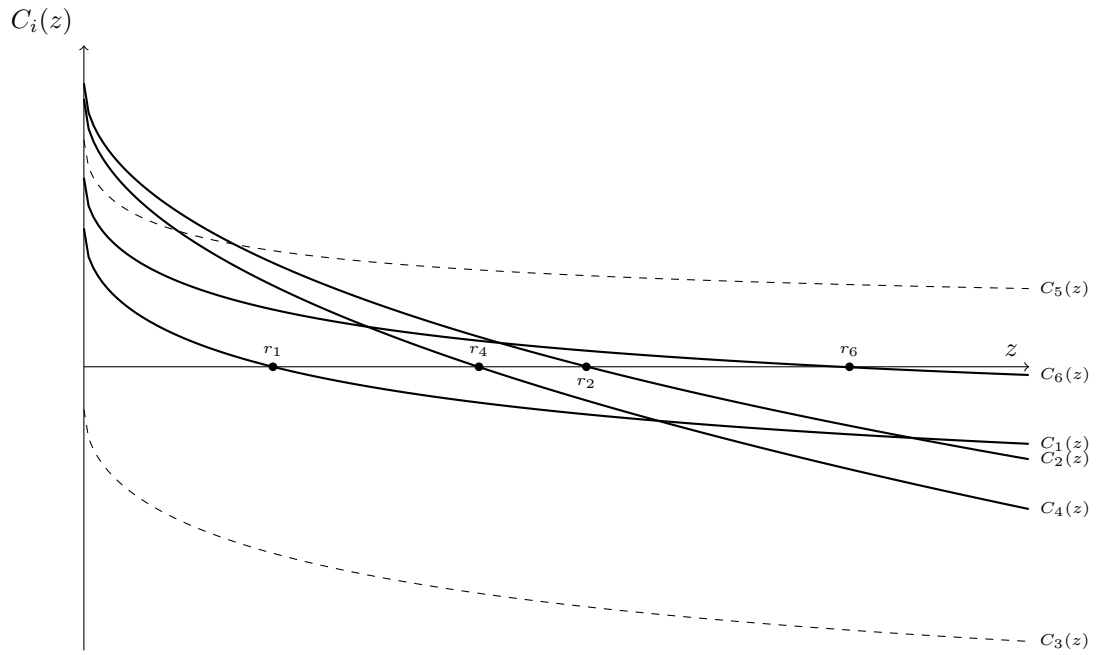

Figure 3.1: Contribution functions.

be negative and would strictly improve the value of the solution. This observation leads to a sorting-based algorithm. However, in a first step we have to eliminate variables i for which C_i has no root, using the following lemma.

Lemma 3.1. *There exists an optimal solution x^* of Problem* (2.21) *with the following properties:*

(i) If $c_{0i} \geq 0$, then $x_i^* = 0$.

(ii) If $c_{0i} \leq -\sqrt{a_i}$, then $x_i^* = 1$.

Proof. The condition in (i) implies that the function C_i is positive everywhere, as $a_i > 0$. This implies that any solution with $x_i = 1$ can be improved by setting $x_i = 0$. The condition in (ii) implies that C_i is non-positive everywhere, as

$$c_{0i} + \sqrt{z + a_i} - \sqrt{z} \leq c_{0i} + \sqrt{a_i} \leq 0$$

by concavity of the square root function. The contribution of the variable i to the value of an arbitrary solution is therefore non-positive, so that it may be fixed to 1 without loss of generality. □

The two cases of Lemma 3.1, in which the contribution functions have no roots, are illustrated by the dashed functions in Figure 3.1. For the remaining variables,

Algorithm 1 To solve the uncorrelated binary mean-risk problem

Input: $c_0 \in \mathbb{R}^n, a \in \mathbb{R}^n_+$
Output: Vector $x^* \in \{0,1\}^n$ minimizing $c_0^\top x + \sqrt{a^\top x}$

▷ Preprocessing

$k = 0$, $A = 0$, $C = 0$
for $i \in \{1, ..., n\}$ **do**
  if $c_{0i} \geq 0$ **then**
    $x_i^* = 0$ ▷ this element cannot improve the solution
  else if $c_{0i} \leq -\sqrt{a_i}$ **then**
    $x_i^* = 1$ ▷ this element will always improve the solution
    $A = A + a_i$, $C = C + c_{0i}$
  else
    $r_i \leftarrow \left(\frac{a_i - c_{0i}^2}{2c_{0i}}\right)^2$
    $k \leftarrow k + 1$
  end if
end for

sort the variables such that $x_1, \ldots, x_k$ are unfixed with $r_1 \leq r_2 \leq \cdots \leq r_k$

$f^* = C + \sqrt{A}$, $i^* = 0$
for $i \in \{1, ..., k\}$ **do** ▷ determining the objective value
  $A = A + a_i$, $C = C + c_{0i}$
  if $C + \sqrt{A} < f^*$ **then**
    $f^* = C + \sqrt{A}$, $i^* = i$
  end if
end for

for $i \in \{1, ..., i^*\}$ **do** ▷ defining the solution
  $x_i^* = 1$
end for
for $i \in \{i^* + 1, ..., k\}$ **do**
  $x_i^* = 0$
end for

return x^*

i.e. the variables with $0 > c_{0i} > -\sqrt{a_i}$, the functions C_i have exactly one positive root each. The entire algorithm is stated in Algorithm 1.

Theorem 3.2. *Algorithm 1 solves Problem* (2.21) *in time* $O(n \log n)$.

Proof. The algorithm runs in linear time except for the sorting of variables which takes $\mathrm{O}(n \log n)$ time. It thus remains to prove the correctness of Algorithm 1.

We first assume $k = n$, i.e. that no variable is fixed by Lemma 3.1. Then it suffices to show that (after sorting) every optimal solution x^* satisfies $x_1^* \geq x_2^* \geq \cdots \geq x_n^*$. Assume on contrary that x^* is an optimal solution with $x_j^* = 0$ and $x_{j+1}^* = 1$ for some $j < n$. Consider the two solutions x^0 and x^1 defined by

$$x_i^0 = \begin{cases} 0 & \text{for } i = j+1 \\ x_i^* & \text{otherwise,} \end{cases} \qquad x_i^1 = \begin{cases} 1 & \text{for } i = j \\ x_i^* & \text{otherwise.} \end{cases}$$

By optimality of x^* we have

$$0 \geq f(x^*) - f(x^0) = C_{j+1}\big(\textstyle\sum_{i \in I} a_i\big),$$

for $I = \{i \in \{1, ..., n\} \setminus \{j+1\} \mid x_i^* = 1\}$ and hence, by definition of r_{j+1} and r_j,

$$\sum_{i \in I} a_i \geq r_{j+1} \geq r_j \ . \tag{3.2}$$

Then, using the concavity of the square-root function we have

$$\begin{aligned} f(x^1) - f(x^*) &= c_{0j} + \sqrt{\sum_{i \in I} a_i + a_{j+1} + a_j} - \sqrt{\sum_{i \in I} a_i + a_{j+1}} \\ &< c_{0j} + \sqrt{\sum_{i \in I} a_i + a_j} - \sqrt{\sum_{i \in I} a_i} \\ &\overset{(3.2)}{\leq} c_{0j} + \sqrt{r_j + a_j} - \sqrt{r_j} = 0, \end{aligned}$$

which contradicts the optimality of x^*.

Now assume $k < n$, i.e. some variables were fixed according to Lemma 3.1. Fixing a variable to 0 (case (i)) causes a trivial dimension reduction. Fixing a variable to 1 (case (ii)) yields a constant $C < 0$ in the linear part of the objective function and a constant $A > 0$ in the square root term. In this case the roots can be computed using the same formula, because an additional constant in the square root term does not affect the order of the roots (the constant C is subtracted anyway). It merely causes a shift of the roots to the left by the same amount for each variable. Thus, fixing variables does not affect the strategy and the correctness of the algorithm in the presence of fixings is easily verified. □

In the procedure only the order of the roots matters. Due to the equivalence

$$-\frac{a_i}{c_{0i}} + c_{0i} \leq -\frac{a_j}{c_{0j}} + c_{0j} \iff \left(\frac{a_i - c_{0i}^2}{2c_{0i}}\right)^2 \leq \left(\frac{a_j - c_{0j}^2}{2c_{0j}}\right)^2$$

for i and j with $c_{0i}, c_{0j} < 0$ and $c_{0i} > -\sqrt{a_i}$ and $c_{0j} > -\sqrt{a_j}$, one can save several monotonous operations in the calculation of the roots and sort the variables by $-\frac{a_i}{c_{0i}} + c_{0i}$ instead of by $\left(\frac{a_i - c_{0i}^2}{2c_{0i}}\right)^2$.

3.1.2 Theoretical Properties

As mentioned above, the most costly part of the algorithm is sorting the variables, which is possible in $O(n \log n)$ runtime (for example with *merge sort*). All of the $n+1$ (or with notation of Algorithm 1, $k+1$) uprising solutions need to be evaluated since we have not found any structure (monotony, (quasi) concavity/convexity) in the arising sequence of values.

Shen et al. in their article "A Joint Location Inventory Model" [69] provide a different sorting rule on variables, which leads to a subset of feasible solutions containing the optimal solution of Problem (2.21). They sort the variables by $\frac{c_{0i}}{a_i}$ (which is the same as sorting by $-\frac{a_i}{c_{0i}}$). It is interesting to note that both sorting rules yield in general different subsets of feasible solutions both containing the optimal solution:

Example 3.3. Consider the instance of Problem (2.21) with

$$c_0 = \left(-2, -10, -\tfrac{1}{2}, -\tfrac{1}{5}, -3, -1\right)^\top \text{ and } a = (10, 2, 5, 1, 16, 25)^\top.$$

By both rules, x_2 has to be fixed to 1. Algorithm 1 yields the order

$$x_2, x_5, x_4, x_1, x_3, x_6$$

of the variables. Hence, the solutions to be compared are

$$\begin{pmatrix}0\\1\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\1\\1\\0\end{pmatrix}, \begin{pmatrix}1\\1\\0\\1\\1\\0\end{pmatrix}, \begin{pmatrix}1\\1\\1\\1\\1\\0\end{pmatrix} \text{ and } \begin{pmatrix}1\\1\\1\\1\\1\\1\end{pmatrix}.$$

The sorting rule from [69] leads to the solutions

$$\begin{pmatrix}0\\1\\0\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}1\\1\\0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}1\\1\\0\\1\\1\\0\end{pmatrix}, \begin{pmatrix}1\\1\\1\\1\\1\\0\end{pmatrix} \text{ and } \begin{pmatrix}1\\1\\1\\1\\1\\1\end{pmatrix}.$$

The optimal solution $(1,1,0,1,1,0)^\top$ is contained in both subsets. □

3.2 Generalization to p-Norm-Uncertainty

As analysed in detail above, the principal uncertainty sets discussed in the literature are ellipsoidal-, discrete scenarios-, interval uncertainty, as well as general polytope- and Γ-uncertainty. Most of the continuous uncertainty sets can be defined by a mean and a range of deviation, i.e. most of them are based on a *norm*. For example, the Γ-uncertainty is based on the 1-norm, the uncorrelated ellipsoidal uncertainty on the Euclidian norm and the interval uncertainty on the ∞-norm. Taking a closer look on the connection between these three uncertainty types, we realize that their considering is not primary caused by the frequency of application in the real life, but by their simplicity. However, in some applications not only these three special cases might be the best models of uncertain reality, but sometimes rather the in-between cases are conceivable.

3.2.1 The Model

We consider now the general p-norm, with $p \in (1,\infty)$, characterizing the uncertainty set

$$\mathcal{U}_p = \left\{ c \in \mathbb{R}^n \mid \sqrt[p]{\frac{1}{a_1^{p-1}}(c-c_0)_1^p + \cdots + \frac{1}{a_n^{p-1}}(c-c_0)_n^p} \leq r \right\},$$

with c_0 being the center of the set and $a_1,\ldots,a_n$ an analogue of the variances in the case of ellipsoids (see Figure 3.2).

The parameter r acts again like a volume-describing constant, which against the background of robust optimization parametrizes the level of risk-aversion of the user. So, now we aim to solve the robust problem

$$\begin{array}{ll} \min & \max\limits_{c\in\mathcal{U}_p} c^\top x \\ \text{s.t.} & x \in \{0,1\}^n \ . \end{array} \tag{3.3}$$

Further on we refer to this uncertainty as to *p-norm-uncertainty*. The case $p=2$, i.e. the uncorrelated ellipsoidal uncertainty, is still contained in this formulation.

Like in the uncorrelated ellipsoidal uncertainty case, we can reformulate the objective function of Problem (3.3) for general $1<p<\infty$ to a closed formula in x.

Theorem 3.4. *Problem* (3.3) *can be formulated as*

$$\begin{array}{ll} \min & c_0^\top x + r\left(a^\top x\right)^{\frac{p-1}{p}} \\ s.t. & x \in \{0,1\}^n \ . \end{array} \tag{3.4}$$

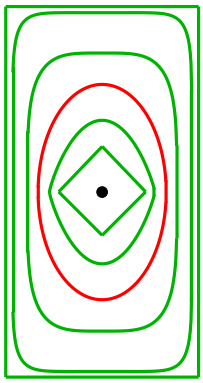

Figure 3.2: p-norm-uncertainty sets for different $p > 1$.

Proof. Like in Section 2.2.1.1 we can use, for example, the Karush-Kuhn-Tucker optimality conditions for the maximization problem: For the considered values of p the set $\mathcal{U}_p$ is described by a norm, i.e. the corresponding unit ball is a convex compact set; a Slater point is given by c_0 and the objective function of the maximization problem is linear for every fixed x.

For the non-trivial case of $x \neq 0$ and for the optimal solution c^* of

$$\max_{c \in \mathcal{U}_p} c^\top x = -\min_{c \in \mathcal{U}_p} -c^\top x \tag{3.5}$$

we find a Lagrangean multiplier $\lambda^* \geq 0$ satisfying the following conditions:

$$\sqrt[p]{\frac{1}{a_1^{p-1}}(c^* - c_0)_1^p + \cdots + \frac{1}{a_n^{p-1}}(c^* - c_0)_n^p} - r \leq 0 \qquad \text{(Primal feasibility)}$$

$$-x + \frac{\lambda^*}{\left(\frac{1}{a_1^{p-1}}(c^* - c_0)_1^p + \cdots + \frac{1}{a_n^{p-1}}(c^* - c_0)_n^p\right)^{\frac{p-1}{p}}} \begin{pmatrix} \frac{1}{a_1^{p-1}}(c^* - c_0)_1^{p-1} \\ \vdots \\ \frac{1}{a_n^{p-1}}(c^* - c_0)_n^{p-1} \end{pmatrix} = 0$$
(Stationarity)

$$\lambda^* \left(\sqrt[p]{\frac{1}{a_1^{p-1}}(c^* - c_0)_1^p + \cdots + \frac{1}{a_n^{p-1}}(c^* - c_0)_n^p} - r \right) = 0$$
(Complementary slackness)

Obviously $\lambda^* > 0$, since otherwise the stationarity would imply $x = 0$. Comple-

mentary slackness yields then

$$\frac{1}{a_1^{p-1}}(c^* - c_0)_1^p + \cdots + \frac{1}{a_n^{p-1}}(c^* - c_0)_n^p = r^p. \tag{3.6}$$

Inserting (3.6) into the stationarity condition implies

$$-x + \lambda^* \frac{1}{r^{p-1}} \begin{pmatrix} \frac{1}{a_1^{p-1}}(c^* - c_0)_1^{p-1} \\ \vdots \\ \frac{1}{a_n^{p-1}}(c^* - c_0)_n^{p-1} \end{pmatrix} = 0$$

and finally

$$c^* - c_0 = \frac{r}{\sqrt[p-1]{\lambda^*}} \begin{pmatrix} a_1 x_1 \\ \vdots \\ a_n x_n \end{pmatrix}, \tag{3.7}$$

where we used the binarity of x.

Substituting $c^* - c_0$ in the constraint of the problem and using again the binarity of x gives

$$r^p = \epsilon_1 \left(\frac{r a_1 x_1}{\sqrt[p-1]{\lambda^*}} \right)^p + \cdots + \epsilon_n \left(\frac{r a_n x_n}{\sqrt[p-1]{\lambda^*}} \right)^p$$

$$\Longleftrightarrow$$

$$\lambda^* = \left(a_1 x_1 + \cdots + a_n x_n \right)^{\frac{p-1}{p}}.$$

Inserting λ^* into (3.7) we get a closed formula for the optimal solution

$$c^* = c_0 + \frac{r}{\sqrt[p]{a_1 x_1 + \cdots + a_n x_n}} \begin{pmatrix} a_1 x_1 \\ \vdots \\ a_1 x_1 \end{pmatrix}$$

of the maximization problem and using once again the binarity of x the objective function of (3.3) turns to

$$c_0^\top x + r \left(a^\top x \right)^{\frac{p-1}{p}}.$$

In the trivial case $x = 0$ the reformulation of the objective function is obviously valid, too. □

The *robust binary problem under p-norm-uncertainty* can thus be stated as Problem (3.4).

An interesting observation is that

$$\lim_{p \to \infty} c_0^\top x + r (a^\top x)^{\frac{p-1}{p}} = (c_0 + ra)^\top x.$$

This is in fact the objective function of (2.11) with interval uncertainty, which in turn is based on the ∞-norm.

3.2.2 Solution Approach

We now show that the idea of Algorithm 1 can be applied to solve Problem (3.4).

Lemma 3.5. *The objective function of* (3.4) *is submodular for* $p \geq 1$.

Proof. The concavity of the root function $f : \mathbb{R} \to \mathbb{R}, z \mapsto z^{\frac{p-1}{p}}$ is easy to verify using the second derivative

$$f''(z) = \frac{1-p}{p^2 z^{\frac{1+p}{p}}},$$

which is non-positive for all $p \geq 1$ and $z > 0$. The objective function of (3.4) is then a sum of a modular function $c_0^\top x$ and a composition of a modular function $a^\top x$, for $a \geq 0$, with a concave function, and thus submodular [6]. □

Also here, due to submodularity, fixing a variable to 1 can only be profitable for contributions of the other variables and we get, in analogy to the case $p = 2$, the following result.

Lemma 3.6. *There exists an optimal solution* x^* *of Problem* (3.4) *with the following properties:*

(i) If $c_{0i} \geq 0$, *then* $x_i^* = 0$.

(ii) If $c_{0i} \leq -a_i^{\frac{p-1}{p}}$, *then* $x_i^* = 1$.

Proof. In analogy to the proof of Lemma 3.1. □

Corollary 3.7. *Problem* (3.4) *can be solved in time* $O(n \log n)$.

Proof. In analogy to the proof of Theorem 3.2. □

Thus, a slight modification of Algorithm 1 concerning computation of the roots (which can easily be done using *Newton's method* [37]) and the preprocessing in the sense of Lemma 3.6 is enough to solve the unrestricted binary problem (3.3) with general p-norm-uncertainty. As a final motivation to this concept we give the following example:

Example 3.8. Consider the 2×3 grid graph on Figure 3.3 and the corresponding values (c_{0ij}, a_{ij}) for every edge (i, j). Only a slight modification of the uncertainty set from the ellipsoidal uncertainty over the more conservative 3- and 4-norm-uncertainty to the 5-norm- and the interval-uncertainty already changes the optimal solution from the path $a - b - c - f$ over the path $a - b - e - f$ to the path $a - d - e - f$ (see Table 3.1).

This example illustrates the concept of p-norm-uncertainty as an intermediate case between the ellipsoidal- and the interval uncertainty.

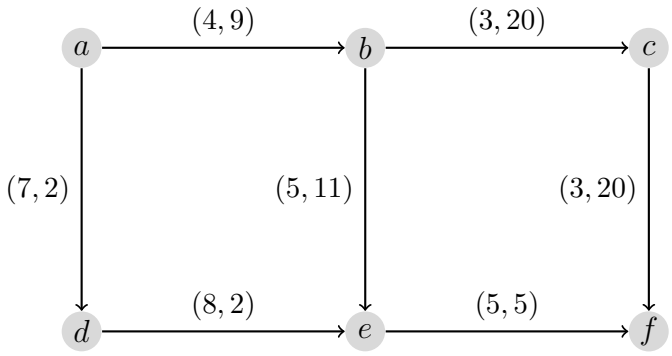

Figure 3.3: Different uncertainty sets yield different robust solutions.

Uncertainty	$a-b-c-f$	$a-b-e-f$	$a-d-e-f$
ellipsoidal	17	19	23
3-norm	23.39	22.55	24.33
4-norm	28.52	25.18	25.20
5-norm	32.50	27.13	25.80
interval	59	39	29

Table 3.1: Objective values of different paths in Figure 3.3 evaluated with respect to different norm-based uncertainty sets.

3.3 Generalization to Integer Optimization

A natural generalization of the unrestricted version of Problem (2.11) deals with integer variables with upper bounds being any integer numbers instead of 1, i.e.

$$\begin{array}{ll} \min & \max\limits_{c\in\mathcal{U}} c^\top x \\ \text{s.t.} & 0 \leq x \leq u \\ & x \in \mathbb{Z}^n, \end{array} \tag{IP}$$

with $u \in \mathbb{Z}^n$ and $\mathcal{U} = \left\{ c \in \mathbb{R}^n \middle| \, (c-c_0)^\top A^{-1} (c-c_0) \leq r^2 \right\}$.

Even though this problem is not a typical representative of the problem set covered by this thesis, its consideration can be seen as a byproduct of the investigation of the binary case which is motivated by the same field of applications. Thinking of the portfolio theory where investment decisions have to be made, represented by independent random variables (see Example 2.18), we might not only restrict to the binary question, whether the investment should be made or not. The amount of items to each investment might be also asked for, which leads to general integer variables.

For $r = 1$ and in complete analogy to the binary case, Problem (IP) can be

reduced to

$$\begin{aligned} \min \quad & c_0^\top x + \sqrt{\sum_{i=1}^{n} a_i x_i^2} \\ \text{s.t.} \quad & 0 \leq x \leq u \\ & x \in \mathbb{Z}^n. \end{aligned} \tag{IUE}$$

In contrast to the binary case, however, the objective function still contains quadratic terms. Like in the binary case, without loss of generality we assume $a \in \mathbb{R}^n_+$ and $c_0 \in \mathbb{R}^n_-$, since for positive coefficients c_{0i} the corresponding variable can directly be fixed to 0.

Due to similarity to the binary Problem (2.21) one might suppose that in the optimal solution each variable x_i is set to 0 or to its upper bound u_i, such that Problem (IUE) could be easily reduced to the binary Problem (2.21). But this supposition is wrong as the following example shows.

Example 3.9. Consider Problem (IUE) with $n = 2$, $c_0 = (-2, -3)^\top$, $a = (10, 6)^\top$ and $u = (2, 2)^\top$. By means of enumeration we can see that the optimal solution is $(1, 2)^\top$, such that not all variables are set to their bounds. □

Nevertheless, we can reduce Problem (IUE) to Problem (2.21). We will now introduce a pseudo-polynomial algorithm for Problem (IUE) based on the idea of the algorithm in the binary case and conclude that Problem (IP) is at most weakly $\mathcal{NP}$-hard.

We consider two solutions x^0 and x^1 which differ by one at exactly one variable i, i.e. $x_i^0 = k - 1$ and $x_i^1 = k$ with $k \in \{1, \ldots, u_i\}$ and $x_j^0 = x_j^1$ for $j \neq i$. The difference between the corresponding objective values is

$$\begin{aligned} f(x^1) - f(x^0) = & \sum_{j \neq i} c_{0j} x_j^1 + c_{0i} k + \sqrt{\sum_{j \neq i} a_j (x_j^1)^2 + a_i k^2} \\ & - \sum_{j \neq i} c_{0j} x_j^0 - c_{0i}(k-1) - \sqrt{\sum_{j \neq i} a_j (x_j^0)^2 + a_i (k-1)^2} \\ = & \, c_{0i} + \sqrt{\sum_{j=1}^{n} a_j (x_j^0)^2 + a_i (2k-1)} - \sqrt{\sum_{j=1}^{n} a_j (x_j^0)^2}. \end{aligned}$$

Based on this observation and similar to the binary case, we introduce *contribution functions for each value increase from $k-1$ to k of each variable i*:

$$C_{i,k}(z) := c_{0i} + \sqrt{z + a_i(2k-1)} - \sqrt{z}.$$

The analytical properties of the contribution functions are equivalent to those in the binary case, so that after the same preprocessing (with respect to $a_i(2k-1)$

instead of a_i) only the functions having exactly one root each remain. It is easy to see that

$$C_{i,k}(z) < C_{i,k+1}(z) \text{ for all } z > 0,$$

which implies $r_{i,k} < r_{i,k+1}$, i.e. the set $(r_{i,1}, \ldots, r_{i,u_i})$ of the roots of one variable is monotonous. We sort all roots (of all variables and their possible values) in a non-decreasing order. We define at most $\sum_{i=1}^{n} u_i + 1$ solutions $x_{i,k}$ by means of increasing of the variable values by one in each step in the order of the roots. In the solution $x_{i,k}$ the variable i has the value k. The other variables j are set to the values corresponding to the largest root $r_{j,l}$, which is smaller than the root $r_{i,k}$, i.e.

$$\begin{aligned} (x_{i,k})_j = \arg \quad & \max_l r_{j,l} \\ \text{s.t.} \quad & r_{j,l} \leq r_{i,k}. \end{aligned} \tag{3.8}$$

Due to monotony of the roots for one variable these solutions are well-defined. Comparing the objective values of these solutions yields an optimal solution of Problem (IUE). Before we formally prove this claim, let us demonstrate the approach on a short example.

Example 3.10. Consider again the instance of Example 3.9. As

$$c_{02} = -3 < -\sqrt{6(2-1)} = -\sqrt{a_2(2k-1)},$$

the value of the variable x_2 is at least 1, such that we do not need to consider the contribution function $C_{2,1}$. For the other functions we have:

The contribution function of the variable x_1 for the value increase from 0 to 1 is

$$C_{1,1}(z) = -2 + \sqrt{z+10} - \sqrt{z},$$

the contribution function of the variable x_1 for the value increase from 1 to 2 is

$$C_{1,2}(z) = -2 + \sqrt{z+30} - \sqrt{z},$$

the contribution function of the variable x_2 for the value increase from 1 to 2 is

$$C_{2,2}(z) = -3 + \sqrt{z+18} - \sqrt{z},$$

see Figure 3.4. Note, that $C_{1,1}(z) < C_{1,2}(z)$ and $C_{2,1}(z) < C_{2,2}(z)$ for all $z \in [0, \infty)$. For the roots we have:

$$r_{1,1} = \tfrac{9}{4},\ r_{1,2} = \tfrac{169}{4} \text{ and } r_{2,2} = \tfrac{9}{4},$$

such that $r_{1,1} = r_{2,2} < r_{1,2}$. As the value of x_2 is at least 1, the first solution we obtain is the "zero solution" $(0,1)^\top$. The smallest roots are $r_{1,1} = r_{2,2}$, such that we next increase the value of the variable x_1 to 1 and the value of the variable x_2 to 2 and obtain the solution $(1,2)^\top$. Finally, the next larger root is $r_{1,2}$, such that we increase the value of the variable x_1 to 2 and obtain the solution $(2,2)^\top$. The comparison of the objective values of the three solutions shows that $(1,2)^\top$ is optimal. □

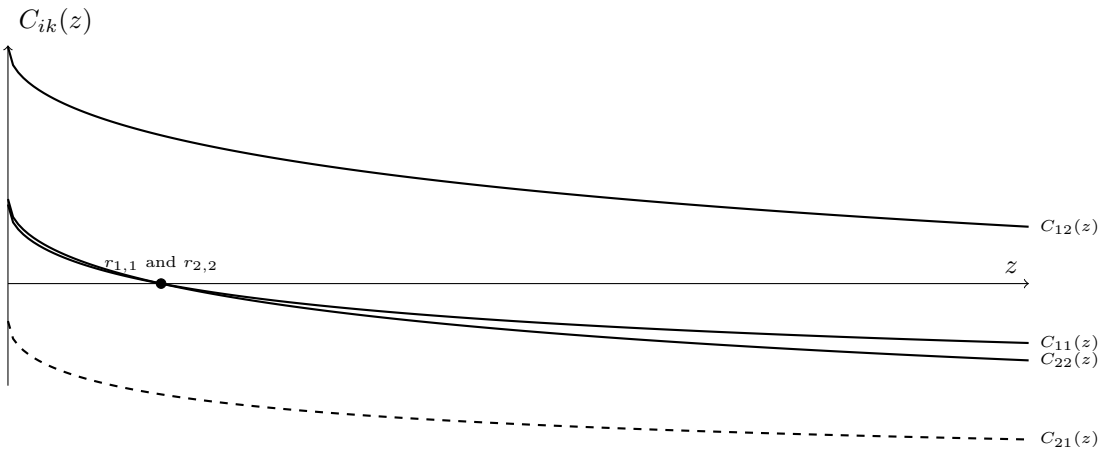

Figure 3.4: The function $C_{2,1}$ runs underneath the z-axis and has no root. The optimal value of the variable x_2 will be at least 1.

Theorem 3.11. *Let*

$$\begin{aligned}(x_{i,k})_j = \arg \quad & \max_l r_{j,l} \\ s.t. \quad & r_{j,l} \leq r_{i,k},\end{aligned}$$

with $r_{i,k}$ and $r_{j,l}$, $i,j \in \{1,\ldots,n\}$, $k \in \{1,\ldots,u_i\}$, $l \in \{1,\ldots,u_j\}$, being the roots of the contribution functions $C_{i,k}$ and $C_{j,l}$, respectively. Then an optimal solution of problem (IUE) *is given by*

$$\arg \quad \min_{x_{i,k}} c_0^\top x_{i,k} + \sqrt{\sum_{j=1}^{n} a_j (x_{i,k})_j^2} \ ,$$

i.e., comparing the at most $\sum_{i=1}^n u_i + 1$ solutions $x_{i,k}$ w.r.t. their objective function values yields an optimal solution of Problem (IUE).

Proof. Let x^* be an optimal solution of (IUE) with $x_i^* = k$ and $x_j^* = l$. Assume on contrary $r_{j,l+1} < r_{i,k}$ for some $i,j \leq n$, $i \neq j$ and $0 \leq l < u_j$, $0 < k \leq u_i$ (the case $r_{i,k+1} < r_{j,l}$ is analogous). This would mean that the optimal solution x^* is not one of the $\sum_{i=1}^n u_i + 1$ solutions we consider in the procedure above. Consider the two solutions x^0 and x^1 defined by

$$x_t^0 = \begin{cases} k-1 & \text{for } t = i \\ x_t^* & \text{otherwise} \end{cases} \qquad \text{and} \qquad x_t^1 = \begin{cases} l+1 & \text{for } t = j \\ x_t^* & \text{otherwise.} \end{cases}$$

By optimality of x^* we have

$$0 \geq f(x^*) - f(x^0) = \sum_{t \neq i,j} c_{0t} x_t^* + c_{0i} k + c_{0j} l + \sqrt{\sum_{t \neq i,j} a_t x_t^{*2} + a_i k^2 + a_j l^2}$$
$$- \sum_{t \neq i,j} c_{0t} x_t^* - c_{0i}(k-l) - c_{0j} l - \sqrt{\sum_{t \neq i,j} a_t x_t^{*2} + a_i (k-1)^2 + a_j l^2}$$

$$= c_{0i} + \sqrt{\sum_{t \neq i,j} a_t x_t^{*2} + a_i k^2 + a_j l^2} - \sqrt{\sum_{t \neq i,j} a_t x_t^{*2} + a_i (k-1)^2 + a_j l^2}$$
$$= c_{0i} + \sqrt{\sum_{t \neq i,j} a_t x_t^{*2} + a_i (k-1)^2 + a_j l^2 + a_i (2k-1)}$$
$$- \sqrt{\sum_{t \neq i,j} a_t x_t^{*2} + a_i (k-1)^2 + a_j l^2}$$
$$= C_{i,k} \left(\sum_{t \neq i,j} a_t x_t^{*2} + a_i (k-1)^2 + a_j l^2 \right)$$

and hence by definition of $r_{i,k}$ and our assumption

$$\sum_{t \neq i,j} a_t x_t^{*2} + a_i (k-1)^2 + a_j l^2 \geq r_{i,k} > r_{j,l+1}. \tag{3.9}$$

Then, using the concavity of the square-root function we have

$$\begin{aligned}
f(x^1) - f(x^*) &= \sum_{t\neq i,j} c_{0t}x_t^* + c_{0i}k + c_{0j}(l+1) + \sqrt{\sum_{t\neq i,j} a_t x_t^{*2} + a_i k^2 + a_j(l+1)^2} \\
&\quad - \sum_{t\neq i,j} c_{0t}x_t^* - c_{0i}k - c_{0j}l - \sqrt{\sum_{t\neq i,j} a_t x_t^{*2} + a_i k^2 + a_j l^2} \\
&= c_{0j} + \sqrt{\sum_{t\neq i,j} a_t x_t^{*2} + a_i k^2 + a_j(l+1)^2} - \sqrt{\sum_{t\neq i,j} a_t x_t^{*2} + a_i k^2 + a_j l^2} \\
&< c_{0j} + \sqrt{\sum_{t\neq i,j} a_t x_t^{*2} + a_i(k-1)^2 + a_j(l+1)^2} \\
&\quad - \sqrt{\sum_{t\neq i,j} a_t x_t^{*2} + a_i(k-1)^2 + a_j l^2} \\
&= c_{0j} + \sqrt{\sum_{t\neq i,j} a_t x_t^{*2} + a_i(k-1)^2 + a_j l^2 + a_i(2(l+1)-1)} \\
&\quad - \sqrt{\sum_{t\neq i,j} a_t x_t^{*2} + a_i(k-1)^2 + a_j l^2} \\
&= C_{j,l+1}\left(\sum_{t\neq i,j} a_t x_t^{*2} + a_i(k-1)^2 + a_j l^2\right) < 0.
\end{aligned}$$

The last inequality is due to (3.9) and yields a contradiction to the optimality of x^*. □

The entire procedure is stated in Algorithm 2.

Corollary 3.12. *Problem* (IUE) *is solvable in pseudo-polynomial time.*

Corollary 3.12 is based on Definition 1.25. In fact, if the values of the parameters $u_i, i = 1, \ldots, n$, are polynomially bounded, Algorithm 2 takes polynomial time, namely $\mathrm{O}(\sum_{i=1}^n u_i \log \sum_{i=1}^n u_i)$.

Algorithm 2 To solve the uncorrelated integer mean-risk problem

Input: $c_0 \in \mathbb{R}, a \in \mathbb{R}^n_+, u \in \mathbb{Z}^n_+$

Output: Vector $x^* \in \mathbb{Z}^n_+$ minimizing $c_0^\top x + \sqrt{\sum_{i=1}^n a_i x_i^2}$

▷ Preprocessing

$k_i^* = 0$ for all $i \in \{1, \ldots, n\}$, $A = 0$, $C = 0$
for $i \in \{1, ..., n\}$ **do**
  if $c_{0i} \geq 0$ **then**
    $x_i^* = 0$ ▷ this element cannot improve the solution
  else if $\frac{c_{0i}^2 + a_i}{2a_i} \in [1, u_i]$ **then** ▷ if $c_{0i} \leq -\sqrt{a_i(2k-1)}$ for some $k \in [1, u_i]$
    $k_i^* = \lfloor \frac{c_{0i}^2 + a_i}{2a_i} \rfloor$ ▷ value of x_i^* is at least k_i^*
    $A = A + a_i(k_i^*)^2$, $C = C + k_i^* c_{0i}$
  else
    for $k \in \{k_i^* + 1, ..., u_i\}$ **do**
      $r_{i,k} \leftarrow \left(\frac{a_i(2k-1) - c_{0i}^2}{2c_{0i}} \right)^2$
      $k \leftarrow k + 1$
    end for
  end if
end for

Sort the roots $r_{i,k}$ for $i \in \{1, ..., n\}$ and $k \in \{k_i^* + 1, ..., u_i\}$ in a non-descending order and obtain the list L with $|L| \leq \sum_{i=1}^n u_i$ of the corresponding indices.

$f^* = C + \sqrt{A}$
for $(i, k) \in L$ **do**
  $A = A + a_i(2k - 1)$, $C = C + c_{0i}$
  if $C + \sqrt{A} < f^*$ **then**
    $f^* = C + \sqrt{A}$, $k_i^* = k_i^* + 1$
  end if
end for

for $i \in \{1, ..., n\}$ **do**
  $x_i^* = k_i^*$
end for

return x^*

Chapter 4

Constrained Uncorrelated Case

We now address Problem (2.17) with an arbitrary combinatorial set $X \subseteq \{0,1\}^n$. We have seen that the specificity of X causes differences in the complexity: While some forms of X lead to proven polynomial solvability, for others, like the robust Shortest Path-problem, the complexity status is yet unclear.

In the first part of this chapter the robust Shortest Path-problem with diagonal covariance matrix A obtains particular attention. We specify the implications of the Nikolova-approach (see Section 2.4.2.2) for this problem and analyze in more detail a labeling approach proposed in [25]. We discuss this approach from the complexity-theoretical perspective and extract a property which might lead to polynomial solvability of problems with certain graph structures.

The second part is dedicated to general combinatorial sets $X \subseteq \{0,1\}^n$. We describe an exact algorithm to solve Problem (2.17). It is a branch and bound-approach based on Lagrangean decomposition, which uses our results for the unconstrained case and computes lower bounds using Algorithm 1 described in Section 3.1. We discuss its theoretical properties and evaluate it afterwards experimentally. One of the applications we address in our tests is again the robust Shortest Path-problem.

4.1 Shortest Path under Ellipsoidal Uncertainty

One of the best-known combinatorial problems with many real-world applications is the *Shortest Path-problem*. As we have taken a deeper interest in combinatorial optimization under ellipsoidal uncertainty, a particular investigation of the robust Shortest Path-problem, known in the literature as the *reliable Shortest Path-problem* [25, 24], is required. We want to examine how the tractability of this special problem behaves under the robust objective function and – no less important – how easy it is to solve it in practice.

Thus, the combinatorial structure we consider in this section is a directed graph $G = (V, E)$ with a node set V and an edge set E and two special nodes $s, t \in V$, being the start node and the terminal node, respectively. We aim to find the shortest $s-t$-path with respect to randomly distributed costs on edges, i.e. we specialize on the problem

$$\begin{array}{ll} \min & c_0^\top x + \sqrt{x^\top A x} \\ \text{s.t.} & x \in X_{st} \end{array} \tag{RSP}$$

with X_{st} defining the set of $s-t$-paths in G. Here we consider the case of uncorrelated edge costs, i.e. the objective function consisting of a linear part $c_0^\top x$ and a non-linear part $\sqrt{a^\top x}$, assuming A to be a diagonal matrix. Also, we make the assumption of a graph with non-negative weights.

The standard Shortest Path-problem is known to be solvable in polynomial time using for example *Dijkstra's algorithm* [31]. We now discuss what causes the difficulty in the reliable Shortest Path-problem in comparison to the linear case.

The most widely applied algorithms for the deterministic problem make use of *Bellman's principal of optimality* [49], the foundation of dynamic programming. The principal states that the optimal solution of some problems is composed of optimal solutions of partial problems. For the Shortest Path-problem it holds in the case of a linear objective function: The shortest path from the node s to the node t is composed of subpaths which in turn are shortest paths between their initial and terminal nodes. Based on that observation, it is sufficient to successively store the shortest path and its value from the source node s to every other node until we get to the terminal node t.

When considering ellipsoidal uncertainty in the edge costs, Bellman's principal of optimality is obviously not valid any more. As we can see in Figure 4.1, the shortest $s-t$-path with respect to the mean-risk costs might even consist of longest subpaths. The non-validity of this fundamental property in the case of the mean-risk objective function makes the algorithms for the standard Shortest Path-problem unsuitable for Problem (RSP).

4.1.1 Nikolova-Approach

Referring to the theory of Nikolova [60] and Bertsimas and Sim [15] from Section 2.4, we observe that the strategy of merely finding all extreme efficient solutions and choosing the best one with respect to the mean-risk objective function does not lead right away to a proven tractability of Problem (RSP) in the uncorrelated case. More precisely, from Section 2.4 we know that there is a bound $n^{\Theta(\log n)}$ [60] on the number of break points in the parametric Shortest Path-problem, i.e. the break points-based approach has a running time of $n^{O(\log n)}$, which is super-polynomial.

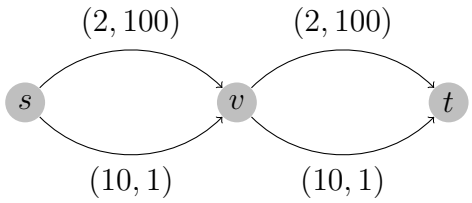

Figure 4.1: In this graph with values (c_{0i}, a_i) for every edge i the longest subpaths (both upper edges), with respect to the mean-risk costs, constitute the shortest $s-t$-path.

However, polynomial techniques have been developed to strongly reduce the number of extreme efficient solutions considered in the process of finding a reliable shortest path with the exact break points-approach, such as *List of Unexplored Triangles* and *Linear Approximation* [24]. We recall that in the break points-approach we aim to find the value of $\lambda \in [0, \infty)$ for which the optimal solution of the deterministic problem

$$\begin{array}{ll} \min & c_0^\top x + \lambda a^\top x \\ \text{s.t.} & x \in X_{sp} \end{array} \tag{4.1}$$

corresponds to the optimal solution of Problem (RSP). In both techniques, similar to the branch and bound-idea, some parts of the λ-domain are excluded from consideration due to certain observations mostly based on the concavity of the function $f : \mathbb{R}^2_+ \to \mathbb{R}_+, f((x, y)) = x + \sqrt{y}$. Unfortunately, the mentioned accelerating strategies do not lead to a theoretically proven polynomial running time, although the practical running time is drastically reduced. Also, these scanning techniques do not use the specificity of a path set, but can be applied for other underlying combinatorial problems.

While the theoretical complexity of the general uncorrelated Problem (RSP) is still an open question, the presence of certain special structures in a graph presumably implies though a polynomial number of extreme efficient paths in the corresponding parametrical problem, so that the break points-approach solves these special cases of Problem (RSP) efficiently. This is the case with the *series-parallel graphs*.

Definition 4.1. A multigraph is called *series-parallel* if it can be constructed recursively from a complete graph with two nodes by the operations of subdividing and of doubling edges.

For this special graph type Chassein et al. [24] proved a polynomial number of break points:

Theorem 4.2. *[24] If $G = (V, E)$ is a series-parallel graph with $|V| = n$ and $|E| = m$, the number of extreme efficient solutions of the bicriteria shortest path problem is bounded by $m - n + 2$.*

For the proof we refer to [24].

Corollary 4.3. *The robust Shortest Path-problem* (RSP) *with uncorrelated ellipsoidal uncertainty on series-parallel graphs is polynomially solvable.*

4.1.2 Labeling Approach

A different approach to the uncorrelated Problem (RSP) is based on the node-labeling algorithms for the *bicriteria Shortest Path-problem* [25]. The idea of the labeling approaches is to successively proceed through the nodes, similarly to Dijkstra's algorithm, but instead of storing one single path in every node, to store all *non-dominated* paths in the sense of Definition 1.32. The tuples of both objective function values of a path from the source node to the current node are called *labels*. While proceeding with respect to certain priority conditions (we refer to Chen et al. [25] for the details) the labeling approach leads to a running time $O(mp + n \log n)$, where n and m are the numbers of nodes and edges in the graph, respectively, and p is the maximum number of labels stored at a single node during the procedure. Therefore, the complexity of the labeling approach depends on the quality of the dominance conditions used to eliminate stored labels in the nodes.

As we know that the set of the non-dominated solutions is a superset of the extreme efficient solutions, this approach also solves Problem (RSP) exactly. However, we also know that the number of non-dominated solutions in the sense of bicriteria optimization is exponential in general (see Section 1.3).

Nevertheless, Chen et al. [25] devise stronger dominance conditions specified for the uncorrelated Problem (RSP) which allow to cut significantly more labels in every node than in the standard labeling algorithms for the bicriteria shortest path problem.

Definition 4.4. [25] A $v-w$-path p *dominates* a $v-w$-path q, $q \neq p$, if p and q satisfy

$$C_p \leq C_q \text{ and } C_p + \sqrt{A_p} < C_q + \sqrt{A_q},$$

where $C_p := \sum_{i \in p} c_{0i}$, $C_q := \sum_{i \in q} c_{0i}$, $A_p := \sum_{i \in p} a_i$ and $A_q := \sum_{i \in q} a_i$.

It is easy to see that if a $v-w$-path q is dominated by a $v-w$-path p in the sense of Definition 4.4, it then can be excluded from further consideration as part of an optimal solution of the uncorrelated Problem (RSP):

Theorem 4.5. *[25] Every subpath of an optimal $s-t$-path is a non-dominated path between its initial and end nodes.*

The proof is straightforward and can be found in [25].

The claim of Theorem 4.5 can be seen as a substitute for Bellman's principal of optimality for the case of the mean-risk objective function. Further on, from the fact that only one optimal solution of Problem (RSP) is required, it is sufficient to store a single representative for every non-dominated tuple of values (C_p, A_p). For the same reason, we can also exclude the case

$$C_p < C_q \text{ and } C_p + \sqrt{A_p} = C_q + \sqrt{A_q}$$

from consideration, since in this case $A_p > A_q$ holds and by adding to both labels further positive edge values (if we are not considering labels in the destination node) the paths p and q would straight away satisfy the initial dominance conditions of Definition 4.4 due to submodularity of the function $c_0^\top x + \sqrt{a^\top x}$. Thus we can assume that for a set of non-dominated labels in a node a proper inequality in both conditions holds:

Definition 4.6 (Mean-risk-dominance)**.** A set P of $v - w$-paths is called *non-dominated in the sense of a mean-risk objective function*, if every pair $p, q \in P$ satisfies one of the following conditions:

(i) $C_p < C_q$ and $C_p + \sqrt{A_p} > C_q + \sqrt{A_q}$;

(ii) $C_p > C_q$ and $C_p + \sqrt{A_p} < C_q + \sqrt{A_q}$.

From now on we will call a set described in Definition 4.6 a *non-dominated set*.

So far a significant reduction of the number of stored labels when using the dominance-conditions of Definition 4.6 is shown experimentally [25]. Unfortunately, no proven polynomial upper bound on the number of labels stored using the conditions is still found.

We examined the mean-risk-dominance w.r.t. the cardinality of the non-dominated sets and regarding possible additional elimination criteria. To this aim we considered the capability of a subpath to be part of an optimal path depending on the upcoming coefficients, which supposed to be added to the given subpath, when proceeding with the algorithm.

Let (C_p, A_p) and (C_q, A_q) be two non-dominated labels in a given node v, and p and q the corresponding $s - v$-paths. We may assume

$$C_p < C_q \text{ and } C_p + \sqrt{A_p} > C_q + \sqrt{A_q}. \tag{4.2}$$

The subpath q is better than p with respect to the objective function in (RSP). Still, the path p cannot be excluded from consideration, since, if the variance value of a $v - t$-path, that we can add to both subpaths to obtain $s - t$-paths, is *big enough*, the relation might reverse. In a given graph a tuple (x, z) of the mean and variance values of a $v - t$-path might exist, such that

$$C_p + x + \sqrt{A_p + z} < C_q + x + \sqrt{A_q + z}. \tag{4.3}$$

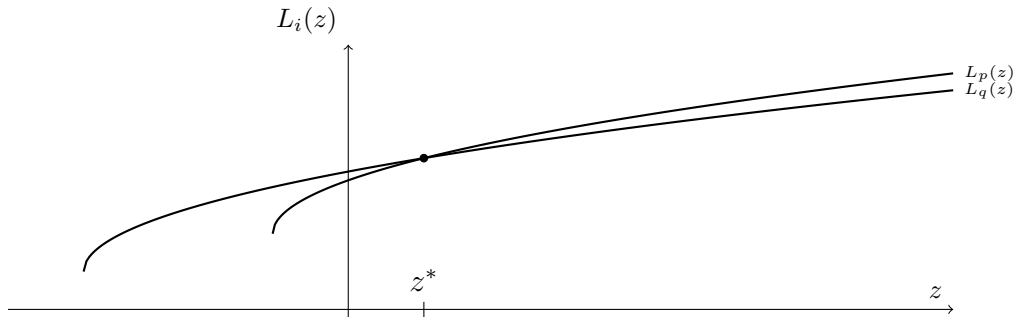

Figure 4.2: Label functions.

We consider the behavior of the two functions L_p and L_q with $L_i(z) : \mathbb{R}_{[-A_i,\infty]} \to \mathbb{R}_+$ and

$$L_i(z) = C_i + \sqrt{A_i + z},$$

for $i = p, q$, which we refer to as the *label function* of the path i (see Figure 4.2).

Obviously, only the non-negative domain is interesting due to the assumption $a \geq 0$ and since z is modeling the sum of variances.

Label functions are normal square root functions beginning at $(-A_i, C_i)$, for $i = p, q$. Two such functions intersect with each other in the positive domain exactly once at the point z^*, due to the condition (4.2). Indeed, if the label functions beginning at $(-A_p, C_p)$ and $(-A_q, C_q)$ intersect in the negative domain, then

$$C_p < C_q \text{ and } C_p + \sqrt{A_p} < C_q + \sqrt{A_q} \tag{4.4}$$

holds. If they do not intersect, then

$$C_p \geq C_q \text{ and } C_p + \sqrt{A_p} > C_q + \sqrt{A_q} \tag{4.5}$$

holds. Both cases imply that one path dominates the other one with respect to Definition 4.6.

Clearly, for $z \in [0, z^*)$ the path q yields a better objective value than p, and for $z \in (z^*, \infty)$ vice versa. We call the corresponding pointwise minimum function, i.e. the function $D_v : [0, \infty) \to [0, \infty)$, with

$$D_v(z) = \min_{p \in X_{sv}} L_p(z),$$

the *dominance function of the set of subpaths in* v.

One possibility to exclude some non-dominated subpaths from further consideration is to find bounds a_{min} and a_{max} on the possible variance values of the upcoming $v-t$-paths, such that the intersection point is outside the corresponding interval $[a_{min}, a_{max}]$. That would mean that one path has a better objective value

on the whole *interesting domain* and there are no $v-t$ subpaths which would improve the value compared to the other subpaths *within the given graph*. Simple bounds are given for example by

$$a_{min} = \min_{x \in X_{vt}} a^\top x,$$

and

$$a_{max} = a^\top x_c \text{ with } x_c = \arg\min_{x \in X_{vt}} c^\top x.$$

It remains a subject of further research to find stronger bounds and to investigate whether these lead to a polynomial number of relevant labels.

The consideration of the dominance functions leads to a further interesting observation. Let $p_1, \ldots, p_k$ be the non-dominated $s-v$-paths, $(C_{p_1}, A_{p_1}), \ldots, (C_{p_k}, A_{p_k})$ the corresponding labels and $C_{p_1} + \sqrt{A_{p_1}}, \ldots, C_{p_k} + \sqrt{A_{p_k}}$ the corresponding objective function values. As well, consider the non-dominated $v-t$-paths $q_1, \ldots, q_l$. Composing the path p_i with the path q_j leads to the label $(C_{p_i} + C_{q_j}, A_{p_i} + A_{q_j})$ in the node t. The corresponding label function $L_{p_i+q_j}$ results from moving the graph of the label function L_{p_i} by the vector $(-A_{q_j}, C_{q_j})^\top$. The same translation by $(-A_{q_j}, C_{q_j})^\top$ occurs while composing every subpath $p_i, i = 1, \ldots k$, with a fixed subpath q_j, such that the whole dominance function of the set of paths $\{p_1 + q_j, \ldots, p_k + q_j\}$ is the dominance function of the set of paths $\{p_1, \ldots, p_k\}$ translated by $(-A_{q_j}, C_{q_j})^\top$. That means that for every set of paths $\{p_1 + q_j, \ldots, p_k + q_j\}$, $j = 1, \ldots, l$, we obtain the same graphs of the dominance functions, merely translated by different vectors and consisting of k pieces each. In particular, these have the same break points and the same slopes.

This observation is the base of the following property:

Theorem 4.7. *Let k be the number of non-dominated $s-v$-paths and l the number of non-dominated $v-t$-paths. The number of non-dominated $s-t$-paths containing v is not greater than $k+l-1$.*

Proof. Let $\tilde{D}_v \subseteq \mathbb{R}^2$ be the graph of the dominance function D_v of the k non-dominated $s-v$-paths in the node v. It consists of exactly k sections and has $k-1$ break points. Furthermore, it is concave and monotonous as the point-wise minimum of k translated square-root functions of the form $C_i + \sqrt{A_i + z}$.

We now add the l non-dominated $v-t$-paths and consider the dominance function D_t of the set of $s-t$-paths containing v in the node t. The graph $\tilde{D}_t \subseteq \mathbb{R}^2$ results from $\tilde{D}_v$ by translating the graph $\tilde{D}_v$ l-times and considering the new point-wise minimum. Two identical but translated concave monotonous functions intersect at most once or coincide (have infinitely many intersection points, but this case can be excluded from consideration since we are interested in the non-dominated paths in the sense of Definition 4.6). This leads to at most $\binom{l}{2} = \frac{l(l-1)}{2}$

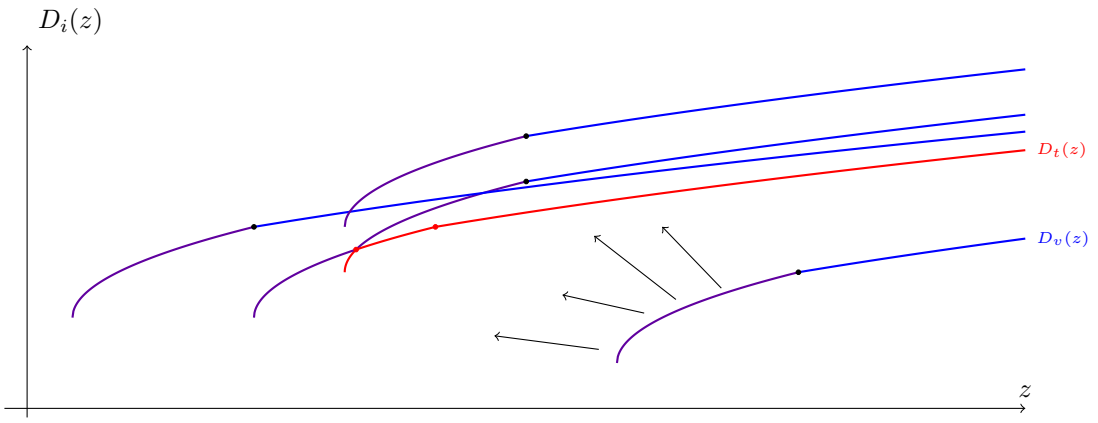

Figure 4.3: Formation of the $s-v-t$-dominance function.

intersection points. All but $l-1$ of these are dominated by the new minimum function, i.e. only $l-1$ of these define a section of $\tilde{D}_t$ (see Figure 4.3).

Thus, the arising dominance function has at most $k+l-1$ sections, which correspond to a non-dominated $s-t$-path containing v each. □

The bound in Theorem 4.7 is a significant reduction compared with the number kl of the possible $s-t$-paths consisting of the k non-dominated $s-v$- and l non-dominated $v-t$-subpaths.

Due to the non-negativity of the variance values, the dominance graph is always shifted to the left when adding subpaths. That means that in the course of the procedure the interesting domain shrinks, which causes exclusion of paths.

With the property of Theorem 4.7 polynomial solvability of the reliable Shortest Path-problem may result for certain graph structures. This is the case for the series-parallel graphs (see Definition 4.1). The bound $m-n+2$ on the number of non-dominated paths in the destination node in a graph with n nodes and m edges can easily be shown. This is the same bound provided in [24] (see Theorem 4.2) on the number of the extreme efficient solutions.

One may wonder if the set of extreme efficient solutions coincides with the set of non-dominated solutions in the destination node. In the following example we observe that this is not the case.

Example 4.8. Consider the graph in Figure 4.4. Here, using e.g. the break points-approach from Section 2.4.2.2, one can calculate the extreme efficient $s-t$-paths. These are

$$s-b-d-f-h-t \text{ with label } (0,64),$$
$$s-b-c-f-h-t \text{ with label } (1,49),$$
$$s-b-d-e-h-t \text{ with label } (4,36),$$

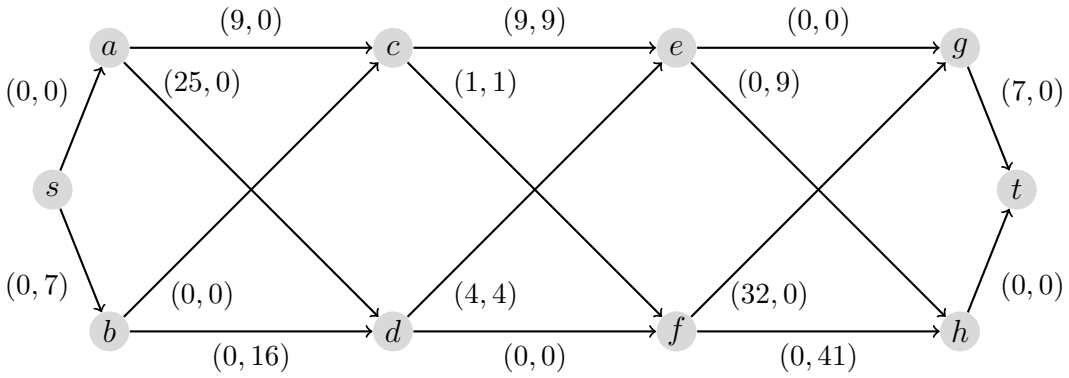

Figure 4.4: The set of extreme efficient solutions differs from the set of non-dominated paths.

$$s-b-c-e-h-t \text{ with label } (9,25),$$

$$s-b-c-e-g-t \text{ with label } (16,16),$$

$$s-a-c-e-g-t \text{ with label } (25,9),$$

$$s-a-d-e-g-t \text{ with label } (36,4),$$

$$s-a-c-f-g-t \text{ with label } (49,1) \text{ and}$$

$$s-a-d-f-g-t \text{ with label } (64,0).$$

But only the path $s-b-d-f-h-t$ with label $(0,64)$ is non-dominated. □

Even though the property in Theorem 4.7 might lead to a proven bound on the number of labels in graphs with some special structures, still, neither the labeling approach is a response to the question about the complexity of the general uncorrelated problem (RSP).

We summarize that the complexity status of the uncorrelated Problem (RSP) is still unknown. There are instances of the parametric Shortest Path-problem with a super-exponential number of break points. There is no proven polynomial upper bound on the number of labels stored using the non-dominance conditions of Definition 4.6. Nevertheless, in general graphs the number of labels in the nodes is now bounded more tightly with respect to the number of labels in the previous nodes. With additional upper bounding of the values of remaining subpaths we expect a significant restriction of the relevant intervals of the corresponding dominance functions. The actual number of labels in the nodes would thus reduce significantly in practice.

We did not evaluate this approach experimentally, instead we consider and evaluate a more general approach for arbitrary combinatorial structures $X \subseteq \{0,1\}^n$.

4.2 General Exact Solution Method

As we have seen, there are robust combinatorial problems with no polynomial algorithm still at hand. Even for matroids the fastest algorithm applicable to Problem (2.17) to the best of our knowledge runs in $O(p^2)$ (if the corresponding linear problem can be solved in time $O(p)$), which is slow for a practical use. Therefore it is desirable to have a faster general tool for both matroids and other combinatorial problems.

There are two aspects of Problem (2.17) to deal with in the search for such a tool – the combinatorial structure of the feasible set and the non-linearity of the objective function resulting from ellipsoidal uncertainty in the cost coefficients. In this regard we implemented the idea to apply *Lagrangean decomposition* [41] to problems with uncertain objective function and combinatorial structure, i.e. to separate the objective from the constraints. The initiative was to address the combinatorial problems with *discrete scenario uncertainty*. However, the basic idea of Lagrangean decomposition is applicable to any other objective provided that the corresponding unconstrained version can be solved in an appropriate time. The Lagrangean decomposition approach fits in so far to the requirements on the tool we are looking for, as it is exact and general, i.e. applicable for any objective function and any type of constraints provided that we can solve the arising subproblems.

We investigated the suitability of the approach for the ellipsoidal uncertainty case. The difficulty was to ensure a quick handling of the unconstrained uncertain subproblem, as at the time we only had at our disposal the general algorithms for submodular function minimization. This was an actual motivation for finding a fast algorithm for the unconstrained uncorrelated problem, which then was developed and introduced in the previous chapter.

In the following we will describe the Lagrangean decomposition approach for uncertain combinatorial problems, provided in [6] in all detail, and specify it for the case of uncorrelated ellipsoidal uncertainty. After that the idea of applying Lagrangean decomposition to the uncorrelated ellipsoidal uncertainty case will be evaluated experimentally. Most results of this section have been published in [9].

4.2.1 Lagrangean Decomposition Approach

For the general Problem (2.17)

$$\begin{array}{ll} \min & c_0^\top x + \sqrt{a^\top x} \\ \text{s.t.} & x \in X \subseteq \{0,1\}^n \end{array}$$

we hold down to have

1. a submodular objective function $c_0^\top x + \sqrt{a^\top x}$, which is known to be minimized efficiently, if there are no additional constraints. Here, however, the feasible set has a combinatorial structure;

2. and the combinatorial domain X, over which we assume the linear minimization to be polynomial. Here, however, it is combined with a non-linear objective function, so that the complexity status in the general case is not known.

There are thus two components, which are easy to deal with, if taken separately, but which might cause a potential difficulty, if considered together. Hence, the idea is to decompose Problem (2.17), i.e. to separate non-linearity from the combinatorial structure.

To this aim in [7] we used Lagrangean decomposition [41], which can be seen as Lagrangean relaxation applied to some artificial constraints coupling two different parts of the model.

Applied to Problem (2.17) the Lagrangean decomposition works as follows. New variables y_i, $i = 1, \dots, n$, are added to the problem formulation and, by means of artificial linking constraints, Problem (2.17) is equivalently formulated as

$$\begin{array}{ll} \min & c_0^\top x + \sqrt{a^\top x} \\ \text{s.t.} & x = y \\ & y \in X \\ & x \in \{0,1\}^n, \end{array} \tag{4.6}$$

with satisfying of the combinatorial constraints now being the task of the variables y. In this formulation the variables $x_i, i = 1, \dots, n$, can be required to stem from any superset of X and the problem remains equivalent to (2.17). We will see that in the context of ellipsoidal uncertainty the requirement $x \in \{0,1\}^n$ is most suitable. Now we build the Lagrangean relaxation applied to the artificial constraints $x = y$, using Lagrangean multipliers $\lambda \in \mathbb{R}^n$, which results in

$$\begin{array}{ll} \min & c_0^\top x + \sqrt{a^\top x} + \lambda^\top (y - x) \\ \text{s.t.} & y \in X \\ & x \in \{0,1\}^n. \end{array} \tag{4.7}$$

According to the Lagrangean theory, this problem yields a *lower bound* on Problem (2.17) for every $\lambda \in \mathbb{R}^n$. Indeed, for an arbitrary function f and an arbitrary $\lambda \in \mathbb{R}^n$ we have:

$$\begin{array}{lll} \min & f(x) & \\ \text{s.t.} & Ax = b & \\ & x \in X & \end{array} \quad = \quad \begin{array}{lll} \min & f(x) + \lambda^\top (Ax - b) & \\ \text{s.t.} & Ax = b & \\ & x \in X & \end{array} \quad \geq \quad \begin{array}{lll} \min & f(x) + \lambda^\top (Ax - b) & \\ \text{s.t.} & x \in X. & \end{array}$$

In Problem (4.7) the variables x and y are not connected with each other any more – neither in the objective function, nor in the constraints, so the problem decomposes to

$$\begin{array}{lll} \min & c_0^\top x + \sqrt{a^\top x} - \lambda^\top x & + \quad \min \quad \lambda^\top y \\ \text{s.t.} & x \in \{0,1\}^n & \text{s.t.} \quad y \in X\,. \end{array} \tag{$LD(\lambda)$}$$

Both subproblems in $(LD(\lambda))$ can be minimized independently and have a beneficial nature. The left-hand side problem

$$\begin{array}{ll} \min & (c_0 - \lambda)^\top x + \sqrt{a^\top x} \\ \text{s.t.} & x \in \{0,1\}^n \end{array} \tag{$LD_{mr}(\lambda)$}$$

is the familiar unconstrained binary mean-risk problem which is quickly solved in time $\mathrm{O}(n \log n)$ by Algorithm 1 (see Section 3.1). The right-hand side problem

$$\begin{array}{ll} \min & \lambda^\top y \\ \text{s.t.} & y \in X \end{array} \tag{$LD_{lin}(\lambda)$}$$

is the corresponding linear instance of the original robust problem for which we assumed to have a polynomial oracle. We will see in the next subsection that every homogeneous inequality which is valid for the objective function coefficients in (2.17), such as non-negativity or triangle inequality, can be assumed to be valid for the objective function of $(LD_{lin}(\lambda))$ as well. This ensures the applicability of the given oracle to the linear instance of (2.17) with the new cost coefficients λ.

All in all, by solving $(LD(\lambda))$ we obtain on the one hand a *primal bound* on Problem (2.17), as a side benefit: An optimal solution of the linear Problem $(LD_{lin}(\lambda))$ is feasible for Problem (2.17) and, evaluated in the original objective function, it yields an upper bound. On the other hand we obtain a *dual bound,* since $(LD(\lambda))$ is a relaxation of (2.17) and induces a lower bound on it, for every $\lambda \in \mathbb{R}^n$. These bounds are used in a *branch and bound-algorithm for Problem* (2.17)*,* which we now specify in more detail.

To get the tightest possible dual bound for (2.17) from the Lagrangean decomposition we compute the *Lagrangean dual*

$$\max_{\lambda \in \mathbb{R}^n} LD(\lambda). \tag{L-Dual}$$

Being a pointwise infimum of affine functions in λ, the objective function $LD(\lambda)$ of (L-Dual) is concave and the maximization can be done for example with the *subgradient method* [17]. In each iteration of the subgradient method computing a search direction is easily done:

Lemma 4.9. *[7]: For a given point $\bar{\lambda} \in \mathbb{R}^n$ let x^* and y^* be minimizers of $(LD_{mr}(\bar{\lambda}))$ and $(LD_{lin}(\bar{\lambda}))$, respectively. Then $y^* - x^*$ is a supergradient of $(LD(\lambda))$ in $\bar{\lambda}$.*

Proof. Let $\lambda \in \mathbb{R}^n$. By definition of $(LD(\lambda))$ we have

$$\begin{aligned} LD(\bar{\lambda}) + (y^* - x^*)^\top(\lambda - \bar{\lambda}) &= LD(\bar{\lambda}) - \bar{\lambda}^\top(y^* - x^*) + \lambda^\top(y^* - x^*) \\ &= c_0^\top x^* + \sqrt{a^\top x^*} + \lambda^\top(y^* - x^*) \\ &\geq LD(\lambda). \end{aligned}$$

The last inequality holds since the tuple (x^*, y^*) is feasible for Problem $(LD(\lambda))$ defined by λ. □

Hence, the two subproblems of $(LD(\lambda))$ have to be solved in each iteration of the subgradient method for a given λ. Here we can see the huge benefit of a fast algorithm for the unrestricted binary mean-risk problem (2.21). The high efficiency is crucial in regards to repeated solving of a slightly modified Problem $(LD_{mr}(\lambda))$ in the course of the subgradient method embedded into a branch and bound-method.

Once the optimal $\lambda \in \mathbb{R}^n$ in a given node of the branch and bound-tree is found, we have an upper and a lower bounds for this node. If the node is still not pruned, the feasible solution given by problem $(LD_{lin}(\lambda))$ will in general differ from the solution of $(LD_{mr}(\lambda))$ in the given node. This suggests a very natural *branching rule* of choosing the *variable in which both solutions differ*.

Within a branch and bound-scheme, we can significantly improve the approach by means of *warmstart* and *reoptimization*. A good choice of the initial multipliers λ is crucial in order to compute the Lagrangean dual (L-Dual) fast. This choice should depend on the objective function, i.e. on the considered uncertainty set $\mathcal{U}$. A good starting guess in the case of ellipsoidal uncertainty could presumably be $\lambda = c_0$, i.e. the center of the ellipsoid. In the following nodes of the branch and bound-tree, the optimal multipliers of the parent node can be used to warmstart the subgradient algorithm.

The whole procedure is stated in Algorithm 3. At the beginning, the feasibility of the given problem is tested and in the affirmative case, a feasible solution and an upper bound are stored. Then we can see the essential structure of the algorithm – a while-loop of the subgradient method is embedded into a while-loop of a branch and bound-algorithm.

Embedding the computation of the Lagrangean dual into a branch and bound-routine, the problem to be solved in the root node of the branch and bound-tree is (L-Dual). In deeper levels of the tree, however, we have to respect variable fixings. This means that the algorithms for both parts of the decomposition have to be adapted to handle fixed variables. For the subproblem $(LD_{mr}(\lambda))$

Algorithm 3 Branch and bound-algorithm for combinatorial problems under uncorrelated ellipsoidal uncertainty

Input: Set $X \subseteq \{0,1\}^n$, linear optimization oracle for X, $c_0 \in \mathbb{R}^n, a \in \mathbb{R}^n_+$
Output: Vector $x^* \in X$ minimizing $c_0^\top x + \sqrt{a^\top x} =: f(x)$
Initialization: Let $\lambda := c_0$, $List := \{LD(\lambda_0)\}$, $ub := \infty$

Solve $LD_{lin}(\lambda)$ using *linear oracle* $\rightarrow x_0$
if $LD_{lin}(\lambda)$ infeasible **then**
  break
  return $infeasible$!
else
  $x^* := x_0$, $ub := f(x_0)$
end if

while $List \neq \emptyset$ **do**
  choose $P(\lambda) \in List$, $List := List \setminus P(\lambda)$
  solve $LD^P_{mr}(\lambda)$ using *Algorithm* 1 $\rightarrow x^*_1$ ▷ the left-hand part of $P(\lambda)$
  solve $LD^P_{lin}(\lambda)$ using *linear oracle* $\rightarrow x^*_2$ ▷ the right-hand part of $P(\lambda)$
  let $lb := c_0^\top x^*_1 + \sqrt{a^\top x^*_1} + \lambda\,(x^*_2 - x^*_1)$ ▷ lower bound on the current node
  if $LD^P_{lin}(\lambda)$ not feasible **or** $lb \geq ub$ **then**
    prune, continue
  else if $f(x^*_2) < ub$ **then**
    $x^* := x^*_2$, $ub := f(x^*_2)$
  end if
  $s := x^*_2 - x^*_1$ ▷ supergradient

  while $s \neq 0$ **do**
    update λ using *subgradient method* and supergradient s
    solve $LD^P_{mr}(\lambda)$ using *Algorithm* 1 $\rightarrow x^*_1$
    solve $LD^P_{lin}(\lambda)$ using *linear oracle* $\rightarrow x^*_2$
    $s := x^*_2 - x^*_1$
    if $f(x^*_2) \leq ub$ **then**
      $x^* := x^*_2$, $ub := f(x^*_2)$
    end if
  end while

  let $lb := c_0^\top x^*_1 + \sqrt{a^\top x^*_1} + \lambda\,(x^*_2 - x^*_1)$
  if $lb \geq ub$ **then**
    prune, continue
  end if
  choose i with $x^*_{1i} \neq x^*_{2i}$ ▷ branching
  let $List := List \cup \left\{LD^P_{i0}(\lambda), LD^P_{i1}(\lambda)\right\}$
end while
return x^*, $f(x^*)$

additional fixings do not cause any problem, as we have seen in Section 3, and do not affect the running time of the combinatorial sorting algorithm. For the subproblem ($LD_{lin}(\lambda)$) adaptation of the used algorithm is also straight-forward in some cases.

For some applications, however, adapting combinatorial algorithms is more complicated and causes sometimes the necessity to use another appropriate algorithm, which can handle fixed variables. Consider, for example, the Shortest Path-problem. The most common approach to its linear non-negative version is Dijkstra's algorithm [31]. We do not know any possible modification of this algorithm to handle fixed edges. However, for $\lambda \geq 0$ we have the following integer linear formulation of the standard Shortest Path-problem:

$$\begin{aligned} \min \quad & \lambda^\top x \\ \text{s.t.} \quad & \textstyle\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = \begin{cases} 1, & \text{if} \quad i = s \\ -1, & \text{if} \quad i = t \\ 0, & \text{otherwise} \end{cases} \quad \forall\, i \in V \\ & x \in \{0,1\}^n . \end{aligned} \tag{4.8}$$

The constraint matrix is *totally unimodular*, such that the explicit requirement of integrality can be dropped in this formulation. Fixing certain variables to 0 or 1 merely causes an extension of the constraint matrix with the corresponding rows of the identity matrix and the corresponding rows of the negative identity matrix. The submatrices of such an extension are contained in the set of submatrices of the extension with a whole identity or a negative identity matrix. We know that extending a total unimodular matrix with a whole identity or a negative identity matrix does not affect total unimodularity, such that the submatrices of our extension have also the determinants -1, 0 or 1. Thus per definition of total unimodularity fixing variables does not affect total unimodularity and Problem (4.8) can be solved efficiently.

However, Problem (4.8) with fixed variables is no longer equivalent to the Shortest Path-problem with the same fixed variables. In particular, its solution might contain loops or separate circles, i.e. be not feasible for the Shortest Path-problem. Nevertheless, the feasible set of the Shortest Path-problem is always contained in the feasible set of Problem (4.8) with the corresponding fixings, such that solving Problem (4.8) yields a lower bound on the right-hand side subproblem of the decomposition, which is sufficient to obtain a lower bound from the decomposed problem.

Hence the idea is to replace the set X through the feasible set of Problem (4.8), since we can assume a non-negative λ, as we will see in the next section. For this special case of the *Minimum Cost Flow-problem* there exists a particularly efficient kind of simplex algorithm, the *network simplex algorithm* [49].

4.2.2 Theoretical Properties

The Lagrangean decomposition approach is one possibility to solve Problem (2.17). For this type of problems no general polynomial algorithm is known, although we still could not find any problem tractable in linear case that would become proven $\mathcal{NP}$-hard, if considered under uncorrelated ellipsoidal uncertainty. Nevertheless, the theoretical benefits of the non-polynomial branch and bound-approach described in the previous section, as well as its particularities and the difference compared to the usual Lagrangean relaxation or common convexifying of the feasible set, should be discussed.

For a start, in comparison to standard relaxations like the normal Lagrangean relaxation or convexifying of the feasible set, the Lagrangean decomposition approach does not require any polyhedral description of the feasible set X or of its convex hull. This is a big advantage since for many considered sets X such a description does not exist or is exponentially large, like for the *Minimum Spanning Tree-problem*.

The feasible set X in this approach is treated by using an existing oracle for the linear version. The oracle, however, is applied to the linear problem $(LD_{lin}(\lambda))$ with special coefficients λ, which raises the question whether the problem properties required for the functioning of the oracle are maintained also with these coefficients. In the following we argue that any homogeneous inequality that is valid for all scenarios $c \in \mathcal{U}$ can be assumed to be valid also for each λ defining the objective function of the right-hand side $(LD_{lin}(\lambda))$ of the decomposition.

The crucial point for this observation is that due to the following theorem the equality $x = y$ in the formulation of Problem (4.6) can be replaced by the restriction $x - y \in \text{cone}(\mathcal{U})$ which is a much weaker requirement, and the arising problem remains equivalent to Problem (2.17).

Theorem 4.10. *Let cone$(\mathcal{U})$ denote the closed convex cone generated by $\mathcal{U}$ and let cone$(\mathcal{U})^*$ be its dual cone. Then*

$$\begin{array}{lll} \min & \max\limits_{c\in\mathcal{U}} c^\top x \\ s.t. & x \in X \end{array} = \begin{array}{lll} \min & \max\limits_{c\in\mathcal{U}} c^\top x \\ s.t. & x \in \{0,1\}^n \\ & y \in X \\ & x - y \in \mathit{cone}(\mathcal{U})^*. \end{array}$$

Proof. By definition of a dual cone we have $x - y \in \text{cone}(\mathcal{U})^*$ if and only if $d^\top(x-y) \geq 0$ for all $d \in \mathcal{U}$, i.e. $d^\top y \leq d^\top x$ for all $d \in \mathcal{U}$ for every feasible solution (x, y) of the right-hand side problem. In particular,

$$\max_{c\in\mathcal{U}} c^\top x \geq \max_{c\in\mathcal{U}} c^\top y$$

is always true, and due to the minimization sense we can even assume equality in the optimal solution (x, y). Hence, replacing $\max_{c\in\mathcal{U}} c^\top x$ by $\max_{c\in\mathcal{U}} c^\top y$, we

obtain the equivalent problem

$$\begin{array}{rl} \min & \max\limits_{c\in\mathcal{U}} c^\top y \\ \text{s.t.} & x \in \{0,1\}^n \\ & y \in X \\ & x - y \in \operatorname{cone}(\mathcal{U})^*. \end{array} \tag{4.9}$$

As for x only binarity is required and since $X \subseteq \{0,1\}^n$, Problem (4.9) can be solved by first choosing $y \in X$ that minimizes $\max_{c\in\mathcal{U}} c^\top y$, i.e. solving

$$\begin{array}{rl} \min & \max\limits_{c\in\mathcal{U}} c^\top y \\ \text{s.t.} & y \in X, \end{array}$$

and then setting $x := y$, as $0 \in \operatorname{cone}(\mathcal{U})^*$ is always true. This shows the equivalence of the latter problem to

$$\begin{array}{rl} \min & \max\limits_{c\in\mathcal{U}} c^\top x \\ \text{s.t.} & x \in X \end{array}$$

and finally proves the statement. □

The Lagrangean relaxation approach can thus be directly applied to the problem

$$\begin{array}{rl} \min & \max\limits_{c\in\mathcal{U}} c^\top x \\ \text{s.t.} & x \in \{0,1\}^n \\ & y \in X \\ & x - y \in \operatorname{cone}(\mathcal{U})^*, \end{array}$$

i.e. the dual multipliers have to be chosen from $\operatorname{cone}(\mathcal{U})$ instead of $\mathbb{R}^n$ which is a dual cone to $\{0\}$. Therefore, all the assumptions on $\operatorname{cone}(\mathcal{U})$ can now be assumed for λ. The remaining question is whether this restriction on λ yields the same bound as (L-Dual) does. As we now maximize over a smaller set the quality of the dual bound might be affected. Nevertheless, the following considerations show that the bound remains the same.

Lemma 4.11. *Let $\mathcal{C} \subseteq \mathbb{R}^n$ be any closed convex cone and assume that the problem*

$$\begin{array}{rl} \min & z \\ \textit{s.t.} & (z,x) \in \mathit{conv}(F) \\ & y \in \mathit{conv}(X) \\ & x - y \in \mathcal{C}^*, \end{array} \tag{4.10}$$

with $F := \{(z,x) \mid x \in \{0,1\}^n, z \geq \max_{c\in\mathcal{U}} c^\top x\}$, satisfies Slater's condition. Then (4.10) *agrees with* $\max_{\lambda\in\mathcal{C}} LD(\lambda)$.

Proof. First note that

$$LD(\lambda) = \left\{ \begin{array}{lll} \min & z - \lambda^\top x & \\ \text{s.t.} & (z,x) & \in \text{conv}(F) \end{array} \right. + \left\{ \begin{array}{lll} \min & \lambda^\top y & \\ \text{s.t.} & y & \in \text{conv}(X). \end{array} \right.$$

By construction, $\max_{\lambda \in \mathcal{C}} LD(\lambda)$ is then the partial Lagrangean dual of (4.10) with respect to the last constraints. As this problem is convex, strong duality follows from Slater's condition. □

From Lemma 4.11 we derive, in particular, that the bound we get from (L-Dual) is at least as good as the one we would get from convexifying the set X:

Corollary 4.12.

$$\max_{\lambda \in \mathbb{R}^n} LD(\lambda) = \left\{ \begin{array}{lll} \min & z & \\ \textit{s.t.} & (z,x) & \in \textit{conv}(F) \\ & x & \in \textit{conv}(X). \end{array} \right.$$

Proof. The claim follows from Lemma 4.11 with $\mathcal{C} = \mathbb{R}^n$, for which $C^* = \{0\}$. □

In the following we can finally see that, under mild conditions, we may assume $\lambda \in \text{cone}(\mathcal{U})$ without weakening the Lagrangean bound.

Theorem 4.13. *Assume that* (4.10) *satisfies Slater's condition for* $\mathcal{C} = \textit{cone}(\mathcal{U})$. *Then*

$$\max_{\lambda \in \textit{cone}(\mathcal{U})} LD(\lambda) = \max_{\lambda \in \mathbb{R}^n} LD(\lambda).$$

Proof. In analogy to the proof of Theorem 4.10, one can show that

$$\begin{array}{lrl} \min & z & \\ \text{s.t.} & (z,x) & \in \text{conv}(F) \\ & y & \in \text{conv}(X) \\ & x - y & \in \text{cone}(\mathcal{U})^* \end{array} = \begin{array}{lrl} \min & z & \\ \text{s.t.} & (z,x) & \in \text{conv}(F) \\ & y & \in \text{conv}(X) \\ & x - y & = 0. \end{array}$$

Applying Lemma 4.11 to both sides, the result follows. □

With that we can transfer the validity of all relevant inequalities from the set $\mathcal{U}$ to λ:

Corollary 4.14. *Let* $A \in \mathbb{R}^{m \times n}$, *such that* $Ac \leq 0$ *for all* $c \in \mathcal{U}$. *Then*

$$\max_{A\lambda \leq 0} LD(\lambda) = \max_{\lambda \in \mathbb{R}^n} LD(\lambda).$$

Proof. If $\mathcal{C}$ from Lemma 4.11 is described by finitely many linear inequalities, i.e. $\mathcal{C} = \{A\lambda \leq 0\}$, Problem (4.10) satisfies Slater's condition and the result follows with Theorem 4.13. □

In particular, this shows that any finite set of such conditions as non-negativity or non-positivity of given objective function coefficients, triangle inequality, non-negativity of the total cost of a given cycle (if variables correspond to edges of a graph), valid for every $c \in \mathcal{U}$, can be assumed to be valid for λ. E.g., if no scenario in $\mathcal{U}$ induces a negative cycle for an instance of the Shortest Path-problem, we may assume that the same holds for all λ produced in the course of the subgradient algorithm.

We presented an exact algorithm to solve general combinatorial problems with uncorrelated ellipsoidal uncertainty. In the next section the algorithm is applied to the Shortest Path- and the Knapsack-problem under uncorrelated ellipsoidal uncertainty and evaluated from the practical point of view.

4.2.3 Experiments

The flexibility of the decomposition approach allows to use it in a wide range of applications. We present the results for the *robust Shortest Path-* and *robust Knapsack-problem*. These experiments were run on SUSE Linux, on an Intel Xeon CPU E5-2640 2.6 GHz processor. For the subproblem ($LD_{mr}(\lambda)$) of the decomposition Algorithm 1 for the unconstrained binary mean-risk problem in all applications was directly applicable. To implement the subgradient method the *conic bundle solver* [44] was embedded. We used the mean values c_0 as initial values of the Lagrangean multipliers, to warmstart the subgradient algorithm.

4.2.3.1 Robust Shortest Path-Problem

The first application we consider is the robust Shortest Path-problem (see Section 1.1). We are thus given a graph and two special nodes between which a shortest path with respect to certain costs has to be found. For our experiments we generated $n \times n$ grid graphs, i.e. graphs with $2n(n-1)$ edges, for $n \in \{100, \ldots, 500\}$. In the uncorrelated case each edge of the graph is associated with a mean value c_{0i} and a variance value a_i. We generated the objective function in accordance to [4], i.e. as follows. The coefficients of the ellipsoid center c_0 were randomly chosen from the interval $[0, 100]$. The variances a_i were then determined as squares of randomly chosen numbers in the corresponding intervals $[0, c_{0i}]$. The uncertain part of the objective function was scaled with $r \in \{0.1, 0.5, 1\}$ (see Model (MR)), which is equivalent to multiplication of the variance coefficients a_i with r^2.

To manage fixings arising during the branch and bound-procedure, the subproblem ($LD_{lin}(\lambda)$) of the Lagrangean decomposition was in this case solved with the

network simplex-method [49] (see Section 4.2.1). We produced 10 instances of each size and set a limit of one CPU hour.

We compared the performance of the decomposition approach to the SOCP solver of CPLEX, using version 12.6 [28]. For that the problem was reformulated to a mixed-binary SOCP, as described in Section 2.3.2.

Table 4.1 shows the comparison of the performance of the two approaches on the instances with up to 499000 variables. We compare the number of solved instances (**solved**), the average number of branch and bound-nodes (**nodes**), the average number of iterations of the subgradient method or, for CPLEX, the average number of simplex iterations (**calls**) and the average total time in CPU-seconds (**time/s**) for every size and scaling. All averages are taken only over the solved instances.

The decomposition approach could solve all but two instances within the time limit, while the SOCP solver reached its limit on 500×500 grids. We can see in all performance parameters the strong impact of scaling: Solving instances with a big ellipsoid volume requires more time and branch and bound-nodes than with a small volume. It is noticeable that the number of iterations, i.e. oracle calls in every node, is significantly smaller in the decomposition approach, while CPLEX takes many simplex-iterations to solve the SOCPs in the nodes. This might relate to a good starting guess for the Lagrangean multipliers λ in the first node as well as in the deeper nodes, such that not many iterations remain to reach the optimal λ in every node. Moreover, as we get a lower bound for *every* λ in a node, on the way to an *optimal* λ for this node, we might get a lower bound which allows to prune already *before* the optimal λ in this node has been reached, such that not all iterations of the subgradient method in every node are necessary.

A comparably small total number of nodes indicates a good quality of the lower bounds obtained through Lagrangean decomposition in every node. Combined with strong primal bounds obtained in every iteration by solving the subproblem ($LD_{lin}(\lambda)$), the decomposition approach turns out to be well suited for the robust uncorrelated Shortest Path-problem. It outperforms the CPLEX solver in all sizes and scaling parameters by a factor between 10 and 100.

4.2.3.2 Robust Knapsack-Problem

We consider again the following *Risk-Averse Capital Budgeting Problem* (see Example 2.7):

$$\begin{array}{lrcl} \max & c_0^\top x - \sqrt{\frac{1-\epsilon}{\epsilon} a^\top x} & & \\ \text{s.t.} & w^\top x & \leq & b \\ & x & \in & \{0,1\}^n \ . \end{array} \tag{4.11}$$

Here, in addition we characterize the level of risk the investor is willing to take by a parameter $\epsilon > 0$. The investment choice ensures that with probability $1 - \epsilon$ the portfolio will return at least the value of the optimal solution (see the value-at-risk model in Section 2.2.1.2).

The instances were produced as proposed in [4]. In particular, the mean and variance values for every variable were generated as described in Section 4.2.3.1 and the coefficients w_i of the knapsack constraint are randomly and uniformly distributed numbers from the interval $[0, 100]$. To prevent generation of trivial instances, the capacity was set to $b = \frac{1}{2}\sum_{i=1}^{n} w_i$, as common in the literature. Each instance was solved for the values $\epsilon \in \{0.01, 0.02, 0.03, 0.05, 0.1\}$. Note that in comparison to the Shortest Path-application ellipsoids of a larger volume are considered here. In particular, here we have $r = \sqrt{(1-\epsilon)/\epsilon} > 1$, so that $r \in [3, 10)$ for the given choice of ϵ.

The subproblem ($LD_{lin}(\lambda)$) of the Lagrangean decomposition was solved with the greedy algorithm by Dantzig [29]. In particular, we do not solve the Knapsack-problem exactly, but compute only a lower bound. Doing this, it is easy to see that ($LD(\lambda)$) still provides a lower bound for (4.11). Of each size 10 instances were generated. We state the results for dimensions $n \in \{1000, \ldots, 6000\}$ in Table 4.2.

In the case of the robust Knapsack-problem the decomposition approach could solve all instances without exception with up to 4000 variables. The scaling parameter has only a slight effect here. As in the previous application we observe that only a few calls per node on average were necessary, mostly less than three. This again can be explained by the warmstart-strategy and earlier pruning due to existing bounds. Combined with fast algorithms to solve arising subproblems it leads to a fast overall running time.

Also here we compared the performance with the SOCP solver of CPLEX [28]. The SOCP solver turned out to be not competitive with the decomposition approach, as it could not even solve all instances of dimension 40 within the given time limit of 1 hour. For that reason we do not specify the results of CPLEX.

A more problem specific approach was proposed in [4]. The authors derive additional cutting planes from the submodularity of the objective function and use them to strengthen the SOCP by adding the cutting planes to the relaxation in each node of the enumeration-tree. This leads to better dual bounds and as a consequence faster solution times compared to the pure SOCP approach. But this approach still turns out to be not competitive with the decomposition approach: For instances with only 100 variables it takes on average more than 800 seconds for $\epsilon = 0.01$.

We summarize that the decomposition approach also here strongly outperforms the CPLEX solver, but also more problem-specific solvers.

			decomposition approach				CPLEX SOCP			
n	**vars**	r	**solved**	**nodes**	**calls**	**time/s**	**solved**	**nodes**	**calls**	**time/s**
100	**19800**	0.1	10	4.0	17.4	0.20	10	1.1	6767.8	10.58
		0.5	10	6.8	45.6	0.47	10	44.5	7297.1	14.60
		1	10	7.2	85.0	0.87	10	796.9	11963.0	38.14
200	**79600**	0.1	10	4.0	24.4	1.42	10	2.1	27410.1	101.92
		0.5	10	24.8	176.3	7.97	10	123.7	28982.2	153.07
		1	10	164.2	1165.8	45.30	9	6067.8	116059.3	891.38
300	**179400**	0.1	10	6.4	22.1	5.75	10	2.4	62384.8	399.66
		0.5	10	14.6	101.7	15.76	10	170.6	65024.3	539.21
		1	10	198.0	1390.6	184.63	6	6903.0	134278.5	2179.16
400	**319200**	0.1	10	5.6	25.4	15.37	10	1.9	112126.9	1198.64
		0.5	10	58.4	411.6	102.14	10	347.4	117252.2	1545.56
		1	9	323.2	2296.0	550.06	0	—	—	—
500	**499000**	0.1	10	6.4	26.0	30.04	10	2.3	176940.3	2531.90
		0.5	10	92.2	638.7	257.19	2	42.5	180003.5	2605.49
		1	9	237.2	1704.0	803.39	0	—	—	—

Table 4.1: Results for the Shortest Path-problem with ellipsoidal uncertainty on grid graphs.

n=vars	ε	**solved**	**nodes**	**calls**	**time/s**
1000	0.10	10	11561.6	25059.9	8.88
	0.05	10	11901.0	25946.6	9.04
	0.03	10	18080.6	40439.2	14.13
	0.02	10	17116.4	38528.3	13.74
	0.01	10	18227.4	40036.7	14.09
2000	0.10	10	38648.8	86407.2	62.46
	0.05	10	43502.8	99245.1	71.46
	0.03	10	83618.2	188819.6	134.42
	0.02	10	54335.8	120607.0	88.38
	0.01	10	45562.8	103159.3	73.70
3000	0.10	10	70660.2	158974.0	181.48
	0.05	10	65901.0	147838.1	167.27
	0.03	10	71944.2	164729.6	185.35
	0.02	10	195983.4	446662.6	502.76
	0.01	10	122424.0	281471.0	315.72
4000	0.10	10	77751.4	173648.2	260.22
	0.05	10	74034.2	167908.2	247.60
	0.03	10	134695.0	311877.6	462.83
	0.02	10	134252.4	303120.6	451.51
	0.01	10	158270.2	370843.7	558.05
5000	0.10	10	105903.4	239099.0	459.70
	0.05	9	229644.3	541620.1	1040.87
	0.03	8	210731.2	478897.1	926.99
	0.02	10	211222.2	508668.6	981.95
	0.01	9	288255.7	668599.0	1302.85
6000	0.10	10	214092.4	494438.2	1172.40
	0.05	8	303761.0	701917.5	1663.84
	0.03	9	294969.7	690688.8	1610.91
	0.02	6	271478.7	621214.5	1445.17
	0.01	9	258226.6	611664.8	1491.80

Table 4.2: Results for the Knapsack-problem with ellipsoidal uncertainty.

Part III

Correlated Case

We consider now the general ellipsoidal uncertainty case, i.e. general, not necessarily axis-parallel ellipsoids, described by a positive semidefinite, not necessarily diagonal matrix:

$$\begin{array}{rl} \min & c_0^\top x + \sqrt{x^\top A x} \\ \text{s.t.} & x \in X \ . \end{array}$$

As already known, this problem is $\mathcal{NP}$-hard (see Section 2), which can be seen by reduction of the weakly $\mathcal{NP}$-hard *Subset Sum-problem.*

Beyond that the general case has been investigated to a limited extent so that the basic approach to solve it remains the common algorithms for general mixed-integer SOCPs, as also mentioned in Section 2.

In this part we specify the complexity of Problem (2.16) more precisely, and propose a new exact solution method. Like in the uncorrelated case we first consider the set $X = \{0,1\}^n$ and then the general case $X \subseteq \{0,1\}^n$.

In particular, in Chapter 5 we prove the strong $\mathcal{NP}$-hardness of this problem, already in the case $X = \{0,1\}^n$, and provide a branch and bound-algorithm, where lower bounds are obtained approximating the quadratic term under the square root of the objective function in (2.16) by a separable underestimator.

The idea goes back to Buchheim and Traversi [20], who used separable underestimators for quadratic binary programs. Applied on binary variables the minimization of the separable underestimator reduces to a linear minimization over the same feasible set. We adapted this idea especially to maintain the positivity of the square root argument, which was not demanded in [20].

Later on, we use this approach to address the unrestricted binary problem, generalizing the Lagrangean decomposition-approach from Chapter 4 for arbitrary sets $X \subseteq \{0,1\}^n$, more precisely, to solve the left-hand side subproblem in the decomposition to get lower bounds.

We examined the performance of the approach and present the results in Sections 5.3 and 6.2.

Chapter 5

Unconstrained Correlated Case

We consider the special case of Problem (2.16)

$$\begin{array}{ll} \min & c_0^\top x + \sqrt{x^\top A x} \\ \text{s.t.} & x \in \{0,1\}^n \ , \end{array} \tag{5.1}$$

with $c_0 \in \mathbb{R}^n$ and $A \in \mathcal{S}^n_{++}$. First we observe that the assumption $c_0 < 0$ we made in the uncorrelated case is not applicable to the correlated case without loss of generality: Obviously, the trivial solution $x = (0, \ldots, 0)^\top$ is optimal if $A \geq 0$ and $c_0 \geq 0$. If this is not the case, however, we cannot fix a variable to 0 only if the corresponding value of c_0 is positive, like in the uncorrelated case, as the following example shows.

Example 5.1. Consider Problem (5.1) with

$$c_0 = \begin{pmatrix} 1 \\ -7 \end{pmatrix} \text{ and } A = \begin{pmatrix} 10 & -20 \\ -20 & 41 \end{pmatrix} \in \mathcal{S}^n_{++}.$$

Enumerating all feasible solutions we can see that the optimal solution is $\big(1,1\big)^\top$, which is not 0 in the first component, even though $c_{01} > 0$. □

Thus, due to possible negative covariances in the correlated case, we do not assume $c_0 < 0$ here.

Also we observe that special structures of A obviously exist, for which Problem (5.1) can be solved efficiently and, as the case, may sometimes be reduced to the uncorrelated case. For example, if the number of the correlated variables is fixed to k, the problem reduces to solving 2^k uncorrelated problems. This is due to the fact that the combinatorial algorithm in Section 3.1 is not affected by an additional constant under the square root, so that we can enumerate the 2^k partial solutions and solve the uncorrelated problem 2^k times with the corresponding arising constants, choosing the best solution for (5.1) afterwards.

Also, Problem (5.1) is polynomially solvable if the matrix A consists of many identical blocks of a fixed size, i.e.

$$A = \begin{pmatrix} B & 0 & \dots & 0 \\ 0 & B & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & B \end{pmatrix},$$

with $B \in \mathcal{S}^k_{++}$ for a fixed k. Dziuron [32] has shown that if A and c consist of identical blocks of a fixed or an unfixed size, there exists an optimal solution x^* of (5.1) composed of correspondingly identical parts, i.e.

$$x^* = \begin{pmatrix} x' \\ \vdots \\ x' \end{pmatrix},$$

with $x' \in \{0,1\}^k$. If additionally the block size of A is fixed, the solutions of this form can obviously be enumerated polynomially.

However, if any of such special structures cannot be found, we cannot even expect pseudo-polynomial time, as we show in the next section.

5.1 Complexity

The following complexity result and its proof are taken from our paper [9].

Theorem 5.2. *Problem* (2.16) *is strongly $\mathcal{NP}$-hard, even for $X = \{0,1\}^n$.*

Proof. We use the well known fact that Binary Quadratic Programming is strongly $\mathcal{NP}$-hard, as can be shown by its equivalence to the Maximum Cut-problem [40]. It thus suffices to describe a polynomial reduction from a problem of the form

$$\min_{x \in \{0,1\}^n} \frac{1}{2} x^\top Q x + L^\top x \,, \tag{5.2}$$

where $Q \in \mathbb{Z}^{n \times n}$ is any symmetric matrix and $L \in \mathbb{Z}^n$, to Problem (2.16) with data of polynomial size. First, compute

$$\lambda := \min_{i=1,\dots,n} \Big(|Q_{ii}| - \sum_{j \neq i} |Q_{ij}| \Big) \in \mathbb{Z} \,.$$

Setting $\bar{Q} := Q - 2(\lambda - 1)I \in \mathbb{Z}^{n \times n}$ and $\bar{L} := L + (\lambda - 1)(1, \dots, 1)^\top \in \mathbb{Z}^n$, we have

$$\frac{1}{2} x^\top Q x + L^\top x = \frac{1}{2} x^\top \bar{Q} x + \bar{L}^\top x \quad \forall x \in \{0,1\}^n \,.$$

By construction, the matrix $\bar{Q} - I$ is diagonally dominant, so that $\bar{Q} - I \succeq 0$ and $\bar{Q} \succ 0$. Next, define $c := \bar{L}^\top \bar{L} + 1 \in \mathbb{Z}$. Then the matrix

$$A := \begin{pmatrix} \bar{Q} & \bar{L} \\ \bar{L}^\top & c \end{pmatrix}$$

is integer and each entry has polynomial size in the entries of Q and L. The Schur complement shows that A is positive definite, as

$$c - \bar{L}^\top \bar{Q}^{-1} \bar{L} = 1 + \bar{L}^\top (I - \bar{Q}^{-1}) \bar{L} \geq 1$$

due to $I - \bar{Q}^{-1} \succeq 0$. Now (5.2) agrees with

$$\begin{array}{rlll} -\frac{1}{2}c + & \min & \frac{1}{2} y^\top A y & \\ & \text{s.t.} & y & \in \{0,1\}^{n+1} \\ & & y_{n+1} & = 1\,, \end{array}$$

which can be reduced to solving

$$\begin{array}{lll} \min & \sqrt{y^\top A y} & \\ \text{s.t.} & y & \in \{0,1\}^{n+1} \\ & y_{n+1} & = 1\,. \end{array}$$

Setting $M := \sum_{ij} A_{ij} + 1 \in \mathbb{Z}$, the latter can be rewritten as

$$\begin{array}{rlll} M + & \min & (-M) y_{n+1} + \sqrt{y^\top A y} & \\ & \text{s.t.} & y & \in \{0,1\}^{n+1}\,, \end{array}$$

since $y^\top A y \in \{0, \ldots, M-1\}$ and hence $0 \leq \sqrt{y^\top A y} \leq \sqrt{M-1} \leq M - 1$ for all $y \in \{0,1\}^{n+1}$. The latter problem is of the form (2.16). □

The result of Theorem 5.2 suggests that no efficient or even pseudo-polynomial algorithm is possible for Problem (5.1), unless $\mathcal{NP} = \mathcal{P}$. To still handle this strongly $\mathcal{NP}$-hard problem in practice we thus specify a branch and bound-procedure in the next section.

5.2 Branch and Bound-Algorithm Based on Uncorrelated Underestimators

To deal with the strongly $\mathcal{NP}$-hard Problem (5.1) we first underestimate the quadratic term under the square root by a linear term. The resulting relaxation is tractable due to Theorem 2.14 and we can solve it using the efficient Algorithm 1. The value is then used as a lower bound in a branch and bound-algorithm.

What requirements should the sought underestimator comply with and how can it be determined? In the following we will constructively deduce an underestimator for Problem (5.1). We will describe our line of thought in its development and the adjustments we undertook while facing difficulties caused by the special problem structure.

We start with the specification of the requirements on the underestimator, necessary for the latter usage of the Algorithm 1, in particular, its linearity and the non-negativity of the variables coefficients.

The embedding into a branch and bound-procedure causes fixed variables in the problem definition. The information about the fixings should be used to improve the bounds in the deeper levels of the branch and bound-tree, which results in the need for a *number* of certainly defined underestimators. We formalize these ideas in Section 5.2.1. The next step will be the computation of the defined underestimators. How the computational concerns affect the form of the underestimators is described in Sections 5.2.2 and 5.2.3. In particular, we will motivate a *fixed branching rule* for our branch and bound-algorithm in Section 5.2.2. The calculation of the underestimators, given this branching rule, reduces to solving $n-1$ semidefinite programs (SDPs), for which an *objective function* should be defined. A reasonable choice of the objective function for the SDPs is discussed in Section 5.2.2.

We now describe the approach in all detail.

5.2.1 The Model

We want to *underestimate* the objective function of Problem (5.1), such that the resulting problem is solvable by Algorithm 1. To this aim, for a given $c_0 \in \mathbb{R}^n$ and $A \in \mathcal{S}^n_{++}$ we are looking for a *diagonal* matrix

$$D = \mathrm{Diag}(d), d \in \mathbb{R}^n_+,$$

such that

$$c_0^\top x + \sqrt{x^\top D x} = c_0^\top x + \sqrt{d^\top x} \leq c_0^\top x + \sqrt{x^\top A x} \tag{5.3}$$

holds for all $x \in \{0,1\}^n$. It is hard to restrict this requirement on the binary domain. Therefore, we agree on vectors d with

$$0 \preceq \mathrm{Diag}(d) \preceq A, \tag{5.4}$$

which can be found by solving an SDP. For such d the condition (5.3) is satisfied even for every $x \in \mathbb{R}^n$. The first inequality in the condition (5.4) is necessary to well define the square root argument in (5.3) and the second condition is necessary to ensure the underestimation of the matrix A.

In a branch and bound-algorithm such underestimator can be used in the root node of the branch and bound-tree. But in the lower levels of the tree it is worthwhile

to take into account the information about the arising fixings, to obtain better bounds in the course of the procedure. There the problem to be solved is

$$\min_{x\in\{0,1\}^{n-k}} c^\top \begin{pmatrix} x_{fix} \\ x \end{pmatrix} + \sqrt{\begin{pmatrix} x_{fix} \\ x \end{pmatrix}^\top A \begin{pmatrix} x_{fix} \\ x \end{pmatrix}}, \tag{5.5}$$

where for simplicity of notation we assume the first k variables to be fixed and x_{fix} to be the vector of corresponding fixed binary values. Here, the only quadratic term under the square root is

$$\begin{pmatrix} 0 \\ x \end{pmatrix}^\top A \begin{pmatrix} 0 \\ x \end{pmatrix}.$$

But the mere underestimation of the corresponding part of A might make the whole square root argument negative, as we allow negative covariances.

The idea is to underestimate the *whole* square root argument, i.e. to find a matrix

$$\begin{pmatrix} q & l \\ l^\top & D \end{pmatrix} \in \mathcal{S}_+^{n-k+1}$$

with the diagonal submatrix $D \in \mathbb{R}_+^{(n-k)\times(n-k)}$, $q \geq 0$ and $l \in \mathbb{R}^{n-k}$, such that

$$0 \preceq \begin{pmatrix} q & l \\ l^\top & D \end{pmatrix} \preceq \begin{pmatrix} \begin{pmatrix} x_{fix} \\ 0 \end{pmatrix}^\top A \begin{pmatrix} x_{fix} \\ 0 \end{pmatrix} & \begin{pmatrix} x_{fix} \\ 0 \end{pmatrix}^\top A \\ A^\top \begin{pmatrix} x_{fix} \\ 0 \end{pmatrix} & A_{n-k} \end{pmatrix},$$

where A_{n-k} represents the part of the matrix A corresponding to the unfixed dimensions and

$$\begin{pmatrix} 1 \\ x \end{pmatrix}^\top \begin{pmatrix} q & l \\ l^\top & D \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} \leq \begin{pmatrix} x_{fix} \\ x \end{pmatrix}^\top A \begin{pmatrix} x_{fix} \\ x \end{pmatrix}$$

is valid for all $x \in \{0,1\}^{n-k}$.

Note that such an underestimator incorporates the information about both which variables are fixed and on which values these are fixed. One could assume it to provide good bounds. On the other hand, getting these bounds requires solving an SDP in every single node of the branch and bound-tree, which, even when the dimension of the SDPs decreases, could result in a very big expense.

5.2.2 Fixed Branching

A solution to the difficulty described in the end of the previous section is to keep the underestimators independent from the specific fixings and use *fixed branching*,

i.e. branch on variables in the order of $1, \dots, n$, such that in the level k of the tree exactly the variables $1, \dots, k$ are fixed.

Also the nature of the lower bound is in support of this decision: The lower bound here does not result from the relaxation (enlargement) of the feasible set, but from the underestimation of the objective function. The optimal solution of the underestimating problem is always feasible for the underestimated unconstrained binary Problem (5.1), such that the popular branching criterion of the "highest unfeasibility" looses its sense here.

Thus, for every $k = 1, \dots, n-1$, we compute in a *preprocessing* an underestimator

$$U_k := \begin{pmatrix} Q & L \\ L^\top & D \end{pmatrix} \in \mathcal{S}^n_{++},$$

with $Q \in \mathbb{R}^{k \times k}$, $L \in \mathbb{R}^{k \times (n-k)}$ and a diagonal matrix $D = \operatorname{Diag}(d) \in \mathbb{R}_+^{(n-k)\times(n-k)}$, such that

$$0 \preceq U_k \preceq A.$$

The underestimator U_n consists of n^2 entries to compute. This complies with the dimension $n \times n$ of A, such that $U_n = A$ would be the optimal underestimator for any objective function of the arising SDP.

A lower bound on (5.5) is then the value of the problem

$$\min_{x \in \{0,1\}^{n-k}} c_0^\top \begin{pmatrix} x_{fix} \\ x \end{pmatrix} + \sqrt{\begin{pmatrix} x_{fix} \\ x \end{pmatrix}^\top U_k \begin{pmatrix} x_{fix} \\ x \end{pmatrix}}, \tag{5.6}$$

which is equivalent to

$$c_0^\top \begin{pmatrix} x_{fix} \\ 0 \end{pmatrix} + \min_{x \in \{0,1\}^{n-k}} c_0^\top \begin{pmatrix} 0 \\ x \end{pmatrix} + \sqrt{(d + 2L^\top x_{fix})^\top x + x_{fix}^\top Q x_{fix}}. \tag{5.7}$$

The argument of the square root is positive due to construction. However, the vector $d + 2L^\top x_{fix}$ might contain negative entries, which Algorithm 1 cannot deal with. Still, we can easily handle Problem (5.7) adjusting it slightly for Algorithm 1:

Lemma 5.3. *Problem* (5.7) *can be solved in time* $O((n-k)\log(n-k))$.

Proof. Let

$$I := \left\{ i \in \{k+1, \dots, n\} \mid (d + 2L^\top x_{fix})_i < 0 \right\}$$

be the index set of the variables corresponding to the negative entries of the vector $d + 2L^\top x_{fix}$. For every $i \in I$ we replace the variable x_i by its *complement* $\bar{x_i} := 1 - x_i$ and define the equivalent problem

$$c_0^\top \begin{pmatrix} x_{fix} \\ 0 \end{pmatrix} + \sum_{i \in I} c_{oi} + \min_{x \in \{0,1\}^{n-k}} \sum_{i \notin I} c_{oi} x_i - \sum_{i \in I} c_{oi} \bar{x_i} +$$

$$\sqrt{\sum_{i\notin I}(d+2L^\top x_{fix})_i x_i - \sum_{i\in I}(d+2L^\top x_{fix})_i \bar{x_i} + \sum_{i\in I}(d+2L^\top x_{fix})_i + x_{fix}^\top Q x_{fix}}. \tag{5.8}$$

This problem is of the form (2.21) and the inequalities

$$x_{fix}^\top Q x_{fix} \geq \sum_{i\in I}(d+2L^\top x_{fix})_i + x_{fix}^\top Q x_{fix} \geq 0$$

still hold, in particular, the constant term under the square root is still non-negative. Thus we can apply Algorithm 1 to Problem (5.8), as it can handle non-negative constants under the square root and the variables entries are now non-negative as well. The running time follows with the complexity of the algorithm. □

5.2.3 Objective Function

The choice of the matrices U_k, for $k = 1, \ldots, n-1$, satisfying the requirements defined in the previous section is crucial for the quality of the lower bounds. The best underestimators are induced by the problem

$$\max_{A\succeq U_k\succeq 0} \min_{x\in\{0,1\}^n} c_0^\top x + \sqrt{x^\top U_k x}. \tag{5.9}$$

But due to Theorem 5.2 a single evaluation of the objective function of (5.9) for a fixed U_k is strongly $\mathcal{NP}$-hard. Thus, instead of solving (5.9) exactly, we use a different objective function in the definition of the SDP. We maximize the sum of the diagonal entries of the underestimator, i.e. we solve the SDP

$$\begin{aligned} \max \quad & \mathbb{1}^\top \mathrm{diag}(U_k) \\ \text{s.t.} \quad & A \succeq U_k \succeq 0, \end{aligned} \tag{5.10}$$

with $\mathbb{1}^\top = (1, \ldots, 1)^\top$. This choice of the objective function is motivated by the following observations:

Lemma 5.4. *Let $A \succeq 0$. The optimal solution U^* of*

$$\begin{aligned} \max \quad & \mathbb{1}^\top \mathit{diag}(U) \\ \mathit{s.t.} \quad & A \succeq U \succeq 0 \end{aligned} \tag{5.11}$$

is $U^* = A$.

Proof. We have $\mathrm{diag}(A) \geq 0$, since $A \succeq 0$, and $\mathrm{diag}(U) \leq \mathrm{diag}(A)$, since $A - U \succeq 0$. This implies that the diagonal of the optimal solution U^* is identical to the diagonal of A, since A is a feasible solution of (5.11). Thus, $\mathrm{diag}(A - U^*) = 0$ and due to restriction $A - U^* \succeq 0$ all other entries of $A - U^*$ are 0 as well. □

Due to the following lemma we can expect that for an increasing k the corresponding underestimator U_k is a better approximation of the matrix A and thus we get better bounds with the growing number of fixed variables:

Lemma 5.5. *The optimal value of the problem*

$$\begin{aligned} \max \quad & \mathbb{1}^\top diag \begin{pmatrix} Q & L \\ L^\top & Diag(d) \end{pmatrix} \\ s.t. \quad & A \succeq \begin{pmatrix} Q & L \\ L^\top & Diag(d) \end{pmatrix} \succeq 0 \\ & Q \in \mathbb{R}^{k \times k} \\ & L \in \mathbb{R}^{k \times (n-k)} \\ & d \in \mathbb{R}^{n-k}, d \geq 0 \end{aligned} \tag{5.12}$$

is monotonously increasing in k.

Proof. The diagonal part $\mathrm{Diag}(d) = D$ of the matrix $\begin{pmatrix} Q & L \\ L^\top & D \end{pmatrix}$ shrinks for an increasing k, i.e. less variables are fixed to 0. That means that the number of degrees of freedom in the matrix $\begin{pmatrix} Q & L \\ L^\top & D \end{pmatrix}$ grows. This leads to an extension of the feasible set and the claim follows. □

As the diagonal of the underestimator is upper bounded by the diagonal of A, we expect a better approximation of the matrix A the more variables are fixed. Hence, we solve in a preprocessing phase $n-1$ SDPs of the form

$$\begin{aligned} \max \quad & \mathbb{1}^\top \mathrm{diag} \begin{pmatrix} Q & L \\ L^\top & D \end{pmatrix} \\ \text{s.t.} \quad & A \succeq \begin{pmatrix} Q & L \\ L^\top & D \end{pmatrix} \succ 0 \,, \end{aligned} \tag{5.13}$$

and expect an improvement of the bounds with progressing fixings.

5.3 Experiments

We provided a new branch and bound-approach for the strongly $\mathcal{NP}$-hard Problem (5.1), with arising theoretical questions that should be examined experimentally: How good are the defined bounds and how fast can these be computed? We want now to examine the practical performance of our approach.

We implemented our idea in C++ and run these experiments on an Intel Xeon CPU E5-2640 2.5 GHz processor.

To compute the underestimators for every branching level we solve in a preprocessing phase $n-1$ semidefinite programs using the version 6.1.1 of the *csdp library*

for semidefinite programming [16]. We recall that we use a fixed branching rule, such that the variables are fixed in the order $1, \ldots, n$. Also we adopted the best node first-strategy.

We aimed to produce binary correlated instances following as far as possible the generation routine of the uncorrelated instances in the applications of Chapter 4. The coefficients of the mean c_0 were generated randomly in the intervals $[-100, 0]$, the eigenvalues of the covariance matrix A were generated as squares of the randomly chosen numbers from the corresponding intervals $[0, -c_{0i}]$ and the eigenvectors were then chosen randomly.

We generated 10 instances of each size. The uncertain part of the objective function was scaled with $r \in \{0.1, 0.5, 1\}$. Since we allow both positive and negative covariances, the scaling of the ellipsoid in the unconstrained correlated case usually does not lead to trivial instances, in contrast to the uncorrelated case.

We again examined the correctness and compared the performance of our approach with the SOCP solver of CPLEX. However, CPLEX clearly outperformed the underestimation algorithm, such that we only state our results to detect the improvement potential and to analyze the impact of covariances on the practical complexity of the problem.

Table 5.1 shows the performance of our algorithm on the instances with up to 70 variables. Besides the number of solved instances (**solved**), the average number of branch and bound-nodes (**nodes**) and the average total time in CPU-seconds (**t-total/s**) for every size and type, we broke down the total running time of our program into the average time consumed in preprocessing on solving the SDPs (**t-SDPs/s**) and the average time spent on the actual branch and bound-procedure (**t-BNB/s**).

Our algorithm could solve all but two instances with up to 60 variables within the time limit of one CPU-hour. We see that a big share of the total time is spent on the calculation of the underestimators. Especially for minor dimensions the preprocessing consumes nearly 100% of the total time. On the other hand the dependency of the SDP-time on the scaling parameter r is completely absent. Here only the growing dimension causes the increase of the calculation time. And in fact this growth rate is polynomial in theory, while the number of nodes in an enumeration-tree grows exponentially with the dimension of the problem. That induces that the time needed for the branch and bound-procedure would exceed at some point the time of the preprocessing. Unfortunately, the range of the problem size reached is still too small to observe this effect clearly.

The performance of the branch and bound-procedure in turn is strongly dependent on the ellipsoid volume. Instances with a big r seem to be very hard to solve with our approach. Obviously, with the growing influence of the hard part of the objective function, i.e. the covariances, the gap between the uncorrelated underestimators and the original ellipsoid grows, and so does the number of the

vars	r	solved	nodes	t-total/s	t-SDPs/s	t-BNB/s
10	0.1	10	17.4	0.036	0.036	0.000
	0.5	10	27.8	0.037	0.037	0.000
	1	10	40.8	0.037	0.037	0.000
20	0.1	10	53.4	0.696	0.696	0.000
	0.5	10	213.6	0.683	0.683	0.000
	1	10	583.2	0.692	0.691	0.001
30	0.1	10	120.0	5.281	5.280	0.001
	0.5	10	802.0	5.381	5.376	0.005
	1	10	4143.0	5.498	5.425	0.073
40	0.1	10	251.8	31.576	31.572	0.004
	0.5	10	3476.4	31.879	31.834	0.045
	1	10	28798.8	34.156	31.001	3.155
50	0.1	10	523.4	129.608	129.600	0.008
	0.5	10	17544.0	130.634	129.145	1.489
	1	10	236847.4	596.343	125.561	470.782
60	0.1	10	1113.0	444.649	444.627	0.022
	0.5	10	69890.6	471.168	425.116	46.052
	1	8	306631.8	1156.234	405.046	751.187
70	0.1	10	1491.6	1165.269	1165.228	0.041
	0.5	10	143457.8	1215.433	1154.990	60.443
	1	2	425028.0	2025.815	1170.395	855.420

Table 5.1: Results for the unconstrained correlated problem. Underestimator approach with SDP-tolerance 10^{-8}

scanned nodes. On the other hand a very small value of r leads to the dominance of the linear part of the objective function such that in this special case $X = \{0,1\}^n$ the correlated problem may become trivial.

Hoping for a reduction of the long preprocessing running times we shrink the accuracy of the SDP solver from 10^{-8} to 10^{-2}. Corresponding results can be found in Table 5.2. As expected, we decreased the SDP computing time (by a factor of 2), while the bounds seem to be hardly influenced by this relaxation, as can be observed on the slightly affected number of nodes and the running time for the branch and bound-routine.

Summarizing our observations, for the first, we must conclude that our approach requires a faster computation of the underestimators. Dropping the practically expensive solution of the arising SDPs it is then also conceivable to go back to a flexible branching, exploiting the information about the concrete fixings, to tighten the lower bound in every branch and bound-node. For the second we might possibly need better underestimators, preferably even more problem-specific. All in all our approach shows a high improvement potential in different directions. Still, it is a new strategy to solve the strongly $\mathcal{NP}$-hard Problem (5.1).

vars	r	solved	nodes	t-total/s	t-SDPs/s	t-BNB/s
10	0.1	10	21.6	0.019	0.019	0.000
	0.5	10	32.2	0.017	0.017	0.000
	1	10	46.0	0.017	0.017	0.000
20	0.1	10	57.4	0.325	0.324	0.001
	0.5	10	217.8	0.327	0.327	0.000
	1	10	588.8	0.329	0.325	0.004
30	0.1	10	123.6	2.623	2.623	0.000
	0.5	10	808.8	2.612	2.603	0.009
	1	10	4159.9	2.684	2.612	0.072
40	0.1	10	256.0	15.026	15.024	0.002
	0.5	10	3484.6	15.126	15.078	0.048
	1	10	28820.8	18.461	15.253	3.208
50	0.1	10	529.0	57.560	57.550	0.010
	0.5	10	17612.8	58.942	57.458	1.484
	1	9	163435.4	224.229	56.518	167.711
60	0.1	10	1116.5	191.468	191.446	0.022
	0.5	10	69855.8	236.639	190.046	46.593
	1	8	307909.6	1042.097	194.352	847.745
70	0.1	10	1494.0	496.556	496.517	0.039
	0.5	10	143375.8	558.518	497.284	61.234
	1	3	586354.3	2125.633	568.477	1557.157
80	0.1	10	2175.6	1153.915	1153.842	0.073
	0.5	10	267819.2	1666.189	1291.392	374.797
	1	1	796169.0	3544.360	1172.730	2371.630
90	0.1	10	4490.6	2549.829	2549.648	0.181
	0.5	5	347856.2	2979.470	2467.144	512.326
	1	0	—	—	—	—

Table 5.2: Results for the unconstrained correlated problem. Underestimator approach with SDP-tolerance 10^{-2}

Chapter 6

Exact Approach for Combinatorial Optimization under Ellipsoidal Uncertainty

We now present an exact approach to solve the general Problem (2.16). In the previous chapters we have seen that parts of this problem taken in isolation already can lead to an enormous growth of complexity. In particular, the problem becomes strongly $\mathcal{NP}$-hard when extending from the uncorrelated to the correlated problem. When extending from the unconstrained uncorrelated to the general uncorrelated case, we proceed from proven optimality to uncertainty about polynomial solvability for most combinatorial problems.

Consider for example again the robust Shortest Path-problem, now with correlations between the edges. Even though there are few special cases where the complexity of Problem (2.16) can be bounded, the general result is negative:

Theorem 6.1. *[24] The correlated Problem* (RSP) *is APX-hard [24].*

For the proof we refer to [24].

Nevertheless, there is a positive result on the *weakly correlated* problem (RSP). Nikolova [60] has shown that if there are only correlations between adjacent edge costs, the instance can be polynomially reduced to the uncorrelated case. Chassein et al. [24] introduced a slightly modified reduction. They create in every node a complete bipartite graph with the first partition standing for the incoming edges and the second partition standing for the outgoing edges (see Figure 6.1).

The edges of the bipartite graph in every node get the variances equal to the double corresponding covariances and the expected values equal to 0. It is easy to see that the uncorrelated instance described by the new graph is equivalent to the original weakly correlated instance and the reduction is polynomial (see [24] for the proof). The original reduction of Nikolova [60] can also be extended for

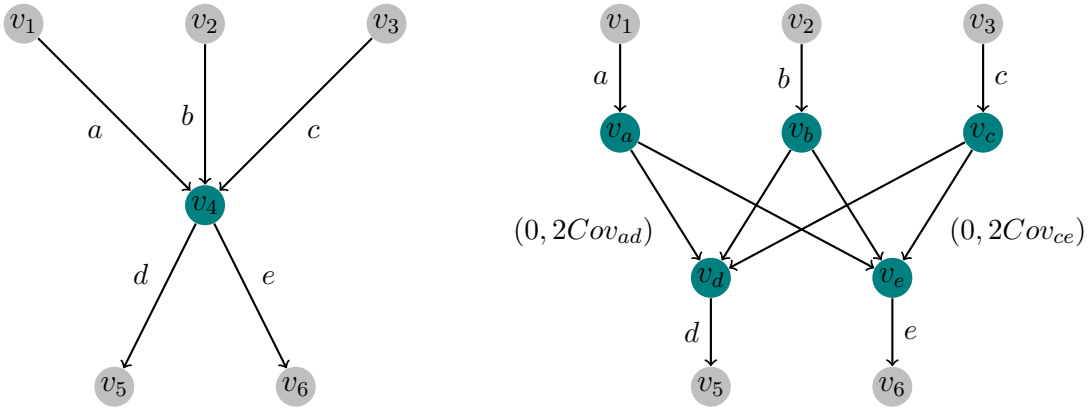

Figure 6.1: Transformation of a weakly-correlated Shortest Path-instance to an uncorrelated instance. Every artificial edge connecting an edge i with an edge j of the bipartite graph gets the variance equal to the double corresponding covariance Cov_{ij}.

correlations between a constant number of consecutive edges, i.e. edges with a constant distance between each other, maintaining polynomiality.

Still the solution approaches for the general Problem (2.16) remain to the best of our knowledge limited to the general mixed-integer SOCP reformulations (see Section 2.3.2).

Here, we combine the procedures presented in previous chapters for the components of the general problem. The sorting Algorithm 1, the Lagrangean decomposition for robust combinatorial problems (Algorithm 3) and the underestimator-approach from Chapter 5 are the building blocks to establish a general procedure for Problem (2.16).

6.1 Underestimation of the Covariance Matrix in the Lagrange-Approach

In analogy to the uncorrelated case (Chapter 4), we decompose Problem (2.16) to

$$\begin{array}{lll} \min & (c_0 - \lambda)^\top x + \sqrt{x^\top A x} & + \quad \min \quad \lambda^\top y \\ \text{s.t.} & x \in \{0,1\}^n & \text{s.t.} \quad y \in X\,. \end{array} \tag{6.1}$$

This decomposition is a lower bound on (2.16) for every λ, which can be computed by separately solving the two corresponding subproblems. We now take advantage of the flexibility of the Lagrangean decomposition-approach, which in principle allows arbitrary constraints and objective functions.

Since for general matrices A the unconstrained problem

$$\begin{array}{ll} \min & (c_0 - \lambda)^\top x + \sqrt{x^\top A x} \\ \text{s.t.} & x \in \{0,1\}^n \end{array} \quad . \tag{6.2}$$

is strongly $\mathcal{NP}$-hard, we abstain from solving it exactly and merely compute a *lower bound* on this subproblem. As we use the solution of the left-hand part of (6.1) only in the computation of a lower bound on (2.16), the overall lower bound, which is the sum of the two parts of (6.1), remains valid if we decrease the left-hand summand for a given λ. So, we underestimate matrix A through a diagonal matrix, to solve the resulting uncorrelated unrestricted problem using Algorithm 1. This may lead to a worse overall lower bound, but the evaluation of the lower bounds in the nodes of the branch and bound-tree becomes significantly faster. Handling the nominal combinatorial problem (the right-hand part of the decomposition) remains the same.

Note that the subgradient method is now applied to the *underestimating* problem

$$\begin{array}{lll} \min & (c_0 - \lambda)^\top x + \sqrt{x^\top U x} & + \quad \min \quad \lambda^\top y \\ \text{s.t.} & x \in \{0,1\}^n & \text{s.t.} \quad y \in X\,. \end{array} \tag{6.3}$$

Therefore, the supergradient $y^* - x^*$ is still valid if x^* and y^* are the optimal solutions of the left-hand and the right-hand subproblems, respectively. This can easily be verified in analogy to Lemma 4.9.

To be able to compute the underestimators in a preprocessing step, like in the unconstrained case, we also here agree on a fixed branching.

6.2 Experiments

We implemented the idea of combining all the building blocks from the previous chapters to solve the general robust Problem (2.16). The Lagrangean decomposition approach exploits the combinatorial structure of the feasible set to solve the right-hand part of the decomposition quickly and obtain good primal bounds. In the unconstrained correlated case our approach did not show an excellent performance. We might be concerned that in the constrained case, where the procedure is even more complicated, the performance might get worse. On the other hand we can reasonably expect no great increase of the running time in the general case compared to the unconstrained case, because we suspect the main difficulty of the general Problem (2.16), also from the practical point of view, to be in the objective function.

We introduce results for the *robust Shortest Path-problem* and the *robust Knapsack-problem*. The set up of the branch and bound-routine remains like in the uncorrelated case (Chapter 4), besides from the fixed branching, which we agreed on for the

n	vars	r	solved	nodes	calls	t-total/s	t-SDPs/s	t-BNB/s
5	**40**	0.1	10	41.4	73.1	29.815	29.809	0.006
		0.5	10	67.2	138.2	31.092	31.080	0.012
		1	10	95.1	239.7	30.459	30.444	0.015
6	**60**	0.1	10	86.0	160.7	417.588	417.574	0.014
		0.5	10	140.8	280.7	427.928	427.908	0.020
		1	10	206.4	502.9	415.364	415.326	0.038
7	**84**	0.1	5	103.6	188.8	3409.558	3409.538	0.020
		0.5	4	227.8	427.0	3401.502	3401.457	0.045
		1	4	377.2	922.0	3356.295	3356.210	0.085

Table 6.1: Results for the reliable Shortest Path-problem on grid graphs. Decomposition approach with SDP-tolerance 10^{-8}. The number of edges is $2(n-1)n$.

underestimator-approach. In particular, also here we apply the warm start-strategy using the mean c_0 as the initial values of the Lagrangean multipliers λ.

All experiments were done on the same processor as in the unconstrained case (Chapter 5). We compared the performance with the SOCP solver of CPLEX, which again overtook our approach.

6.2.1 Robust Shortest Path-Problem

In the case of the reliable Shortest Path-problem we solved the combinatorial deterministic problem in the left-hand part of the decomposition in analogy to the uncorrelated case, i.e. with the network simplex-method [49]. For this application we aimed to generate instances in a way, which is as similar as possible to the generation of the uncorrelated instances in Section 4.2.3, to see the effect of additional correlations. In particular, we produced $n \times n$ grid graphs, where every edge i is associated with a mean c_{0i}. These values were randomly chosen from the interval $[0, 100]$. In analogy to generating variances in the uncorrelated case, we have chosen the eigenvalues of the covariance matrix as squares of randomly chosen numbers from the corresponding intervals $[0, c_{0i}]$. We scaled the ellipsoids with $r \in \{0.1, 0.5, 1\}$ and generated 10 instances of each size and type.

The results are stated in Table 6.1. As in the uncorrelated case we compare the number of solved instances (**solved**), the average number of branch and bound-nodes (**nodes**) and the average number of iterations of the subgradient method or, for CPLEX, the average number of simplex iterations (**calls**). Additionally, as in the unconstrained correlated case, we broke down the total running time of our program into the average time consumed in preprocessing on solving the SDPs (**t-SDPs/s**) and the average time spent on the actual branch and bound-procedure (**t-BNB/s**). All averages are taken only over the solved instances.

Unfortunately, our approach could only reach instances with 60 variables. On all parameters we can observe similar tendencies as in the unconstrained case.

However, the crucial importance of a faster preprocessing, i.e. solving the $n-1$ semidefinite programs while computing the underestimators, is evident here. We see that the preprocessing exhausts the time limit of 3600 CPU seconds for instances with 84 variables, while after the preprocessing the branch and bound-routine is done quickly.

The effect of using different r again can only be observed after the preprocessing, but in all performance parameters. This is explicable by the invariance of the SDP solver towards big parameter values.

It is interesting to observe that for comparable dimensions the branch and bound-routine for the robust Shortest Path-problem takes considerably less time than in the unconstrained case. This could possibly be due to the special structure of the grid graphs, where covariances between many edges are not relevant as they cannot occur together in one path.

Moreover, in the unconstrained case the primal bounds result from the solution of the underestimating problem, while here we obtain reasonable and strong primal bounds from the solution of the right-hand part subproblem of the decomposition.

We conclude here that it is absolutely necessary to compute the underestimators in a different way. The bounds though appear to be strong enough, such that it is imaginable to accept weaker bounds if their computation would take shorter time.

6.2.2 Robust Knapsack-Problem

Like in the unconstrained case we present here also numerical results for the robust Knapsack-problem, i.e. Problem

$$\begin{array}{lrcl} \max & c_0^\top x - \sqrt{\frac{1-\epsilon}{\epsilon} x^\top A x} & & \\ \text{s.t.} & w^\top x & \leq & b \\ & x & \in & \{0,1\}^n \ . \end{array} \tag{6.4}$$

(compare with Problem (4.11) without correlations). We initially produced the instances following the generation of the instances in Section 4.2.3. In particular, the mean values were randomly chosen from the interval $[0, 100]$. The eigenvalues of the covariance matrix were then generated as squares of randomly chosen numbers from the corresponding intervals $[0, c_{0i}]$ and the eigenvectors were chosen randomly.

The deterministic part of the decomposition was solved exactly by dynamic programming, such that for the coefficients w of the knapsack constraint we took randomly chosen integers from the interval $[0, 100]$. The capacity is $b = \frac{1}{2}\sum_{i=1}^{n} w_i$, in analogy to the unconstrained case.

Each instance was solved for the values $\epsilon \in \{0.1, 0.05, 0.03, 0.02, 0.01\}$. We recall that this corresponds to a scaling of the covariance matrix by a factor $(1-\epsilon)/\epsilon$, i.e.

vars	ϵ	solved	nodes	calls	t-total/s	t-SDPs/s	t-BNB/s
10	0.1	10	75.4	225.3	0.043	0.037	0.006
	0.05	10	80.7	252.8	0.043	0.033	0.010
	0.03	10	72.1	249.2	0.041	0.033	0.008
	0.02	10	68.7	252.0	0.041	0.035	0.006
	0.01	10	59.9	233.0	0.043	0.035	0.008
20	0.1	10	2193.4	7478.6	0.797	0.516	0.281
	0.05	10	4449.5	16040.3	1.117	0.514	0.603
	0.03	10	4849.9	18494.0	1.216	0.516	0.700
	0.02	10	4668.8	18781.0	1.225	0.512	0.713
	0.01	10	3868.5	17098.4	1.168	0.516	0.652
30	0.1	10	125414.5	444526.1	190.278	4.637	185.641
	0.05	9	127248.6	496918.8	276.764	4.718	272.047
	0.03	9	131239.6	540277.8	180.999	4.669	176.330
	0.02	10	222580.9	941409.4	414.165	4.655	409.510
	0.01	10	158088.9	697552.6	133.171	4.633	128.538
40	0.1	6	307075.5	1080834.5	483.422	26.232	457.190
	0.05	3	249381.7	998811.0	262.237	26.180	236.057
	0.03	3	516601.3	2236966.7	1150.047	26.253	1123.793
	0.02	2	541996.0	2357665.0	1783.820	26.570	1757.250
	0.01	1	287730.0	1419479.0	275.630	25.140	250.490

Table 6.2: Results for the robust Knapsack-problem. Generation of instances in analogy to the uncorrelated case. Decomposition approach with SDP-tolerance 10^{-7}.

here by the factors 9, 19, 32.3, 49 and 99. As in the uncorrelated case we compare the number of solved instances (**solved**), the average number of branch and bound-nodes (**nodes**) and the average number of iterations of the subgradient method or, for CPLEX, the average number of simplex iterations (**calls**). Additionally, as in the unconstrained correlated case, we broke down the total running time of our program into the average time consumed in preprocessing on solving the SDPs (**t-SDPs/s**) and the average time spent on the actual branch and bound-procedure (**t-BNB/s**). All averages are taken only over the solved instances.

Unfortunately, with this scaling of the ellipsoids our approach did not go as far as we hoped (see Table 6.2).

While the SDP time is not affected by a growing ellipsoid volume, the instances with a big scaling parameter appear to be very hard to handle with the following branch and bound-routine.

Fortunately, in the majority of the conceivable applications we can assume the coefficients to not vary to the extent of up to the expected value c_0. Here, however, we generated the eigenvalues (in analogy to the variances in the uncorrelated case) as squares of randomly chosen numbers in the intervals $[0, c_{0i}]$, which means that the corresponding costs can be allowed to deviate from the expected value up to the expected value. One can argue that the variances in the instances generated

in this way are extremely over-estimated from the practical point of view.

After this consideration we generated new, in our opinion more reasonable instances, by scaling the old covariance matrices by a factor of $\frac{1}{100}$ and used the original values ϵ. This scaling of the matrix is offset against the scaling factor $\frac{1-\epsilon}{\epsilon}$, such that comparable sizes of the ellipsoids are considered as in the application to the robust Shortest Path-problem and the unconstrained binary problem. The factor r in the Model (MR) is in this case $\frac{1}{10}\sqrt{\frac{1-\epsilon}{\epsilon}} \in \{0.3, 0.44, 0.57, 0.7, 0.995\}$. The results for these instances can be found in Table 6.3. Here all instances could be solved up to the dimension 50. In the following dimensions the time limit was consumed by the SDP solver. The effect of a growing ellipsoid volume can be observed in the performance parameters of the branch and bound-algorithm. The total number of branch and bound-nodes is the crucial factor for the long running time. The warmstart-strategy, however, appears to work properly further on, as we can judge looking onto the number of calls per node. For all dimensions and scalings only 2-3 iterations of the subgradient method were necessary to reach a near-optimal λ or to prune. For all performed dimensions the running time consumed is to the largest extent still attributed to the computation of the underestimators in preprocessing.

In both applications the effects of the building blocks of our algorithm can be observed. The tests show that the approach is hardly applicable for extremely large ellipsoid volumes, since even if being able to reduce the preprocessing time, the branch and bound-routine takes a lot of time in these cases. Provided certain improvement strategies mostly concerning the computation of the underestimators the approach is though applicable for a less risk-averse user.

vars	ϵ	solved	nodes	calls	t-total/s	t-SDPs/s	t-BNB/s
10	0.1	10	22.6	45.9	0.037	0.036	0.001
	0.05	10	26.4	57.1	0.037	0.034	0.003
	0.03	10	29.2	65.2	0.037	0.033	0.004
	0.02	10	31.3	76.3	0.036	0.033	0.003
	0.01	10	38.9	95.9	0.040	0.034	0.006
20	0.1	10	172.6	361.8	0.531	0.514	0.017
	0.05	10	254.1	580.7	0.542	0.519	0.023
	0.03	10	348.3	832.3	0.548	0.514	0.034
	0.02	10	486.1	1218.0	0.563	0.516	0.047
	0.01	10	853.9	2294.3	0.603	0.514	0.089
30	0.1	10	280.4	595.7	4.563	4.530	0.033
	0.05	10	580.2	1314.5	4.664	4.589	0.075
	0.03	10	1017.1	2420.0	4.697	4.565	0.132
	0.02	10	1685.4	4235.4	4.804	4.577	0.227
	0.01	10	4237.0	11599.1	5.179	4.564	0.615
40	0.1	10	1393.4	2706.6	25.497	25.285	0.212
	0.05	10	3420.5	7137.3	25.929	25.390	0.539
	0.03	10	6687.9	14790.1	26.524	25.424	1.100
	0.02	10	13968.8	32733.4	27.983	25.477	2.506
	0.01	10	52976.4	134303.7	42.119	25.362	16.757
50	0.1	10	2054.8	4125.2	102.718	102.323	0.395
	0.05	10	5182.6	11211.7	102.562	101.505	1.057
	0.03	10	11061.6	25734.1	103.884	101.419	2.465
	0.02	10	24754.9	61746.4	108.238	101.438	6.800
	0.01	10	100008.5	278295.3	169.212	101.779	67.433
60	0.1	10	19138.9	38023.0	345.971	341.295	4.676
	0.05	10	74432.8	156748.9	369.034	339.306	29.728
	0.03	9	220136.0	488869.1	555.343	344.274	211.069
	0.02	9	393779.0	920658.4	984.951	335.004	649.947
	0.01	4	435189.2	1087465.8	1209.720	326.355	883.365
70	0.1	10	30009.4	58103.1	953.429	942.485	10.944
	0.05	10	124088.7	250347.9	1047.280	939.293	107.987
	0.03	9	205564.7	428071.2	1255.046	1012.863	242.182
	0.02	7	247408.7	543120.6	1214.899	944.690	270.209
	0.01	3	231291.3	586763.3	1094.880	904.943	189.937
80	0.1	10	68662.3	135831.7	2268.158	2223.987	44.171
	0.05	9	163106.1	329809.6	2360.852	2219.807	141.046
	0.03	7	188277.1	401219.0	2342.336	2216.190	126.146
	0.02	6	403808.7	891797.3	2763.785	2290.643	473.142
	0.01	0	-	-	-	-	-

Table 6.3: Results for the robust Knapsack-Problem. Decomposition approach with SDP-tolerance 10^{-7}.

Summary, Conclusions and Outlook

Combinatorial optimization under ellipsoidal uncertainty turns out to be very diverse from theoretical and practical perspectives. We distinguished two main cases in this thesis, namely the uncorrelated and the general correlated case. For the uncorrelated case we gained many different insights and results. We proposed a new combinatorial algorithm for the unconstrained binary case, which we derived from the geometrical view onto the problem. We visualized the diminishing returns-property of submodular functions for this special problem and obtained a sorting rule for variables and an optimality condition. This gave us an efficient $\mathrm{O}(n \log n)$-algorithm, which we then successfully used in a Lagrangean decomposition approach for general subsets $X \subseteq \{0,1\}^n$, i.e. for the constrained uncorrelated case.

The idea of the Lagrangean decomposition approach is to separate the two characteristic aspects of the problem, namely the mean-risk objective function and the combinatorial constraints. This is done through the Lagrangean relaxation of the artificial constraints connecting the two parts and has the advantage that the combinatorial structure is not getting useless through the non-linear objective function, but is effectively used further on. The two isolated aspects manifest themselves in two arising subproblems, one of which is exactly the linear version of the combinatorial problem with the second being the unconstrained binary mean-risk problem, for which the sorting algorithm mentioned above is immediately applicable. The decomposition is embedded into a branch and bound-procedure, which is implemented using the subgradient method to obtain lower bounds in every node. We discussed the theoretical issues of the approach and established the convenient property that any homogeneous inequality, valid for the scenarios from the uncertainty set, can be assumed to be valid for the Lagrangean multipliers as well. The method was tested on two applications and proved itself very fast and suitable, demonstrating by its performance all the theoretical advantages.

We extended the combinatorial algorithm for the unconstrained binary mean-risk problem to general integer variables. The extension is based on the same diminishing returns-property, applied to each value-increase of each variable. As

we look at each value of each variable, the sorting algorithm in this case is only pseudo-polynomial, with the resultant running time depending on the bounds of the variables. Whether there exists a faster algorithm for the unconstrained integer mean-risk problem, remains unclear.

Furthermore, we introduced a new uncertainty set, which was not considered in the literature before, the p-norm-uncertainty. It describes basically the in-between cases within the uncorrelated ellipsoidal and the interval uncertainty. We transformed the objective function using the Karush-Kuhn-Tucker optimality conditions and have shown that after minimal modifications, the sorting algorithm can be easily applied.

Whereas the constrained case of uncorrelated ellipsoidal uncertainty was successfully solved in practice, the theoretical complexity of the reliable Shortest Path-problem remains an open question. We studied several approaches to this problem, in particular a node-labeling approach with some problem-specific non-dominance-conditions. We again used a geometrical illustration to understand the development of the set of non-dominated subpaths in every node. We formalized the dependency of the dominance-capacity on the upcoming subpaths in the forms of a label-function and a dominance-function. This gave us the insight that the actual number of the stored labels in every node is small. In particular, we could determine a bound on the number of labels corresponding to the set of subpaths over a given node.

After an effective examination of the uncorrelated case we turned to the correlated case and considered first the unconstrained case. Initially we gave a tighter lower bound on the complexity of the problem, showing its strong $\mathcal{NP}$-hardness. The method of solving this difficult problem exactly was to develope a new branch and bound-algorithm, where we constructed a linear underestimator on the covariance matrix, i.e. approximate a given general ellipsoid by an axis-parallel ellipsoid, to gain lower bounds. One of the special features of this branch and bound-procedure is a fixed branching, which we use to compute only one underestimator for each level of the enumeration-tree, and not for every node, to reduce computational expenditure. The actual computation of the underestimators is performed by solving the corresponding semidefinite programs.

As with the uncorrelated case, we use the approach to solve the arising unconstrained problem in a Lagrangean decomposition for general combinatorial problems under ellipsoidal uncertainty. With that a new exact method for the strongly $\mathcal{NP}$-hard problem is established.

The practical evaluation of the approach has confirmed that determining the underestimators solving SDPs is not beneficial. Secondly, large ellipsoid volumes affect the performance of the algorithms: for an extreme risk-averse user our approach is not suitable.

During the extensive research on ellipsoidal uncertainty in combinatorial opti-

mization we also gained some beneficial side-results, such as a natural approach for a multicriteria robust optimization. Whilst expanding our knowledge we simultaneously expanded our curiosity, as many interesting and promising directions became evident.

For example, an adaptation of the sorting algorithm for the unconstrained binary mean-risk problem to more general classes of submodular functions is conceivable. The diminishing returns-property, on which the functioning of the algorithm is based, is characteristic for all submodular functions. That means that by abstracting the definition of the contribution functions and their roots, a sorting rule can possibly be found that would promise a running time of $\mathrm{O}(n \log n)$ for minimization of further submodular functions over an unconstrained binary domain, which is worthwhile considering.

Another direction is the improvement of the underestimator approach for general correlated problems. To maintain the good performance of the Lagrangean decomposition approach also in the correlated case, the embedding of a different and a more problem-specific underestimator is required.

Since the center of the underestimating and the underestimated ellipsoids is the same in our approach, we are very limited in getting closer to the relevant part of the ellipsoidal boundary, typical for the robust Problem

$$\begin{array}{ll} \min & \max\limits_{c \in \mathcal{U}} c^\top x \\ \text{s.t.} & x \in X \subseteq \{0,1\}^n \ . \end{array} \tag{6.5}$$

Consider for illustration Figure 6.2. Due to the non-negativity of x, the *worst case*-scenarios of the ellipsoid $\mathcal{U}$ are located in the piece of the ellipsoidal boundary (red) between the optimal solutions of the inner maximization problem of (6.5) over the ellipsoid for fixed *non-negative directions* x. The other points in $\mathcal{U}$ are not relevant for the worst case-minimization since these are always dominated as bad cases by one of the scenarios from this part of the ellipsoidal boundary. If we did not fix the center of the underestimating ellipsoid, the approximation of this relevant part might significantly improve.

Furthermore, with these observations, we are not limited in underestimating the whole ellipsoid. Due to the condition (5.4) the underestimating ellipsoid is completely contained in the original ellipsoid. The only requirement is to approximate the mentioned boundary-section, with the one restriction being to not pass beyond the black borders (see Figure 6.2). Once we managed to compute the underestimators fast, we can move away from the fixed branching and use the information given by concrete fixings.

The most relevant open question is the theoretical complexity of the uncorrelated reliable Shortest Path-problem. Unfortunately, we could not classify this problem, even though we considered it from several different perspectives. In our opinion, most promising appears to be the labeling approach. The derived property in

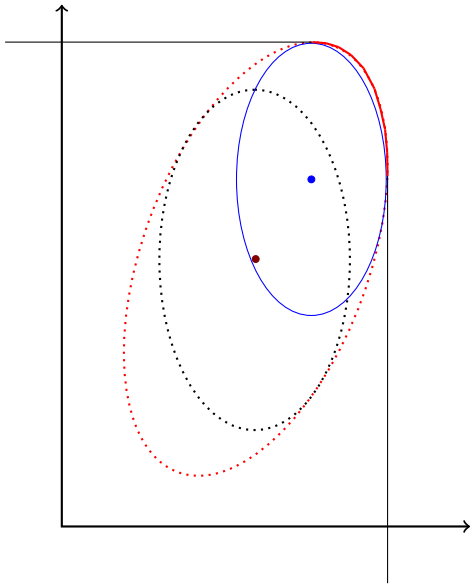

Figure 6.2: Approximating of the worst case subset (red) of the uncertainty set (red, dotted) through a more problem specific underestimator (blue).

connection with reasonably defined bounds on relevant intervals should be tested experimentally, as the number of actually relevant labels is likely to shrink rapidly. Many possibilities for a reasonable choice of the interval bounds are imaginable. Furthermore, other special graph structures should be examined, where the property can possibly induce a polynomial total number of labels. The geometrical illustration, which our results in labeling are based on, can also be helpful to further restrict this number. The next step would be the aggregation of two different dominance functions, i.e. of two functions proceeding from two different nodes. Theoretically the arising aggregated function can summarize all the labels from both nodes. By using additional information about the formation of the corresponding subsets of labels, we can possibly reduce the number of labels.

All in all, the reliable Shortest Path-problem revealed itself as particularly intriguing. Despite its comprehensible form, the theoretical complexity remains unclear, though maybe the mathematical community is closer to the answer. It remains uncertain if and how it will be classified, but what is certain is that the truth is out there.

References

[1] URL `http://www.stromnetzplanung.de`.

[2] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009.

[3] Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical Programming*, 95(1):3–51, 2003.

[4] Alper Atamtürk and Vishnu Narayanan. Polymatroids and mean-risk minimization in discrete optimization. *Operations Research Letters*, 36(5):618–622, 2008.

[5] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer Science & Business Media, 2012.

[6] Frank Baumann. *Exact Methods for Nonlinear Combinatorial Optimization*. PhD thesis, Technische Universität Dortmund, 2014.

[7] Frank Baumann, Sebastian Berckey, and Christoph Buchheim. Exact algorithms for combinatorial optimization problems with submodular objective functions. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization*, pages 271–294. Springer Berlin Heidelberg, 2013.

[8] Frank Baumann, Christoph Buchheim, and Anna Ilyina. Lagrangean decomposition for mean-variance combinatorial optimization. In P. Fouilhoux, E.N. Gouveia, A.R. Mahjoub, and V.T. Paschos, editors, *International Symposium on Combinatorial Optimization – ISCO 2014*, volume 8596 of *Lecture Notes in Computer Science*, pages 62–74. Springer, Cham, 2014.

[9] Frank Baumann, Christoph Buchheim, and Anna Ilyina. A Lagrangean decomposition approach for robust combinatorial optimization. In *Technical Report*. Optimization Online, 2014.

[10] Aharon Ben-Tal and Arkadi Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[11] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[12] Piotr Berman and Toshihiro Fujito. On approximation properties of the independent set problem for degree 3 graphs. In S.G. Akl, F. Dehne, JR. Sack, and N. Santoro, editors, *Workshop on Algorithms and Data Structures*, volume 955 of *Lecture Notes in Computer Science*, pages 449–460. Springer Berlin Heidelberg, 1995.

[13] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1):49–71, September 2003.

[14] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[15] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization under ellipsoidal uncertainty sets. Technical report, MIT, 2004.

[16] Brian Borchers. Csdp, ac library for semidefinite programming. *Optimization Methods and Software*, 11(1-4):613–623, 1999.

[17] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[18] Christoph Buchheim. Robust optimization. Lecture script, Technische Universität Dortmund, 2014.

[19] Christoph Buchheim and Jannis Kurtz. Min-max-min robust combinatorial optimization subject to discrete uncertainty. Technical report, Technische Universität Dortmund, 2016.

[20] Christoph Buchheim and Emiliano Traversi. Separable non-convex underestimators for binary quadratic programming. In V. Bonifaci, C. Demetrescu, and A. Marchetti-Spaccamela, editors, *12th International Symposium on Experimental Algorithms – SEA 2013*, volume 7933 of *Lecture Notes in Computer Science*, pages 236–247. Springer Berlin Heidelberg, 2013.

[21] Kevin Buehler, Andrew Freeman, and Ron Hulme. The risk revolution. *The Strategy: Owning the Right Risks.–Harvard Business Review*, September 2008.

[22] Patricia June Carstensen. *The complexity of some problems in parametric linear and combinatorial programming*. PhD thesis, University of Michigan, 1983.

[23] André Chassein and Marc Goerigk. A bicriteria approach to robust optimization. *Computers & Operations Research*, 66:181–189, 2016.

[24] André Chassein, Michael Hopf, and Marc Goerigk. Reliable and quadratic shortest path problems. Technische Universität Kaiserslautern, Germany, September 2015.

[25] Bi Yu Chen, William H. K. Lam, Agachai Sumalee, and Zhi-lin Li. Reliable shortest path finding in stochastic networks with spatial correlated link travel times. *International Journal of Geographical Information Science*, 26(2): 365–386, 2012.

[26] Serafino Cicerone, Gianlorenzo D'Angelo, Gabriele Di Stefano, Daniele Frigioni, and Alfredo Navarra. On the interaction between robust timetable planning and delay management. In Boting Yang, Ding-Zhu Du, and Cao An Wang, editors, *2nd Annual International Conference on Combinatorial Optimization and Applications (COCOA'08)*, volume 4598 of *Lecture Notes in Computer Science*. Springer, 2007.

[27] Serafino Cicerone, Gianlorenzo D'Angelo, Gabriele Di Stefano, Daniele Frigioni, Alfredo Navarra, Michael Schachtebeck, and Anita Schöbel. Recoverable robustness in shunting and timetabling. *Robust and Online Large-Scale Optimization*, 5868:28–60, 2009.

[28] cplex. IBM ILOG CPLEX optimizer 12.6, 2015. www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[29] George Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[30] Reinhard Diestel. *Graph Theory*, volume 173. Springer-Verlag Berlin and Heidelberg GmbH & amp, 2000.

[31] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[32] Mathias Dziuron. Robust 0-1 optimization under ellipsoidal uncertainty. Master's thesis, Technische Universität Dortmund, February 2016.

[33] Jack Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, 1971.

[34] Matthias Ehrgott. *Multicriteria Optimization*, volume 491. Springer Berlin Heidelberg New York, 2013.

[35] Laurent El Ghaoui and Hervé Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997.

[36] Matteo Fischetti and Michele Monaci. Light robustness. In *Robust and Online Large-Scale Optimization*, pages 61–84. Springer Berlin Heidelberg, 2009.

[37] Otto Forster. *Analysis 1.* Springer-Verlag, 2006.

[38] Michael Friendly, Georges Monette, John Fox, et al. Elliptical insights: understanding statistical methods through elliptical geometry. *Statistical Science*, 28(1):1–39, 2013.

[39] Satoru Fujishige. *Submodular functions and optimization*, volume 58. Elsevier Amsterdam, 2005.

[40] Michael Garey and David Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co Ltd, 1979.

[41] Monique Guignard and Siwhan Kim. Lagrangean decomposition: A model yielding stronger Lagrangean bounds. *Mathematical Programming*, 39(2): 215–228, 1987.

[42] Dan Gusfield. Parametric combinatorial computing and a problem of program module distribution. *Journal of the ACM (JACM)*, 30(3):551–563, 1983.

[43] Grani A. Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage distributionally robust binary programming. *Operations Research Letters*, 44(1):6–11, 2016.

[44] Christoph Helmberg. *ConicBundle 0.3.11.* Fakultät für Mathematik, Technische Universität Chemnitz, 2012. URL `http://www.tu-chemnitz.de/~helmberg/ConicBundle`.

[45] Carsten Homburg and Dipl-Kfm Jörg Stephan. Kennzahlenbasiertes Risikocontrolling in Industrie- und Handelsunternehmen. *Controlling und Management*, 48(5):313–325, 2004.

[46] Roger A. Horn and Charles R. Johnson. *Matrix Analysis.* Cambridge University Press, 2012.

[47] Hans Kellerer, Ulrich Pferschy, and David Pisinger. Knapsack problems. *Journal of the Operational Research Society*, 56(11):1346–1346, 2005.

[48] Bernhard Korte and Jens Vygen. *Combinatorial Optimization.* Springer Berlin Heidelberg, 2002.

[49] Bernhard Korte and Jens Vygen. Combinatorial Optimization. Theory and Algorithms. 2000. page 84, 2005.

[50] Panos Kouvelis and Gang Yu. *Robust Discrete Optimization and Its Applications*, volume 14. Springer Science & Business Media Dordrecht, 2013.

[51] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956.

[52] Jannis Kurtz. *Min-max-min Robust Combinatorial Optimization*. PhD thesis, Technische Universität Dortmund, 2016.

[53] Christian Liebchen, Marco Lübbecke, Rolf Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery and railway applications. In *Robust and Online Large-Scale Optimization*, pages 1–27. Springer Berlin Heidelberg, 2009.

[54] Miguel Sousa Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear Algebra and Its Applications*, 284(1):193–228, 1998.

[55] Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[56] R. Timothy Marler and Jasbir S. Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization*, 41(6):853–862, 2010.

[57] Alessandro Mauro. Price risk management in the energy industry: The value at risk approach. *Proceedings of the XXII Annual International Conference of the International Association for Energy Economics*, June 1999.

[58] George L. Nemhauser, A. H. G. Rinnooy Kan, and Michael J. Todd. Optimization. Vol. 1. *Handbooks in Operations Research and Management Science*, 1989.

[59] Arkadi Nemirovski and Alexander Shapiro. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2006.

[60] Evdokia Nikolova. Approximation algorithms for offline risk-averse combinatorial optimization. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 338–351, 2010.

[61] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Berlin Heidelberg, 2000.

[62] James G. Oxley. *Matroid Theory*, volume 3. Oxford University Press, USA, 2006.

[63] Vilfredo Pareto. *Manuale di Economia Politica*, volume 13. Societa Editrice, 1906.

[64] Georg Ch. Pflug. Some remarks on the value-at-risk and the conditional value-at-risk. In *Probabilistic Constrained Optimization*, pages 272–281. Springer US, 2000.

[65] Andrzej P. Ruszczynski and Alexander Shapiro. *Stochastic Programming*, volume 10. Elsevier Amsterdam, 2003.

[66] Anita Schöbel. Generalized light robustness and the trade-off between robustness and nominal quality. *Mathematical Methods of Operations Research*, pages 1–31, 2014.

[67] Anita Schöbel and Albrecht Kratz. A bicriteria approach for robust timetabling. *Robust and Online Large-Scale Optimization*, 5868:119–144, 2009.

[68] Petra Schuurman and Gerhard J. Woeginger. Approximation schemes-a tutorial. In *Lectures on Scheduling*. Citeseer, 2000.

[69] Zuo-Jun Max Shen, Collette Coullard, and Mark S. Daskin. A joint location-inventory model. *Transportation Science*, 37(1):40–55, 2003.

[70] Allen L. Soyster. Technical note – convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

[71] Peter Stobbe and Andreas Krause. Efficient minimization of decomposable submodular functions. In *Advances in Neural Information Processing Systems 23*, pages 2208–2216. Curran Associates, Inc., 2010.