Min-max-min Robust Combinatorial Optimization

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

Der Fakultät für Mathematik der
Technischen Universität Dortmund
vorgelegt von

Jannis Kurtz

im Juli 2016

Dissertation

Min-max-min Robust Combinatorial Optimization

Fakultät für Mathematik
Technische Universität Dortmund

Erstgutachter: Prof. Dr. Christoph Buchheim

Zweitgutachter: Prof. Dr. Anita Schöbel

Tag der mündlichen Prüfung: 21. Oktober 2016

Abstract

In this thesis we introduce a robust optimization approach which is based on a binary min-max-min problem. The so called *Min-max-min Robust Optimization* extends the classical min-max approach by calculating k different solutions instead of one.

Usually in robust optimization we consider problems whose problem parameters can be uncertain. The basic idea is to define an uncertainty set U which contains all relevant problem parameters, called scenarios. The objective is then to calculate a solution which is feasible for every scenario in U and which optimizes the worst objective value over all scenarios in U.

As a special case of the K-adaptability approach for robust two-stage problems, the min-max-min robust optimization approach aims to calculate k different solutions for the underlying combinatorial problem, such that, considering the best of these solutions in each scenario, the worst objective value over all scenarios is optimized. This idea can be modeled as a min-max-min problem.

In this thesis we analyze the complexity of the afore mentioned problem for convex and for discrete uncertainty sets U. We will show that under further assumptions the problem is as easy as the underlying combinatorial problem for convex uncertainty sets if the number of calculated solutions is greater than the dimension of the problem. Additionally we present a practical exact algorithm to solve the min-max-min problem for any combinatorial problem, given by a deterministic oracle. On the other hand we prove that if we fix the number of solutions k, then the problem is NP-hard even for polyhedral uncertainty sets and the unconstrained binary problem. For the case when the number of calculated solutions is lower or equal to the dimension we present a heuristic algorithm which is based on the exact algorithm above. Both algorithms are tested and analyzed on random instances of the knapsack problem, the vehicle routing problem and the shortest path problem.

For discrete uncertainty sets we show that the min-max-min problem is NP-hard for a selection of combinatorial problems. Nevertheless we prove that it can be solved in pseudopolynomial time or admits an FPTAS if the min-max problem can be solved in pseudopolynomial or admits an FPTAS respectively.

Zusammenfassung

In dieser Arbeit führen wir einen Ansatz in der robusten Optimierung ein, der auf einem binären Min-max-min Problem basiert. Die sogenannte *Min-max-min robuste Optimierung* erweitert die klassische robuste Optimierung, indem k verschiedene Lösungen anstatt einer Einzigen berechnet werden.

Im Allgemeinen betrachtet man in der robusten Optimierung Probleme, deren Problemparameter unsicher sein können. Die Grundidee ist eine Unsicherheitsmenge U zu definieren, die alle relevanten Problemparameter enthält, die man auch Szenarien nennt. Das Ziel ist es eine Lösung zu berechnen, die für alle Szenarien in U zulässig ist und die den schlechtesten Zielfunktionswert bezüglich aller Szenarien in U optimiert.

Als Spezialfall des K-adaptability Ansatzes für zweistufige robuste Probleme, ist es das Ziel des robusten Min-max-min Ansatzes k verschiedene Lösungen für das zugrundeliegende kombinatorische Problem zu berechnen, sodass der schlechteste Fall über alle Szenarien optimiert wird, während wir für jedes Szenario die beste der k Lösungen betrachten. Diese Idee kann als Min-max-min Problem modelliert werden.

In dieser Arbeit untersuchen wir die Komplexität des zuvor beschriebenen Problems für konvexe und für diskrete Unsicherheitsmengen U. Wir zeigen, dass das Problem für konvexe Unsicherheitsmengen unter weiteren Voraussetzungen genau so einfach ist wie das zugrundeliegende kombinatorische Problem, falls die Anzahl der zu berechnenden Lösungen größer als die Dimension des Problems ist. Desweiteren präsentieren wir einen praktischen exakten Algorithmus, der das Min-max-min Problem für alle kombinatorischen Probleme löst, die durch ein Orakel gegeben sind. Andererseits beweisen wir, dass das Problem für eine feste Anzahl von Lösungen *NP*-schwer ist, sogar für polyedrische Unsicherheitsmengen und das unrestringierte binäre Problem. Für den Fall, dass die Anzahl der Lösungen kleiner oder gleich der Dimension des Problem ist, präsentieren wir einen heuristischen Algorithmus, der auf dem zuvor erwähnten exakten Algorithmus basiert. Beide Algorithmen wurden an zufälligen Instanzen des Rucksackproblems, des Vehicle-Routing-Problems und des kürzesten Wege Problems getestet und analysiert.

Für diskrete Unsicherheitsmengen zeigen wir, dass das Min-max-min Problem *NP*-schwer ist für eine Auswahl an kombinatorischen Problemen. Dennoch konnten wir zeigen, dass das Problem in pseudopolynomieller Zeit gelöst werden kann bzw. in polynomieller Zeit beliebig genau approximiert werden kann, falls das zugehörige min-max Problem die jeweilige Eigenschaft hat.

Partial Publications and Collaboration Partners

Most of the results on convex and discrete uncertainty sets in Chapter 4 and 5 were developed together with my supervisor Christoph Buchheim from TU Dortmund University and can be found in [23, 24].

The computations for the vehicle routing problem in Section 6.1.2 were performed in a cooperation with Uwe Clausen and Lars Eufinger from the Institute of Transport Logistics at TU Dortmund University.

Acknowledgements

I would like to thank all my colleagues at the University of Dortmund who I met during my time at the university and who worked together with me for at least a small period. Thank you for helping me with my mathematical problems but also for helping me and giving me advice in all other situations in life. Especially I want to thank my supervisor Christoph Buchheim who always supported me and my research work and who always had an open door to answer questions and giving advices.

Furthermore I want to thank my family and all my friends for being always there for me.

Contents

Chapter 1

Introduction

Combinatorial optimization problems occur in many real-world applications e.g. in the industry or in natural sciences and are intensively studied in the optimization literature. A wide range of problems has been solved yet and the algorithms are continuously improved to run faster and to solve larger instances. A famous problem which is often solved e.g. in the logistic industry is the *vehicle routing problem*. Consider a parcel service which owns a fleet of vehicles and which has to deliver parcels to its customers every day. The vehicle routing problem calculates a tour for each vehicle such that each customer is served by one vehicle and such that on each tour the total demand of the customers does not exceed the capacity of the vehicle. The objective is to find such a set of tours which has minimal cost e.g. minimal travel time. This problem is known to be hard to solve in practice in the sense that an optimal solution can not be calculated in a reasonable time for high-dimensional instances. In this case often approximation algorithms or even heuristic algorithms are used.

In this thesis we study combinatorial problems which can be formulated as

$$\min_{x \in X} \; c^\top x + c_0, \tag{M}$$

where $X \subseteq \{0,1\}^n$ are the incidence vectors of all feasible solutions of the problem, $c \in \mathbb{R}^n$ is a cost-vector and $c_0 \in \mathbb{R}$ a constant. We call (M) the *deterministic problem* or the *underlying combinatorial problem*.

If we want to solve a combinatorial problem in a real-world application we often have to deal with uncertainty in the problem parameters. For example the traveling times of a vehicle can be uncertain because every day a different traffic situation can occur. Depending on the size of the problem, calculating an optimal solution for the vehicle routing problem in the morning after

the traffic scenario is observed, can be inefficient. One approach to tackle uncertainty in the problem parameters is the *robust optimization* approach. Here for an *uncertainty set* U which contains all relevant *scenarios* of the uncertain parameters, the objective is to find a solution which is feasible for every scenario and which optimizes the worst objective value over all scenarios. In this thesis we assume that the uncertainty only affects the objective function, i.e. every scenario is given by a cost vector $(c, c_0) \in U \subseteq \mathbb{R}^{n+1}$. Then the *robust counterpart* of (M) is the problem

$$\min_{x \in X} \max_{(c,c_0) \in U} c^\top x + c_0. \tag{M^2}$$

This problem is known to be *NP*-hard for several combinatorial problems and several classes of uncertainty sets. Furthermore in practical applications an optimal solution of (M^2) is often too conservative in the sense that its objective value can be very bad in many scenarios. Especially in practical applications robust solutions are often not useful.

In the robust optimization literature many different approaches have been presented to find less conservative solutions for problems with uncertain parameters. For example the K-adaptability approach aims to calculate K different solutions to approximate a robust two stage problem [41]. Here the variables of the problem are divided into first-stage variables x which have to be calculated before the scenario is known and second-stage variables y which can be calculated afterwards. In this thesis we consider the special case where only one stage exists and propose to solve deterministic problems (M) with uncertain objective functions $(c, c_0) \in U$ by adressing the problem

$$\min_{x^{(1)},\dots,x^{(k)} \in X} \max_{(c,c_0) \in U} \min_{i=1,\dots,k} c^\top x^{(i)} + c_0 \tag{M^3}$$

where $k \in \mathbb{N}$ is a given number of solutions. Clearly the approach is an extension of the min-max approach (M^2), which is obtained for $k = 1$, and yields a better objective value in general. As a motivation consider the parcel service again. Instead of calculating a solution every morning after the scenario is known, by Problem (M^3) a set of k solutions can be calculated once in a perhaps expensive preprocessing. Afterwards each morning when the actual scenario is known the best of the calculated solutions can be easily chosen by comparing the objective values. Furthermore the min-max-min approach (M^3) hedges against uncertainty in a robust way and even gives more flexibility to the user, since he can choose the solution which has most of the properties which he prefers from his personal experience and the solutions do not change over time.

In this thesis the main objective is to analyze the complexity of Problem (M^3) for several combinatorial problems and uncertainty classes. In Chapter 2 we present all necessary definitions and results for linear, multicriteria and convex optimization problems and we define all combinatorial problems which we will study in this thesis. Furthermore we give a short introduction to complexity theory. In Chapter 3 we will present a selection of robust optimization approaches and discuss the complexity of the related robust counterparts. The main contribution of this thesis is the analysis of the complexity of Problem (M^3) for convex uncertainty sets in Chapter 4 and for discrete uncertainty sets in Chapter 5. Additionally to the theoretical results we provide several algorithms to solve Problem (M^3) for the specific uncertainty classes and for several combinatorial problems. Finally in Chapter 6 we present the results of our experiments for the knapsack problem, the shortest path problem and the vehicle routing problem and give a practical proof that the algorithms presented in Chapter 4 work well on random instances for the afore mentioned problems.

Chapter 2

Preliminaries

2.1 Linear Optimization

In this section we will give a short introduction to linear optimization and polyhedral theory. We will quote basic definitions and results which are used in the following chapters. For a detailed description of the results in this section see [45, 40, 54].

We define a *linear program* as a minimization problem of the form

$$\begin{aligned} \min \quad & c^\top x \\ s.t. \quad & Ax \leq b \\ & x \in \mathbb{R}^n \end{aligned} \tag{$\mathcal{P}$}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Any $x \in \mathbb{R}^n$, which fulfills the *constraints*, i.e. it is contained in the set $P := \{x \in \mathbb{R}^n \mid Ax \leq b\}$, is called a *feasible solution*. Note that by the transformation

$$\min_{x \in P} c^\top x = -\max_{x \in P} -c^\top x$$

any linear program of the form $(\mathcal{P})$ can be transformed to an equivalent maximization problem. The main objective in linear programming is to find an optimal solution of problem $(\mathcal{P})$. Here a feasible solution $x^* \in P$ is called *optimal solution* if $c^\top x^* \leq c^\top x$ for all $x \in P$. We then say $v := c^\top x^*$ is the *optimal value* of $(\mathcal{P})$. The function $f : P \to \mathbb{R}$ with $f(x) = c^\top x$ is called *objective function*. If no feasible solution exists, i.e. P is empty, then the problem is called *infeasible*. The problem is called *unbounded* if for any $\alpha \in \mathbb{R}$ there exists an $x \in P$ with $c^\top x < \alpha$. We then define the optimal value by $-\infty$.

Example 2.1. Consider the linear program

$$\begin{aligned} \max \quad & x_1 + x_2 \\ s.t. \quad & -x_1 + x_2 \leq 0 \\ & x_1 \leq 1 \\ & -x_2 \leq 2 \\ & x_1, x_2 \in \mathbb{R} \end{aligned} \tag{2.1}$$

which is of the form $(\mathcal{P})$. The unique optimal solution is $(x_1, x_2) = (1, 1)$ with an optimal value of 2 (see Figure 2.1).

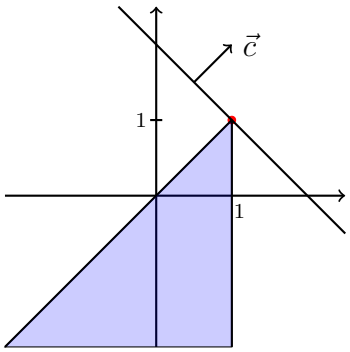


Figure 2.1: The feasible set and the optimal solution of the Problem (2.1).

In literature many different methods and algorithms have been developed to solve linear optimization problems. In general any linear program can be solved theoretically efficient by the *ellipsoid method* [40]. But since this method is not very practical to implement it is rarely used to solve problems of the form $(\mathcal{P})$. A practically efficient method to solve general linear problems is the *simplex method* which is very fast for many problems in practice and which is easier to implement. Nonetheless in the worst case its theoretical run-time can be exponential [40]. Besides the latter methods many problem-specific algorithms have been developed, which make use of the structure and the properties of $(\mathcal{P})$ for a given problem.

2.1.1 Polyhedra and Cones

We now define two classes of sets, *polyhedra* and *cones*, which are of high importance for optimization theory as well as for the framework of robust

optimization. A *polyhedron* is a set of the form

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\} \tag{2.2}$$

with a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $b \in \mathbb{R}^m$. A polyhedron is called *polytope* if it is bounded, i.e. if there exists a radius $R > 0$ such that

$$P \subset B_R(0) := \{x \in \mathbb{R}^n \mid \|x\|_2 \leq R\},$$

where $\|\cdot\|_2$ is the Euclidean norm. The description (2.2) is called an *outer description* of P. For a nonzero vector $d \in \mathbb{R}^n$ and $\delta \in \mathbb{R}$, we call a set of the form

$$H := \left\{x \in \mathbb{R}^n \mid d^\top x = \delta\right\}$$

a *hyperplane*. Furthermore we define

$$H^- := \left\{x \in \mathbb{R}^n \mid d^\top x \leq \delta\right\}.$$

A *face* of P is a subset $F \subseteq P$ such that a hyperplane H exists with $F = P \cap H$ and $P \subseteq H^-$. A face with a dimension of $\dim(P)-1$ is called *facet*. A face with dimension 0 is called *vertex*. The dimension here is defined by the concept of *affine independence*. We say the vectors $x_0, x_1 \ldots, x_k$ are affinely independent if $x_1 - x_0, \ldots, x_k - x_0$ are linearly independent. The dimension of a set $S \subseteq \mathbb{R}^n$ is then defined by

$$\dim(S) := \max\left\{k \in \mathbb{N} \cup \{0\} \mid x_0, \ldots, x_k \in S \text{ affinely independent}\right\}.$$

If the dimension is n, then the set S is called *full-dimensional*. For such a full-dimensional set S there always exists a point $s_0 \in S$ and a radius $r > 0$ such that

$$B_r(s_0) := \{x \in \mathbb{R}^n \mid \|x - s_0\|_2 \leq r\} \subseteq S.$$

We call $B_r(s_0)$ the *ball around s_0 with radius r*.

A set $C \subseteq \mathbb{R}^n$ is called *(convex) cone* if for any $x, y \in C$ and $\lambda, \mu \geq 0$ the point $\lambda x + \mu y$ is also contained in C.

Example 2.2. The set

$$\mathbb{R}^n_+ := \{x \in \mathbb{R}^n \mid x \geq 0\}$$

is a cone. Furthermore $\mathbb{R}^n_+$ is a polyhedron with exactly one vertex 0 and facets $\left\{x \in \mathbb{R}^n_+ \mid x_i = 0\right\}$ for each $i = 1, \ldots, n$. Note that $\mathbb{R}^n_+$ is not a polytope since it is not bounded. An example of a cone which is not a polyhedron is the second-order cone

$$\mathcal{K}_n := \left\{(x_0, x) \in \mathbb{R} \times \mathbb{R}^{n-1} \mid x_0 \geq \|x\|_2\right\}.$$

Both, cones and polyhedra, are convex sets. Here a set $U \subseteq \mathbb{R}^n$ is *convex* if for each $x, y \in U$ and $0 \leq \lambda \leq 1$ the point $\lambda x + (1-\lambda)y$ is also contained in U. We define the *convex hull* of a set $X \subseteq \mathbb{R}^n$ as

$$\operatorname{conv}(X) := \left\{ x = \sum_{i=1}^{k} \lambda_i x_i \mid \lambda_i \geq 0,\ x_i \in X,\ \sum_{i=1}^{k} \lambda_i = 1,\ k \in \mathbb{N} \right\}$$

and the *conic hull* of $X \subseteq \mathbb{R}^n$ as

$$\operatorname{cone}(X) := \left\{ x = \sum_{i=1}^{k} \lambda_i x_i \mid \lambda_i \geq 0,\ x_i \in X,\ k \in \mathbb{N} \right\}.$$

The convex hull (conic hull) of X is the smallest convex set (cone) which contains X. The famous *Theorem of Carathéodory* states that any point in the convex hull of $X \subseteq \mathbb{R}^n$ can be obtained as a *convex combination* of at most $n+1$ points in X.

Theorem 2.3 (Theorem of Carathéodory)**.** For any set $X \subseteq \mathbb{R}^n$ and any point $x \in \operatorname{conv}(X)$ there exist $x_1, \ldots, x_k \in X$ with $k \leq n+1$ such that $x \in \operatorname{conv}(x_1, \ldots, x_k)$.

It is easy to verify that any polytope with vertices $x_1, \ldots, x_k$ is equal to the set $\operatorname{conv}(x_1, \ldots, x_k)$. In particular a set P is a polytope if and only if it is the convex hull of a finite set of points [45]. For general polyhedra the following theorem holds [54].

Theorem 2.4 (Theorem of Weyl-Minkowski)**.** A set $P \subseteq \mathbb{R}^n$ is a polyhedron if and only if

$$P = \operatorname{conv}(x_1, \ldots, x_k) + \operatorname{cone}(y_1, \ldots, y_m) \tag{2.3}$$

for $x_1, \ldots, x_k, y_1, \ldots, y_m \in \mathbb{R}^n$.

The representation (2.3) of P in the latter theorem is called *inner description* of P. It is always possible to transform an outer description of a polyhedron into an inner description of the same polyhedron and the other way round. Nevertheless the size of the two descriptions can be different as the following example shows.

Example 2.5. Consider the cube $B := [0,1]^n \subset \mathbb{R}^n$. Clearly an outer description of B is given by

$$B = \{x \in \mathbb{R}^n \mid 0 \leq x_i \leq 1\}$$

and B can therefore be described by $2n$ inequalities. On the other hand an inner description of B is given by

$$B = \operatorname{conv}\left(\{0,1\}^n\right)$$

which involves 2^n vectors. It is easy to see that no inner description exists which uses a smaller number of vectors. To show the other direction if we consider the inner description

$$P := \operatorname{conv}\left(\{e_i, -e_i \mid i = 1, \ldots, n\}\right),$$

where e_i is the i-th unit-vector, then it can be shown that P has an exponential number of facets, and therefore any outer description must have an exponential number of inequalities. But the inner description is given by $2n$ vectors.

Since the run-time of an algorithm can depend on the size of the description it is important to mention which type of description we use.

A polyhedron P is called *rational* if any face of P contains a rational point. From the definition it follows directly that any vertex of P must be rational. Therefore a polytope is rational if and only if it can be described as the convex hull of rational points. Equivalently a rational polytope can be described by an outer description which has only rational entries. We call a polyhedron P *well-described* if an outer description

$$P = \left\{x \in \mathbb{R}^n \mid a_i^\top x \leq b_i,\ i = 1, \ldots, m\right\}$$

exists such that the binary encoding-length of each vector (a_i, b_i) is bounded by a value $\varphi \in \mathbb{Q}$. We then say P has *facet-complexity of at most* φ. The following lemma shows a relation between the facet complexity and the encoding size of the vertices of P.

Lemma 2.6 ([40])**.** If there exists an inner description $P = \operatorname{conv}(V) + \operatorname{cone}(E)$ such that the binary encoding-length of each vector in V and E is bounded by $\nu \in \mathbb{Q}$, then P has facet-complexity of at most $3n^2\nu$.

As a corollary we obtain that if each vertex of a polytope P has encoding-length of polynomial size, then P has facet-complexity of polynomial size.

Example 2.7. Consider the polytope $P := \operatorname{conv}(X)$ with $X \subseteq \{0,1\}^n$. Since the binary encoding length of each vertex in X is bounded by n it follows from Lemma 2.6 that P has facet-complexity of at most $3n^3$. Therefore P is well-described and an outer description of P exists such that each inequality has encoding length of at most $3n^3$.

Note that the definition of the facet-complexity above does not take into account the number of inequalities m in the outer description. Hence if each outer description of a polyhedron has an exponential number of inequalities, it can still have a polynomial facet-complexity. We will go into detail about this topic in Section 2.3.

2.1.2 Duality

Duality is an important concept which is applied to linear optimization problems but also to general optimization problems. For a given minimization problem, the *primal problem*, the basic idea is to find a related, so called *dual problem*, which computes the best lower bound on the optimal value of the primal problem. Depending on the class of optimization problems one can give conditions under which the optimal values of the dual and the primal problem coincide. The *dual problem* of $(\mathcal{P})$ is given by

$$\begin{aligned} \max \quad & -b^\top y \\ s.t. \quad & A^\top y = -c \\ & y \geq 0. \end{aligned} \tag{$\mathcal{D}$}$$

In Section 2.2 we show how the dual problem can be derived for general convex optimization problems and, as a special case, how $(\mathcal{D})$ can be derived for linear problems. Clearly for any feasible solution x of $(\mathcal{P})$ and any feasible solution y of $(\mathcal{D})$ it holds

$$y^\top b \leq y^\top Ax = c^\top x$$

since $y \geq 0$ and y and x are feasible. Therefore the optimal value of the dual problem gives always a lower bound on the optimal value of the primal problem. In fact both optimal values are equal if both problems have feasible solutions.

Theorem 2.8 ([45])**.**

(i) If $(\mathcal{P})$ and $(\mathcal{D})$ are both feasible, then their optimal values are the same.

(ii) If $(\mathcal{P})$ has an optimal solution then $(\mathcal{D})$ has an optimal solution and both optimal values are the same.

(iii) If $(\mathcal{P})$ is unbounded then $(\mathcal{D})$ is infeasible and vice versa.

Example 2.9. The dual problem of the problem in Example 2.1 is

$$\begin{aligned} \min \quad & y_2 + 2y_3 \\ s.t. \quad & -y_1 + y_2 = 1 \\ & y_1 - y_3 = 1 \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

with the optimal solution $(1, 2, 0)$ and an optimal value of 2.

2.2 Convex Optimization

In this section we will give a short introduction to problems and tools of the theory of convex optimization. For a detailed description of the topic see [22].

A general *convex optimization problem* is a problem of the form

$$\begin{aligned} \min \quad & f_0(x) \\ s.t. \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \\ & x \in \mathbb{R}^n \end{aligned} \tag{$\mathcal{COP}$}$$

where the functions $f_0, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$ are convex, i.e. they satisfy

$$f_i(\gamma x + (1-\gamma)y) \leq \gamma f_i(x) + (1-\gamma) f_i(y)$$

for all $x, y \in \mathbb{R}^n$ and all $\gamma \in [0, 1]$. Here $A \in \mathbb{R}^{p \times n}$ and $b \in \mathbb{R}^p$. There are many sub-classes of convex problems which can be solved by special methods and algorithms e.g. linear programs. Another sub-class are the so called *quadratically constrained quadratic problems* which are of the form

$$\begin{aligned} \min \quad & x^\top P_0 x + q_0^\top x + r_0 \\ s.t. \quad & x^\top P_i x + q_i^\top x + r_i \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \\ & x \in \mathbb{R}^n \end{aligned} \tag{$\mathcal{QCQP}$}$$

where each P_i is a symmetric positive semidefinite $n \times n$ matrix, $q_i \in \mathbb{R}^n$, $r_i \in \mathbb{R}$ and A, b are defined like above. Note that because all P_i are positive semidefinite the functions

$$f_i(x) := x^\top P_i x + q_i^\top x + r_i$$

are convex and hence the latter problem is a convex problem.

Example 2.10. Let $E = \left\{x \in \mathbb{R}^n \mid (x-\bar{x})^\top \Sigma (x-\bar{x}) \leq \Omega^2\right\}$ be an ellipsoid with center-point $\bar{x} \in \mathbb{R}^n$, positive definite matrix $\Sigma \in \mathbb{R}^{n\times n}$ and $\Omega \in \mathbb{R}$. Then optimizing a linear function over the ellipsoid U can be modeled as a $(\mathcal{QCQP})$ by

$$\begin{aligned} \max \quad & c^\top x \\ s.t. \quad & x^\top \Sigma x - (2\Sigma\bar{x})^\top x + \bar{x}^\top \Sigma \bar{x} - \Omega^2 \leq 0 \\ & x \in \mathbb{R}^n. \end{aligned}$$

It was shown in [40], that

$$\bar{x}^\top c + \Omega\sqrt{c^\top \Sigma^{-1} c}$$

is the optimal value of the latter problem (see Figure 2.2).

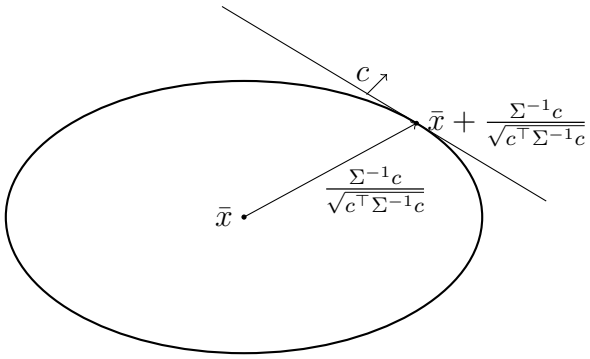


Figure 2.2: Maximization of $c^\top x$ over $E = \left\{x \in \mathbb{R}^n \mid (x-\bar{x})^\top \Sigma (x-\bar{x}) \leq 1\right\}$.

A more general problem than $(\mathcal{QCQP})$ is the *second-order cone problem*

$$\begin{aligned} \min \quad & c_0^\top x \\ s.t. \quad & \|P_i x + q_i\|_2 \leq c_i^\top x + d_i, \quad i = 1, \ldots, m \\ & Ax = b \\ & x \in \mathbb{R}^n \end{aligned} \tag{$\mathcal{SOCP}$}$$

where $P_i \in \mathbb{R}^{n_i \times n}$, $q_i \in \mathbb{R}^{n_i}$, $c_i \in \mathbb{R}^n$, $d_i \in \mathbb{R}$ and A, b are defined like above. Note that each of the first m constraints describes a second-order cone in dimension $n_i + 1$. Since for any positive semidefinite matrix P there exists a matrix B such that $P = B^\top B$, we can transform any quadratic constraint of the form $x^\top P x + q^\top x + r \leq 0$ into the equivalent second-order cone constraint

$$\left\| \begin{pmatrix} B \\ \frac{1}{2} q^\top \end{pmatrix} x + \begin{pmatrix} 0 \\ \frac{1}{2}(1+r) \end{pmatrix} \right\|_2 \leq \frac{1}{2}\left(1 - q^\top x - r\right).$$

Hence ($\mathcal{QCQP}$) is a special case of ($\mathcal{SOCP}$). In general Problem ($\mathcal{SOCP}$) and therefore Problem ($\mathcal{QCQP}$) can be solved in polynomial time by the interior-point method up to an arbitrary accuracy [8].

2.2.1 Duality

In this section we will give an instruction how to dualize convex problems and we will give a condition under which strong duality holds. For the convex problem ($\mathcal{COP}$) the *Lagrange dual function* $L : \mathbb{R}^m_+ \times \mathbb{R}^p \to \mathbb{R}$ is defined by

$$L(\lambda, \nu) = \inf_{x \in \mathbb{R}^n} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i (a_i^\top x - b_i) \right)$$

where a_i is the i-th row of A.

Lemma 2.11 ([22])**.** Let p^* be the optimal value of problem ($\mathcal{COP}$). Then for any $\lambda \in \mathbb{R}^m_+$ and $\nu \in \mathbb{R}^p$ the inequality

$$L(\lambda, \nu) \leq p^*$$

is valid.

Hence the Lagrange dual function gives a lower bound on the optimal value of Problem ($\mathcal{COP}$) for any $\lambda \in \mathbb{R}^m_+$ and $\nu \in \mathbb{R}^p$. Now a natural question is: what is the best of these lower bounds? The answer is the optimal value of the so called *Lagrange dual problem*

$$\begin{array}{rl} \max & L(\lambda, \nu) \\ s.t. & \lambda \in \mathbb{R}^m_+ \\ & \nu \in \mathbb{R}^p. \end{array} \tag{2.4}$$

In the following let d^* be the optimal value of (2.4). From Lemma 2.11 follows $d^* \leq p^*$. The following theorem gives a sufficient condition, called *Slater's Condition*, under which *strong duality* holds, i.e. $d^* = p^*$.

Theorem 2.12 ([22])**.** If there exists a vector $x \in \mathbb{R}^n$ such that $f_i(x) < 0$ for all $i = 1, \ldots, m$ and $Ax = b$ then $d^* = p^*$ holds.

In fact a weaker version of Slater's condition can be proved. To obtain strong duality it suffices if there exists a point $x \in \mathbb{R}^n$ such that strict inequality $f_i(x) < 0$ holds for all non-affine functions f_i while for the affine functions f_j only $f_j(x) \leq 0$ is required.

In the following example we derive the dual problem $(\mathcal{D})$ presented in Section 2.1.2 for linear optimization problems. The idea of the following calculations will also be used in Section 4.1 to transform problem (M^3).

Example 2.13. The Lagrange dual function of the primal problem $(\mathcal{P})$ is given by

$$\begin{aligned} L(\lambda) &= \inf_{x\in\mathbb{R}^n} \left(c^\top x + \lambda^\top (Ax - b)\right) \\ &= -\lambda^\top b + \inf_{x\in\mathbb{R}^n} \left(c^\top + \lambda^\top A\right) x \\ &= \begin{cases} -\lambda^\top b, & c^\top + \lambda^\top A = 0 \\ -\infty & \text{otherwise} \end{cases} \end{aligned}$$

for all $\lambda \geq 0$. Therefore the Lagrange dual problem is

$$\begin{aligned} \max \quad & -\lambda^\top b \\ s.t. \quad & c^\top = -\lambda^\top A \\ & \lambda \geq 0 \end{aligned}$$

which is exactly problem $(\mathcal{D})$. Note that either $(\mathcal{P})$ has no feasible solution or we can apply the weaker version of Slater's condition and obtain strong duality which was already given in Theorem 2.8.

Finally we will state a famous result from convex analysis called *minimax theorem*, which allows us to dualize min-max problems.

Theorem 2.14 ([51])**.** Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$ be non-empty closed convex sets and let $f : X \times Y \to \mathbb{R}$ be a continuous function which is concave in X for every $y \in Y$ and convex in Y for every $x \in X$. If either X or Y is bounded then

$$\inf_{y\in Y} \sup_{x\in X} f(x, y) = \sup_{x\in X} \inf_{y\in Y} f(x, y).$$

2.3 Complexity Theory

In this thesis our main objective is to analyze the complexity of problem (M^3). Hence in this section we give a short introduction to complexity theory and present results which we use in the following chapters. To get a precise and detailed description of this topic see [45].

In complexity theory the aim is to analyze how difficult it is to solve a problem. On the one hand if an algorithm to solve a problem is known, we

are interested in upper bounds on the worst-case run-time of the algorithm to get an idea how fast the algorithm is. On the other hand we want to classify problems by their difficulty. If a problem A can be used to solve a different problem B without doing additional expensive calculations, then we can say that Problem B is not harder than Problem A, since we can always solve it by solving Problem A.

In this thesis an *algorithm* is defined for a set of valid *inputs* and is a sequence of instructions which calculate for any input a certain *output* in a finite number of steps. A step in an algorithm can be arithmetic operations like addition, subtraction, multiplication, division and comparison of numbers, but also variable assignments. For an exact definition of this see the definition of Turing machines in [45].

In the following let $L := \{0,1\}^*$ be the set of all finite strings of 0's and 1's. Any subset of L is called *language*. We will use L to describe all necessary objects of our problems. For any $x \in L$ we define the *size* of x, denoted by $\langle x \rangle$, as the number of 0's and 1's in x. For the description of numerical values we assume the binary encoding scheme in this thesis, i.e. every value is described by its binary string $v \in L$. We say $f : \mathbb{N} \to \mathbb{N}$ is a *run-time function* for an algorithm $\mathcal{A}$ if for any input $i \in L$ of size n, the algorithm calculates an output in at most $f(n)$ steps. The algorithm is called *polynomial-time algorithm* (or has *polynomial run-time*) if a run-time function f exists such that $f(n) \leq p(n)$ for all $n \in \mathbb{N}$ for some polynomial p. An algorithm has *constant run-time* if $f(n) \leq c$ for all $n \in \mathbb{N}$ and a constant $c \in \mathbb{N}$. We often write $\mathcal{O}(f)$ to denote the run-time of an algorithm.

Given a language $I \subseteq \{0,1\}^*$, a function $f : I \to \{0,1\}^*$ and an algorithm A which calculates for any $i \in I$ the ouput $f(i) \in \{0,1\}^*$, then we say A *computes* f. If for a given f a polynomial time algorithm exists which computes f then we say f is *computable in polynomial time*. If $f : \{0,1\}^* \to \{0,1\}$ and A computes f then we say A decides the language

$$L' := \{l \in \{0,1\}^* \mid f(l) = 1\}.$$

If a polynomial time algorithm exists which decides a language L' then we say L' is *decidable in polynomial time*.

We define an *optimization problem* as a quadruple $\Pi = (I, (S_i)_{i \in I}, (c_i)_{i \in I}, \text{goal})$ where

- $I \subseteq \{0,1\}^*$ is a language decidable in polynomial time;
- $S_i \subseteq \{0,1\}^*$ is nonempty for each $i \in I$ and a polynomial p exists with $\langle y \rangle \leq p(\langle i \rangle)$ for all $i \in I$ and $y \in S_i$; furthermore the language $\{(i,y) \mid i \in I, y \in S_i\}$ is decidable in polynomial time;

- $c_i : S_i \to \mathbb{Q}$ for each $i \in I$ is a function computable in polynomial time;
- goal $\in \{\max, \min\}$.

The elements of I are called *instances*. For each instance $i \in I$ we call S_i the set of *feasible solutions* and c_i the *objective function* of i. Note that by the assumptions given in the listing above we make sure that an algorithm exists which can decide in polynomial time if a given string $i \in \{0,1\}^*$ is a valid instance of the problem and for any $y \in S_i$ if y is a feasible solution. Furthermore we assume that we can calculate the objective value for each feasible solution in polynomial time.

An *(exact) algorithm* $\mathcal{A}$ for Π is an algorithm which computes for each instance $i \in I$ a feasible solution $y \in S_i$ such that

$$c_i(y) = \text{goal}\{c_i(y') : y' \in S_i\}.$$

The computed solution is called *optimal solution* and its objective function value $c_i(y)$ is denoted by $\text{opt}(i)$. We define the size of an instance i of an optimization problem by $\langle i \rangle = \langle S_i \rangle + \langle c_i \rangle + 1$. The size of the latter objects depends on the description of the object which is used. As we have seen in Example 2.5 the size of a description of the same object can vary a lot.

Example 2.15. For an inner description

$$P = \text{conv}\,(x_1, \ldots, x_m) + \text{cone}\,(e_1, \ldots, e_l)$$

the size of P is

$$\langle P \rangle = \sum_{i=1}^{m} \langle x_i \rangle + \sum_{i=1}^{l} \langle e_i \rangle.$$

If P is given by an outer description $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ then the size of P is

$$\langle P \rangle = \sum_{i=1}^{m} \langle (a_i, b_i) \rangle.$$

As we have seen in Section 2.2 the feasible set can be an ellipsoid given by

$$E = \left\{ x \in \mathbb{R}^n \mid (x - \bar{x})^\top \Sigma (x - \bar{x}) \leq \Omega^2 \right\}.$$

Then the size of E is

$$\langle E \rangle = \langle \Sigma \rangle + \langle \bar{x} \rangle + \langle \Omega \rangle.$$

If a feasible set U is given by a list of vectors $U = \{c_1, \ldots, c_m\}$ then the size of U is $\langle U \rangle = \sum_{i=1}^{m} \langle c_i \rangle$. The latter classes of sets of feasible solutions will be studied later in this thesis.

We say $f : \mathbb{N} \to \mathbb{N}$ is a run-time function for an optimization problem Π if for any instance i of size n, an algorithm for Π exists which calculates an optimal solution of instance i in at most $f(n)$ steps. Note that in general the actual run-time for an instance of size n must not depend exactly on the parameter n but can depend on a much smaller parameter which is part of the instance description. For example as we will see later, certain problems over a polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ can be solved in polynomial time in the maximum of $\langle (a_i, b_i) \rangle$ over all row-vectors (a_i, b_i). Therefore in contrast to the size of P the run-time of the algorithm does not depend on the number of rows in A. Note that the run-time of an algorithm can depend on additional parameters given in the input which are not part of the instance description, e.g. an accuracy parameter $\varepsilon > 0$ for the computations. In this case we have to mention that the run-time function is given in the size of the instance and the respective additional parameters.

The basic classes P and NP are originally defined for so called *decision problems*. A decision problem is a pair $\Pi = (I, Y)$, where $I \subseteq \{0,1\}^*$ is a language decidable in polynomial time, called *instances*, and $Y \subseteq I$ are the so-called *yes-instances*. An algorithm solves Π if it decides for any instance $i \in I$, if $i \in Y$ or if $i \in I \setminus Y$, i.e. the algorithm computes the function

$$f : I \to \{0,1\}$$

with $f(i) = 1$ if $i \in Y$ and $f(i) = 0$ otherwise. A detailed description of the following results and an extensive list of decision and optimization problems can be found in [37].

Definition 2.16. The class of all decision problems which can be solved by a polynomial-time algorithm is denoted by P.

Definition 2.17. A decision problem $\Pi = (I, Y)$ belongs to the class NP if there is a polynomial p and a decision problem $\Pi' = (I', Y')$ in P where $I' := \{ic : i \in I, c \in \{0,1\}^{\lfloor p(\langle i \rangle) \rfloor}\}$, such that

$$Y = \{y \in I : \exists\, c \in \{0,1\}^{\lfloor p(\langle i \rangle) \rfloor} : yc \in Y'\}.$$

Here c is called a *certificate* for y.

In other words a problem is in NPif for any yes-instance y there exists a certificate of polynomial size, with the help of which we can decide in polynomial time that y is a yes-instance.

Example 2.18. The *subset-sum problem* is defined as follows: An instance is given by integers $c_1, \ldots, c_n \in \mathbb{Z}$ and $K \in \mathbb{Z}$ and an instance is a yes-instance if there exists a set $S \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in S} c_j = K$. If we

choose Π' in the latter definition as the subset-sum problem as well then for each yes-instance the set S is a certificate and by calculating $\sum_{j \in S} c_j$ we can decide in polynomial time if the given instance is a yes-instance. Therefore the subset-sum problem is in NP.

It is easy to prove that $P \subseteq NP$ by choosing $p \equiv 0$ and $\Pi' = \Pi$ in Definition 2.17. On the other hand the question if $P = NP$ or not, is one of the most important open problems in complexity theory.

Since we will only work with optimization problems in this thesis we will omit further definitions which are only related to decision problems. Substantial results again can be found in [45].

Definition 2.19. A problem Π_1 *polynomially reduces* to an optimization problem $\Pi_2 = (I, (S_i)_{i \in I}, (c_i)_{i \in I}, \text{goal})$ if for a given function f defined by

$$f(i) = \{y \in S_i : c_i(y) = opt(i)\},$$

there exists a polynomial-time algorithm to solve Π_1 using f, where calculating $f(i)$ for any i is assumed to consume constant run-time.

In other words, if we have an oracle which solves problem Π_2 in constant time, then we can solve problem Π_1 in polynomial time. A consequence of the latter definition is that, if we have a polynomial-time algorithm for Π_2, then we can solve Π_1 in polynomial time. On the other hand if we would know that Π_1 can not be solved in polynomial-time then Π_2 can not be solved in polynomial time since this would lead to a contradiction. It is easy to see that the latter construction is transitive i.e. if Π_1 polynomially reduces to Π_2 and Π_2 polynomially reduces to Π_3 then Π_1 polynomially reduces to Π_3.

Definition 2.20. An optimization problem Π is called *NP-hard* if all problems in NP polynomially reduce to Π.

The latter definition implies that the class of NP-hard problems contains the optimization problems which are at least as hard as the hardest problems in NP. To show that a problem Π is NP-hard, by the transitivity of the reduction, it suffices to reduce any NP-hard problem to Π.

Proposition 2.21. If an NP-hard problem Π_1 polynomially reduces to an optimization problem Π_2 then Π_2 is NP-hard.

If i consists of a list of integers then in the following we denote by largest(i) the largest of these integers.

Definition 2.22. Let Π be an optimization problem such that each instance i consists of a list of integers. An algorithm for Π is called *pseudopolynomial* if its run-time is bounded by a polynomial in $\langle i \rangle$ and largest(i). An *NP*-hard optimization problem for which a pseudopolynomial algorithm exists is called *weakly NP-hard*.

In other words the run-time of a pseudopolynomial algorithm depends on the values of the numbers in i. It can have exponential run-time in its worst-case even if all numbers can be encoded in polynomial size. But if the value of all occurring numbers has polynomial size, the algorithm is a polynomial-time algorithm.

Example 2.23. The knapsack problem, which we will define properly in Section 2.4, is known to be *NP*-hard (see Theorem 2.46). Nevertheless there exists a dynamic programming algorithm to solve the problem (see [44]) which has run-time $\mathcal{O}(nb)$ where b is the knapsack capacity. Clearly this is a pseudopolynomial algorithm. Note that if we choose $b = 2^n$ then the size of b is linear in n, since we assume binary encoding, but the run-time of the algorithm is $n2^n$ in its worst-case which is exponential in the size of b.

Problems which can not have a pseudopolynomial algorithm, unless $P = NP$, are the following:

Definition 2.24. Let Π be an optimization problem such that each instance consists of a list of integers and for any polynomial p let Π_p be the same problem restricted to only the instances $i \in I$ with largest$(i) \leq p(\langle i \rangle)$. Problem Π is called *strongly NP-hard* if there is a polynomial p such that Π_p is *NP*-hard.

Lemma 2.25. Let Π_1 and Π_2 be optimization problems. If Π_1 is strongly *NP*-hard and polynomially reduces to the problem $(\Pi_2)_p$ for a polynomial p, then Π_2 is strongly *NP*-hard.

In other words the latter lemma states that if all the numbers used by the reduction algorithm have polynomial size then reducing a strongly *NP*-hard problem yields strongly *NP*-hardness. For problems which are hard to solve, sometimes also non-optimal solutions are accepted by a user if we can give a certain guarantee for the quality of the solution.

Definition 2.26. Let Π be an optimization problem with a non-negative optimal value and $\varepsilon > 0$. An algorithm $\mathcal{A}$ is an *ε-approximation algorithm* if for any instance i of Π it calculates a feasible solution $y \in S_i$ with

$$\frac{1}{1+\varepsilon}\text{opt}(i) \leq c_i(y) \leq (1+\varepsilon)\text{opt}(i).$$

Problem Π has a *fully polynomial approximation scheme* if for any $\varepsilon > 0$ there exists an ε-approximation algorithm whose run-time as well as the maximum size of any number occurring in the computation is bounded by a polynomial in $\langle i \rangle + \langle \varepsilon \rangle + \frac{1}{\varepsilon}$. We then say that Π admits an FPTAS.

2.3.1 Oracles

In Section 4.1.1 we will give an oracle-based algorithm which uses several results from [40]. Therefore we will give a short introduction to the results which we use later. For a detailed description see [40].

We define an *oracle* as an algorithm $\mathcal{O}$ with constant run-time which returns, for an input $\sigma \in \{0,1\}^*$ of size n, an output of size at most $p(n)$ for a polynomial p. We can consider an oracle as a procedure which gives an answer to a question or a solution for a problem without increasing the run-time except by a constant number of steps. An *oracle-polynomial algorithm* $\mathcal{A}$ is an polynomial-time algorithm which uses $\mathcal{O}$. We also say $\mathcal{A}$ runs in oracle-polynomial time. In Section 4.1.1 we will use oracles for optimization and separation problems. The latter problems are defined as follows for convex and compact sets $K \subseteq \mathbb{R}^n$:

Problem 2.27 (The Strong Optimization Problem)**.** Given a vector $c \in \mathbb{R}^n$, find a vector $y \in K$ that maximizes $c^\top x$ over K or assert that K is empty.

Problem 2.28 (The Strong Separation Problem)**.** Given a vector $y \in \mathbb{R}^n$, decide whether $y \in K$, and if this is not the case, find a vector $c \in \mathbb{R}^n$ such that $c^\top y > \max\{c^\top x \mid x \in K\}$ holds.

In fact the latter problem is to find a so-called *cutting-plane*, i.e. a hyperplane H which cuts off the point y if it is not contained in K. More precisely we want to find a hyperplane H such that $K \subseteq H^-$ but $y \notin H^-$ if it is not contained in K.

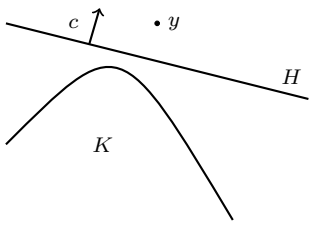


Figure 2.3: Solution c of the separation problem.

Based on the latter two problems a very famous result from [40] can be cited, which states that optimization and separation are polynomially equivalent. In the following for a well-described polyhedron $P \subseteq \mathbb{R}^n$ we define the *facet-size* of P by $\langle P \rangle_F = n+\varphi$, where φ is the facet-complexity of P. This is motivated by the fact that the algorithms in [40] iteratively call the separation oracle or the optimization oracle respectively a polynomial number of times no matter how many inequalities are given in the description of P. The computations only depend on an upper bound on the size of each inequality on its own.

Theorem 2.29 ([40])**.** Given an oracle for the strong separation (optimization) problem, the strong optimization (separation) problem can be solved in oracle-polynomial time in $\langle P \rangle_F$ for any well-described polyhedron P.

Example 2.30. As we have seen in Example 2.7 the facet-complexity of $P := \operatorname{conv}(X)$ with $X \subseteq \{0,1\}^n$ is at most $3n^3$. An optimization oracle for the deterministic problem

$$\min_{x \in X} c^\top x$$

yields an optimization oracle for the equivalent problem

$$\min_{x \in P} c^\top x$$

and hence from Theorem 2.29 follows that we can solve the separation problem over P in time polynomial in $3n^3 + n$.

Given an optimization oracle it is possible to calculate a convex combination for any rational point in a well-described polyhedron in polynomial time.

Theorem 2.31 ([40])**.** For any well-described polyhedron P given by a strong optimization oracle and for any rational vector $y_0 \in P$, there exists an oracle-polynomial time algorithm in $\langle P \rangle_F + \langle y_0 \rangle$, that finds affinely independent vertices $x_0, \ldots, x_k$ of P and rational $\lambda_0, \ldots, \lambda_k \geq 0$ with $\sum_{i=0}^k \lambda_i = 1$ such that $y_0 = \sum_{i=0}^k \lambda_i x_i$.

Note that since the vertices calculated by the latter algorithm are affinely independent the algorithm calculates at most $n+1$ vertices. The latter theorem especially states that for each rational point $x^* \in \operatorname{conv}(X)$ we can calculate a convex combination of points in X for x^* in polynomial time if we can linearly optimize over $\operatorname{conv}(X)$ in polynomial time, which is equivalent to linear optimization over X.

For general convex problems the equivalence in Theorem 2.29 is not true. If we are considering arbitrary convex sets K, we even have to take into account

irrational values, which can occur for example if we consider expressions involving a norm. Therefore we need a parameter $\varepsilon > 0$ which determines the accuracy of calculations. We define

$$B_\varepsilon(K) := \{x \in \mathbb{R}^n \mid \|x - y\|_2 \leq \varepsilon \text{ for some } y \in K\}$$

and

$$B_{-\varepsilon}(K) := \{x \in K \mid B_\varepsilon(x) \subseteq K\}.$$

By definition $B_\varepsilon(K)$ contains all points which have a distance to K of at most ε. The set $B_{-\varepsilon}(K)$ contains all points which have a distance of at least ε to the boundary of K (see Figure 2.4). It always holds $B_{-\varepsilon}(K) \subset K \subset B_\varepsilon(K)$. Note that if K is not full-dimensional then $B_{-\varepsilon}(K) = \emptyset$.

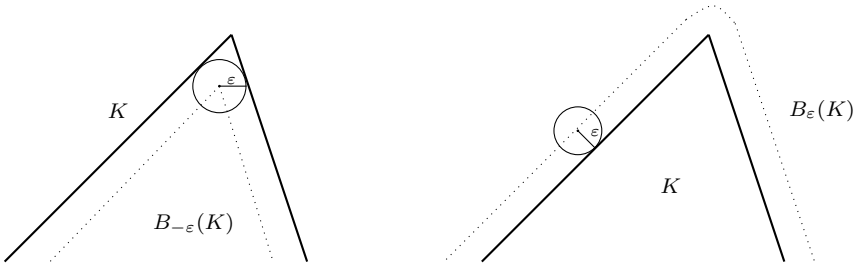


Figure 2.4: The sets $B_{-\varepsilon}(K)$ and $B_\varepsilon(K)$

Using the latter definitions we define the weak version of the so called membership problem.

Problem 2.32 (The Weak Membership Problem)**.** Given a vector $y \in \mathbb{Q}^n$ and any rational value $\varepsilon > 0$, assert that $y \in B_\varepsilon(K)$ or that $y \notin B_{-\varepsilon}(K)$.

Note that both conditions, $y \in B_\varepsilon(K)$ and $y \notin B_{-\varepsilon}(K)$ can be true at the same time. If we can solve the strong separation problem it directly follows that we can solve the weak membership problem.

In the following a full-dimensional compact convex set K is called a *centered convex body* if the following information is explicitly given: the integer n such that $K \subseteq \mathbb{R}^n$, a positive rational number R such that $K \subseteq B_R(0)$ and a rational number r and a vector $a_0 \in \mathbb{Q}^n$ such that $B_r(a_0) \subseteq K$. We then write $K(n, R, r, a_0)$ and the size of K is

$$\langle K \rangle := \langle n \rangle + \langle \varepsilon \rangle + \langle R \rangle + \langle r \rangle + \langle a_0 \rangle \ .$$

As mentioned before the equivalence of the strong optimization problem and the strong separation problem is not true if we consider convex problems. But,

as the following theorem states, if we consider convex objective functions, given by an oracle and defined over a centered convex body, then solving the weak membership problem in polynomial time yields a polynomial-time optimization-algorithm in a weak version. More precisely:

Theorem 2.33 ([40])**.** Let $\varepsilon > 0$ be a rational number, $K(n, R, r, a_0)$ a centered convex body given by a weak membership oracle and $f : \mathbb{R}^n \to \mathbb{R}$ a convex function given by an oracle which returns for every $x \in \mathbb{Q}^n$ and $\delta > 0$ a rational number t such that $|f(x) - t| \leq \delta$. Then there exists a oracle-polynomial time algorithm in $\langle K \rangle$ and $\langle \delta \rangle$, that returns a vector $y \in B_\varepsilon(K)$ such that $f(y) \leq f(x) + \varepsilon$ for all $x \in B_{-\varepsilon}(K)$.

In other words the theorem states that we can find a vector almost in K which almost maximizes the objective function over all vectors which are deep in K. In Section 4.1.1 the latter theorem will be combined with a rounding procedure. The idea is to round the point y which is calculated in Theorem 2.33 to a denominator such that it is guaranteed to be contained in K. Under certain assumptions, this can also be done in polynomial time, which is shown by the following lemma.

Lemma 2.34 ([40])**.** Let $P \subseteq \mathbb{R}^n$ be a polyhedron such that for $\varphi \in \mathbb{N}$ we have $\langle P \rangle_F \leq \varphi$ and let $v \in B_{2^{-6n\varphi}}(P)$. Then we can calculate $q \in \mathbb{Z}$ with $0 < q < 2^{4n\varphi}$ and a vector $w \in \mathbb{Z}^n$ in polynomial time in $\langle \varphi \rangle$ and $\langle v \rangle$ such that

$$\|qv - w\| < 2^{-3\varphi}$$

and such that $\frac{1}{q}w$ is contained in P.

2.4 Combinatorial Optimization

In this section we will define the combinatorial optimization problems which will appear in this thesis. While our results in Section 4 hold for any combinatorial problem which can be described through a set $X \subseteq \{0, 1\}^n$, the complexity results in Chapter 5 are proved for the combinatorial problems defined in this section. We will not give explicit polyhedral formulations for the feasible sets $X \subseteq \{0, 1\}^n$ of these problems. This is due to the fact that our results in Section 4.1.1 can be applied to an arbitrary optimization routine for the underlying combinatorial problem and therefore we do not need any specific formulation of the feasible set X. A detailed analysis of the following results can be found in [45, 54].

Most of the combinatorial problems we are considering are defined on graphs. An *undirected graph* $G = (V, E)$ consists of a finite set of *nodes* $V = \{v_1, \ldots, v_m\}$ and a finite set of *edges* $E = \{e_1, \ldots, e_n\}$. An edge $e = \{v, w\}$ is a set of two nodes $v, w \in V$. In a *directed* graph, each edge has a direction i.e. it has a *head* v and a *tail* w. In this case we write $e = (v, w)$ as a 2-tuple to clarify that the order of the nodes has to be considered. An undirected graph can be easily transformed into a directed graph by replacing each edge $e = \{v, w\}$ by two directed edges $e_1 = (v, w)$ and $e_2 = (w, v)$. An undirected graph is called *complete* if for any two nodes $v, w \in V$ there exists an edge $\{v, w\} \in E$ or in the directed case if both edges (v, w) and (w, v) exist. Two edges are called *parallel* if they are defined for the same pair of nodes and, in the directed case, have the same direction. In this thesis we are considering only *simple* graphs i.e. graphs without parallel edges. Furthermore we assume that no edges of the form (v, v) exist. Two edges are called *adjacent* if they have a common node. An undirected graph is called *bipartite* if V can be partitioned in two disjunctive sets V_1 and V_2 such that for each edge $\{v, w\} \in E$ holds $v \in V_1$ and $w \in V_2$. For any subset of edges $X \subseteq E$ we define the *incidence vector* $\mathbf{1}_X : E \to \{0, 1\}$ of X by

$$\mathbf{1}_X(e) = \begin{cases} 1 & e \in X \\ 0 & e \notin X. \end{cases}$$

2.4.1 The Shortest Path Problem

Let $G = (V, E)$ be a directed graph, $c : E \to \mathbb{R}$ a cost function on the edges of G and $s, t \in V$ two nodes. A path from s to t is a sequence of edges

$$p_{st} := (e_1, e_2, \ldots, e_l)$$

where the head of e_1 is s, the tail of e_l is t and the tail of e_j is equal to the head of e_{j+1} for each $j = 1, \ldots, l-1$ while any node in V is traversed at most once. The cost of path p_{st} is defined by

$$c(p_{st}) := \sum_{j=1}^{l} c(e_j).$$

A graph is called *connected* if for any pair of vertices $s, t \in V$ a path from s to t exists. A *cycle* C in G is a sequence of edges $(p_{st}, (t, s))$ for a path p_{st}. A graph is *conservative* if each cycle has non-negative cost. The latter definitions can be applied analogously to undirected graphs. The shortest path problem is defined as follows:

Problem 2.35 (Shortest Path Problem)**.** Given a conservative directed graph G together with a cost function $c : E \to \mathbb{R}$ and two vertices $s, t \in V$, find a path from s to t with minimal cost or decide that no such path exists.

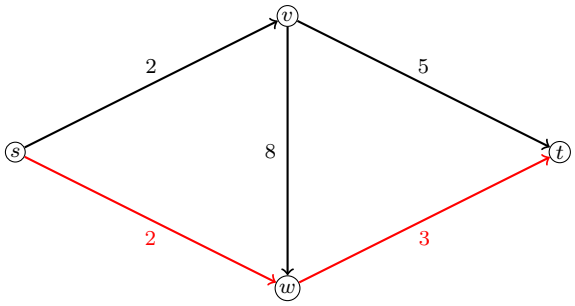


Figure 2.5: The shortest path with cost 5

The set of all paths in G can be described by a binary set $X_{SP} \subseteq \{0,1\}^E$ which contains all incidence vectors of paths in G:

$$X_{SP} = \{\mathbf{1}_P \mid P \subseteq E \text{ is a path from } s \text{ to } t \text{ in } G\}.$$

The following theorem gives two important complexity results on the shortest path problem.

Theorem 2.36 ([54])**.** The shortest-path problem on conservative graphs can be solved in polynomial time while for arbitrary cost functions it is *NP*-hard.

Proof. Statement (i) can be verified among others by the by the *Moore-Bellmann-Ford Algorithm* which has polynomial run-time. For further algorithms see [54].

The second statement is proved by reducing the Hamiltonian path problem to the shortest path problem. Given a directed graph $G = (V, E)$ and $s, t \in V$, the Hamiltonian path problem gives an answer to the question if there exists a path from s to t in G which traverses each node in V exactly once. The Hamiltonian path problem is known to be *NP*-complete [54]. Define the shortest path problem on G by defining cost $c(e) = -1$ on each edge $e \in E$. The shortest path then has cost $|V| - 1$ if and only if a Hamiltonian path exists in G. □

2.4.2 The Minimum Cut Problem

Let $G = (V, E)$ be an undirected and connected graph and $c : E \to \mathbb{R}$ a cost function on the edges of G. A *cut* in G is a set of edges of the form

$$\delta(S) := \left\{ \{v, w\} \in E \;\middle|\; v \in S,\ w \in V \setminus S \right\}.$$

for any nonempty set of nodes $S \subsetneq V$. For $s, t \in V$ an *s-t-cut* in G is a cut $\delta(S)$ where $s \in S$ and $t \in V \setminus S$. The cost of a cut are defined by

$$c\left(\delta(S)\right) := \sum_{e \in \delta(S)} c(e).$$

Problem 2.37 (Minimum Cut Problem)**.** Given an undirected and connected graph $G = (V, E)$ and a cost function $c : E \to \mathbb{R}$, find a cut in G with minimal cost.

Problem 2.38 (Minimum s-t-Cut Problem)**.** Given an undirected and connected graph $G = (V, E)$, two nodes $s, t \in V$ and a cost function $c : E \to \mathbb{R}$, find a s-t-cut in G with minimal cost.

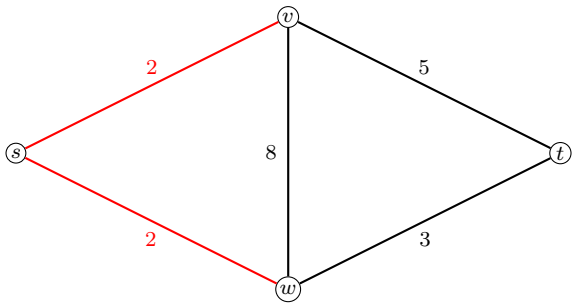


Figure 2.6: The minimum s-t cut with cost 4

The set of all cuts in G can be described by a binary set $X_C \subseteq \{0, 1\}^E$ which contains all incidence vectors of cuts in G:

$$X_C = \{\mathbf{1}_{\delta(S)} \mid \emptyset \neq S \subsetneq V\}.$$

Theorem 2.39. The minimum cut problem and the minimum s-t-cut problem can be solved in polynomial time.

Proof. By the famous Max-Flow-Min-Cut-Theorem the minimum s-t-cut problem can be solved by the maximum flow problem in polynomial time [54]. On the other hand the minimum cut problem can be solved by calculating the minimum s-t-cut for each pair of nodes $s, t \in V$ and return the cut with minimum cost over all pairs. Note that the number of pairs of nodes is $\frac{1}{2}|V|\,(|V|-1)$ and is therefore polynomial in the input. □

2.4.3 The Minimum Spanning Tree Problem

Let $G = (V, E)$ be an undirected graph and $c : E \to \mathbb{R}$ a cost function on the edges of G. A *spanning tree* in G is a sub-graph $T = (V, E')$ with $E' \subseteq E$ such that T contains no cycles and is connected. The cost of a spanning-tree is defined by $c(T) := \sum_{e \in E'} c(e)$.

Problem 2.40 (Minimum Spanning Tree Problem)**.** Given a connected undirected graph $G = (V, E)$ and a cost function $c : E \to \mathbb{R}$, find a spanning tree in G with minimal cost.

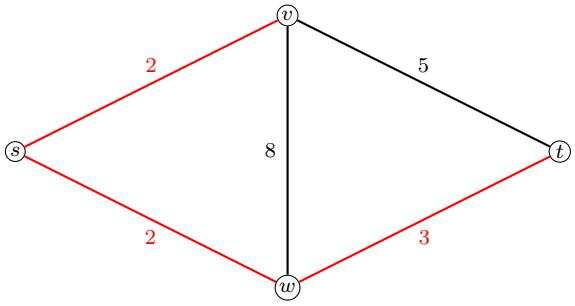


Figure 2.7: The minimum spanning-tree with cost 7

The set of all possible spanning trees in G can be described by a binary set $X_{ST} \subseteq \{0,1\}^E$ which contains all incidence vectors of spanning trees in G:

$$X_{ST} = \{\mathbf{1}_{E'} \mid T = (V, E') \text{ is a spanning tree in } G\}.$$

Theorem 2.41 ([45])**.** The spanning-tree problem can be solved in polynomial time.

Proof. The spanning-tree problem can be solved e.g. by the famous algorithm of Kruskal which has polynomial run-time. □

2.4.4 The Matching Problem

Let $G = (V, E)$ be an undirected graph and $w : E \to \mathbb{R}$ a weight function on the edges of G. A *matching* in G is a set of pairwise non-adjacent edges $M \subseteq E$. A *perfect matching* in G is a matching $M_p \subseteq E$ such that for each node $i \in V$ there exists an edge $e \in M_p$ with $i \in e$. A perfect matching in a bipartite graph is called *assignment*. The weight of a matching is defined by $w(M) := \sum_{e \in M} w(e)$. In literature different variants of the matching problem are studied [45]. In this thesis we will consider the following two variants.

Problem 2.42 (Minimum Weight Perfect Matching Problem)**.** Given an undirected graph G and a weight function $w : E \to \mathbb{R}$, find a perfect matching M with minimum weight or decide that G has no perfect matching.

Problem 2.43 (Assignment Problem)**.** Given a bipartite graph G, find a perfect matching M_p in G with minimum weight or decide that G has no perfect matching.

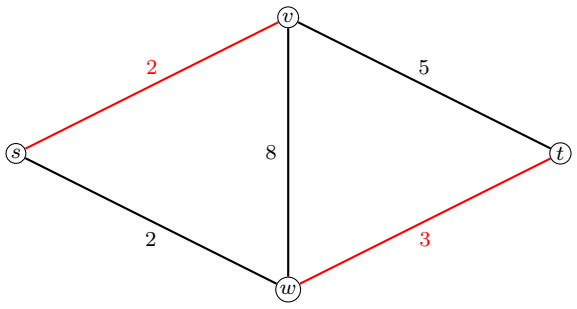


Figure 2.8: The minimum perfect matching with cost 5

The set of all perfect matchings in G can be described by a binary set $X_M \subseteq \{0, 1\}^E$ which contains all incidence vectors of perfect matchings in G.

$$X_M = \{\mathbf{1}_M \mid M \subseteq E \text{ is a matching in } G\}.$$

Theorem 2.44 ([45])**.** The minimum weight perfect matching problem and the assignment problem can be solved in polynomial time.

Proof. The minimum weight perfect matching problem can be solved in polynomial time by the famous algorithm of Edmonds [45]. The assignment problem is a special case of the minimum weight perfect matching problem which proves the result. □

2.4.5 The Knapsack Problem

The Knapsack Problem is defined as follows:

Problem 2.45 (Knapsack Problem)**.** Given profits $c_1, \dots, c_n \in \mathbb{N}$, weights $a_1, \dots, a_n \in \mathbb{N}$ and a capacity b, find a subset $S \subseteq \{1, \dots, n\}$ such that $\sum_{j \in S} a_j \leq b$ and $\sum_{j \in S} c_j$ is maximal.

The set of all feasible subsets S can be described by a binary set $X_{KP} \subseteq \{0,1\}^n$, which contains all incidence vectors of feasible subsets $S \subseteq \{1, \dots, n\}$:

$$X_{KP} = \{\mathbf{1}_S \mid S \subseteq \{1, \dots, n\} : \sum_{j \in S} a_j \leq b\}.$$

Theorem 2.46 ([45])**.** The Knapsack Problem is weakly *NP*-hard.

Proof. To prove that Problem 2.45 is *NP*-hard we reduce the subset-sum problem to the knapsack problem. For given $c_1, \dots, c_n \in \mathbb{Z}$ and $K \in \mathbb{Z}$ the subset-sum problem asks if there exists a set $S \subseteq \{1, \dots, n\}$ such that $\sum_{j \in S} c_j = K$. Define an instance of the knapsack problem by choosing $c_1, \dots, c_n$ as profits and setting $a_j = c_j$ and $b = K$. Then the knapsack problem has optimal value K if and only if the answer to the subset sum problem is yes.

A pseudopolynomial algorithm for the knapsack problem which has a runtime of $\mathcal{O}(nb)$ is given by a dynamic programming approach and can be found in [44]. □

2.4.6 The Unconstrained Binary Problem

In this subsection we introduce the unconstrained binary problem. As we will see the deterministic version of this problem can be solved trivially. But since the robust version (M^2) of the problem is *NP*-hard as we will see later we will proof complexity results for the min-max-min version as well. The unconstrained binary problem is defined as follows.

Problem 2.47 (Unconstrained Binary Problem)**.** Given profits $c_1, \dots, c_n \in \mathbb{Q}$ find a subset $S \subseteq \{1, \dots, n\}$ such that $\sum_{j \in S} c_j$ is minimal.

Since all subsets $S \subseteq \{1, \dots, n\}$ are feasible, the set of feasible incidence vectors is

$$X_{UB} = \{0,1\}^n.$$

Theorem 2.48. The binary unconstrained problem can be solved in linear time.

Proof. Clearly an optimal solution of the binary unconstrained problem is

$$S = \{i \in \{1, \ldots, n\} \mid c_i < 0\}$$

which proves the result. □

2.5 Multicriteria Optimization

In Section 3.1.1 we derive a relation between robust optimization and multicriteria optimization which we will extend for Problem (M^3) in Section 5.1. Therefore we give a short introduction to multicriteria optimization in this section. Detailed results about this topic can be found in [30, 31].

We define a *linear multicriteria optimization problem* by

$$\min_{x \in X} \left(c_1^\top x, \ldots, c_m^\top x\right) \tag{2.5}$$

where $c_i \in \mathbb{R}^n$. Since in this thesis we are interested in combinatorial optimization problems we assume $X \subseteq \{0,1\}^n$. Obviously in general there does not exist a point which minimizes all objective functions c_i at the same time. So the main objective in multicriteria optimization is to find *efficient solutions* i.e. solutions for which no other solution exists which is better in every criteria c_i. Formally a solution $x \in X$ is called efficient if no other solution $y \in X$ exists, such that $c_i^\top y \leq c_i^\top x$ for all $i = 1 \ldots, m$ and $c_j^\top y < c_j^\top x$ for at least one objective function c_j. If such a y exists, then x is called *dominated* by y. One objective in multicriteria optimization can be to find the set of all efficient solutions X_E. Note that for $X \subseteq \{0,1\}^n$ the set X_E can have exponential size, as it was shown for several combinatorial problems [31]. There are several different methods to find efficient solutions for multicriteria problems. A well-studied method is the weighted-sum method [30]. Here the idea is to solve the deterministic problem

$$\min_{x \in X} \sum_{i=1}^{m} \lambda_i c_i^\top x \tag{2.6}$$

for $\lambda_i \geq 0$ and $\sum_{i=1}^m \lambda_i = 1$. It can be proved that any optimal solution of the latter problem is efficient for the related multicriteria problem. Nevertheless in general not for every efficient solution exists a λ like above such that

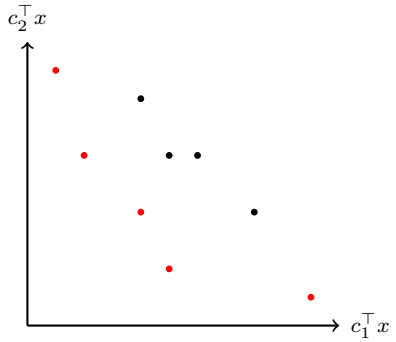


Figure 2.9: Objective function values $(c_1^\top x, c_2^\top x)$ for non-efficient solutions and efficient solutions $x \in X$.

the solution is obtained as the minimum of Problem (2.6). The number of solutions which can not be obtained by the weighted-sum method can even be of exponential size. In this thesis we are interested in multicriteria problems with an efficient set X_E of (pseudo)polynomial size, which can be calculated in (pseudo)polynomial time. For most of the problems in Section 2.4 the latter requirements are not fulfilled. One exception is the minimum cut problem.

Theorem 2.49 ([9])**.** For a fixed number m of objective functions, the set of efficient solutions X_E for the multicriteria minimum cut problem can be calculated in pseudopolynomial time.

If the set X_E has exponential size one may also be interested in approximations of this set, i.e. subsets of X_E such that each efficient solution is approximated by one of the solutions in the subset. Formally the latter approximation concept for multicriteria problems is described as follows.

Definition 2.50. For an instance of a multicriteria problem of the form (2.5) with positive objective values $c_i^\top x$ for each feasible solution $x \in X$, we say an algorithm $\mathcal{A}$ is an *ε-approximation algorithm* if it calculates a set $F \subseteq X$ of solutions such that for each efficient solution $x \in X$ there exists a solution $y \in F$ with

$$c_i^\top y \leq (1+\varepsilon) c_i^\top x \quad \forall i = 1, \ldots, m.$$

A multicriteria problem has a *fully polynomial approximation scheme* if for any $\varepsilon > 0$ there exists an ε-approximation algorithm whose run-time as well as the maximum size of any number occurring in the computation is bounded by a polynomial in $\langle x \rangle + \langle \varepsilon \rangle + \frac{1}{\varepsilon}$. We then say problem (2.5) admits an FPTAS.

Note that the polynomial run-time of the algorithm includes that the calculated set F has polynomial size. In literature several approximation methods

have been presented. Two general approaches to obtain approximation algorithms are local search methods in the objective space and population based methods ([31]). Furthermore it has been proved that some of the combinatorial problems presented in Section 2.4 admit an FPTAS.

Theorem 2.51 ([50, 34])**.** For a fixed number m of objective functions Problem (2.5) admits an FPTAS for the shortest path problem, the minimum spanning tree problem and the knapsack problem.

Chapter 3

Combinatorial Robust Optimization

In this section we will give an introduction to the robust optimization framework and present a selection of different models which are studied in robust optimization literature.

In many real world applications the input data of an optimization problem can be subject to uncertainty. For example the travel times for the shortest path problem, the traveling salesman problem, or the vehicle routing problem can be uncertain because of unknown traffic situations. Other problems can be influenced by measurement or rounding errors while many financial optimization problems use information about uncertain demands or uncertain returns of assets.

In the literature there are three main approaches to tackle uncertainty in optimization problems. On the one hand there is the approach of *stochastic optimization*, which requires a probability distribution on the uncertain data and which aims to optimize the expected value of the objective function [52]. A well known class of stochastical optimization problems are the *stochastic two-stage problems* where the variables are divided into first-stage variables x and second-stage variables y. A stochastic two-stage problem is of the form

$$\min_{x \in X} f(x) + \mathbb{E}_{\mathbb{P}} \left(g(x, \tilde{c}) \right)$$

with feasible set $X \subseteq \mathbb{R}^n$, objective function $f : X \to \mathbb{R}$, a probability vector $\tilde{c} : \Omega \to \mathbb{R}^n$ with probability distribution $\mathbb{P}$ and $\mathbb{E}_{\mathbb{P}}$ denoting the expected value under $\mathbb{P}$. The function $g : X \times \mathbb{R}^n \to \mathbb{R}$ is of the form

$$g(x, c) = \min_{y \in Y(x,c)} h(x, y, c)$$

where $c \in \mathbb{R}^n$ is any outcome of the probability vector $\tilde{c}$ and $Y(x,c) \subseteq \mathbb{R}^m$ is the feasible set which depends on the first-stage solution x and the outcome c. Hence for each first-stage solution x and each outcome c the best second-stage solution is selected for a given objective function h and under all feasible solutions $Y(x,c)$. The expected value of this best second-stage value is optimized over all first-stage solutions x. Often the latter two-stage problem is considered to have uncertainty in the constraints which is modeled by so called *chance-constraints*. A chance constraint is of the form

$$\mathbb{P}\left(f(x,\tilde{a}) > \nu\right) \leq \alpha$$

with $f : X \times \mathbb{R}^n \to \mathbb{R}$ and given parameters $\alpha \in (0,1)$ and $\nu \in \mathbb{R}$. Here $\tilde{a} : \Omega \to \mathbb{R}^n$ is another probability vector with probability distribution $\mathbb{P}$. If we choose $f(x,\tilde{a}) := \tilde{a}^\top x$ then the chance constraint guarantees that the linear constraint $\tilde{a}^\top x \leq \nu$ is violated with probability of at most α. One of the main drawbacks of the stochastic optimization approach in practice is to find a probability distribution which models the uncertain data properly.

The approach of *distributional robustness* avoids the latter problem by assuming that only the expected value or some other parameters of the unknown probability distribution are given. The objective in this approach is to optimize the expected value of the worst-case probability distribution under all which have the required parameters [58, 28]. Formally a *combinatorial distributionally robust problem* is of the form

$$\min_{x \in X} \sup_{\mathbb{P} \in \mathcal{P}} \mathbb{E}_{\mathbb{P}}\left(c^\top x\right) \tag{3.1}$$

where $\mathcal{P}$ is a set of probability distributions of the probability vector c. Often instead of the linear objective function $c^\top x$ other functions are consider e.g. disutility functions (see [42]). The main task in the latter approach is to define the *ambiguity set* $\mathcal{P}$. There are numerous different definitions of ambiguity sets in literature. Often the set includes only probability distributions which have a given expected value and a given support set. Mostly additional parameters on the probability distribution are required e.g. bounds on given moments, bounds on the probability over given subsets of the support set and many more. A very general definition of $\mathcal{P}$ can be found in [58]. Surprisingly many problems of the form (3.1) for appropriate sets $\mathcal{P}$ can be reformulated by using conic optimization duality which yields problems which are often closely related to the *robust optimization approach* which is mainly studied in this thesis.

The *robust optimization approach* was first introduced by Soyster [56] in 1973. The idea is to define an *uncertainty set* U which contains all relevant

scenarios of the uncertain parameters. The aim is to find a solution which is feasible for all scenarios in U and which optimizes the worst-case value over all scenarios in U. The robust optimization approach received increasing attention the first time in the late 1990s. Kouvelis and Yu studied the robust optimization approach for problems with finite sets of scenarios for several combinatorial optimization problems in [46]. Almost at the same time Ben-Tal and Nemirovski analyzed the approach for convex problems where the uncertainty set is a cone or an ellipsoid [14, 15]. Furthermore El Ghaoui et al. studied semi-definite problems and least-square problems with uncertain data [33, 32]. Some years later Bertsimas and Sim introduced budgeted uncertainty sets and described and analyzed what they call the *Price of Robustness* [19].

The main difference in robust optimization compared to stochastic optimization is that we do not assume the uncertain parameters to behave according to a probability distribution. The uncertainty set U, for example, can contain all relevant observed scenarios but can also be modeled as an infinite set. Instead of optimizing the expected value we optimize the worst objective value which a solution can have considering all scenarios in U. For the so called *deterministic problem*

$$\begin{aligned} \min \quad & f_\xi(x) \\ s.t. \quad & x \in X_\xi \end{aligned} \tag{3.2}$$

where the objective function and the feasible set depend on the uncertain parameters $\xi \in U$, the *robust counterpart* of the problem is defined by

$$\begin{aligned} &\min_x \max_{\xi \in U} \quad f_\xi(x) \\ &s.t.\ x \in X_\xi \quad \forall\, \xi \in U. \end{aligned}$$

In other words we are looking for solutions which are feasible for every scenario $\xi \in U$, i.e. x is contained in X_ξ for all $\xi \in U$. Under all these solution we want to minimize the worst case objective value. i.e. we want to minimize the objective function

$$\begin{aligned} g : \bigcap_{\xi \in U} X_\xi &\to \mathbb{R} \\ x &\mapsto \max_{\xi \in U} f_\xi(x). \end{aligned}$$

Depending on the properties of f_ξ, X_ξ and U, many different classes of the latter general problem can be defined and are studied in the robust optimization literature [15]. In this thesis for the deterministic problem we always consider

linear combinatorial problems of the form (M) where the uncertainty only affects the cost vector i.e. problems of the form (3.2) where $f_\xi(x) = \xi^\top x + \xi_0$ and $X_\xi = X$. Since each scenario for (M) in this case is given by an explicit cost vector $(c, c_0) \in U \subseteq \mathbb{R}^{n+1}$, instead of using the parameter ξ we formulate the robust counterpart as problem (M^2).

Uncertainty Sets

In the literature different classes of uncertainty sets are studied. The most common classes will be presented in the following. Since we assume the cost vector $(c, c_0) \in \mathbb{R}^{n+1}$ in (M) to be uncertain, we define all uncertainty sets in the space $\mathbb{R}^{n+1}$. For each $c \in \mathbb{R}^{n+1}$ we always assume the last entry c_{n+1} to be the constant c_0.

A natural way to define scenarios is given by *discrete uncertainty sets*, where $U = \{c_1, \dots, c_m\} \subset \mathbb{R}^{n+1}$ is a finite set of scenarios. Furthermore there are several classes of infinite uncertainty sets. One well-studied class in literature are *polyhedral uncertainty sets* where U is either given by an outer description

$$U = \{c \in \mathbb{R}^n \mid Ac \leq b\}$$

with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, or by an inner description

$$U = \operatorname{conv}(V) + \operatorname{cone}(E)$$

where $V, E \subset \mathbb{R}^{n+1}$ are finite sets. If U is bounded we also say that U is a *polytopal uncertainty set*. As we have seen in Example 2.5 the size of both descriptions can be substantially different and since U is part of the input for most of the algorithms in this thesis we always have to mention by which description U is given. A special case of polyhedral uncertainty sets are the *interval uncertainty sets* where

$$U = [a, b] := \left\{c \in \mathbb{R}^{n+1} \mid a \leq c \leq b\right\}$$

for $a, b \in \mathbb{R}^{n+1}$. As we have seen in Example 2.5, the box U can be described by an outer description given by $2(n+1)$ inequalities or by an inner description given by 2^{n+1} vertices. Another special case of polyhedral uncertainty sets are the so called *budgeted uncertainty sets*. Here for a given parameter $\Gamma \in \mathbb{N}$ the uncertainty set is given by

$$U = \left\{ \tilde{c} = c + \delta^\top d \mid 0 \leq \delta_j \leq 1,\ \forall j = 1, \dots, n+1;\ \sum_{j=1}^{n+1} \delta_j \leq \Gamma \right\}$$

where $c \in \mathbb{R}^{n+1}$ is the *mean vector* and $d \in \mathbb{R}^{n+1}$ the *deviation vector*. Note that in our case, where $X \subseteq \{0,1\}^n$, for minimization problems it suffices to restrict to deviation vectors $d \geq 0$, while for maximization problems we can assume that $d \leq 0$. The idea is to restrict the maximum number of uncertain parameters which are allowed to deviate from their mean value at the same time by a parameter Γ, since it is very unlikely in practice that all uncertain parameters deviate at the same time. The latter set is therefore an extension of interval uncertainty, which can be obtained by choosing $\Gamma = n + 1$.

Another well-studied class of uncertainty sets are *ellipsoidal uncertainty sets* where

$$U = \left\{ c \in \mathbb{R}^{n+1} \mid (c - \bar{c})^\top \Sigma (c - \bar{c}) \leq \Omega^2 \right\}$$

with a symmetric positive definite matrix $\Sigma \in \mathbb{R}^{(n+1)\times(n+1)}$, a given center point $\bar{c} \in \mathbb{R}^{n+1}$ and $\Omega \in \mathbb{R}$. Note that since Σ is positive definite, U is always full-dimensional and bounded. In some applications we also want to allow *flat ellipsoids*, i.e. an ellipsoid which is contained in a lower dimensional affine subspace of $\mathbb{R}^{n+1}$. For example this is the case if we assume that not every entry of the cost vector c is uncertain. Therefore we also allow sets $U' \subset \mathbb{R}^l$ which are the image of a full-dimensional ellipsoid $U \subset \mathbb{R}^{n+1}$ defined like above under an affine embedding $B : \mathbb{R}^{n+1} \to \mathbb{R}^l$ with $l > n + 1$. If U is an axis-parallel ellipsoid then we call U an *uncorrelated ellipsoidal uncertainty set*. Note that for full-dimensional ellipsoids in the definition above U is axis-parallel if and only if Σ is a diagonal matrix.

Robust solutions can be very conservative and not very effective in practice since every scenario in U is considered with the same probability, even if it is very unlikely to happen. Furthermore as we will see in Section 3.1, the robust counterparts are *NP*-hard for many tractable combinatorial problems. To address these drawbacks many new robust models were introduced in literature. A selection of approaches is given in the following.

3.1 Strict Robustness

The main idea of robust optimization, which was already described above, is also known as *strict robustness*. In our case, when the deterministic problem (M) is only affected by uncertainty in the objective function, the *strictly robust counterpart* is the problem

$$\min_{x \in X} \max_{(c,c_0) \in U} c^\top x + c_0. \tag{M^2}$$

Most of the results in literature, which we present in the following sections, are shown for problems of the form

$$\min_{x \in X} \max_{c \in U} c^\top x \tag{M_0^2}$$

i.e. for Problem (M^3) without an uncertain constant. In the following we prove that this case is equivalent to Problem (M^2) for all combinatorial problems which we defined in Section 2.4.

Lemma 3.1. Problem (M^2) is equivalent to problem (M_0^2) for the shortest path problem, the minimum spanning-tree problem and the minimum perfect weighted matching problem.

Proof. Given an instance of problem (M^2), i.e. a graph $G = (V, E)$ such that $X \subseteq \{0,1\}^E$ is the set of incidence vectors of all feasible solutions of the underlying problem and an uncertainty set $U \subseteq \mathbb{R}^{|E|+1}$, define the following instances of the respective problems in dimension $|E| + 1$ by extending G as follows: for the shortest path problem define the graph $G_{SP} = (V_{SP}, E_{SP})$ with $V_{SP} = V \cup \{t'\}$ and $E_{SP} = E \cup \{(t, t')\}$. For the spanning tree problem define $G_{ST} = (V_{ST}, E_{ST})$ with $V_{ST} = V \cup \{w\}$ and $E_{ST} = E \cup \{\{v, w\}\}$ for one arbitrary node $v \in V$. For the matching problem define $G_M = (V_M, E_M)$ with $V_M = V \cup \{v, w\}$ and $E_M = E \cup \{\{v, w\}\}$ (see Figure 3.1).

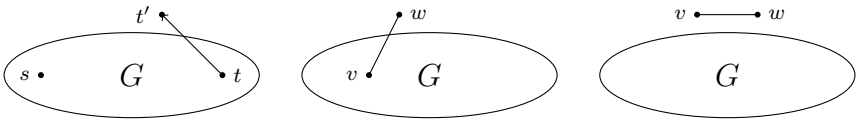


Figure 3.1: The graphs G_{SP}, G_{ST} and G_M.

Note that the new graphs are constructed such that each feasible solution must use the added edge. Hence an optimal solution of problem (M^2) for the respective problem can be obtained by solving (M_0^2) on the respective new graph with the same uncertainty set U and projecting the optimal solution to the edges of G. Here the shortest path problem has to be solved for the new target node t'. On the other hand any instance of Problem (M_0^2) can be solved by solving problem (M^2) with $U' = U \times \{0\}$. Note that for all uncertainty classes defined in the previous section U' remains in the same class as U. This also holds if we consider general convex uncertainty sets U. □

The same result holds for further combinatorial problems of Section 2.4 if the uncertainty set is bounded.

Lemma 3.2. If U is bounded, then Problem (M^2) is equivalent to Problem (M_0^2) for the minimum (s-t-)cut problem, the knapsack problem and the unconstrained binary problem.

Proof. Let $U \subseteq B_R(0)$ for a radius $R > 0$. Given an instance of problem (M^2) for the respective underlying problem in dimension n, define the following instances for the respective problems in dimension $n+1$. For the minimum (s-t-)cut problem extend the given connected graph $G = (V, E)$ to the graph $G_C = (V_C, E_C)$ with $V_C = V \cup \{w\}$ and $E_C = E \cup \{\{v, w\}\}$ where $v \in V$ is an arbitrary node (see Figure 3.2). Note that G_C is still connected.

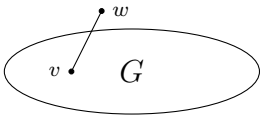


Figure 3.2: The graph G_C.

For the instance of the knapsack problem, given by weights $a \in \mathbb{R}^n$ and a capacity b, define the new instance by the weight-vector $\bar{a} = (a, 0) \in \mathbb{R}^{n+1}$ and capacity $\bar{b} = b$. For the unconstrained binary problem just add one dimension. First note, that for any feasible solution $x \in X$ of the original instance the vectors $(x, 1)$ and $(x, 0)$ are feasible for the new generated instances in dimension $n+1$. We define

$$U' = \{(c, c_0 - (R+1)) \mid (c, c_0) \in U\}.$$

Then any optimal solution x^* of (M_0^2) over U' and $X' \subseteq \{0,1\}^{n+1}$, where X' is the set of all feasible solutions in dimension $n+1$ of the instances created above, fulfills $x^*_{n+1} = 1$. To see this let x be any solution with $x_{n+1} = 1$ and $y = x - e_{n+1}$. Let $c_x \in U'$ be the maximum scenario of x. Then we have

$$\max_{c \in U'} c^\top y - \max_{c \in U'} c^\top x \geq c_x^\top y - c_x^\top x = -(c_x)_0 + R + 1 > 0$$

and hence y can not be optimal. Therefore we have

$$\min_{x \in X'} \max_{c \in U'} c^\top x + R + 1 = \min_{x \in X} \max_{(c, c_0) \in U} c^\top x + c_0.$$

The other direction is proved by the observation that any instance of Problem (M_0^2) can be solved by solving problem (M^2) with $U' = U \times \{0\}$. □

In the following subsections we give an overview of recent complexity results of Problem (M^2) regarding discrete and convex uncertainty sets.

3.1.1 Discrete Uncertainty

The discrete uncertainty case was intensively studied in [46]. It was shown that for discrete uncertainty sets $U = \{c_1, \dots, c_m\}$ Problem (M_0^2) is weakly *NP*-hard for several combinatorial problems if the number of scenarios is constant and strongly *NP*-hard if the number of scenarios is non-constant i.e. it is part of the input. One exception here is the minimum cut problem. Surprisingly Problem (M_0^2) even has different complexity for the minimum cut and the minimum *s*-*t*-cut problem, while the deterministic versions of both problems are closely related [9, 5]. Recently it was shown that even for the unconstrained binary problem (M_0^2) is *NP*-hard, although the deterministic version can be solved trivially [12]. Furthermore it was shown that for several combinatorial problems (M_0^2) admits an FPTAS if the number of scenarios is constant [3]. An overview of the complexity results regarding the combinatorial problems introduced in Section 2.4 can be found in Table 3.1. All reductions and proofs can be found in the references shown in the table. The results for the shortest path problems were proved for non-negative scenarios and the approximation results were proved for a constant number of scenarios. An overview of the discrete min-max problem can also be found in [7]. Note that all results in the table were shown for Problem (M_0^2) but can be extended to (M^2) by Lemma 3.1, 3.2 and Corollary 3.4. To the best of my knowledge no results exist for the entries marked with a question mark.

Problem	Constant $\|U\|$	Non-constant $\|U\|$	Approximation
Shortest Path	weakly *NP*-hard [46]	strongly *NP*-hard [46]	FPTAS [3]
Spanning Tree	weakly *NP*-hard [46][3]	strongly *NP*-hard [46]	FPTAS [3]
Assignment	*NP*-hard [46]	strongly *NP*-hard [4]	?
Knapsack	weakly *NP*-hard [46]	strongly *NP*-hard [46]	FPTAS [3]
Min-Cut	polynomial [9]	strongly *NP*-hard [5]	FPTAS
Min *s*-*t*-Cut	strongly *NP*-hard [5]	strongly *NP*-hard [5]	?
Unconstrained	*NP*-hard [12]	*NP*-hard [12]	?

Table 3.1: Complexity of problem (M^2) for discrete U. All *NP*-hardness results for constant $|U|$ even hold for $|U| = 2$.

To extend the approximation results in [3] to Problem (M^2), we will prove the following theorem, which was already proved in [3] for Problem (M_0^2).

Theorem 3.3. For a given uncertainty set $U = \{(c_1, (c_1)_0), \dots, (c_m, (c_m)_0)\}$ Problem (M^2) has a fully polynomial approximation scheme if the multicriteria problem

$$\min_{x \in X} \left(c_1^\top x, \dots, c_m^\top x\right)$$

has a fully polynomial approximation scheme.

Proof. Without loss of generality we may assume that $(c_i)_0 \geq 0$ for all $i = 1, \ldots, m$, since otherwise, if we define

$$M := \max_{i=1\ldots,m} \{|(c_i)_0|\}$$

and $U' = U + Me_{n+1}$ we have

$$\min_{x \in X} \max_{(c,c_0) \in U} c^\top x + c_0 = -M + \min_{x \in X} \max_{(c,c_0) \in U'} c^\top x + c_0$$

and $(c_i)_0 + M \geq 0$ for all $i = 1, \ldots, m$. By assumption for any $\varepsilon > 0$ there exists an ε-approximation algorithm for the multicriteria problem, i.e. there exists an algorithm which calculates a set $F \subseteq X$ in polynomial time in $n + \langle U \rangle + \langle \varepsilon \rangle + \frac{1}{\varepsilon}$, such that for each efficient solution $x \in X$ there exists a solution $y \in F$ with

$$c_i^\top y \leq (1 + \varepsilon) c_i^\top x \quad \forall i = 1, \ldots, m.$$

Note that F must have polynomial size since it has been calculated in polynomial time. At least one optimal solution x of (M^2) must be an efficient solution of the multicriteria problem since, if not, there exists an efficient solution $y \in X$ which dominates x and therefore

$$c_i^\top y + (c_i)_0 \leq c_i^\top x + (c_i)_0$$

for all $i = 1 \ldots, m$. But then y is optimal for (M^2) which is a contradiction. So let x^* be an optimal solution of (M^2) which is efficient. Then there exists a $y \in F$ with

$$c_i^\top y \leq (1 + \varepsilon) c_i^\top x^* \quad \forall i = 1, \ldots, m.$$

Now choose the solution $z \in F$ with minimum value $\max_{(c,c_0) \in U} c^\top z + c_0$ under all solutions in F, which can be done in polynomial time in the input. Then we have

$$\begin{aligned} \max_{(c,c_0) \in U} c^\top z + c_0 &\leq \max_{(c,c_0) \in U} c^\top y + c_0 \\ &\leq \max_{(c,c_0) \in U} (1 + \varepsilon) c^\top x^* + c_0 \\ &\leq (1 + \varepsilon) \left(\max_{(c,c_0) \in U} c^\top x^* + c_0 \right) \end{aligned}$$

while the last inequality holds since $c_0 \geq 0$. Therefore z is a an ε-approximate solution of (M^2). □

Corollary 3.4. Problem (M^2) admits an FPTAS for the shortest path problem, the minimum spanning tree problem and the knapsack problem if the number of scenarios is fixed.

Proof. By Theorem 2.51 the multicriteria versions of the listed problems have fully polynomial approximation schemes for a constant number of linear objective functions. The result follows by Theorem 3.3. □

3.1.2 Convex Uncertainty

The strictly robust problem (M_0^2) has been intensively studied for ellipsoidal uncertainty sets in [14, 15, 20] and for budgeted uncertainty in [19, 18, 55]. It is well known that for general ellipsoidal and polyhedral uncertainty sets Problem (M_0^2) is at least as hard as the same problem for discrete uncertainty sets. In this section we will present extensions to Problem (M^2) for two well-known proofs of the latter two results. On the other hand we will show some special cases for which Problem (M^2) can be solved in polynomial time.

Polyhedral Uncertainty

If we consider polyhedral uncertainty sets $U = \{c \in \mathbb{R}^n \mid Ac \leq b\}$ then the strictly robust problem (M^2) can be reformulated as

$$\begin{aligned} \min \quad & z \\ s.t. \quad & \max_{(c,c_0)\in U} c^\top x + c_0 \leq z \\ & x \in X,\ z \in \mathbb{R}. \end{aligned}$$

The maximum expression in the first constraint is a linear program with variables (c, c_0) and can therefore be dualized (see Section 2.1.2) which leads to the equivalent constraint

$$\min_{\substack{y^\top A = (x^\top, 1)^\top \\ y \in \mathbb{R}^m_+}} y^\top b \leq z\ .$$

The latter constraint is fulfilled if and only if there exists a feasible solution y of the minimization problems which has objective value $y^\top b \leq z$. Hence

Problem (M^2) is equivalent to the problem

$$\begin{aligned} \min \quad & z \\ s.t. \quad & y^\top b \leq z \\ & y^\top A = (x^\top, 1)^\top \\ & x \in X,\ z \in \mathbb{R},\ y \in \mathbb{R}^m_+ \end{aligned}$$

which is a linear problem with continuous variables z, y and binary variables x. To prove the *NP*-hardness of Problem (M^2) for several combinatorial problems we first show that we can reduce the discrete uncertainty case to the polytopal uncertainty case.

Theorem 3.5. Problem (M^2) with $|U| = 2$ can be reduced to the same problem with polytopal uncertainty given by an inner description or by an outer description.

Proof. Let

$$\min_{x \in X} \max_{(c,c_0) \in U} \quad c^\top x + c_0$$

with $U = \{(a, a_0), (b, b_0)\}$ be an instance of the problem with two scenarios. Then this is equivalent to the same problem with $U = \operatorname{conv}((a, a_0), (b, b_0))$. The latter set is a polytope given by an inner description and can also be described by an outer description of polynomial size which shows the result. □

We now extend the well known result that Problem (M^2_0) is as easy as the underlying deterministic problem if U is a box to Problem (M^3).

Theorem 3.6. Problem (M^2) with interval uncertainty is equivalent to the underlying deterministic problem.

Proof. Let $U = [(a, a_0), (b, b_0)] \subseteq \mathbb{R}^{n+1}$. Since $X \subseteq \{0, 1\}^n$ and hence $x \geq 0$ for any $x \in X$, the maximum over U in direction x is obtained in scenario (b, b_0) (see Figure 3.3), and therefore

$$\min_{x \in X} \max_{(c,c_0) \in U} \quad c^\top x + c_0 = \min_{x \in X} \quad b^\top x + b_0$$

which proves the result. □

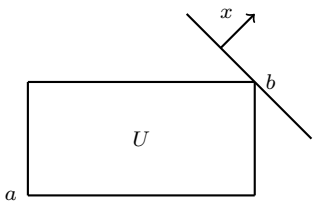


Figure 3.3: Maximization over a box in direction x.

The latter theorem includes that Problem (M^2) with interval uncertainty can be solved in polynomial time for all problems discussed in Section 2.4, except for the knapsack problem.

Another important special case of polyhedral uncertainty sets are the budgeted uncertainty sets, which were first introduced by Bertsimas and Sim in [19]. The interval uncertainty approach allows every parameter to vary in a given range independently of each other. This often leads to solutions which are too conservative and far from optimal in many scenarios. Furthermore in practice it is very unlikely that all uncertain parameters attain the worst value at the same time. The idea in the budgeted uncertainty approach is to fix a parameter Γ and allow at most Γ many parameters to differ from their nominal value. The parameter Γ controls the so called *price of robustness*, which is the trade-off between the probability that more than Γ uncertain parameters differ from their mean value and the effect to the optimal value. For deterministic combinatorial problems of the type (M), the robust counterpart with budgeted uncertainty is the problem

$$\min_{x \in X} \max_{(c,c_0) \in U^\Gamma} c^\top x + c_0 \tag{BR}$$

where

$$U^\Gamma = \left\{ \tilde{c} = c + \delta^\top d \mid 0 \leq \delta_j \leq 1,\ \forall j = 1, \ldots, n+1;\ \sum_{j=1}^{n+1} \delta_j \leq \Gamma \right\}$$

with $c, d \in \mathbb{R}^{n+1}$ and $\Gamma \in \mathbb{R}_+$. Note that U is a polyhedron and therefore Problem (BR) is a special case of Problem (M^2) with polyhedral uncertainty (see Figure 3.4). The outer description of U^Γ is given by

$$U^\Gamma = \left\{ \tilde{c} \in [c, c+d] \;\middle|\; \sum_{i=1}^{n+1} \tfrac{1}{d_j} \tilde{c}_j \leq \Gamma - \sum_{i=1}^{n+1} \tfrac{c_j}{d_j} \right\}$$

and can therefore be described by $2n+1$ constraints. On the other hand the inner description has exponential size since it can be proved that for each

subset $S \subset \{1, \ldots, n+1\}$ with $|S| = \Gamma$ the point

$$v_S = \begin{cases} c_j + d_j & \text{if } j \in S \\ c_j & \text{otherwise} \end{cases}$$

is a vertex of U^Γ. Therefore U^Γ has $\mathcal{O}(n^\Gamma)$ vertices.

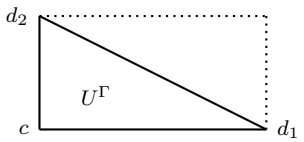


Figure 3.4: The set U^Γ in two dimensions for $\Gamma = 1$.

The important difference to general polyhedral uncertainty is that Problem (BR) can be solved very efficiently for tractable combinatorial problems.

Theorem 3.7 ([18])**.** Problem (BR) for budgeted uncertainty can be solved by solving a deterministic version (M) of the same underlying problem $n+1$ times.

It follows directly from the latter theorem that if the deterministic problem can be solved in polynomial time, then Problem (BR) can be solved in polynomial time. An overview of complexity results of the combinatorial problems described in Section 2.4 for the presented classes of polyhedral uncertainty sets can be found in Table 3.2. The results follow from the theorems in this section together with the results from Table 3.1 in the last section. For the shortest path problem we always assume $U \subseteq \mathbb{R}^n_+$. To the best of my knowledge no complexity results were presented in the literature for the entry marked with a question mark.

Problem	Polyhedral U	Interval U	Budgeted U
Shortest Path	*NP*-hard	polynomial	polynomial
Spanning Tree	*NP*-hard	polynomial	polynomial
Assignment	*NP*-hard	polynomial	polynomial
Knapsack	*NP*-hard	weakly *NP*-hard	weakly *NP*-hard
Min-Cut	?	polynomial	polynomial
Min *s*-*t*-Cut	strongly *NP*-hard	polynomial	polynomial
Unconstrained	*NP*-hard	polynomial	polynomial

Table 3.2: Complexity of Problem (M^2) for polyhedral classes of U.

Ellipsoidal Uncertainty

We now analyze the case where

$$U = \left\{(c, c_0) \in \mathbb{R}^{n+1} \mid ((c, c_0) - (\bar{c}, \bar{c}_0))^\top \Sigma ((c, c_0) - (\bar{c}, \bar{c}_0)) \leq \Omega^2\right\}$$

is an ellipsoid with positive definite matrix $\Sigma \in \mathbb{R}^{(n+1)\times(n+1)}$, center point $(\bar{c}, \bar{c}_0) \in \mathbb{R}^{n+1}$ and $\Omega \in \mathbb{R}$. As we have seen in Example 2.10 the objective function of (M^2) can then be reformulated as

$$\max_{(c,c_0)\in U} c^\top x + c_0 = \bar{c}^\top x + \bar{c}_0 + \Omega\sqrt{(x^\top, 1)\Sigma^{-1}(x^\top, 1)^\top}.$$

Since Σ^{-1} is a positive definite matrix there exists a matrix $B \in \mathbb{R}^{n+1}$ such that $\Sigma^{-1} = B^\top B$ and therefore the latter expression can be reformulated as

$$\bar{c}^\top x + \bar{c}_0 + \Omega\left\|B(x^\top, 1)^\top\right\|_2.$$

Hence (M^2) is equivalent to the problem

$$\begin{aligned} \min \quad & z \\ s.t. \quad & \Omega\left\|B(x^\top, 1)^\top\right\|_2 \leq z - \bar{c}^\top x - \bar{c}_0 \\ & x \in X,\ z \in \mathbb{R} \end{aligned}$$

which is of the form ($\mathcal{SOCP}$) with binary variables x and continuous variable z. As in the polyhedral case this reformulation gives us a possibility to solve Problem (M^2). Before we prove the *NP*-hardness for several combinatorial problems under ellipsoidal uncertainty we give an example for a problem from financial optimization which can be transformed to a problem of the form (M^2) with ellipsoidal uncertainty.

Example 3.8. Robust optimization problems under ellipsoidal uncertainty often occur in financial optimization problems [27]. For example in portfolio optimization the return of a set of n assets can be uncertain. We assume that each asset has a price of w_i and we have a total budget of B. If we assume that the return vector $r \in \mathbb{R}^n$ of the assets is normally distributed with mean $\mu \in \mathbb{R}^n$ and covariance matrix $\Sigma = A^\top A \in \mathbb{R}^{n\times n}$, then one problem could be to find the maximum return value of a portfolio which can be realized by the given budget such that the probability to fall below the return value is at most $\varepsilon > 0$. This problem can be modeled by a binary problem with one knapsack constraint and a chance constraint of the form

$$\begin{aligned} \max \quad & z \\ s.t. \quad & P(r^\top x \leq z) \leq \varepsilon \\ & w^\top x \leq B \\ & x \in \{0, 1\}^n. \end{aligned}$$

The chance constraint is equivalent to the constraint

$$z \leq \mu^\top x + \Phi^{-1}(\varepsilon)\|Ax\|_2 = \mu^\top x - \Phi^{-1}(1-\varepsilon)\sqrt{x^\top \Sigma x}$$

where Φ is the cumulative distribution function of the standard normal distribution. Therefore we can rewrite the problem as

$$\begin{aligned} \max \quad & \mu^\top x - \Phi^{-1}(1-\varepsilon)\sqrt{x^\top \Sigma x} \\ s.t. \quad & w^\top x \leq B \\ & x \in \{0,1\}^n, \end{aligned}$$

which by Example 2.10 is equivalent to

$$\begin{aligned} \max_{x\in\{0,1\}^n} \min_{c\in U} \quad & c^\top x \\ s.t. \quad & w^\top x \leq B \end{aligned}$$

where

$$U = \left\{ c \in \mathbb{R}^n \mid (c-\mu)^\top \Sigma^{-1}(c-\mu) \leq \left(\Phi^{-1}(1-\varepsilon)\right)^2 \right\} .$$

The latter problem is a strictly robust knapsack problem under ellipsoidal uncertainty.

We can show that Problem (M^2) under ellipsoidal uncertainty is *NP*-hard for most of the combinatorial problems we considered in the section before. The following theorem was already proved for Problem (M_0^2) in [55].

Theorem 3.9. Problem (M^2) with $|U| = 2$ can be reduced to the same problem with ellipsoidal uncertainty.

Proof. Given an instance of (M^2) with $U = \{(a, a_0), (b, b_0)\}$ define $U' = \operatorname{conv}((a, a_0), (b, b_0))$. Then U' is an one-dimensional ellipsoid since it is the image of the affine embedding

$$B : \left\{ x \in \mathbb{R} \mid x^2 \leq 1 \right\} \to \mathbb{R}^{n+1}$$

with

$$B(c) = \frac{1}{2}\left((a, a_0) + (b, b_0)\right) + \frac{1}{2}\left((a, a_0) - (b, b_0)\right) x \ .$$

The result then follows analogously to the proof of Theorem 3.5. □

Note that the unconstrained binary problem with general ellipsoidal uncertainty is even strongly *NP*-hard, as it was shown in [12]. An interesting special case is uncorrelated ellipsoidal uncertainty. In this case the the unconstrained

binary problem as well as the spanning-tree problem can be solved in polynomial time. An overview of complexity results of the combinatorial problems described in Section 2.4 for the presented classes of ellipsoidal uncertainty can be found in Table 3.3. If no citations are given, the result follows from Theorem 3.9 in this section together with the results from Table 3.1 in the last section. Again for the shortest path problem we always assume $U \subseteq \mathbb{R}^n_+$. To the best of my knowledge no complexity results were presented in the literature for the entries marked with a question mark.

Problem	Ellipsoidal U	Uncorr. ellips. U
Shortest Path	*NP*-hard	?
Spanning Tree	*NP*-hard	polynomial [49]
Assignment	*NP*-hard	?
Knapsack	*NP*-hard	*NP*-hard
Min-Cut	?	?
Min s-t-Cut	strongly *NP*-hard	?
Unconstrained	strongly *NP*-hard [12]	polynomial [12]

Table 3.3: Complexity of Problem (M^2) for ellipsoidal classes of U.

3.2 Regret Robustness

In this section we introduce the regret robust approach. Most of the results on this topic presented in the literature consider uncertainty which only affects the objective function. The approach has been intensively studied for discrete uncertainty sets in [46] and for interval uncertainty in [11, 43]. A survey can be found in [7]. Besides the afore-mentioned uncertainty sets there are only a few results for ellipsoidal uncertainty sets. Recently the complexity of the regret robust problem for the unconstrained binary problem and the shortest path problem was studied in [26]. The basic idea of regret robustness is to find a solution which minimizes the worst-case deviation of the solution's objective value and the objective value of the optimal solution in a scenario. Formally let x^*_c be the optimal solution of the certain problem

$$\min_{x \in X} c^\top x,$$

for scenario $c \in U$. Then the *regret robust counterpart* is

$$\min_{x \in X} \max_{c \in U} |c^\top x - c^\top x^*_c|. \tag{RR}$$

While for discrete uncertainty sets (RR) has the same complexity as Problem (M^2) for most of the problems introduced in Section 2.4, for interval

uncertainty the regret robust versions are much harder to solve than the strictly robust versions. An overview of the complexity results regarding the combinatorial problems from Section 2.4 can be found in Table 3.2 and in [7]. To the best of my knowledge no complexity results were presented in the literature for the unconstrained binary problem.

Problem	$\|U\|$ constant	$\|U\|$ non-constant	Interval U
Shortest Path	weakly *NP*-hard [46]	strongly *NP*-hard [46]	strongly *NP*-hard [10]
Spanning Tree	weakly *NP*-hard [46][3]	strongly *NP*-hard [3]	strongly *NP*-hard [10]
Assignment	*NP*-hard [46]	strongly *NP*-hard [4]	strongly *NP*-hard [4]
Knapsack	weakly *NP*-hard [46]	strongly *NP*-hard [3]	*NP*-hard
Min-Cut	polynomial [5]	strongly *NP*-hard [5]	polynomial [5]
Min *s*-*t*-Cut	strongly *NP*-hard [5]	strongly *NP*-hard [5]	strongly *NP*-hard [5]
Unconstrained	?	?	?

Table 3.4: Complexity of (RR) for different classes of U.

For general ellipsoidal uncertainty the authors in [26] prove that Problem (RR) for the shortest path problem and the unconstrained binary problem is *NP*-hard. Furthermore they show that it is polynomial for the unconstrained problem if the ellipsoidal uncertainty is uncorrelated.

3.3 Light Robustness

In this section we present the approach of *light robustness* which was first introduced in [35]. The main idea is to fix a nominal scenario $\hat{\xi}$, which can be chosen as the mean scenario of an uncertainty set (e.g. the center of the ellipsoid), and to calculate a solution which is feasible for this scenario with an objective value which deviates at most a given value ρ from the optimal value in this scenario. The objective is then to find a solution which, additionally to the latter condition, minimizes the violation of the constraints for all other scenarios. The optimal lightly robust solution therefore must not be feasible for all scenarios. This approach was motivated by an application in railway timetabling and was combined with the approach of budgeted uncertainty [36, 35]. The constraints of timetabling problems can often include restrictions depending on the delay time, where the delay time is an uncertain parameter. Using a strictly robust approach for a timetabling problem would lead to solutions which are feasible for every possible delay scenario which is contained in U. Since this often yields very conservative solutions which are not efficient in practical applications, the idea of the lightly robust approach is to find a timetable which works well in the mean scenario when no disturbances occur but which is allowed to be slightly infeasible for some delay

scenarios. The authors in [35] analyze the approach for linear programs of the type $(\mathcal{P})$ with budgeted uncertainty which only appears in the constraints. This approach was later extended for general types of optimization problems and arbitrary uncertainty sets in [53]. This approach called *generalized light robustness* assumes a deterministic problem of the form

$$\begin{aligned} \min \quad & f(x,\xi) \\ s.t. \quad & F_i(x,\xi) \leq 0 \quad \forall i = 1,\dots,m \\ & x \in \mathbb{R}^n. \end{aligned}$$

where $\xi \in U$ are the uncertain parameters and F_i and f are functions which map from $\mathbb{R}^n \times U$ to $\mathbb{R}$ for each $i = 1,\dots,m$. The *generalized lightly robust counterpart* of the latter problem is then defined as

$$\begin{aligned} \min \quad & \|\gamma\| \\ s.t. \quad & f(x,\hat{\xi}) \leq \hat{z} + \rho \\ & F_i(x,\hat{\xi}) \leq 0 \\ & F_i(x,\xi) \leq \gamma_i \quad \forall \xi \in U \\ & x \in \mathbb{R}^n,\ \gamma \in \mathbb{R}^m_+ \end{aligned} \tag{GLR}$$

where $\|\cdot\|$ is an arbitrary norm, $\rho \in \mathbb{R}$ and $\hat{z}$ is the optimal value of the deterministic problem in the nominal scenario $\hat{\xi}$. The first two constraints ensure that the calculated solution x is feasible in scenario $\hat{\xi}$ and that its objective value in scenario $\hat{\xi}$ deviates at most ρ from the optimal value in $\hat{\xi}$. The remaining constraints make sure that the violation of the constraint for all scenarios is bounded by the parameters γ_i which are minimized in an arbitrary norm. The author proves the following theorem.

Theorem 3.10 ([53])**.** If the deterministic problem is of the form $(\mathcal{P})$ with polyhedral uncertainty in the constraints and $\|\cdot\|$ is a norm for which the unit ball is a polytope then the generalized lightly robust counterpart of $(\mathcal{P})$ is a linear program. If we consider $(\mathcal{P})$ with ellipsoidal uncertainty in the constraints and if $\|\cdot\|$ is a norm for which the unit ball is an ellipsoid or a polytope, then the lightly robust counterpart of $(\mathcal{P})$ is a second-order cone problem.

Examples of norms for which the unit ball is a polyhedron are

$$\|x\|_1 := \sum_{i=1}^{n} |x_i|$$

and

$$\|x\|_\infty := \max_{i=1,\dots,n} |x_i|$$

while for the Euclidean norm $\|\cdot\|_2$ and the ellipsoidal norm

$$\|x\|_\Sigma := \sqrt{x^\top \Sigma x}$$

with positive definite $\Sigma \in \mathbb{R}^{n\times n}$ the unit ball is an ellipsoid.

3.4 Recoverable Robustness

The approach of *recoverable robustness* was first introduced in [48] and, besides other problems, was used to solve timetabling problems. The approach is similar to the stochastic two-stage approach in the sense that a solution has to be calculated in the first stage and after the scenario is known this solution can be adjusted under certain conditions. Formally the main idea behind this approach is that a set $\mathcal{A}$ of *recovery algorithms* is given such that each algorithm can be applied to a solution x and a scenario ξ to construct a feasible solution for the scenario. The objective is then to find a solution x which is feasible in a nominal scenario $\hat{\xi}$ together with an algorithm $A \in \mathcal{A}$, such that for every scenario $\xi \in U$, the algorithm constructs a feasible solution $A(x,\xi) \in X_\xi$ and the worst-case objective value over all scenarios is optimized. Formally for a given uncertain problem

$$\begin{aligned} \min \quad & f_\xi(x) \\ s.t. \quad & x \in X_\xi \end{aligned}$$

with $\xi \in U$, the *recoverable robust counterpart* is the problem

$$\begin{aligned} \min_{(x,A)\in X_{\hat{\xi}}\times\mathcal{A}} \max_{\xi\in U} \quad & f_\xi(A(x,\xi)) \\ s.t. \quad & A(x,\xi) \in X_\xi \quad \forall \xi \in U. \end{aligned} \tag{RCR}$$

The complexity of the latter problem depends heavily on the underlying problem and on the choice of $\mathcal{A}$.

The concept was later applied to the shortest path problem in [25]. As a motivating example consider a long-term construction process for a railway system or a highway. In the first stage we have to find a construction plan which minimizes the actual cost. But since during the construction process the cost can change, it can be better to change the construction plan after the future price scenario is known. In this case it would be efficient to consider

possible changes of the cost in the future already in the first stage. For the application in [25] in the first stage a path on a graph G is calculated before the scenario is known. Afterwards, when the scenario is known, it is allowed to change at most k edges of the calculated path. Formally the *k-dist recoverable robust shortest path problem* is defined as follows: for a given directed graph $G = (V, A)$ and nodes $s, t \in V$, let $c^1 : \mathcal{P}(A) \to \mathbb{N}$ be the first stage cost, which has to be paid for the implementation of the calculated path before the scenario is known. For each scenario $s \in S$, the cost is given by a function $c^s : \mathcal{P}(A) \to \mathbb{N}$. Let k be the given recovery parameter. Let $\mathcal{P}$ be the set of all s-t-paths in G and for any path $p \in \mathcal{P}$ define

$$P_p^k = \{p' \in \mathcal{P} \mid |p' \setminus p| \leq k\}.$$

Then the problem can be formalized by

$$\min_{p \in \mathcal{P}} \; c^1(p) + \max_{s \in S} \min_{p' \in P_p^k} \; c^s(p') \; . \tag{3.3}$$

Besides other results, the following theorem was proved in [25].

Theorem 3.11 ([25])**.** Problem (3.3) is strongly *NP*-hard for discrete, interval and budgeted uncertainty.

3.5 Adjustable Robustness

Adjustable robustness was first introduced in [13] and is, like recoverable robustness, a robust version of the two-stage approach in stochastic optimization. The idea is to distinguish between *here and now* variables x which have to be calculated in advance and *wait and see* variables y which can be defined after the scenario is known. The x variables are also called *first stage solutions* and y are the *second stage solutions*. The objective is to calculate a solution x in the first stage such that for every possible scenario ξ there exists a solution y such that (x, y) is feasible and such that (x, y) minimizes the worst-case objective value over all scenarios. After the scenario is known the best of the feasible solutions y is chosen. For a deterministic problem of the form

$$\begin{aligned} \min \quad & f_\xi(x, y) \\ s.t. \quad & (x, y) \in X_\xi \times Y_\xi \end{aligned}$$

with $X_\xi \subset \mathbb{R}^m$ and $Y_\xi \subset \mathbb{R}^n$ the *adjustable robust counterpart* is the problem

$$\min_{x \in \bar{X}} \max_{\xi \in U} \min_{y:(x,y) \in X_\xi \times Y_\xi} \quad f_\xi(x, y) \tag{AR}$$

where the feasible set $\bar{X}$ is defined by

$$\bar{X} := \left\{ x \in \bigcap_{\xi \in U} X_\xi \mid \forall \xi \in U \; \exists y \in Y_\xi : (x, y) \in X_\xi \times Y_\xi \right\}.$$

In other words $\bar{X}$ contains all first stage solutions x such that x is feasible for all scenarios ξ and such that for each scenario ξ there exists a second stage solution y such that (x, y) is feasible for the scenario ξ.

Example 3.12. Consider the network design problem where for a given graph $G = (V, E)$ each edge $e \in E$ has a maximal capacity u_e with total price c_e. Furthermore d_{ij} are the uncertain demands of units which have to be sent from node i to j. Each unit which is sent from i to j yields a profit of p_{ij}. In the first stage a fraction x_e of u_e can be bought by paying the price $x_e c_e$. This fraction $x_e u_e$ can then be used in the second stage to send f_e units over the edge where $f_e \leq x_e u_e$ must hold. In the second stage when the demands d_{ij} are known we can calculate the flow which satisfies all demands and all capacities $x_e u_e$ and which has the maximal profit. Therefore the problem can be formulated as an adjustable robust problem with first stage variables x_e and second stage variables f_e [17].

The adjustable robust problem is hard to solve in general, that is why there are several variations and special cases of the problem which are considered in the literature. One approach, which was presented in [13], is called *affinely adjustable robustness*. In this approach the authors assume the wait and see variables y to be given by an affine function of the uncertainty parameters, i.e. $y = a + A\xi$ with $A \in \mathbb{R}^{n \times q}$ and $a \in \mathbb{R}^n$. We consider the problem

$$\begin{aligned} \min \quad & c^\top x \\ s.t. \quad & T_\xi x + W y \leq h_\xi, \end{aligned} \tag{3.4}$$

where $\xi \in U \subset \mathbb{R}^q$, $T_\xi \in \mathbb{R}^{l \times m}$, $W \in \mathbb{R}^{l \times n}$ and $h_\xi \in \mathbb{R}^l$. Here the uncertainty parameters only occur in T_ξ and h_ξ and we assume that T_ξ and h_ξ are affine linear in ξ. The authors prove the following theorem:

Theorem 3.13 ([13])**.** The adjustable robust counterpart of Problem (3.4), where the second stage variables y are affine functions of the uncertain parameters, is equivalent to the strictly robust problem

$$\begin{aligned} \min \quad & c^\top x \\ s.t. \quad & B_\xi x + b_\xi \geq 0 \quad \forall \xi \in U \end{aligned}$$

for a matrix $B_\xi \in \mathbb{R}^{l' \times m}$ and $b_\xi \in \mathbb{R}^{l'}$ and it has the same complexity as the strictly robust problem for any U.

For combinatorial problems of the form

$$\begin{aligned} \min \quad & c^\top x + d^\top y \\ s.t. \quad & (x, y) \in X \times Y \end{aligned}$$

where $X \subseteq \{0,1\}^m$ and $Y \subseteq \{0,1\}^n$ without uncertainty in the constraints and $(c,d) \in U \subseteq \mathbb{R}^{m+n}$, the adjustable robust counterpart reduces to the min-max-min problem

$$\min_{x \in X} \max_{(c,d) \in U} \min_{y \in Y} \; c^\top x + d^\top y.$$

Note that Problem (M^3) has the same min-max-min structure and will be analyzed later. In fact (M^3) is a special case of the min-max-min problem discussed in the following section.

3.5.1 K-Adaptability

As mentioned before the adjustable robust counterpart (RCR) is hard to solve in general. One approach to approximate the latter problem is called *K-adaptability* and was first introduced in [16] for general linear problems with uncertainty in the objective function and in the constraints. The idea of the approach is to choose K second stage solutions $y_1, \ldots, y_K$ in the first stage and then choose the best of them after the scenario is revealed. The authors analyze the gaps between the adjustable robust counterpart, the K-adaptability problem and the strictly robust problem in [16] and give necessary conditions to obtain a certain improvement. They also present a bilinear optimization formulation to solve the 2-adaptability problem and prove that the latter problem is *NP*-hard. Later the approach of K-adaptability was analyzed for binary problems in [41]. We consider problems of the form

$$\begin{aligned} \min \quad & \xi^\top C x + \xi^\top Q y \\ s.t. \quad & Tx + Wy \leq h \\ & (x, y) \in X \times Y \end{aligned}$$

where $\xi \in U = \{\xi \in \mathbb{R}^q \mid A\xi \leq b\}$ are the uncertain parameters, $X \subseteq \mathbb{R}^m$ is a bounded polyhedral set, $Y \subseteq \{0,1\}^n$, $C \in \mathbb{R}^{q \times m}$, $Q \in \mathbb{R}^{q \times n}$, $T \in \mathbb{R}^{l \times m}$, $W \in \mathbb{R}^{l \times n}$ and $h \in \mathbb{R}^l$. The *K-adaptability robust counterpart* is then the problem

$$\begin{aligned} \min \; & \max_{\xi \in U} \min_{k=1,\ldots,K} \xi^\top C x + \xi^\top Q y^{(k)} \\ s.t. \; & Tx + Wy^{(k)} \leq h \quad k = 1, \ldots, K \\ & x \in X, \; y^{(k)} \in Y \quad k = 1, \ldots, K \end{aligned} \tag{K-AR}$$

for a given positive integer K. Besides other results the authors prove the following theorem.

Theorem 3.14 ([41])**.** For all

$$K \geq \min\{n, \operatorname{rank} Q\} + 1,$$

Problem (K-AR) has the same optimal value as the adjustable robust counterpart

$$\min_{x \in X} \max_{\xi \in U} \min_{y \in Y : Tx + Wy \leq h} \xi^\top Cx + \xi^\top Qy. \tag{3.5}$$

Proof. Assume $n \leq \operatorname{rank} Q$ and let $K = |Y| < \infty$. Then Problem (3.5) and (K-AR) are equivalent. The objective function of (K-AR) can be written as

$$\begin{aligned} &\max_{\xi \in U} \min_{\lambda \in \mathbb{R}^K_+ : e^\top \lambda = 1} \xi^\top Cx + \sum_{k=1}^{K} \lambda_k \xi^\top Q y^{(k)} \\ = &\max_{\xi \in U} \min_{\lambda \in \mathbb{R}^K_+ : e^\top \lambda = 1} \xi^\top Cx + \xi^\top Q \sum_{k=1}^{K} \lambda_k y^{(k)}. \end{aligned}$$

Since

$$y := \sum_{k=1}^{K} \lambda_k y^{(k)} \in \operatorname{conv}\left(y^{(1)}, \ldots, y^{(K)}\right)$$

and by the Theorem of Caratheodory 2.3 it follows that for any point y it suffices to choose $n+1$ solutions in $\left\{y^{(1)}, \ldots, y^{(K)}\right\}$ to describe y as a convex combination. Since all $y^{(i)}$ are chosen in the first stage it suffices to set $K = n + 1$ to reach the same optimal value which proves the theorem. The case rank $Q < n$ can be proved analogously by considering the convex combination

$$y := \sum_{k=1}^{K} \lambda_k Q y^{(k)}.$$

□

Furthermore the authors in [41] provide a mixed-integer linear program formulation for Problem (K-AR).

Theorem 3.15 ([41])**.** The K-adaptability robust counterpart (K-AR) is equivalent to problem

$$\begin{aligned}
&\min\ b^\top \alpha \\
&s.t.\ \ A^\top \alpha = Cx + \sum_{k=1,\dots,K} Qz^k \\
&\qquad \lambda^\top \mathbf{1} = 1 \\
&\qquad Tx + Wy^k \leq h \quad k = 1,\dots,K \\
&\qquad z^k \leq y^k,\ z^k \leq \lambda_k \mathbf{1} \quad k = 1,\dots,K \\
&\qquad z^k \geq (\lambda_k - 1)\,\mathbf{1} + y^k \quad k = 1,\dots,K \\
&\qquad x \in X, \alpha \in \mathbb{R}^r_+,\ \lambda \in \mathbb{R}^K_+ \\
&\qquad y^k \in Y,\ z^k \in \mathbb{R}^n \quad k = 1,\dots,K\ .
\end{aligned}$$

Problem (M^3), which will be studied in Chapters 4 and 5, is the special case of the Problem (K-AR) where no first stage exists, i.e. $X = \{0\}$. In Section 4.1.1 we will show that for convex U this case is tractable for all tractable combinatorial problems if we choose $K \geq n + 1$. Additionally we show that this problem is *NP*-hard for any fixed K and for polyhedral uncertainty, which also proves the *NP*-hardness of Problem (K-AR).

3.6 Bulk Robustness

The concept of *Bulk Robustness* was presented in [1] and studied for the shortest path problem and the minimum matroid basis problem. In contrast to the previous models, this approach considers a set of failure scenarios, where each failure scenario is a set of edges which can break down simultaneously. The aim is to calculate a set of edges such that, if we remove the edges of any failure scenario from this set, it still contains the edge set of a feasible solution of the combinatorial problem. As an application consider a railway system for which a shortest path has to be calculated. Because of possible constructions or accidents it can happen that a section of the railway system is not passable anymore. In this case we can use the bulk robust idea to calculate a set of sections which always contain a feasible path no matter which of the failure scenarios occurs. The optimal solution of the bulk robust counterpart is a set of edges and must not be a solution of the deterministic problem as it is the case in the previously presented robust approaches.

Given a deterministic combinatorial problem of the form (M) we define $\bar{X} \subseteq \{0,1\}^n$ as the set of feasible solutions of the bulk robust counterpart. We

assume that $X \subseteq \bar{X}$ and that if $x_1 \in \bar{X}$, then for any $x_2 \geq x_1$ also $x_2 \in \bar{X}$. Hence a feasible solutions in $\bar{X}$ must not be feasible for the deterministic combinatorial problem as it was the case in the previous sections. But any feasible solution of the bulk robust problem is always a superset of a feasible solution of the deterministic problem. Given a cost vector $c \in \mathbb{R}^n$ and a set of failure scenarios $\Omega = \{F_1, \ldots, F_m\}$, where $F_i \subseteq \{1, \ldots, n\}$ for each $i = 1, \ldots, m$, the *bulk robust counterpart* is defined as

$$\begin{aligned} &\min\ c^\top x \\ &s.t.\ x_{F_i} \in \bar{X} \quad \forall F_i \in \Omega \\ &\qquad x \in \{0,1\}^n \end{aligned} \tag{BR}$$

where we define

$$(x_{F_i})_j := \begin{cases} 0 & \text{if } j \in F_i \\ x_i & \text{otherwise.} \end{cases}$$

As an example consider the graph in Figure 3.5 with failure scenarios $\bar{\Omega} = \{\{e_1\}, \ldots, \{e_5\}\}$ i.e. in any scenario exactly one edge is not passable. The optimal solution of the related bulk robust problem is shown in Figure 3.5. Besides other results in [1] the authors present a polynomial-time approxima-

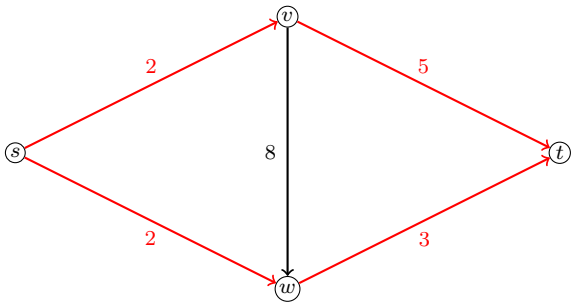


Figure 3.5: The optimal bulk-robust solution for the shortest path problem and failure scenarios $\bar{\Omega}$ with cost 12.

tion algorithm with an approximation guarantee of $\mathcal{O}(\log m + \log r)$ for the minimum matroid basis problem. Furthermore a polynomial-time $\mathcal{O}(\log m)$-approximation algorithm for the shortest path problem for fixed

$$k = \max_{F \in \Omega} |F|$$

is presented.

A two-stage version of the bulk robust approach was already presented in [2] for the shortest-path problem. The idea is that in a second stage, after the

failure scenario is known, it is allowed to add r edges to the pre-calculated edge-set to connect the given nodes s and t. Formally for a given graph $G = (V, A)$ and $s, t \in V$ and $f, r \in \mathbb{Z}_+$, the objective is to find a set of edges $X \subseteq A$ of minimum cardinality which connects s and t such that for any $F \subseteq A$ with $|F| \leq f$ there exists $R \subseteq A$ with $|R| \leq r$ such that s and t are connected in $(X \setminus F) \cup R$. In other words the set of failure scenario here is $\Omega = \{F \subset A \mid |F| \leq f\}$ and in the second stage it is possible to add at most r arbitrary edges which are not contained in the actual scenario F to obtain a feasible path. The authors show that the latter problem in its general version is NP-hard and that there does not exist a polynomial-time α-approximation algorithm for the problem if $\alpha < 2$ unless $P = NP$. Besides other results they present a polynomial-time algorithm for the case $f = r = 1$.

Chapter 4

Min-max-min Robustness under Convex Uncertainty

The main contribution of this thesis is the analysis of problem

$$\min_{x^{(1)},\dots,x^{(k)}\in X} \max_{(c,c_0)\in U} \min_{i=1,\dots,k} c^\top x^{(i)} + c_0 \qquad (\mathrm{M}^3)$$

where $X \subseteq \{0,1\}^n$ is the set of incidence vectors of all feasible solutions of the given deterministic problem (M) and $U \subseteq \mathbb{R}^{n+1}$ is an uncertainty set which contains all relevant cost-vectors $(c, c_0) \in \mathbb{R}^{n+1}$ of the deterministic problem. The motivation behind this approach is to calculate $k \in \mathbb{N}$ solutions in a perhaps expensive preprocessing before the actual scenario is known. This preprocessing has to be performed only once and afterwards, when the scenario is known, the best of the calculated solutions for the actual scenario can be implemented. In contrast to the standard robust approach the objective function for any scenario attains the best value of all k solutions under the respective scenario which leads to a better objective value in general. Like in the standard robust approach the worst case over all scenarios is considered. Note that for $k = 1$ we obtain the min-max Problem (M^2).

Obviously there is a relation between adjustable robustness and Problem (M^3) since both problems solve a similar min-max-min problem. In fact Problem (M^3) is the special case of the K-adaptability problem (K-AR) when no first stage exists i.e. if $X = \{0\}$. This case is appropriate for many combinatorial problems since many problems are modeled using only one stage in general. As we will see, this case is even very interesting, since it is as easy as the underlying problem if we calculate $k \geq n + 1$ solutions. This is exactly the number of solutions for which the K-adaptability problem yields the exact value of the adjustable robust problem (see Theorem 3.14). In this

thesis we do not consider the more general objective function $c^\top Qx$ as in Problem (K-AR) since we can replace the latter objective function by $d^\top x$ with $d \in Q^\top U$. The set $Q^\top U$ remains convex if U is convex and therefore all results in this section hold for sets of the form $Q^\top U$. The same idea holds for the discrete case in the next chapter. Although Problem (M^3) is a special case of K-adaptability, we propose to use it as a one-stage robust model to solve combinatorial problems under uncertainty.

In this section we analyze the complexity of problem (M^3) for convex uncertainty sets U, which among others include polyhedral or ellipsoidal uncertainty sets. We will show that Problem (M^3) has the same complexity as the underlying problem if we calculate $k \geq n+1$ solutions and if we can linearly optimize over U. In contrast to this we will also show that the problem for any fixed $k \in \mathbb{N}$ is *NP*-hard even for a polyhedral uncertainty set given by an inner description and for the unconstrained binary problem. We will also present an exact algorithm to solve the problem for $k \geq n+1$ and a heuristic algorithm for any $k < n+1$, which is mainly based on the algorithm of the former case. An overview of the complexity results for convex uncertainty sets can be found in Table 4.1. Again, for the shortest path problem we assume that $U \subseteq \mathbb{R}^{n+1}_+$. Surprisingly Problem ($M^3$) with $k \geq n+1$ can be solved in polynomial time for every combinatorial problem which can be solved in polynomial time, which is in contrast to the *NP*-hardness of Problem (M^2) for most of the problems (see Table 3.2 and 3.3). This even holds for general convex uncertainty sets over which we can optimize linearly. Nevertheless, if k is part of the input or if k is fixed, Problem (M^3) is *NP*-hard. Most of the results in this section were already published in [23] and are joint work with Christoph Buchheim.

	$k \geq n+1$	k constant	k input
Problem	Convex U	Polyhedral U (i.d.)	Convex U
Shortest Path	polynomial	?	*NP*-hard
Spanning Tree	polynomial	?	*NP*-hard
Assignment	polynomial	?	*NP*-hard
Knapsack	weakly *NP*-hard	*NP*-hard	*NP*-hard
Min-Cut	polynomial	?	?
Min s-t-Cut	polynomial	?	strongly *NP*-hard
Unconstrained	polynomial	*NP*-hard	strongly *NP*-hard

Table 4.1: Complexity of Problem (M^3) for convex U.

The results in this section are based on the following reformulation which also follows from a generalization of the proof of Theorem 1 in [41].

Lemma 4.1. Let $U \subseteq \mathbb{R}^{n+1}$ be a non-empty convex set. Then

$$\min_{x^{(1)},\dots,x^{(k)} \in X} \max_{(c,c_0)\in U} \min_{i=1,\dots,k} c^\top x^{(i)} + c_0 = \min_{x \in X(k)} \max_{(c,c_0)\in U} c^\top x + c_0$$

where

$$X(k) := \left\{ \sum_{i=1}^k \lambda_i x^{(i)} \mid \lambda_i \geq 0,\ \sum_{i=1}^k \lambda_i = 1,\ x^{(i)} \in X \ \text{ for } i = 1, \dots k \right\}$$

is the set of all convex combinations of k elements of X.

Proof. The main idea of the proof is to calculate the Lagrangian dual problem (see Section 2.2.1) of the inner maximization problem of (M^3). To this end, for any fixed $x^{(1)}, \dots, x^{(k)}$ we reformulate the problem by a level set transformation as

$$\max_{(c,c_0)\in U} \min_{i=1,\dots,k} c^\top x^{(i)} + c_0 = \begin{array}{ll} \max & z \\ s.t. & c^\top x^{(i)} + c_0 \geq z \ \ \forall i = 1, \dots, k \\ & (c, c_0) \in U,\ z \in \mathbb{R}. \end{array}$$

The Lagrange dual function of the latter convex problem is

$$\begin{aligned} L(\lambda) &= \sup_{(c,c_0)\in U, z\in\mathbb{R}} z + \sum_{i=1}^k \lambda_i \left(c^\top x^{(i)} + c_0 - z \right) \\ &= \sup_{(c,c_0)\in U, z\in\mathbb{R}} z \left(1 - \sum_{i=1}^k \lambda_i \right) + \sum_{i=1}^k \lambda_i (c^\top x^{(i)} + c_0) \\ &= \begin{cases} \max_{(c,c_0)\in U} c^\top \sum_{i=1}^k \lambda_i x^{(i)} + c_0, & \text{if } \sum_{i=1}^k \lambda_i = 1 \\ \infty, & \text{otherwise} \end{cases} \end{aligned}$$

Therefore the Lagrange dual problem is

$$\begin{aligned} &\min_{\lambda \geq 0} \max_{(c,c_0)\in U} \ c^\top \sum_{i=1}^k \lambda_i x^{(i)} + c_0 \\ &s.t. \ \sum_{i=1}^k \lambda_i = 1 \end{aligned}$$

and by substituting the latter expression into Problem (M^3) we obtain exactly the problem given in the Lemma. Obviously Slater's condition is fulfilled for the primal problem, since z can be chosen arbitrarily in $\mathbb{R}$. Therefore strong duality holds which proves the result. □

From now on, we will thus consider the problem

$$\min_{x \in X(k)} \max_{(c,c_0) \in U} c^\top x + c_0 \qquad (\mathrm{M}^2[k])$$

instead of Problem (M^3). Note that $X(1) = X$ and for each $k \in \mathbb{N}$ we have $X(k) \subseteq X(k+1)$ (see Figure 4.1). Therefore the objective value of Problem ($\mathrm{M}^2[k]$) decreases or remains constant with growing k.

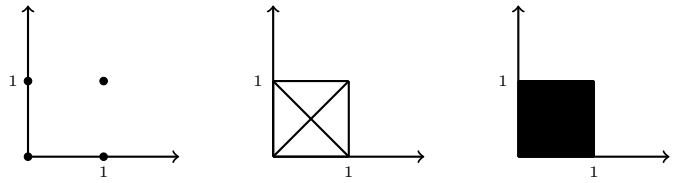


Figure 4.1: The sets $\{0,1\}^2(1)$, $\{0,1\}^2(2)$ and $\{0,1\}^2(3)$.

Using the result from Lemma 4.1 we propose the following algorithm to solve Problem (M^3).

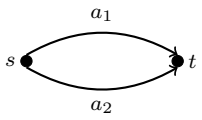
Algorithm 1 Algorithm to solve Problem (M^3) for $k \in \mathbb{N}$

Input: convex $U \subseteq \mathbb{R}^{n+1}$, $X \subseteq \{0,1\}^n$, $k \in \mathbb{N}$
Output: optimal solution of Problem (M^3)
1: calculate an optimal solution x^* of Problem ($\mathrm{M}^2[k]$).
2: determine $x^{(1)}, \ldots, x^{(k)} \in X$ such that $x^* \in \operatorname{conv}\left(x^{(1)}, \ldots, x^{(k)}\right)$.
3: **return** $x^{(1)}, \ldots, x^{(k)}$

Clearly following the proof of Lemma 4.1, Algorithm 2 calculates an optimal solution of Problem (M^3). It follows directly from Theorem 2.31 that Step (2) only depends on the deterministic problem and can be calculated in polynomial time if the deterministic problem can be solved in polynomial time and if x^* has polynomial size. Note that the algorithm which proves the latter theorem in [40] computes a convex combination with the smallest possible m. Hence for any k and $x^* \in X(k)$, the algorithm returns at most k solutions in X and therefore the related optimal solution of (M^3). The remaining task and the main part of Algorithm 1 which we are interested in is therefore to calculate the optimal solution x^* of problem ($\mathrm{M}^2[k]$). This step depends, besides the set X, on the uncertainty set U. The complexity of this step for different uncertainty sets will be analyzed in this thesis. Note that solving ($\mathrm{M}^2[k]$) already yields the optimal value of Problem (M^3).

First we show in the following example that the difference of the objective values of Problem (M^3) and Problem (M^2) can be arbitrary large.

Example 4.2. Consider the graph $G = (V, A)$ with $V = \{s, t\}$ and $A = \{a_1, a_2\}$ where $a_i = (s, t)$ for each i



together with the ellipsoidal uncertainty set $U_\alpha = \left\{c \in \mathbb{R}^2 \mid c^\top \Sigma_\alpha c \leq 1\right\}$ with positive definite matrix

$$\Sigma_\alpha = \begin{pmatrix} 4 + \frac{1}{\alpha} & 2 - \frac{2}{\alpha} \\ 2 - \frac{2}{\alpha} & 1 + \frac{4}{\alpha} \end{pmatrix} = 9(x^*)(x^*)^\top + \tfrac{1}{\alpha} v v^\top$$

for $\alpha \geq 1$. Here $x^* = (\frac{2}{3}, \frac{1}{3})^\top$ and $v = (-1, 2)^\top$. The corresponding ellipsoid is shown in Figure 4.2. The parameter α can be used to scale the ellipsoid in direction of its extreme ray v.

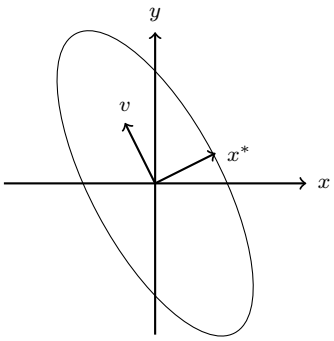


Figure 4.2: The ellipsoid U_α.

The set of all paths in G can be modeled by $X = \{e_1, e_2\}$ where e_i is the i-th unit vector. As in Example 2.10, for any $x \in \mathbb{R}^n$ we can reformulate the maximum over the ellipsoid U by

$$\max_{c \in U} c^\top x = \sqrt{x^\top \Sigma_\alpha^{-1} x}. \tag{4.1}$$

The optimal solution of (M^2) is then e_1 with an optimal value of $\frac{1}{5}\sqrt{4 + \alpha}$ which can be arbitrarily large for $\alpha \geq 1$. A feasible solution of Problem ($M^2[k]$) with $k = 2$ is

$$x^* = \left(\tfrac{2}{3}, \tfrac{1}{3}\right)^\top = \tfrac{2}{3} e_1 + \tfrac{1}{3} e_2 \in X(2)$$

with an objective value of $\frac{1}{3}$ for any α. Therefore the difference between the optimal values of (M^2) and (M^3) can be arbitrarily large if α is increased.

4.1 The Case $k \geq n+1$

If we consider the case $k \geq n+1$, then from Theorem 2.3 we immediately obtain that $X(k) = \text{conv}\,(X)$. This proves the following Corollary.

Corollary 4.3. For $k \geq n+1$ and for each non-empty convex set U we have

$$\min_{x^{(1)},\ldots,x^{(k)} \in X} \max_{(c,c_0)\in U} \min_{i=1,\ldots,k} c^\top x^{(i)} + c_0 = \min_{x \in \text{conv}(X)} \max_{(c,c_0)\in U} c^\top x + c_0 \,.$$

In the special case where no uncertain constant c_0 is considered, it holds

$$\max_{c\in U} c^\top(\lambda x) = \lambda \max_{c\in U} c^\top x$$

and therefore an optimum is obtained over the boundary of $\text{conv}\,(X)$. Since the boundary of $\text{conv}\,(X)$ is contained in $X(n)$, Corollary 4.3 even holds for $k \geq n$ in the case where no uncertain constant is considered.

In general Corollary 4.3 implies that the objective value of Problem (M^3) does not improve if we consider more than $n+1$ solutions, which was also shown in [41]. On the other hand, calculating $k = n+1$ solutions is a reasonable choice since, after the scenario is known, the best of the pre-calculated $n+1$ solutions can be determined in time $\mathcal{O}(n^2)$ by calculating the $n+1$ values $c^\top x^{(1)} + c_0, \ldots, c^\top x^{(n+1)} + c_0$ for the actual scenario (c, c_0). Furthermore as we will see the problem

$$\min_{x \in \text{conv}(X)} \max_{(c,c_0)\in U} c^\top x + c_0 \qquad (M^2[n+1])$$

can be solved efficiently if the deterministic problem can be solved efficiently. By Algorithm 1 we then obtain an efficient algorithm for Problem (M^3). In the following sections we will investigate the complexity of Problem ($M^2[n+1]$) for convex uncertainty sets and finally give another practical and oracle-based algorithm to solve Problem (M^3) for any combinatorial problem.

Remark 4.4. If $k \geq n+1$, then Problem ($M^2[n+1]$) is a convex problem and therefore the Minimax Theorem 2.14 can be applied. From this we obtain the equality

$$\begin{aligned}\min_{x \in \text{conv}(X)} \max_{(c,c_0)\in U} c^\top x + c_0 &= \max_{(c,c_0)\in U} \min_{x \in \text{conv}(X)} c^\top x + c_0 \\ &= \max_{(c,c_0)\in U} \min_{x \in X} c^\top x + c_0.\end{aligned}$$

Hence there always exists a scenario $(c, c_0) \in U$ such that the optimal value of Problem (M^3) is equal to the optimal value of the deterministic problem in scenario (c, c_0).

4.1.1 Complexity

In this section we will analyze the complexity of Problem (M^3) for the case $k \geq n + 1$. In contrast to the *NP*-hardness of the strictly robust problem for many combinatorial problems (see Table 3.2 and 3.3) we show that Problem (M^3) is solvable in polynomial time for general convex uncertainty sets U whenever the deterministic problem is solvable in polynomial time, if $k \geq n + 1$ and if we can linearly optimize over U. The latter property holds for polyhedral and ellipsoidal uncertainty sets.

Polyhedral Uncertainty

For polyhedral uncertainty sets U, we show that Problem (M^3) has the same complexity as the underlying problem if we choose $k \geq n+1$. Clearly if the underlying deterministic problem is *NP*-hard, then Problem (M^3) is *NP*-hard, since we can solve the deterministic problem with cost vector c by choosing $U = \{c\}$ in Problem (M^3). On the other hand, if the deterministic problem can be solved in polynomial time, then we can solve (M^3) in polynomial time which follows from the theorem below.

Theorem 4.5. Given an optimization oracle for the problem

$$c \mapsto \min_{x \in X} c^\top x,$$

for any polyhedron

$$U = \{(c, c_0) \in \mathbb{R}^{n+1} \mid A(c^\top, c_0)^\top \leq b\}$$

with $A \in \mathbb{Q}^{m \times (n+1)}$ and $b \in \mathbb{Q}^m$ and $k \geq n+1$ we can solve (M^3) in polynomial time in $\langle U \rangle$.

Proof. The basic idea of the proof is to dualize the inner maximization problem in ($M^2[n + 1]$) to derive a linear minimization problem, which by Theorem 2.29 can be solved in polynomial time if we can separate the corresponding feasible set in polynomial time. Let $A(c^\top, c_0)^\top = \bar{A}c + ac_0$. Then Problem ($M^3$) is equivalent to

$$\min_{x \in \text{conv}(X)} \max \{c^\top x + c_0 \mid \bar{A}c + ac_0 \leq b,\ (c, c_0) \in \mathbb{R}^{n+1}\} \tag{4.2}$$

for $k \geq n+1$ by Corollary 4.3. The dual problem of the inner maximization problem is

$$\begin{aligned} \min_{y \geq 0} \quad & b^\top y \\ s.t. \quad & y^\top \bar{A} = x \\ & y^\top a = 1 \end{aligned}$$

which can be derived by the results in Section 2.2.1. Substituting into Problem (4.2) we obtain the problem

$$\min_{(x,y) \in P} \; b^\top y \tag{4.3}$$

with

$$P := \{(x,y) \in \mathbb{R}^n \times \mathbb{R}^m \mid x \in \operatorname{conv}(X),\ y^\top \bar{A} = x,\ y^\top a = 1,\ y \geq 0\}\ .$$

By Theorem 2.29 the latter problem can be solved in polynomial time in $\langle P \rangle_F$, if we can solve the separation problem over P in polynomial time. By assumption we have an oracle for the deterministic problem, which yields an oracle for the equivalent problem

$$\min_{x \in \operatorname{conv}(X)} \; c^\top x.$$

Hence, again using Theorem 2.29, we can solve the separation problem over $\operatorname{conv}(X)$ in polynomial time in $\langle \operatorname{conv}(X) \rangle_F = 3n^3 + n$. On the other hand

$$Q := \{(x,y) \in \mathbb{R}^n \times \mathbb{R}^m \mid y^\top \bar{A} = x,\ y^\top a = 1,\ y \geq 0\}$$

is a rational polyhedron and every point (x,y) can be separated by checking whether all equations are satisfied and if not choosing the corresponding equation as separating hyperplane. This can be done in polynomial time in $\langle U \rangle$. Both algorithms together yield a polynomial time separation algorithm for P which proves that we can solve (4.2) in polynomial time in $\langle U \rangle$. The optimal solution then is rational and has polynomial size in the input, and by Theorem 2.31 the result follows. □

Proposition 4.6. Assume we have an outer description of $\operatorname{conv}(X)$, i.e.

$$\operatorname{conv}(X) = \{x \in \mathbb{R}^n \mid Tx \leq w\}\,,$$

and $T \in \mathbb{Q}^{l \times n}$ and $w \in \mathbb{Q}^l$ have polynomial size. Then following the latter proof we obtain a linear problem in (4.3) which can be solved by the ellipsoid method in polynomial time in $\langle T \rangle + \langle w \rangle + \langle U \rangle$. Note that the result in the theorem is much more general since every algorithm which solves the deterministic problem in polynomial time yields a polynomial time algorithm for (M^3) even if the outer description above for $\operatorname{conv}(X)$ does not exist or is not known.

General Convex Uncertainty

In this section we prove that Problem (M^3) with general convex uncertainty has the same complexity as the underlying problem if we choose $k \geq n+1$. By the same reasoning as for polyhedral uncertainty sets Problem (M^3) is *NP*-hard if the underlying deterministic problem is *NP*-hard. In the following we show that if the deterministic problem can be solved in polynomial time, we can solve (M^3) in polynomial time if the following assumptions are fulfilled: we assume that U is a non-empty convex set for which we have a weak optimization oracle, i.e., for a given $x \in \mathbb{Q}^n$ and a rational $\varepsilon > 0$ we can compute in polynomial time a vector $(c, c_0) \in U \cap \mathbb{Q}^{n+1}$ with

$$c^\top x + c_0 \geq d^\top x + d_0 - \varepsilon \quad \text{for all } (d, d_0) \in U.$$

Furthermore, we assume that U is bounded by a constant M, i.e.

$$\|(c, c_0)\|_2 \leq M \quad \text{for all } (c, c_0) \in U.$$

Note that the latter assumptions hold for polyhedral and ellipsoidal uncertainty sets. In contrast to the case of polyhedral uncertainty, for general convex uncertainty sets we will define an accuracy parameter $\varepsilon > 0$ for the optimal value, since calculations involving irrational numbers can only be done up to a given accuracy in finite time.

Theorem 4.7. Let $\operatorname{conv}(X)$ be full-dimensional, $\varepsilon \in (0, 1)$ and U as above. Given an optimization oracle for the certain problem

$$c \mapsto \min_{x \in X} c^\top x,$$

we can solve Problem (M^3) up to an accuracy of at most ε in time polynomial in $\langle n \rangle + \langle M \rangle + \langle \varepsilon \rangle$ if $k \geq n + 1$.

To prove the latter theorem we need the following lemmata.

Lemma 4.8. Let $f : \mathbb{R}^n \to \mathbb{R}$ be defined by

$$f(x) := \max_{(c,c_0) \in U} c^\top x + c_0$$

where U is a convex set bounded by M. If $x, y \in \mathbb{R}^n$ with $\|x - y\| \leq \varepsilon$ then

$$f(x) - f(y) \leq M\varepsilon \, .$$

Proof. Let $x, y \in \mathbb{R}^n$ with $\|x - y\| \leq \varepsilon$. We can reformulate

$$c^\top x + c_0 = c^\top (x - y) + c^\top y + c_0$$

and hence

$$\max_{(c,c_0)\in U} c^\top x + c_0 \leq \max_{(c,c_0)\in U} c^\top (x - y) + \max_{(c,c_0)\in U} c^\top y + c_0.$$

By the definition of f and the Cauchy-Schwarz inequality we obtain

$$f(x) - f(y) \leq \max_{(c,c_0)\in U} \|c\|\|x - y\| \leq M\varepsilon$$

which proves the result. □

The proof of Theorem 4.7 is mainly based on the result of Theorem 2.33. The latter theorem calculates a solution which is (up to an accuracy of $\varepsilon > 0$) optimal over all elements deep in the feasible set. To verify theorem 4.7 we need to prove optimality over all feasible solutions. To this end we prove in the following lemma that if we can optimize over all elements deep in a convex set K with an arbitrary accuracy then we can optimize over all elements in K with an arbitrary accuracy.

Lemma 4.9. Let f and U be defined as in Lemma 4.8. Let K be any convex set for which we know $R > 0$ with $K \subseteq B_R(0)$ and for which we know that K contains a ball with radius r. Additionally, let $0 < \varepsilon < r$ and $x^* \in B_\varepsilon(K)$ such that for all $y \in B_{-\varepsilon}(K)$

$$f(x^*) \leq f(y) + \varepsilon.$$

Then for all $y \in K$

$$f(x^*) \leq f(y) + \varepsilon \left(1 + \tfrac{2R}{r} M\right) .$$

Proof. For all $0 < \varepsilon < r$ formula (0.1.14) in [40], states that for all $y \in K$ there exists a point $z_y \in B_{-\varepsilon}(K)$ such that $\|z_y - y\| \leq \frac{2R}{r}\varepsilon$. From Lemma 4.8 we obtain

$$f(z_y) - f(y) \leq M\tfrac{2R}{r}\varepsilon .$$

By our assumption on x^*, we derive

$$f(x^*) \leq f(z_y) + \varepsilon \leq f(y) + \varepsilon \left(1 + \tfrac{2R}{r} M\right).$$

which proves the result. □

Proof of Theorem 4.7. Since $k \geq n + 1$, Problem (M^3) is equivalent to

$$\min_{x \in \text{conv}(X)} \max_{(c,c_0) \in U} c^\top x + c_0$$

by Corollary 4.3. Define

$$f(x) := \max_{(c,c_0) \in U} c^\top x + c_0$$

like above. As the maximum of affine-linear functions, f is a convex function. The basic idea of the proof is to use Theorem 2.33 to calculate a point $x^* \in B_{\varepsilon'}(\text{conv}\,(X))$ which is optimal over all elements $y \in B_{-\varepsilon'}(K)$ up to an accuracy of ε' for an appropriately defined ε'. Then we use Lemma 4.9 to obtain optimality over all elements in K and afterwards we use Lemma 2.34 to round x^* to a point in $\text{conv}\,(X)$.

As $X \subseteq \{0,1\}^n$ and $\text{conv}\,(X)$ is full-dimensional, there exist values $r, R > 0$ such that

$$B_r(x_0) \subseteq \text{conv}\,(X) \subseteq B_R\,(0)$$

for a rational center point $x_0 \in \text{conv}\,(X)$. By the results in [40] all parameters r, R, x_0 can be determined in polynomial time and have polynomial size in n i.e. $\text{conv}\,(X)$ is a centered convex body. For an integer φ, with $2^{-3\varphi} \leq \frac{\varepsilon}{2M}$ we define

$$\varepsilon' := \min \left\{ \frac{\frac{\varepsilon}{2}}{1 + \frac{2R}{r} M}, 2^{-6n\varphi} \right\}$$

which has encoding length polynomial in $\langle n \rangle + \langle M \rangle + \langle \varepsilon \rangle$ since φ can be chosen such that $\langle \varphi \rangle$ is polynomial in $\langle \varepsilon \rangle + \langle M \rangle$. Since we have an optimization oracle for the deterministic problem, from Theorem 2.29 follows that we can solve the separation problem for $\text{conv}\,(X)$ in polynomial time (see also Example 2.30) which also yields an oracle for the membership problem. Hence by the latter results and the assumptions on U we can use Theorem 2.33 to compute a rational vector $x^* \in B_{\varepsilon'}(\text{conv}\,(X))$ with $f(x^*) \leq f(y) + \varepsilon'$ for all $y \in B_{-\varepsilon'}(\text{conv}\,(X))$, in polynomial time. To prove the theorem we have to show that x^* can be rounded to a vector $x' \in \text{conv}\,(X)$ such that the inequality holds for all $y \in \text{conv}\,(X)$. First we may assume $M, R \geq 1$, then $\varepsilon' < r$ so that we can apply Lemma 4.9 and it follows that

$$f(x^*) \leq f(y) + \varepsilon'(1 + \tfrac{2R}{r} M) \leq f(y) + \tfrac{\varepsilon}{2} \tag{4.4}$$

holds for all $y \in \text{conv}\,(X)$. To round x^* to a point in $\text{conv}\,(X)$ we use Lemma 2.34 which provides a polynomial time algorithm in the input and $\langle \varphi \rangle$ to calculate $q \in \mathbb{Z}$ and $w \in \mathbb{Z}^n$ with $\|qx^* - w\| < 2^{-3\varphi}$ and $1 \leq q < 2^{4n\varphi}$ such

that $x' := \frac{1}{q}w$ is contained in $\operatorname{conv}(X)$ if $x^* \in B_{2^{-6n\varphi}}(\operatorname{conv}(X))$. The latter condition is true since $\varepsilon' \leq 2^{-6n\varphi}$ (see Figure 4.3). Moreover

$$\|x^* - x'\| = \tfrac{1}{q}\|qx^* - w\| < \tfrac{1}{q}2^{-3\varphi}$$

and by the choice of φ above we obtain $2^{-3\varphi} \leq \frac{\varepsilon}{2M}$. Hence $\|x^* - x'\| < \frac{\varepsilon}{2M}$ and therefore by Lemma 4.8 we have $f(x') - f(x^*) \leq \frac{\varepsilon}{2}$. Together with inequality (4.4) we obtain

$$f(x') \leq \frac{\varepsilon}{2} + f(x^*) \leq f(y) + \varepsilon$$

for all $y \in \operatorname{conv}(X)$. Since x' is rational and $\langle x' \rangle$ is polynomial in the input, from Theorem 2.31 follows the result. □

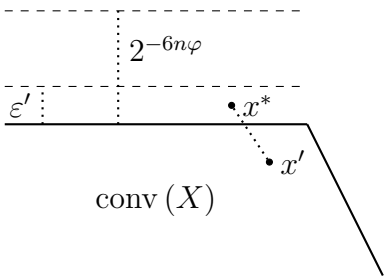


Figure 4.3: The rounding procedure in the proof of Theorem 4.7. The nearly optimal point $x^* \in B_\varepsilon(\operatorname{conv}(X))$ is rounded to the point $x' \in \operatorname{conv}(X)$.

In fact the result of the theorem even holds if $\operatorname{conv}(X)$ is not full-dimensional. As explained in Section 6.1.2 in [40] we can achieve a separation algorithm (and therefore a membership algorithm) for lower dimensional sets by the following idea which we explain here for the case if $\operatorname{conv}(X)$ is $(n-1)$-dimensional. In [39] it was shown that we can calculate a hyperplane $H \subseteq \mathbb{R}^n$ which contains $\operatorname{conv}(X)$ in polynomial time by using a combination of the ellipsoid method and diophantine approximation. Then every point which is not contained in the hyperplane can be separated by using the hyperplane itself. So the problem is reduced to the full-dimensional separation problem in dimension $n-1$ which is equivalent to the optimization problem in dimension $n-1$. The latter results can then be applied to the appropriate lower-dimensional space with a full-dimensional feasible set.

Proposition 4.10. Assume we have an outer description of $\operatorname{conv}(X)$, i.e.

$$\operatorname{conv}(X) = \{x \in \mathbb{R}^n \mid Tx \leq w\},$$

and let $T \in \mathbb{Q}^{l\times n}$ and $w \in \mathbb{Q}^l$ have polynomial size. Then as in Example 2.10 we can reformulate Problem ($\mathrm{M}^2[n+1]$) as

$$\min_{\{x\in\mathbb{R}^n | Tx\leq w\}} \bar{c}^\top x + \bar{c}_0 + \Omega\sqrt{(x^\top, 1)\Sigma^{-1}\begin{pmatrix} x \\ 1 \end{pmatrix}}$$

for ellipsoidal uncertainty sets of the form

$$U = \left\{(c, c_0) \in \mathbb{R}^{n+1} \mid ((c, c_0) - (\bar{c}, \bar{c}_0))^\top \Sigma ((c, c_0) - (\bar{c}, \bar{c}_0)) \leq \Omega^2\right\}.$$

Analogously to Section 3.1.2 this can be transformed to problem

$$\begin{array}{ll} \min & z \\ s.t. & \Omega\left\|B(x^\top, 1)^\top\right\|_2 \leq z - \bar{c}^\top x - \bar{c}_0 \\ & Tx \leq w \\ & x \in \mathbb{R}^n,\ z \in \mathbb{R} \end{array}$$

where B is the matrix such that $\Sigma^{-1} = B^\top B$. The latter problem is of the form ($\mathcal{SOCP}$) which can be solved in polynomial time in $\langle B\rangle + \langle\Omega\rangle + \langle(\bar{c}, \bar{c}_0)\rangle + \langle T\rangle + \langle w\rangle$ by the interior point method up to an arbitrary accuracy. Note that as it is the case for polyhedral uncertainty sets the result in the theorem is much more general.

4.1.2 Exact Algorithm

The construction in the proof of Theorem 4.7 is mainly based on the results in [40] which make use of the ellipsoid method. Thus the implicit algorithm given in the proof is not very practical. In this section we present an algorithm which is mainly based on the idea of column generation and which works very well in practice though it is not guaranteed to have polynomial run-time. In fact the main idea of the algorithm was already presented in [21]. The algorithm uses two oracles, one for solving the deterministic problem (M) in Step (5) and one for optimizing over U in Step (4). Hence the algorithm works for any uncertainty set for which the latter problem can be solved, e.g. for polyhedral and ellipsoidal uncertainty. Furthermore we can use every combinatorial algorithm for solving the deterministic problem and therefore no polyhedral description of conv (X) is needed.

The algorithm, stated below, calculates besides the optimal solution x^* of Problem ($\mathrm{M}^2[k]$) the corresponding solution set $\{x^{(1)}, \ldots, x^{(n+1)}\}$ of Prob-

lem (M^3) and the coefficients $\lambda_1, \ldots, \lambda_{n+1}$ of the related convex combination

$$x^* = \sum_{i=1}^{n+1} \lambda_i x^{(i)}.$$

The latter are important for the heuristic algorithm in the next section, which is based on Algorithm 2.

Algorithm 2 Algorithm to solve Problem (M^3) for $k \geq n+1$

Input: $U \subset \mathbb{R}^{n+1}$, $X \subseteq \{0,1\}^n$
Output: optimal solution of Problem (M^3) and Problem ($M^2[n+1]$)

1: $i := 0$
2: choose any $x_0^* \in X$
3: **repeat**
4: calculate optimal solution (z^*, c^*, c_0^*) of
$$\max \{z \mid c^\top x_j^* + c_0 \geq z \ \forall j = 0, \ldots, i, \ z \in \mathbb{R}, \ (c, c_0) \in U\}$$
5: calculate optimal solution x_{i+1}^* of
$$\min \{(c^*)^\top x + c_0^* \mid x \in X\}$$
6: $i := i + 1$
7: **until** $(c^*)^\top x_i^* + c_0^* \geq z^*$
8: calculate a basic feasible solution of the linear system
$$z^* - c_0^* = \textstyle\sum_{j=0}^{i} \lambda_j (c^*)^\top x_j^*, \ \sum_{j=0}^{i} \lambda_j = 1, \ \lambda \geq 0$$
9: $X^* := \{x_j^* \mid \lambda_j > 0, \ j = 0, \ldots, i\}$
10: **return** X^* and $x^* := \sum_{j=0}^{i} \lambda_j x_j^*$

Theorem 4.11. Algorithm 2 is correct and terminates in finite time.

Proof. First note that for every subset $X' \subseteq X$, by Theorem 2.14 and by level set transformation we have

$$\begin{aligned}
\min_{x \in \operatorname{conv}(X')} \max_{(c,c_0) \in U} c^\top x + c_0 &= \max_{(c,c_0) \in U} \min_{x \in \operatorname{conv}(X')} c^\top x + c_0 \\
&= \max_{(c,c_0) \in U} \{z \mid z \leq c^\top x + c_0 \ \forall x \in \operatorname{conv}(X')\} \\
&= \max_{(c,c_0) \in U} \{z \mid z \leq c^\top x + c_0 \ \forall x \in X'\}.
\end{aligned}$$

After the termination of the loop in the algorithm $(c^*)^\top x + c_0^* \geq z^*$ holds for all $x \in X$. Hence

$$\max_{(c,c_0) \in U} \{z \mid z \leq c^\top x_j^* + c_0 \ \forall j = 0, \ldots, i\} = \max_{(c,c_0) \in U} \{z \mid z \leq c^\top x + c_0 \ \forall x \in X\}$$

and therefore, by the equivalence above

$$\min_{x \in \operatorname{conv}\left(x_0^*, \ldots, x_i^*\right)} \max_{(c,c_0) \in U} c^\top x + c_0 = \min_{x \in \operatorname{conv}(X)} \max_{(c,c_0) \in U} c^\top x + c_0.$$

Any point $x^* \in \operatorname{conv}(x_0^*, \ldots, x_i^*)$ which has objective value z^* in scenario (c^*, c_0^*) is an optimal solution of the problem on the left hand side and hence of Problem ($\mathrm{M}^2[n+1]$). Note that x^* calculated by the algorithm has all the latter properties which follows from the calculations in Step 8. Since λ is a basic solution of the linear system in Step 8, at most $n+1$ of the variables λ_j are strictly positive and hence $|X^*| \leq n+1$. Finite run-time follows directly from the finiteness of X. □

Note that the dual problem which has to be solved in Step (4) is continuous. In particular, for polyhedral uncertainty the problem is a linear problem and for ellipsoidal uncertainty a quadratic constrained problem. These problems can be solved very efficiently by standard solvers like CPLEX. In Section 6 the results of our implementations of the algorithm for some combinatorial problems are presented.

4.2 The Case $k < n + 1$

In the latter section we showed that Problem (M^3) is as easy as the underlying problem if we choose $k \geq n + 1$. A natural question is: what is the complexity of Problem (M^3) if $k < n + 1$? Here we have to distinguish between the cases where k is part of the input or not. Clearly, if k is part of the input, then the case $k = 1$ is a special case of the problem. From Table 3.2 and 3.3 we then obtain that Problem (M^3) with ellipsoidal and polyhedral uncertainty is *NP*-hard for most of the problems in Section 2.4. On the other hand we will show in this section if k is fixed then Problem (M^3) is *NP*-hard, even for polyhedral uncertainty sets given by an inner description and for the unconstrained binary problem. Note that both linear optimization over a polyhedron given by an inner description and solving the deterministic unconstrained binary problem can be done very efficiently. Even in this elementary case Problem (M^3) for any fixed k is *NP*-hard.

4.2.1 Complexity

Before we prove the latter result we consider the special case of interval uncertainty.

Proposition 4.12. If $U = [(a, a_0), (b, b_0)] \subset \mathbb{R}^{n+1}$, then for any fixed k, Problem (M^3) is equivalent to the deterministic problem (M).

Proof. Clearly for any $x \in X$ the worst case in Problem (M^3) is always obtained in scenario (b, b_0) since $X \subseteq \{0,1\}^n$ and therefore $x \geq 0$. Hence Problem ($M^2[k]$) is equivalent to the deterministic problem

$$\min_{x \in X} b^\top x + b_0.$$

which proves the result. □

From now on instead of (M^3) we consider the equivalent Problem ($M^2[k]$) for fixed k. As a special case of the latter problem we will show that the problem

$$\min_{x \in \{0,1\}^n(k)} \max_{(c,c_0) \in U} c^\top x + c_0 \,, \qquad (M_B^2[k])$$

is *NP*-hard for any fixed k if

$$U = \operatorname{conv}(v_1, \ldots, v_r) + \operatorname{cone}(w_1, \ldots, w_s)$$

with vectors $v_1, \ldots, v_r, w_1, \ldots, w_s \in \mathbb{R}^{n+1}$. In the main proof below we solve Problem ($M_B^2[k]$) over $X(k) \cap B$ instead of $X(k)$, where B is an appropriate box. In other words, we add linear inequalities to the feasible set $X(k)$. The following Lemma shows that adding linear inequalities does not make the problem harder.

Lemma 4.13. Let $Y \subseteq \mathbb{R}^n$. Then the problem

$$\min_{x \in Y,\, Ax \leq b} \max_{(c,c_0) \in U} c^\top x + c_0 \qquad (4.5)$$

with $U = \operatorname{conv}(v_1, \ldots, v_r) + \operatorname{cone}(w_1, \ldots, w_s)$ is equivalent to the problem

$$\min_{x \in Y} \max_{(c,c_0) \in V} c^\top x + c_0 \qquad (4.6)$$

with $V = \operatorname{conv}(v_1, \ldots, v_r) + \operatorname{cone}\big(w_1, \ldots, w_s, (a_1, -b_1)^\top, \ldots, (a_m, -b_m)^\top\big)$.

Proof. Given an instance of Problem (4.5), let x be any solution of Problem (4.6). If $x \notin P := \{x \in \mathbb{R}^n \mid Ax \leq b\}$, then there exists a row a_i of A with $a_i^\top x > b_i$. Hence for any $\lambda \geq 0$

$$\max_{(c,c_0) \in V} c^\top x + c_0 \geq \lambda \left(a_i^\top x - b_i\right) > \lambda$$

and therefore $\max_{(c,c_0)\in V} \; c^\top x + c_0 = \infty$. On the other hand, for any $x \in P$ and for any scenario

$$(c, c_0) = v + \sum_{i=1}^{s} \mu_i w_i + \sum_{i=1}^{m} \lambda_i \, (a_i, -b_i) \in V$$

where $\lambda_i, \mu_i \geq 0$ and $v \in \operatorname{conv}(v_1, \ldots, v_r)$, the objective value is

$$c^\top x + c_0 = v^\top x + \sum_{i=1}^{s} \mu_i w_i^\top x + \sum_{i=1}^{m} \lambda_i \left(a_i^\top x - b_i\right) \leq v^\top x + \sum_{i=1}^{s} \mu_i w_i^\top x.$$

Therefore the optimal solution is contained in P and its worst case scenario is obtained in U, which proves the result. □

Remark 4.14. Following the proof we observe that the latter lemma is even true if we replace $\operatorname{conv}(v_1, \ldots, v_r)$ by an arbitrary convex set $C \subseteq \mathbb{R}^{n+1}$.

Note that if we add an exponential number of constraints to the problem, then Lemma 4.13 states that we also have to add an exponential number of vectors to U to obtain an equivalent problem without additional constraints.

Remark 4.15. The latter lemma allows to add problem-specific constraints to the original problem. For example if we want to solve (M^3) for the shortest path problem and if we want all paths to use a certain edge e, we can solve the problem over $Y = X_{SP}(k)$ and add the constraint $x_e = 1$. On the other hand assume we have failure scenarios like in the bulk robust approach, e.g. we know a set of edges $\Omega \subseteq E$ such that the failure scenarios are $F_e = \{e\}$ for all $e \in \Omega$ (see Section 3.6). Then we could calculate a set of k paths such that for each edge $e \in \Omega$ there always exists at least one of the paths, which does not use the edge. This could be modeled by solving the problem over $Y = X_{SP}(k)$ and adding the constraints $x_e \leq 1 - \varepsilon$ for all $e \in \Omega$ for an appropriate small $\varepsilon > 0$. Let $\bar{E} \subseteq E$ be the set which contains all edges which are used of at least one $x^{(i)}$ in the solution of the min-max-min problem. Then $\bar{E}$ is a feasible solution to the bulk robust shortest path problem.

The following result was already shown in Section 3.1.2 but can also be obtained as a corollary of the latter lemma.

Corollary 4.16. Problem ($\mathrm{M}_B^2[1]$) is *NP*-hard.

Proof. The deterministic knapsack problem 2.45 can be formulated as

$$\min_{\substack{x\in\{0,1\}^n \\ a^\top x \leq b}} \; \max_{(c,c_0)\in U} \; c^\top x + c_0$$

with $U = \{(c, 0)\}$. Applying Lemma 4.13 with $Y = \{0,1\}^n$ to the latter reformulation, we can model the deterministic knapsack problem in the form of Problem ($\mathrm{M}_B^2[1]$) with a polyhedral uncertainty set. □

The basic idea to prove that Problem ($\mathrm{M}_B^2[k]$) is *NP*-hard is to reduce ($\mathrm{M}_B^2[k]$) to ($\mathrm{M}_B^2[k+1]$) and to conclude by induction together with Corollary 4.16.

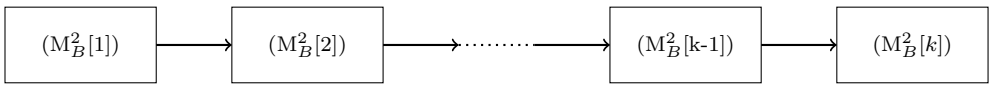


Figure 4.4: Reduction of ($\mathrm{M}_B^2[1]$) to ($\mathrm{M}_B^2[k]$)

In the proof below we will refer to the following result which was proved in [40].

Lemma 4.17 ([40])**.** Let P be a polyhedron with facet-complexity of at most φ and let $y \in \mathbb{Q}^n$ be a vector whose entries have a common denominator which is bounded by q. If $y \in B_{\frac{1}{q}2^{-2\varphi}}(P)$ then $y \in P$.

In other words, if the facets of a polyhedron have a bounded encoding length, then any point which is not contained in P must have at least a certain distance to the polyehdron, depending on the common denominator of the entries. We can now prove the following theorem which is the basic result of this section.

Theorem 4.18. For any $k \in \mathbb{N}$, Problem ($\mathrm{M}_B^2[k]$) can be polynomially reduced to Problem ($\mathrm{M}_B^2[k+1]$).

Proof. The idea of the proof is to reduce minimization over $\{0,1\}^n(k)$ to minimization over $\{0,1\}^{n+1}(k+1)$. Applying Lemma 4.17 to the center point $\bar{c} := \frac{1}{2}\mathbf{1} \in \{0,1\}^{n+1}(k+1)$ and the polyhedron $P := \operatorname{conv}\left(x^{(1)}, \ldots, x^{(k+1)}\right)$, which has facet-complexity of at most $3(n+1)^3$ by Example 2.7, we obtain that either $\bar{c} \in P$ or the Euclidean distance between $\bar{c}$ and P is at least $2^{-6(n+1)^3-1}$. In particular since

$$\max\{|x_1|, \ldots, |x_{n+1}|\} \geq \frac{1}{n+1}\|x\|_2$$

we obtain from the latter observation that each set $\operatorname{conv}\left(x^{(1)}, \ldots, x^{(k+1)}\right)$ either contains $\bar{c}$ or does not intersect the box

$$B := [\bar{c} - \tfrac{1}{2}d\mathbf{1}, \bar{c} + \tfrac{1}{2}d\mathbf{1}] \cap \{x \in \mathbb{R}^{n+1} \mid x_{n+1} = \tfrac{1}{2} - \tfrac{1}{2}d\}$$

where $d := \frac{1}{n+1} 2^{-6(n+1)^3-1}$. The encoding length of d is polynomial in n. The main step in the proof is now to define an affine bijection

$$f : \{0,1\}^{n+1}(k+1) \cap B \to \{0,1\}^n(k) \times \{0\}$$

with $f(x) = (1-\frac{1}{d})\bar{c}+\frac{1}{d}x$ which is used to reduce minimization over $\{0,1\}^n(k)$ to minimization over $\{0,1\}^{n+1}(k+1)\cap B$ (see Figure 4.5). Using Lemma 4.13 to model the box B proves then the result.

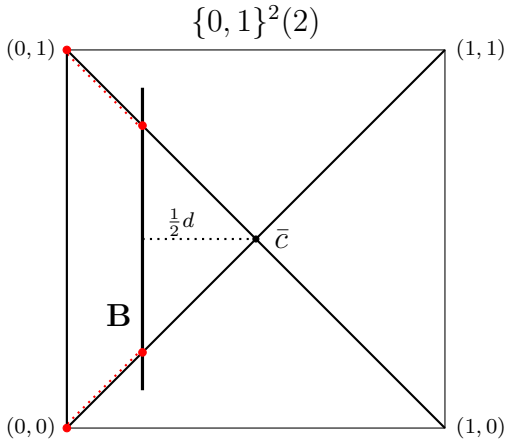


Figure 4.5: Bijection f between $\{0,1\}^2(2) \cap B$ and $\{0,1\}^1(1) \times \{0\}$

To prove that f induces a bijection between the two sets $\{0,1\}^{n+1}(k+1)\cap B$ and $\{0,1\}^n(k) \times \{0\}$ let $x^* \in \{0,1\}^{n+1}(k+1) \cap B$ with $x^* = \sum_{i=1}^{k+1} \lambda_i x^{(i)}$. Since $x^* \in B$, we obtain

$$(f(x^*))_{n+1} = (1 - \frac{1}{d})\bar{c}_{n+1} + \frac{1}{d}(\tfrac{1}{2} - \tfrac{1}{2}d) = 0.$$

Moreover, by the observations above, the center point $\bar{c}$ is contained in conv $\left(x^{(1)}, \ldots, x^{(k+1)}\right)$. Let $\bar{c} = \sum_{i=1}^{k+1} \mu_i x^{(i)}$. Then

$$f(x^*) = (1 - \tfrac{1}{d})\bar{c} + \tfrac{1}{d}x^* = \sum_{i=1}^{k+1} \left((1 - \tfrac{1}{d})\mu_i + \tfrac{1}{d}\lambda_i\right) x^{(i)} \ . \tag{4.7}$$

Since $x^*_{n+1} > 0$, there must exist some j with $(x^{(j)})_{n+1} = 1$. As $(f(x^*))_{n+1} = 0$, we derive $(1 - \frac{1}{d})\mu_j + \frac{1}{d}\lambda_j = 0$ from (4.7). Hence $f(x^*) \in \{0,1\}^{n+1}(k)$ and since $f(x^*)_{n+1} = 0$ we obtain $f(x^*) \in \{0,1\}^n(k) \times \{0\}$. To show the other

direction, consider any $y^* \in \{0,1\}^{n+1}(k)$ with $y^*_{n+1} = 0$. Let $y^* = \sum_{i=1}^{k} \nu_i x^{(i)}$, then we have

$$\begin{aligned} f^{-1}(y^*) &= -(d-1)\bar{c} + dy^* \\ &= \left(\tfrac{1}{2} - \tfrac{1}{2}d\right)\mathbf{1} + d\sum_{i=1}^{k} \nu_i x^{(i)} \\ &= \left(\tfrac{1}{2} - \tfrac{1}{2}d\right)\left(x^{(1)} + \bar{x}^{(1)}\right) + d\sum_{i=1}^{k} \nu_i x^{(i)} \end{aligned}$$

where $\bar{x}^{(1)}$ is defined by $\bar{x}^{(1)}_i = 1 - x^{(1)}_i$ for all i. This is a convex combination of the binary vectors $x^{(1)}, \ldots, x^{(k)}, \bar{x}^{(1)}$, hence $f^{-1}(y^*) \in \{0,1\}^{n+1}(k+1)$. It is easy to verify that $f^{-1}(y^*) \in B$. To conclude the proof, we have

$$\begin{aligned} \min_{x \in \{0,1\}^n(k)} \max_{(c,c_0)\in U} c_0 + c^\top x &= \min_{x \in B \cap \{0,1\}^{n+1}(k+1)} \max_{(c,c_0)\in U} c_0 + c^\top f(x) \\ &= \min_{x \in B \cap \{0,1\}^{n+1}(k+1)} \max_{(c,c_0)\in U} c_0 + c^\top \bar{c}(1 - \frac{1}{d}) + c^\top \frac{1}{d} x \\ &= \min_{x \in B \cap \{0,1\}^{n+1}(k+1)} \max_{(c',c'_0)\in U'} c'_0 + (c')^\top x \end{aligned}$$

where U' is the image of U under the linear map

$$(c, c_0) \mapsto (\tfrac{1}{d}c, c_0 + (1 - \tfrac{1}{d})c^\top \bar{c})) \; .$$

Together with Lemma 4.13, modeling the box B by linear inequalities, this proves the result. □

By induction, the result of Theorem 4.18 together with Corollary 4.16 yields the final result.

Corollary 4.19. Problem (M^3) is *NP*-hard for any fixed $k \in \mathbb{N}$, even if U is a polyhedron given by an inner description and $X = \{0,1\}^n$.

To conclude this section we remark that by the latter result we also obtain *NP*-hardness of the K-adaptability problem (K-AR) for polyhedral uncertainty sets, since ($\mathrm{M}^2_B[k]$) is the special case where no first stage variables exist.

4.2.2 Heuristic Algorithm

In the previous section we showed that Problem (M^3) is *NP*-hard for any fixed k, even for the binary unconstrained problem, which can be solved trivially in linear time in its deterministic version. This gives the motivation for

a heuristic algorithm to solve Problem (M^3) for any $k < n+1$. In our computations it is often the case that an optimal solution x^* of Problem ($M^2[n+1]$) only uses significantly less than $n+1$ solutions (see Chapter 6). This is the case if the optimal solution x^* is contained in a lower-dimensional convex hull $x^* \in \operatorname{conv}\left(x^{(1)}, \ldots, x^{(m)}\right)$ with $m < n+1$. The main idea of the heuristic algorithm which we present in this section is to first calculate an optimal solution $\{x^{(1)}, \ldots, x^{(n+1)}\}$ of Problem (M^3) for $k = n+1$, which can be done efficiently for tractable combinatorial problems. Afterwards we choose the k vectors with the largest coefficients λ_i in the convex combination of x^* in Step (2) of Algorithm 1.

Algorithm 3 Heuristic algorithm for Problem (M^3) for $k < n+1$

Input: convex $U \subseteq \mathbb{R}^{n+1}$, $X \subseteq \{0,1\}^n$, $k \in \mathbb{N}$ with $1 \leq k < n+1$, $\varepsilon \in (0,1)$
Output: feasible solution of Problem (M^3)

1: calculate an optimal solution x^* of ($M^2[n+1]$) up to accuracy ε, using Theorem 4.7
2: calculate $x^{(1)}, \ldots, x^{(n+1)} \in X$, coefficients $\lambda_1, \ldots, \lambda_{n+1} \geq 0$ with $\sum_{i=1}^{n+1} \lambda_i = 1$, $x^* = \sum_{i=1}^{n+1} \lambda_i x^{(i)}$, using Lemma 2.31
3: sort the λ_i in decreasing order $\lambda_{i_1} \geq \ldots \geq \lambda_{i_{n+1}}$
4: **return** $\{x^{(i_1)}, \ldots, x^{(i_k)}\}$

Theorem 4.20. Let $k \in \{1, \ldots, n\}$ and $\varepsilon \in (0,1)$. Let $\operatorname{conv}(X)$ be full-dimensional and U as in Section 4.1.1. Given an optimization oracle for the certain problem

$$c \mapsto \min_{x \in X} c^\top x,$$

Algorithm 3 calculates in polynomial time a solution of Problem (M^3) with an additive error of at most $M\frac{\sqrt{n}(n+1-k)}{k+1} + \varepsilon$.

Proof. By Theorem 4.7 and 2.31, Algorithm 3 has polynomial run-time under the given assumptions. Let $x_k \in X(k)$ be an optimal solution for Problem ($M^2[k]$) and let

$$x_a := \sum_{j=1}^{k-1} \lambda_{i_j} x^{(i_j)} + \Big(\sum_{j=k}^{n+1} \lambda_{i_j}\Big) x^{(i_k)} \in X(k).$$

Since $X(k) \subseteq X(n+1)$ and by the optimality of x^*, we have

$$\begin{aligned}
\max_{(c,c_0)\in U} c^\top x_a + c_0 - \max_{c\in U} c^\top x_k + c_0 &\leq \max_{(c,c_0)\in U} c^\top x_a + c_0 - \max_{(c,c_0)\in U} c^\top x^* + c_0 \\
&\leq \max_{c\in U} c^\top (x_a - x^*) \\
&\leq \max_{c\in U} \|c\| \|x_a - x^*\| \\
&\leq M \left\| \sum_{j=k+1}^{n+1} \lambda_{i_j} (x^{(i_k)} - x^{(i_j)}) \right\|.
\end{aligned}$$

By the decreasing order of λ_{i_j}, we have $\lambda_{i_j} \leq \frac{1}{j}$. Additionally, as $X \subseteq \{0,1\}^n$, we know $\|x^{(i_k)} - x^{(i_j)}\| \leq \sqrt{n}$. Therefore

$$\left\| \sum_{j=k+1}^{n+1} \lambda_{i_j} (x^{(i_k)} - x^{(i_j)}) \right\| \leq \sqrt{n} \sum_{j=k+1}^{n+1} \frac{1}{j} \leq \sqrt{n} \frac{n+1-k}{k+1} .$$

Together with the acurracy $\varepsilon > 0$ from Step (1) this implies the result. □

For fixed U and n, the error bound given in Theorem 4.20 is strictly monotonously decreasing with growing $k \leq n+1$. For $k = n+1$, it coincides with ε. In Section 6.2 we give a practical proof that the heuristic algorithm performs very well for the shortest path problem on the instances studied in [41]. Nevertheless the following example shows that the additive error from Theorem 4.20 can be arbitrarily large.

Example 4.21. Consider the set $X := \{x \in \{0,1\}^n \mid \mathbf{1}^\top x \leq 1\}$, so that $\operatorname{conv}(X)$ is a simplex. For $\alpha > 0$, we define an ellipsoid

$$U_\alpha = \{c \in \mathbb{R}^n \mid (c + \mathbf{1})^\top \Sigma_\alpha (c + \mathbf{1}) \leq 1\} ,$$

where the inverse matrix of Σ_α is defined by

$$\Sigma_\alpha^{-1} = \frac{1}{4} \mathbf{1}\mathbf{1}^\top + \alpha \left(\sum_{i=2}^{n} v_i v_i^\top \right)$$

and $\mathbf{1}, v_2, \ldots, v_n$ is an orthogonal basis in $\mathbb{R}^n$ (see Figure 4.6). Note that Σ_α^{-1} is positive definite. Then the objective function of Problem ($\mathrm{M}^2[n+1]$) can be reformulated as

$$\max_{c\in U_\alpha} c^\top x = -\mathbf{1}^\top x + \sqrt{\frac{1}{4}(x^\top \mathbf{1})^2 + \alpha \left(\sum_{i=2}^{n} (x^\top v_i)^2 \right)}. \tag{4.8}$$

We define $\mathbf{1}^\perp = \left\{x \in \mathbb{R}^n \mid x^\top \mathbf{1} = 0\right\}$. Since $\operatorname{conv}(X) \subset \mathbb{R}^n_+$ we have

$$\mathbf{1}^\perp \cap \operatorname{conv}(X) = \{0\}.$$

Therefore any point $x \in \operatorname{conv}(X)$ which is not 0 has a representation

$$x = \lambda_1 \mathbf{1} + \sum_{i=2}^{n} \lambda_i v_i$$

with $\lambda_1 \neq 0$. Substituting in (4.8) we obtain that for any x of the latter form the vector $\bar{x} = \lambda_1 \mathbf{1}$ has a strictly smaller objective value than x. Therefore any optimal solution x^* is of the form $x^* = \lambda \mathbf{1}$ with $\lambda \in \mathbb{R}$. Substituting again in (4.8) we obtain that for an optimal solution $\lambda \geq 0$ must hold and

$$\max_{c \in U_\alpha} c^\top (\lambda \mathbf{1}) = -\frac{1}{2} \lambda n < 0.$$

Then the unique λ which minimizes the latter expression under the condition that $\lambda \mathbf{1} \in \operatorname{conv}(X)$ is $\lambda = \frac{1}{n}$. Hence $x^* = \frac{1}{n}\mathbf{1}$ is the unique optimal solution. The only possible representation as a convex combination of elements of X is $x^* = \sum_{i=1}^{n} \frac{1}{n} e_i$. Hence, for $k \leq n$, the heuristic will choose a random set $X^* \subseteq \{e_1, \ldots, e_n\}$. Assume without loss of generality that $X^* = \{e_1, \ldots, e_k\}$. Since $\operatorname{conv}(X^*)$ has at least a certain distance $\delta > 0$ to $\mathbf{1}^\perp$ and to 0 there must exist a $\mu > 0$ such that for each $x \in \operatorname{conv}(X^*)$ at least one v_i exists such that $x^\top v_i \geq \mu$. From this observation follows that

$$\sum_{i=2}^{n} (x^\top v_i)^2 \geq \mu^2$$

for each $x \in \operatorname{conv}(X^*)$. Furthermore $x^\top \mathbf{1} = 1$ for each $x \in \operatorname{conv}(X^*)$, and therefore the objective value of the heuristic solution X^*, which is the optimum of

$$\min_{x \in X^*(k)} \max_{c \in U_\alpha} c^\top x,$$

becomes arbitrarily large with growing α. On contrary every $X(k)$ contains the zero vector with objective value zero.

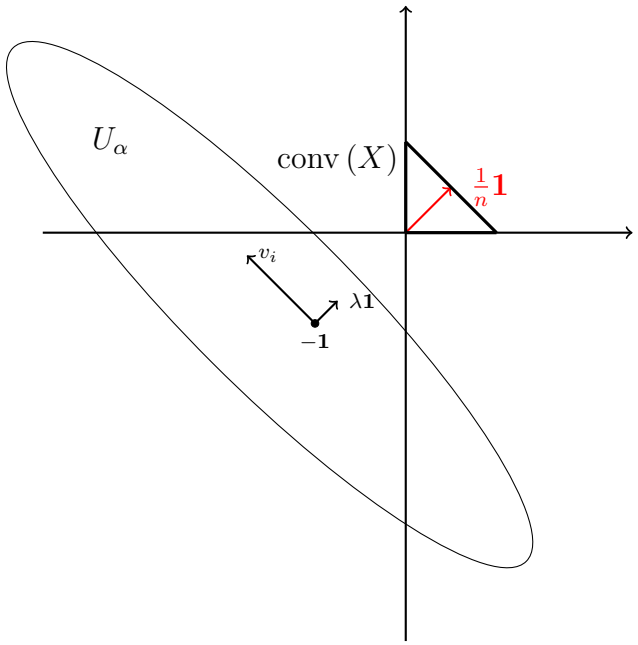


Figure 4.6: The ellipsoid U_α.

Chapter 5

Min-max-min Robustness under Discrete Uncertainty

In this section we analyze the complexity of Problem (M^3) if the uncertainty set $U = \{c_1, \ldots, c_m\} \subset \mathbb{R}^{n+1}$ is finite. The strictly robust approach (M^2) with discrete uncertainty is *NP*-hard for many combinatorial problems, as we have seen in Section 3.1. In this section we show that these results can be extended to Problem (M^3). This is in contrast to the results for convex uncertainty sets in Section 4.1. Nevertheless we will show that for a fixed number of scenarios a pseudopolynomial algorithm exists for all combinatorial problems whose strictly robust version can be solved in pseudopolynomial time. Furthermore we show that (M^3) admits an FPTAS if the strictly robust version of the problem admits an FPTAS or if the related multicriteria problem admits an FPTAS. An overview of the complexity results proved in this section can be found in Table 5.1. For the shortest path problem we assume $U \subset \mathbb{R}^{n+1}_+$. Note that the approximation results are only valid for a constant number of scenarios. Before we prove the before mentioned complexity results we give

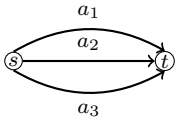
Problem	Constant $\|U\|$	Non-constant $\|U\|$	Approximation
Shortest Path	weakly *NP*-hard	strongly *NP*-hard	FPTAS
Spanning Tree	weakly *NP*-hard	strongly *NP*-hard	FPTAS
Assignment	*NP*-hard	strongly *NP*-hard	?
Knapsack	weakly *NP*-hard	strongly *NP*-hard	FPTAS
Min-Cut	polynomial	strongly *NP*-hard	FPTAS
Min *s*-*t*-Cut	strongly *NP*-hard	strongly *NP*-hard	?
Unconstrained	*NP*-hard	*NP*-hard	?

Table 5.1: Complexity of problem (M^3) for discrete U.

a motivating example which shows that Problem (M^3) can yield an optimal

value which is arbitrarily better than the optimal value of Problem (M^2) even for $k = 2$.

Example 5.1. Consider the graph $G = (V, A)$ with $V = \{s, t\}$ and $A = \{a_1, a_2, a_3\}$ where $a_i = (s, t)$ for each i:



Define the following scenarios $U = \{c_1, c_2, c_3\}$, where the j-th component of c_i is the cost of edge a_j in scenario i,

$$c_1 = \begin{pmatrix} 1 \\ m \\ m+1 \end{pmatrix}, \quad c_2 = \begin{pmatrix} m+1 \\ m \\ 1 \end{pmatrix}, \quad c_3 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

where m is any positive integer. The optimal solution of Problem (M^2) is a_2 with objective value m. Problem (M^3) with $k = 2$ yields the optimal solution $\{a_1, a_3\}$ and an optimal value of 1.

5.1 Complexity

In this section we investigate the complexity of Problem (M^3) for discrete uncertainty sets. Note that the important result of Lemma 4.1, which is the basis of the complexity results of Section 4.1.1, is not applicable for discrete uncertainty sets, since U is not convex. For the strictly robust problem (M^2) it is possible to replace U by its convex hull (see Theorem 3.5) to obtain a convex uncertainty set. This approach does not yield an equivalent problem for (M^3) as the following example shows.

Example 5.2. Consider again Example 5.1, with an optimal value of 1 for the min-max-min problem (M^3) with $k = 2$. If we replace U by its convex hull and consider the scenario

$$\frac{1}{2}(c_1 + c_2) = \begin{pmatrix} \frac{1}{2}m + 1 \\ m \\ \frac{1}{2}m + 1 \end{pmatrix} \in \operatorname{conv}(U) \ ,$$

then we derive

$$\max_{c \in \operatorname{conv}(U)} \min\{c^\top x^{(1)}, c^\top x^{(2)}\} \geq \frac{1}{2}m + 1$$

for all $x^{(1)}, x^{(2)} \in X$ which yields a strictly larger optimal value for any $m \in \mathbb{N}$.

Finally we prove for all combinatorial problems defined in Section 2.4, except for the general min-cut problem, that their min-max-min versions (M^3) are *NP*-hard for a fixed number of scenarios. All the proofs rely on the same idea. We reduce Problem (M^2) to Problem (M^3) by adding $k-1$ new solutions to the given instance of the min-max version of the problem. Then we define new scenarios, such that each of the $k-1$ new solutions must be contained in an optimal solution of (M^3). For appropriately chosen scenarios the remaining k-th solution in any optimal solution of (M^3) is then the optimal solution of Problem (M^2). Since the latter construction depends heavily on the given combinatorial problem, we show the result for all problems separately in the following subsections. An easy observation which is true for any combinatorial problem is the following.

Proposition 5.3. Let $U = \{(c_1, (c_1)_0), \ldots, (c_m, (c_m)_0)\} \subset \mathbb{R}^{n+1}$. If $k \geq m$, then Problem (M^3) can be reduced to the underlying deterministic problem.

Proof. Let $x^{(i)} \in X$ be an optimal solution of the deterministic problem

$$\min_{x \in X} c_i^\top x + (c_i)_0$$

for all $i = 1, \ldots, m$. Then $\left\{x^{(1)}, \ldots, x^{(m)}\right\}$ is clearly an optimal solution of Problem (M^3). □

Hence in the following we restrict ourselves to the case $k < m$.

5.1.1 Shortest Path Problem

Theorem 5.4. For the shortest path problem on a graph G and for a discrete uncertainty set $U = \{c_1, \ldots, c_m\}$, we can polynomially reduce Problem (M^2) to Problem (M^3) with at least $m + k - 1$ scenarios, for any fixed $k < m$.

Proof. Instead of (M^2) we consider the equivalent Problem (M_0^2) (see Theorem 3.1). For any given instance of the min-max shortest path problem, i.e., a directed graph $G = (V, E)$, nodes $s, t \in V$ and an uncertainty set $U = \{c_1, \ldots, c_m\}$ with $c_i \geq 0$ for all i, we define the graph $G_k^{sp} := (V_k^{sp}, E_k^{sp})$ with $V_k^{sp} := V \cup \{v_1, \ldots, v_{k-1}\}$ and

$$E_k^{sp} := E \cup \{l_i = (s, v_i), r_i = (v_i, t) : i = 1, \ldots, k-1\}.$$

In fact we add $k-1$ new paths $p^i_{st} := (l_i, r_i)$ to the graph G. The latter construction is shown in Figure 5.1. The idea of the proof is to define scenarios on G^{sp}_k that force the min-max-min problem to choose each of the new paths as one solution. The remaining k-th solution then is the optimal min-max solution in G. To this end, we define scenarios $\bar{c}_1, \ldots, \bar{c}_m$ on G^{sp}_k by extending every scenario $c_i \in U$ by M on the edges $r_1, \ldots, r_{k-1}$, where M is a sufficiently large number that can be chosen as

$$M := \sum_{i=1}^{m} \sum_{j=1}^{|E|} (c_i)_j + 1.$$

Then we add scenarios $d_1, \ldots, d_{k-1}$ such that in scenario d_i all edges in G and all edges $r_1, \ldots, r_{k-1}$ have cost M, except for edge r_i which has cost zero. The edges $l_1, \ldots, l_{k-1}$ have zero cost in all scenarios. Note that we only added the $k-1$ new nodes and therefore the edges $l_1, \ldots, l_{k-1}$ to avoid multigraphs. Now if we choose a solution $\left\{x^{(1)}, \ldots x^{(k)}\right\}$ of the min-max-min shortest-path

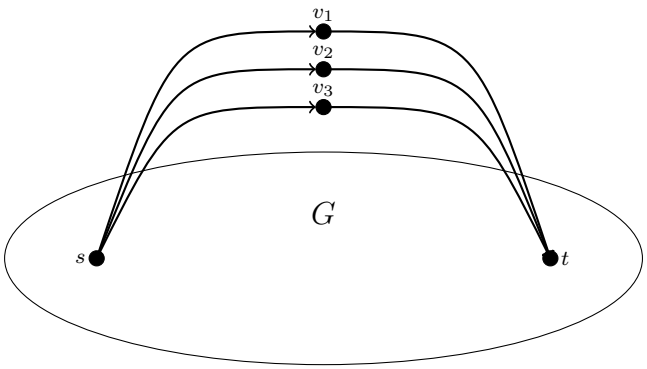


Figure 5.1: The graph G^{sp}_k for $k = 4$.

problem on G^{sp}_k where $x^{(1)}, \ldots, x^{(k-1)}$ are the added paths p^i_{st} and $x^{(k)}$ is any feasible solution in G then for the objective value holds

$$\max_{c \in \{\bar{c}_1, \ldots, \bar{c}_m, d_1, \ldots d_{k-1}\}} \min_{i=1,\ldots,k} c^\top x^{(i)} < M,$$

since on $\bar{c}_i$ the minimum is attained by $x^{(k)}$ and therefore, by the definition of M, it is strictly lower than M. On the other hand on d_i the minimum is attained by $x^{(i)}$ and hence is 0. So we know that every optimal solution of the min-max-min problem on the graph G^{sp}_k must contain the paths p^i_{st} since otherwise, if for any i_0 the path $p^{i_0}_{st}$ is not contained, then in scenario d_{i_0} the

minimum $\min_{i=1,\dots,k} c^\top x^{(i)}$ is greater or equal M and therefore

$$\max_{c\in\{\bar{c}_1,\dots,\bar{c}_m,d_1,\dots d_{k-1}\}} \min_{i=1,\dots,k} c^\top x^{(i)} \geq M$$

which cannot be optimal by the observation above. On the other hand, by the same reasoning applied to the scenarios $\bar{c}_i$, in every optimal solution there must be contained a solution which only uses edges in G. So let $\left\{x^{(1)},\dots,x^{(k)}\right\}$ be an optimal solution where w.l.o.g. $x^{(k)}$ is the path in G and $x^{(i)}$ is the path p^i_{st} for $i = 1,\dots k-1$. Then we have

$$\max_{c\in\{d_1,\dots,d_{k-1}\}} \min_{i=1,\dots,k} c^\top x^{(i)} = 0$$

by definition of the scenarios $\{d_1,\dots,d_{k-1}\}$. On the other hand

$$\max_{c\in\{\bar{c}_1,\dots,\bar{c}_m\}} \min_{i=1,\dots,k} c^\top x^{(i)} = \max_{c\in\{c_1,\dots,c_m\}} c^\top x^{(k)} > 0$$

and therefore

$$\min_{x^{(1)},\dots,x^{(k)}\in X} \max_{c\in\{\bar{c}_1,\dots,\bar{c}_m,d_1,\dots d_{k-1}\}} \min_{i=1,\dots,k} c^\top x^{(i)} = \min_{x\in X} \max_{c\in\{c_1,\dots,c_m\}} c^\top x.$$

and the min-max optimal solution must be contained in $\left\{x^{(1)},\dots,x^{(k)}\right\}$. □

Corollary 5.5. For any fixed $k \in \mathbb{N}$ and fixed $m > k$, Problem (M^3) is *NP*-hard for the shortest path problem for uncertainty sets U with $|U| = m$.

Proof. Problem (M^2) for the shortest path problem is *NP*-hard for $m \geq 2$ (see Table 3.1). From Theorem 5.4 we derive that the min-max-min variant of the same problem is *NP*-hard if the number of scenarios is at least $k+1$. □

Note that by Proposition 5.3 for $k \geq m$ Problem (M^3) can be solved in polynomial time for the shortest path problem. Therefore the restriction to $m > k$ is necessary.

Corollary 5.6. For any fixed $k \in \mathbb{N}$, Problem (M^3) with discrete U is strongly *NP*-hard for the shortest path problem.

Proof. The min-max variant of the shortest path problem is strongly *NP*-hard if the number of scenarios is not constant (see Table 3.1). The result follows since all numbers in the construction in the proof of Theorem 5.4 are polynomial in $|U|$. □

5.1.2 Spanning Tree Problem

Theorem 5.7. For the minimum spanning-tree problem on a graph G with at least k nodes and for a discrete uncertainty set $U = \{c_1, \ldots, c_m\}$, we can polynomially reduce Problem (M^2) to Problem (M^3) with at least $m + k - 1$ scenarios, for any fixed $k < m$.

Proof. Instead of (M^2) we consider the equivalent Problem (M_0^2) (see Theorem 3.1). The proof is similar to the proof of Theorem 5.4. Let a graph $G = (V, E)$ with $|V| \geq k$ and an uncertainty set $U = \{c_1, \ldots, c_m\}$ be given. We then define the graph $G_k^{st} := (V_k^{st}, E_k^{st})$ with $V_k^{st} := V \cup \{w\}$ and $E_k^{st} := E \cup \{f_i = \{v_i, w\} : i = 0 \ldots, k-1\}$ where the v_i are arbitrary pairwise different nodes in G. In fact we add one node and connect it to k different nodes with exactly one edge. The latter construction is shown in Figure 5.2. Again the idea of the proof is to define scenarios on G_k^{st} that force

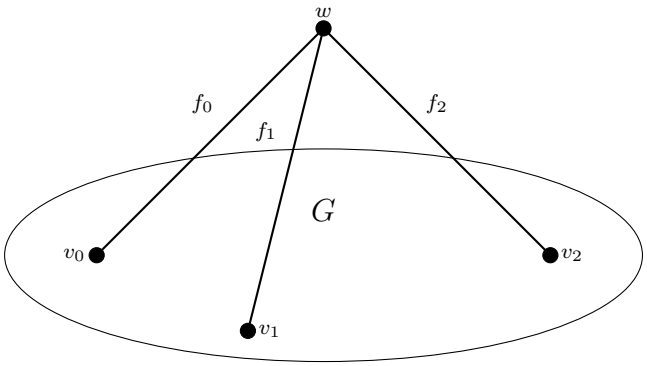


Figure 5.2: The graph G_k^{st} for $k = 3$.

the min-max-min problem to choose solutions which use exactly one of the new edges each. Note that a feasible solution could use all of the new edges simultaneously. To this end, we define scenarios $\bar{c}_1, \ldots, \bar{c}_m$ on G_k^{st} by extending every scenario $c_i \in U$ by zero on f_0 and by $2M$ on the edges $f_1, \ldots, f_{k-1}$, where M can be chosen as

$$M := \sum_{i=1}^{m} \sum_{j=1}^{|E|} |(c_i)_j| + 1.$$

Then we add scenarios $d_1, \ldots d_{k-1}$ such that in scenario d_i all edges in G have cost M and all edges $f_0, \ldots, f_{k-1}$ have cost $(|V| + 1)\, M$, except for edge f_i which has cost $-|V|M$. Now if we choose a solution $\left\{x^{(0)}, \ldots x^{(k-1)}\right\}$ of the

min-max-min spanning-tree problem on G_k^{st}, where $x^{(i)}$ uses edge f_i and none of the other new edges to connect node w, then for the objective value holds

$$\max_{c\in\{\bar{c}_1,\ldots,\bar{c}_m,d_1,\ldots d_{k-1}\}} \min_{i=0,\ldots,k-1} c^\top x^{(i)} < M,$$

since on $\bar{c}_i$ the minimum is attained by $x^{(0)}$ and therefore, by the definition of M, it is strictly lower than M. On the other hand on d_i the minimum is attained by $x^{(i)}$ and is exactly $-M$ since any spanning-tree in G uses exactly $|V|-1$ edges. Therefore we know that every optimal solution of the min-max-min problem on the graph G_k^{st} must contain for each edge f_i a solution which only uses edge f_i under the new edges. Otherwise, if for any f_{i_0} this is not true, then, if $i_0 \geq 1$, in scenario d_{i_0} the minimum $\min_{i=1,\ldots,k} c^\top x^{(i)}$ is greater or equal to M and therefore

$$\max_{c\in\{\bar{c}_1,\ldots,\bar{c}_m,d_1,\ldots d_{k-1}\}} \min_{i=0,\ldots,k-1} c^\top x^{(i)} \geq M$$

which cannot be optimal by the observation above. If $i_0 = 0$ by the same reasoning applied to the scenarios $\bar{c}_i$ every optimal solution must contain a solution which only uses edge f_0. So let $\left\{x^{(0)},\ldots,x^{(k-1)}\right\}$ be an optimal solution like above. Then we have

$$\max_{c\in\{d_1,\ldots,d_{k-1}\}} \min_{i=0,\ldots,k-1} c^\top x^{(i)} = -M$$

by definition of the scenarios $\{d_1,\ldots,d_{k-1}\}$. On the other hand

$$\max_{c\in\{\bar{c}_1,\ldots,\bar{c}_m\}} \min_{i=0,\ldots,k-1} c^\top x^{(i)} = \max_{c\in\{c_1,\ldots,c_m\}} c^\top x^{(0)} > -M$$

and therefore

$$\min_{x^{(1)},\ldots,x^{(k)}\in X} \max_{c\in\{\bar{c}_1,\ldots,\bar{c}_m,d_1,\ldots d_{k-1}\}} \min_{i=1,\ldots,k} c^\top x^{(i)} = \min_{x\in X} \max_{c\in\{c_1,\ldots,c_m\}} c^\top x.$$

and the min-max optimal solution must be contained in $\left\{x^{(0)},\ldots,x^{(k-1)}\right\}$. □

Note that the spanning-tree problem on graphs with less than k nodes can be extended to graphs with at least k nodes by adding extra nodes and connecting them with exactly one edge to one of the nodes of the original graph. A solution for the original problem can be obtained by projecting the solution on the extended graph to the original graph. Therefore it is no restriction to consider only graphs with at least k nodes.

Corollary 5.8. For any fixed $k \in \mathbb{N}$ and fixed $m > k$, Problem (M^3) is *NP*-hard for the minimum spanning-tree problem for uncertainty sets U with $|U| = m$.

Proof. Problem (M^2) for the minimum spanning-tree problem is *NP*-hard for $m \geq 2$ (see Table 3.1). From Theorem 5.7 we derive that the min-max-min variant of the same problem is *NP*-hard if the number of scenarios is at least $k + 1$. □

Note that by Proposition 5.3 for $k \geq m$ Problem (M^3) can be solved in polynomial time for the minimum spanning-tree problem. Therefore the restriction to $m > k$ is necessary.

Corollary 5.9. For any fixed $k \in \mathbb{N}$, Problem (M^3) with discrete U is strongly *NP*-hard for the minimum spanning-tree problem.

Proof. The min-max variant of the minimum spanning-tree problem is strongly *NP*-hard if the number of scenarios is not constant (see Table 3.1). The result follows since all numbers in the construction in the proof of Theorem 5.7 are polynomial in $|U|$. □

5.1.3 Assignment Problem

Theorem 5.10. For the assignment problem on a graph G and for a discrete uncertainty set $U = \{c_1, \ldots, c_m\}$, we can polynomially reduce Problem (M^2) to Problem (M^3) with at least $m + k - 1$ scenarios, for any fixed $k < m$.

Proof. Instead of (M^2) we consider the equivalent Problem (M_0^2) (see Theorem 3.1). The proof is similar to the previous proofs. For a given instance of the min-max assignment problem, i.e. a bipartite graph $G = (V, W, E)$ and an uncertainty set $U = \{c_1, \ldots, c_m\}$, we define the graph $G_k^{as} := (V_k^{as}, W_k^{as}, E_k^{as})$ with $V_k^{as} := V \cup \{v_1, \ldots, v_{k-1}\}$, $W_k^{as} = W \cup \{w_1, \ldots, w_{k-1}\}$ and

$$E_k^{as} := E \cup \{f_{ij} = \{v_i, w_j\} : i, j = 1, \ldots, k-1\}.$$

The latter construction is shown in Figure 5.3. Note that any feasible solution on G_k^{as} must induce a perfect matching in G and a perfect matching in the subgraph with nodes $v_1, \ldots, v_{k-1}$ and $w_1, \ldots, w_{k-1}$. The idea of the proof is to define scenarios on G_k^{as} that force the min-max-min problem to choose k solutions $x^{(i)}$ such that for each $i = 1, \ldots, k-1$ solution $x^{(i)}$ uses edge f_{ii} but non of the edges f_{jj} for $j \neq i$ and solution $x^{(k)}$ uses all the edges f_{ii}. To

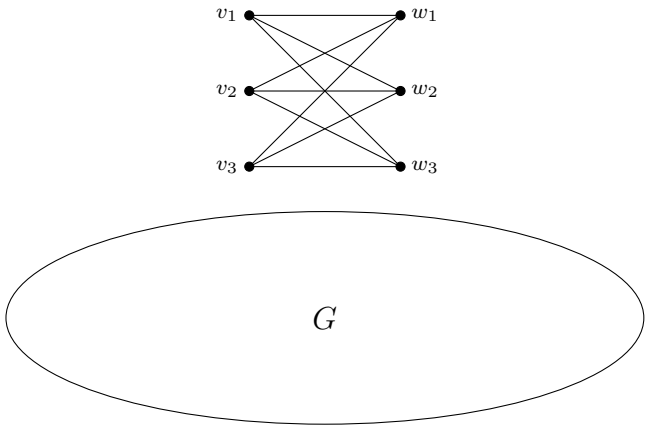


Figure 5.3: The graph G_k^{as} for $k = 4$.

this end, we define scenarios $\bar{c}_1, \ldots, \bar{c}_m$ on G_k^{as} by extending every scenario $c_i \in U$ by zero on the edges f_{ii} for all $i = 1 \ldots, k-1$ and by $2M$ on all other edges. Here M can be chosen as

$$M := \sum_{i=1}^{m} \sum_{j=1}^{|E|} |(c_i)_j| + 1.$$

Then we add scenarios $d_1, \ldots d_{k-1}$ such that in scenario d_i the edges f_{jj} have cost $3M$ for $j \neq i$ and edge f_{ii} has cost $-2M$. Furthermore for any i each edge f_{ij} for $j = 1, \ldots, k-1$ and $j \neq i$ has cost $3M$. All other edges have cost 0. Now we choose a solution $\left\{x^{(1)}, \ldots x^{(k)}\right\}$ of the min-max-min assignment problem on G_k^{as}, such that for each $i = 1, \ldots, k-1$ solution $x^{(i)}$ uses edge f_{ii} and none of the edges f_{jj} for $j \neq i$ and solution $x^{(k)}$ uses all edges f_{jj} for each $j = 1, \ldots, k-1$. Note that there always exists a perfect matching in G_k^{as} with the latter properties if a perfect matching in the original graph G exists. For solutions with the latter properties the objective value is

$$\max_{c \in \{\bar{c}_1, \ldots, \bar{c}_m, d_1, \ldots d_{k-1}\}} \min_{i=1,\ldots,k} c^\top x^{(i)} < M,$$

since on $\bar{c}_i$ the minimum is attained by $x^{(k)}$ and therefore, by the definition of M, it is strictly lower than M. On the other hand on d_i the minimum is attained by $x^{(i)}$ and is smaller than $-M$. Therefore we know that every optimal solution of the min-max-min problem on the graph G_k^{as} must contain for each edge $i = 1 \ldots, k-1$ a solution with the property of $x^{(i)}$ since otherwise if this is not true every solution has to use one of the edges f_{ij} with $j \neq i$

which all have cost $3M$ in scenario d_i. Therefore in scenario d_i the minimum $\min_{i=1,\dots,k} c^\top x^{(i)}$ is greater than M and therefore

$$\max_{c\in\{\bar{c}_1,\dots,\bar{c}_m,d_1,\dots d_{k-1}\}} \min_{i=1,\dots,k} c^\top x^{(i)} \geq M$$

which cannot be optimal by the observation above. By the same reasoning applied to the scenarios $\bar{c}_i$, in every optimal solution there must be contained a solution which uses all edges f_{ii} for $i = 1,\dots,k-1$. So let $\left\{x^{(1)},\dots,x^{(k)}\right\}$ be an optimal solution with the properties like above. Then we have

$$\max_{c\in\{d_1,\dots,d_{k-1}\}} \min_{i=1,\dots,k} c^\top x^{(i)} < -M$$

by the reasoning above. On the other hand

$$\max_{c\in\{\bar{c}_1,\dots,\bar{c}_m\}} \min_{i=1,\dots,k} c^\top x^{(i)} = \max_{c\in\{c_1,\dots,c_m\}} c^\top x^{(k)} > -M$$

and therefore

$$\min_{x^{(1)},\dots,x^{(k)}\in X} \max_{c\in\{\bar{c}_1,\dots,\bar{c}_m,d_1,\dots d_{k-1}\}} \min_{i=1,\dots,k} c^\top x^{(i)} = \min_{x\in X} \max_{c\in\{c_1,\dots,c_m\}} c^\top x.$$

and the min-max optimal solution must be contained in $\left\{x^{(1)},\dots,x^{(k)}\right\}$. □

Corollary 5.11. For any fixed $k \in \mathbb{N}$ and fixed $m > k$, Problem (M^3) is *NP*-hard for the assignment problem for uncertainty sets U with $|U| = m$.

Proof. The Problem (M^2) for the minimum assignment problem is *NP*-hard for $m \geq 2$ (see Table 3.1). From Theorem 5.10 we derive that the min-max-min variant of the same problem is *NP*-hard if the number of scenarios is at least $k + 1$. □

Note that by Proposition 5.3 for $k \geq m$ Problem (M^3) can be solved in polynomial time for the assignment problem. Therefore the restriction to $m > k$ is necessary.

Corollary 5.12. For any fixed $k \in \mathbb{N}$, Problem (M^3) with discrete U is strongly *NP*-hard for the assignment problem.

Proof. The min-max variant of the assignment problem is strongly *NP*-hard if the number of scenarios is not constant (see Table 3.1). The result follows since all numbers in the construction in the proof of Theorem 5.10 are polynomial in $|U|$. □

5.1.4 Minimum Cut Problem

Theorem 5.13. For the minimum s-t-cut problem on a connected graph G and for a discrete uncertainty set $U = \{c_1, \dots, c_m\}$, we can polynomially reduce Problem (M^2) to Problem (M^3) with at least $m + k - 1$ scenarios, for any fixed $k < m$.

Proof. Instead of (M^2) we consider the equivalent Problem (M_0^2) (see Theorem 3.2). The proof is similar to the proofs in the previous sections. Let a connected graph $G = (V, E)$ and an uncertainty set $U = \{c_1, \dots, c_m\}$ be given. We then define the graph $G_k^{cut} := (V_k^{cut}, E_k^{cut})$ with $V_k^{cut} := V \cup \{v_0, \dots, v_{k-1}\}$ and

$$E_k^{cut} := E \cup \{l_i = (s, v_i), r_i = (v_i, t) : i = 0, \dots, k-1\}.$$

Note that the graph is constructed analogously to the graph G_{sp} in the proof of Theorem 5.4 but we add one more path. The latter construction is shown in Figure 5.4. Note that for any feasible solution $\delta(S)$ either the edge l_i or the

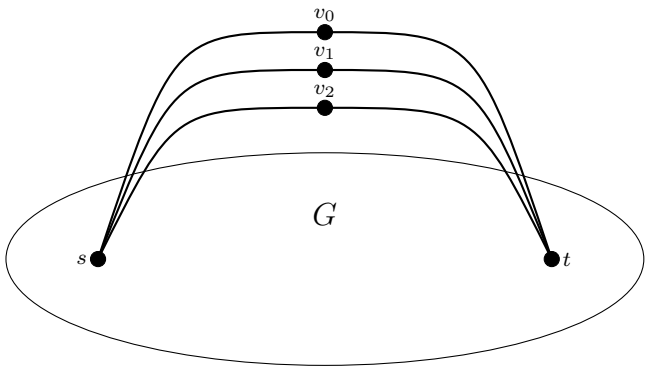


Figure 5.4: The graph G_k^{cut} for $k = 3$.

edge r_i is contained in $\delta(S)$ for each $i = 0, \dots, k-1$, but not both. The idea of the proof is to define scenarios on G_k^{cut} that force the min-max-min problem to choose k solutions which use only the edge r_i for each $i = 0, \dots, k-1$. Note that a feasible solution could use all of the edges r_i simultaneously. To this end, we define scenarios $\bar{c}_1, \dots, \bar{c}_m$ on G_k^{cut} by extending every scenario $c_i \in U$ by zero on $l_1, \dots, l_{k-1}$ and r_0 and by $2M$ on $r_1, \dots, r_{k-1}$ and l_0. Here M can be chosen as

$$M := \sum_{i=1}^{m} \sum_{j=1}^{|E|} |(c_i)_j| + 1.$$

Then we add scenarios $d_1, \dots d_{k-1}$ such that in scenario d_i all edges in G and all edges $l_0, \dots, l_{k-1}$ have cost 0 except edge l_i which has cost $2M$. Furthermore all edges $r_0, \dots, r_{k-1}$ have cost $2M$, except for edge r_i which has cost $-M$. Now if we choose a solution $\left\{x^{(0)}, \dots x^{(k-1)}\right\}$ of the min-max-min s-t-cut problem on G_k^{cut}, where $x^{(i)}$ uses edge r_i and all edges l_j with $j \neq i$, then for the objective value holds

$$\max_{c \in \{\bar{c}_1, \dots, \bar{c}_m, d_1, \dots d_{k-1}\}} \min_{i=0,\dots,k-1} c^\top x^{(i)} < M,$$

since on $\bar{c}_i$ the minimum is attained by $x^{(0)}$ and therefore, by the definition of M, it is strictly lower than M. On the other hand on d_i the minimum is attained by $x^{(i)}$ and is exactly $-M$. Therefore we know that every optimal solution of the min-max-min problem on the graph G_k^{cut} must contain for each edge r_i a solution which only uses edge r_i under the new edges $r_0, \dots, r_{k-1}$. Otherwise, if for any r_{i_0} this is not true, then edge l_{i_0} must be used. In the case if $i_0 \geq 1$, in scenario d_{i_0} the minimum $\min_{i=1,\dots,k} c^\top x^{(i)}$ is then greater or equal to M and therefore

$$\max_{c \in \{\bar{c}_1, \dots, \bar{c}_m, d_1, \dots d_{k-1}\}} \min_{i=0,\dots,k-1} c^\top x^{(i)} \geq M$$

which cannot be optimal by the observation above. In the case if $i_0 = 0$ by the same reasoning applied to the scenarios $\bar{c}_i$, in every optimal solution there must be contained a solution which only uses edge r_0. So let $\left\{x^{(0)}, \dots, x^{(k-1)}\right\}$ be an optimal solution with the properties like above. Then we have

$$\max_{c \in \{d_1, \dots, d_{k-1}\}} \min_{i=0,\dots,k-1} c^\top x^{(i)} = -M$$

by the reasoning above. On the other hand

$$\max_{c \in \{\bar{c}_1, \dots, \bar{c}_m\}} \min_{i=0,\dots,k-1} c^\top x^{(i)} = \max_{c \in \{c_1, \dots, c_m\}} c^\top x^{(0)} > -M$$

and therefore

$$\min_{x^{(1)}, \dots, x^{(k)} \in X} \max_{c \in \{\bar{c}_1, \dots, \bar{c}_m, d_1, \dots d_{k-1}\}} \min_{i=1,\dots,k} c^\top x^{(i)} = \min_{x \in X} \max_{c \in \{c_1, \dots, c_m\}} c^\top x.$$

and the min-max optimal solution must be contained in $\left\{x^{(0)}, \dots, x^{(k-1)}\right\}$. □

Corollary 5.14. For any fixed $k \in \mathbb{N}$ and fixed $m > k$, Problem (M^3) is strongly *NP*-hard for the minimum s-t-cut problem for uncertainty sets U with $|U| = m$.

Proof. Problem (M^2) for the minimum s-t-cut problem is strongly *NP*-hard for $m \geq 2$ (see Table 3.1). From Theorem 5.13 and since all numbers in the proof remain of polynomial size, we derive that the min-max-min variant of the same problem is strongly *NP*-hard if the number of scenarios is at least $k + 1$. □

Note that by Proposition 5.3 for $k \geq m$ Problem (M^3) can be solved in polynomial time for the minimum s-t-cut problem. Therefore the restriction to $m > k$ is necessary.

As we have seen in Table 3.1 Problem (M^2) for the normal minimum cut problem is strongly *NP*-hard if we assume the number of scenarios to be part of the input. In the following we show that this also holds for Problem (M^3).

Theorem 5.15. For the minimum cut problem on a connected graph G and for a discrete uncertainty set $U = \{c_1, \ldots, c_m\}$, we can polynomially reduce Problem (M^2) to Problem (M^3) with at least $m + k - 1$ scenarios.

Proof. The result can be proved analogously to the proof of Theorem 5.13 by using the graph $\bar{G}_k^{cut} := \left(\bar{V}_k^{cut}, \bar{E}_k^{cut}\right)$ with $\bar{V}_k^{cut} := V \cup \{v_0, \ldots, v_{k-1}\}$ and

$$\bar{E}_k^{cut} := E \cup \{f_i = (w, v_i) : i = 0, \ldots, k-1\}$$

for an arbitrary node $w \in V$. The latter construction is shown in Figure 5.5. □

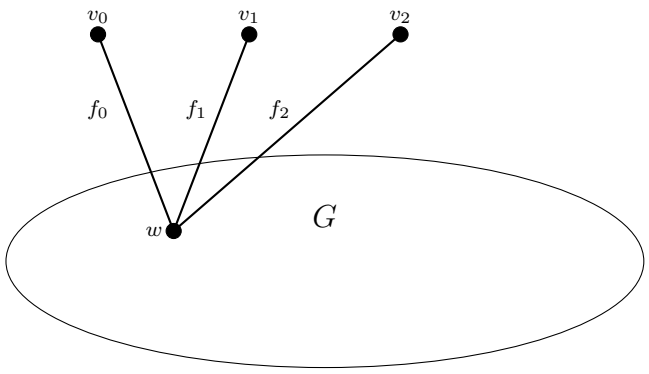


Figure 5.5: The graph $\bar{G}_k^{cut}$ for $k = 3$.

Corollary 5.16. For any fixed $k \in \mathbb{N}$, Problem (M^3) is strongly *NP*-hard for the minimum cut problem for discrete uncertainty sets U.

Proof. Problem (M^2) for the minimum cut problem is strongly *NP*-hard if the number of scenarios is part of the input (see Table 3.1). From Theorem 5.13 and since all numbers in the proof remain of polynomial size, we derive that the min-max-min variant of the same problem is strongly *NP*-hard if the number of scenarios is at least $k+1$. □

5.1.5 Unconstrained Binary Problem

Theorem 5.17. For the unconstrained binary problem and for a discrete uncertainty set $U = \{c_1, \ldots, c_m\}$, we can polynomially reduce Problem (M^2) to Problem (M^3) with at least $m + k - 1$ scenarios, for any fixed $k < m$.

Proof. Instead of (M^2) we consider the equivalent Problem (M_0^2) (see Theorem 3.2). The proof relies on the same idea as the previous proofs. Given an instance of the min-max version of the unconstrained binary problem

$$\min_{x\in\{0,1\}^n} \max_{c\in U} c^\top x$$

where $U = \{c_1, \ldots, c_m\} \subset \mathbb{R}^n$, we define an instance of the unconstrained binary problem in dimension $n + k - 1$ with the scenario set

$$U' = \{\bar{c}_1, \ldots, \bar{c}_m, d_1, \ldots, d_{k-1}\} \subset \mathbb{R}^{n+k-1}.$$

Here we define the scenarios by

$$\bar{c}_i = \begin{pmatrix} c_i \\ 3M \\ \vdots \\ 3M \end{pmatrix}$$

for $i = 1 \ldots, m$ and

$$d_i = 2M\mathbf{1} - 4Me_{n+i}$$

for $i = 1, \ldots, k-1$ where $\mathbf{1} \in \mathbb{R}^{n+k-1}$. The parameter M can be chosen as

$$M := \sum_{i=1}^{m} \sum_{j=1}^{|E|} |(c_i)_j| + 1.$$

Now consider a feasible solution $\left\{x^{(1)}, \ldots x^{(k)}\right\} \subset \{0,1\}^{n+k-1}$ of the min-max-min problem

$$\min_{x^{(1)},\ldots,x^{(k)}\in\{0,1\}^{n+k-1}} \max_{c\in U'} \min_{i=1,\ldots,k} c^\top x^{(i)}$$

with $x^{(i)} = e_{n+i}$ for each $i = 1, \ldots, k-1$ and a solution $x^{(k)}$ which fulfills $x_j^{(k)} = 0$ for all $j = n+1, \ldots, n+k-1$ and $\bar{c}_i^\top x^{(k)} \leq 0$ for all $i = 1, \ldots, m$. Since the zero vector is a feasible solution we can always find a solution which fulfills the latter conditions. Then for the objective value holds

$$\max_{c \in \{\bar{c}_1, \ldots, \bar{c}_m, d_1, \ldots d_{k-1}\}} \min_{i=1,\ldots,k} c^\top x^{(i)} \leq 0,$$

since on $\bar{c}_i$ the minimum is attained by $x^{(k)}$ and therefore it is lower or equal to 0. On the other hand on d_i the minimum is attained by $x^{(i)}$ and is exactly $-2M$. If the optimal value of (M^3) is 0 then an optimal solution of Problem (M_0^2) must be the zero vector since if there would exist a solution with objective value strictly lower than 0 for (M_0^2) we could replace $x^{(k)}$ above by this solution. This would yield a strictly negative objective value for (M^3) as well. In the case that the objective value of (M^3) is strictly lower than 0 we can show that every optimal solution of the min-max-min problem must contain all solutions e_{n+i} for $i = 1, \ldots, k-1$. Otherwise, if any e_{n+i_0} is not contained, then in scenario d_{i_0} the minimum $\min_{i=1,\ldots,k} c^\top x^{(i)}$ is greater or equal to 0 and therefore

$$\max_{c \in \{\bar{c}_1, \ldots, \bar{c}_m, d_1, \ldots d_{k-1}\}} \min_{i=1,\ldots,k} c^\top x^{(i)} \geq 0$$

which cannot be optimal by the assumption above. On the other hand a solution x with $x_{n+i} = 0$ for $i = 1, \ldots, k-1$ must be also contained in the optimal solution by the same reasoning for scenarios $\bar{c}_i$. So let $\left\{x^{(1)}, \ldots, x^{(k)}\right\}$ be an optimal solution with the properties like above. Then we have

$$\max_{c \in \{d_1, \ldots, d_{k-1}\}} \min_{i=1,\ldots,k} c^\top x^{(i)} = -2M$$

by the reasoning above. On the other hand

$$\max_{c \in \{\bar{c}_1, \ldots, \bar{c}_m\}} \min_{i=1,\ldots,k} c^\top x^{(i)} = \max_{c \in \{c_1, \ldots, c_m\}} c^\top x^{(k)} > -2M$$

by the definition of M and therefore

$$\min_{x^{(1)}, \ldots, x^{(k)} \in X} \max_{c \in \{\bar{c}_1, \ldots, \bar{c}_m, d_1, \ldots d_{k-1}\}} \min_{i=1,\ldots,k} c^\top x^{(i)} = \min_{x \in X} \max_{c \in \{c_1, \ldots, c_m\}} c^\top x$$

and the min-max optimal solution must be contained in $\left\{x^{(1)}, \ldots, x^{(k)}\right\}$. □

Corollary 5.18. For any fixed $k \in \mathbb{N}$ and fixed $m > k$, Problem (M^3) is *NP*-hard for the binary unconstrained problem for uncertainty sets U with $|U| = m$.

Proof. The Problem (M^2) for the binary unconstrained problem is *NP*-hard for $m \geq 2$ (see Table 3.1). From Theorem 5.17, we derive that the min-max-min variant of the same problem is *NP*-hard if the number of scenarios is at least $k+1$. □

Note that by Proposition 5.3 for $k \geq m$ Problem (M^3) can be solved in polynomial time for the binary unconstrained problem. Therefore the restriction to $m > k$ is necessary.

5.1.6 Knapsack Problem

Theorem 5.19. For the knapsack problem and for a discrete uncertainty set $U = \{c_1, \ldots, c_m\}$, we can polynomially reduce Problem (M^2) to Problem (M^3) with at least $m + k - 1$ scenarios, for any fixed $k < m$.

Proof. Instead of (M^2) we consider the equivalent Problem (M_0^2) (see Theorem 3.2). The proof is very similar to the proof of Theorem 5.17. Given an instance of the min-max version of the knapsack problem

$$\min_{\substack{x \in \{0,1\}^n \\ a^\top x \leq b}} \max_{c \in U} c^\top x$$

where $U = \{c_1, \ldots, c_m\} \subset \mathbb{R}^n$, we define an instance of the knapsack problem in dimension $n + k - 1$ with weight vector $\bar{a} = (a, 0)$ and the same b as for the original instance. We define the scenario set

$$U' = \{\bar{c}_1, \ldots, \bar{c}_m, d_1, \ldots, d_{k-1}\} \subset \mathbb{R}^{n+k-1}$$

where all scenarios are defined analogously like in the proof of Theorem 5.17 considering a maximization problem now. We can show analogously to the proof of Theorem 5.17 that, if the optimal value is not 0, each optimal solution of problem

$$\min_{\substack{x^{(1)},\ldots,x^{(k)} \in \{0,1\}^n \\ \bar{a}^\top x^{(i)} \leq b \;\; i=1,\ldots,k}} \max_{c \in U'} \min_{i=1,\ldots,k} c^\top x^{(i)}$$

must contain solution $x^{(i)} = e_{n+i}$ for each $i = 1, \ldots, k-1$ and a solution $x^{(k)}$ which fulfills $x_j^{(k)} = 0$ for all $j = n+1, \ldots, n+k-1$ and $\bar{c}_i^\top x^{(k)} \leq 0$ for all $i = 1, \ldots, m$. Again we can show that $x^{(k)}$ projected to the first n dimensions then has to be the optimal solution of (M_0^2). If the optimal value of the created min-max-min instance is 0 then again the zero vector must be the optimal solution of Problem (M_0^2). □

Corollary 5.20. For any fixed $k \in \mathbb{N}$ and fixed m, Problem (M^3) is *NP*-hard for the knapsack problem for uncertainty sets U with $|U| = m$.

Proof. Since the deterministic knapsack problem is *NP*-hard, we can reduce it to the min-max-min problem by adding the given cost-vector of the deterministic problem to U and then by adding an appropriate number of redundant scenarios to U. An optimal solution for any k then only contains the optimal solution of the deterministic knapsack problem which proves the result. □

Note that by Proposition 5.3 for $k \geq m$ Problem (M^3) can be solved in pseudopolynomial time for the knapsack problem.

Corollary 5.21. For any fixed $k \in \mathbb{N}$, Problem (M^3) with a discrete uncertainty set U is strongly *NP*-hard for the knapsack problem.

Proof. The min-max variant of the knapsack problem is strongly *NP*-hard if the number of scenarios is not constant (see Table 3.1). The result follows since all numbers in the construction in the proof of Theorem 5.19 are polynomial in $|U|$. □

5.2 Pseudopolynomial Algorithms

As we have seen in Table 3.1, Problem (M^2) is weakly *NP*-hard for many of the listed combinatorial problems if the number of scenarios is constant, i.e. it can be solved in pseudopolynomial time for these problems. In this section we show that we can reduce Problem (M^3) to Problem (M^2) if the number of scenarios is constant. As a corollary we can solve Problem (M^3) in (pseudo-)polynomial time, if we can solve (M^2) in (pseudo-)polynomial time.

Theorem 5.22. Algorithm 4 calculates an optimal solution for Problem (M^3).

Proof. Let $\{x^{(1)}, \ldots, x^{(k)}\} \subseteq X$ be an optimal solution of Problem (M^3). Choose a partition $U_1, \ldots, U_k$ of U such that

$$U_i \subseteq \{(c, c_0) \in U \mid c^\top x^{(i)} + c_0 \leq c^\top x^{(j)} + c_0 \ \forall j \in \{1, \ldots, k\}\}$$

for all $i = 1, \ldots, k$. Thus U_i is the set of scenarios covered by the solution $x^{(i)}$. Let $\{\bar{x}^{(1)}, \ldots, \bar{x}^{(k)}\}$ be the solution calculated by Algorithm 4 and $\bar{U}_1, \ldots, \bar{U}_k$

Algorithm 4 Reduction from Problem (M^3) to Problem (M^2) for $k < m$

Input: $U = \{(c_1, (c_1)_0), \ldots, (c_m, (c_m)_0)\} \subset \mathbb{R}^{n+1}$, $X \subseteq \{0,1\}^n$, $k < m$
Output: optimal solution $\left\{x^{(1)}, \ldots, x^{(k)}\right\}$ of Problem (M^3)

1: $v := \infty$
2: **for** all k-partitions $U_1, \ldots, U_k$ of U **do**
3: **if** $\max_{i=1,\ldots,k} \left\{\min_{x\in X} \max_{(c,c_0)\in U_i} c^\top x + c_0\right\} < v$ **then**
4: $v = \max_{i=1,\ldots,k} \left\{\min_{x\in X} \max_{(c,c_0)\in U_i} c^\top x + c_0\right\}$
5: $x^{(i)} = \arg\min_{x\in X} \max_{(c,c_0)\in U_i} c^\top x + c_0 \quad \forall i = 1, \ldots, k$
6: **end if**
7: **end for**
8: **return** $\left\{x^{(1)}, \ldots, x^{(k)}\right\}$

be the related partition for which the $\bar{x}^{(i)}$ were calculated in Step (5). Then for the objective value of $\left\{\bar{x}^{(1)}, \ldots, \bar{x}^{(k)}\right\}$ holds

$$\begin{aligned}
\max_{(c,c_0)\in U} \min_{j=1,\ldots,k} c^\top \bar{x}^{(j)} + c_0 &= \max_{i=1,\ldots,k} \max_{(c,c_0)\in \bar{U}_i} \min_{j=1,\ldots,k} c^\top \bar{x}^{(j)} + c_0 \\
&\leq \max_{i=1,\ldots,k} \max_{(c,c_0)\in \bar{U}_i} c^\top \bar{x}^{(i)} + c_0 \\
&= \max_{i=1,\ldots,k} \left\{\min_{x\in X} \max_{(c,c_0)\in \bar{U}_i} c^\top x + c_0\right\} \\
&\leq \max_{i=1,\ldots,k} \left\{\min_{x\in X} \max_{(c,c_0)\in U_i} c^\top x + c_0\right\}
\end{aligned} \tag{5.1}$$

where the last inequality follows since $\bar{U}_1, \ldots, \bar{U}_k$ is the partition which minimizes the last expression according to Step (3). Furthermore we have

$$\begin{aligned}
\min_{x\in X} \max_{(c,c_0)\in U_i} c^\top x + c_0 &\leq \min_{j=1,\ldots,k} \max_{(c,c_0)\in U_i} c^\top x^{(j)} + c_0 \\
&\leq \max_{(c,c_0)\in U_i} c^\top x^{(i)} + c_0 \\
&= \max_{(c,c_0)\in U_i} \min_{j=1,\ldots,k} c^\top x^{(j)} + c_0
\end{aligned} \tag{5.2}$$

where the last equality follows from the choice of the sets U_i. Hence applying the maximum on both sides of the latter inequality we obtain

$$\max_{i=1,\ldots,k} \left\{\min_{x\in X} \max_{(c,c_0)\in U_i} c^\top x + c_0\right\} \leq \max_{(c,c_0)\in U} \min_{j=1,\ldots,k} c^\top x^{(j)} + c_0 \ ,$$

where the right hand side is the optimal value of Problem (M^3). Equation (5.1) together with the last inequality then proves that $\left\{\bar{x}^{(1)}, \ldots, \bar{x}^{(k)}\right\}$ is an optimal solution of (M^3). □

Corollary 5.23. There exist pseudopolynomial algorithms for Problem (M^3) for the shortest path problem, the spanning tree problem, the perfect matching problem in planar graphs, and the knapsack problem if the number of scenarios is constant.

Proof. Algorithm (4) solves an instance of Problem (M^2) a polynomial number of times if the number of scenarios is constant. By Table 3.1 and by [6] there exist pseudopolynomial algorithms for all the problems listed in the theorem which proves the result. □

Corollary 5.24. There exists a polynomial time algorithm for Problem (M^3) with fixed number of scenarios for the minimum cut problem.

Proof. By Table 3.1 the min-max version of the minimum cut problem can be solved in polynomial time for a fixed number of scenarios. By the same argument as above Algorithm 4 provides a polynomial time algorithm for the min-max-min version of the minimum cut problem. □

5.3 Approximation Complexity

In Section 5.1 we showed that Problem (M^3) is *NP*-hard for many combinatorial problems. Therefore it would be interesting to know if an FPTAS exists for these problems. In this section we prove that an ε-approximation algorithm for Problem (M^3) with uncertainty set $U = \{(c_1, (c_1)_0), \ldots, (c_m, (c_m)_0)\}$ can be derived by replacing the exact min-max problems in Algorithm 4 by any ε-approximation algorithm for the min-max problem. To this end let A be an ε-approximation algorithm of Problem (M^2) with a discrete uncertainty set, i.e. an algorithm which returns for each discrete uncertainty set U and each parameter $\varepsilon > 0$ a solution $\bar{x} := A(U, \varepsilon) \in X$ such that

$$\max_{(c,c_0)\in U} c^\top \bar{x} + c_0 \leq (1+\varepsilon)\left(\min_{x\in X} \max_{(c,c_0)\in U} c^\top x + c_0\right).$$

In the following we define for any feasible solution $x \in X$

$$\mathrm{val}(x) := \max_{(c,c_0)\in U} c^\top x + c_0.$$

We prove in the following that under the latter assumptions Algorithm 5 calculates an ε-approximate solution of Problem (M^3).

Algorithm 5 Approximation algorithm for Problem (M^3)

Input: $U = \{(c_1, (c_1)_0), \ldots, (c_m, (c_m)_0)\} \subset \mathbb{R}^{n+1}$, $X \subseteq \{0,1\}^n$, $k < m$, $\varepsilon > 0$ and an ε-approximation algorithm A.

Output: ε-approximate solution $\left\{x^{(1)}, \ldots, x^{(k)}\right\}$ of Problem (M^3)

1: $v := \infty$
2: **for** all k-partitions $U_1, \ldots, U_k$ of U **do**
3: **if** $\max_{i=1,\ldots,k} \{\text{val}(A(U_i, \varepsilon))\} < v$ **then**
4: $v = \max_{i=1,\ldots,k} \{\text{val}(A(U_i, \varepsilon))\}$
5: $x^{(i)} = A(U_i, \varepsilon) \quad \forall i = 1, \ldots, k$
6: **end if**
7: **end for**
8: **return** $\left\{x^{(1)}, \ldots, x^{(k)}\right\}$

Theorem 5.25. If A is an ε-approximation algorithm defined like above, then Algorithm 5 calculates a feasible solution $\left\{\bar{x}^{(1)}, \ldots, \bar{x}^{(k)}\right\}$ of Problem (M^3) such that

$$\max_{(c,c_0)\in U} \min_{i=1,\ldots,k} c^\top \bar{x}^{(i)} + c_0 \leq (1+\varepsilon) \left(\min_{x^{(1)},\ldots,x^{(k)}\in X} \max_{(c,c_0)\in U} \min_{i=1,\ldots,k} c^\top x^{(i)} + c_0 \right)$$

Proof. The result can be proved by following the proof of Theorem 5.22. Let $\left\{x^{(1)}, \ldots, x^{(k)}\right\} \subseteq X$ be an optimal solution of Problem (M^3). Choose a partition $U_1, \ldots, U_k$ of U such that

$$U_i \subseteq \left\{(c, c_0) \in U \mid c^\top x^{(i)} + c_0 \leq c^\top x^{(j)} + c_0 \ \forall j \in \{1, \ldots, k\}\right\}$$

for all $i = 1, \ldots, k$. Thus U_i is the set of scenarios covered by the solution $x^{(i)}$. Let $\left\{\bar{x}^{(1)}, \ldots, \bar{x}^{(k)}\right\}$ be the solution calculated by Algorithm 5 and $\bar{U}_1, \ldots, \bar{U}_k$ be the related partition for which the $\bar{x}^{(i)}$ were calculated in Step (5). Then by equation (5.1) for the objective value of $\left\{\bar{x}^{(1)}, \ldots, \bar{x}^{(k)}\right\}$ holds

$$\max_{(c,c_0)\in U} \min_{j=1,\ldots,k} c^\top \bar{x}^{(j)} + c_0 \leq \max_{i=1,\ldots,k} \{\text{val}(A(U_i, \varepsilon))\}.$$

Furthermore by equation (5.2) we have

$$\begin{aligned} \text{val}(A(U_i, \varepsilon)) &\leq (1+\varepsilon) \left(\min_{x\in X} \max_{(c,c_0)\in U_i} c^\top x + c_0 \right) \\ &\leq (1+\varepsilon) \left(\max_{(c,c_0)\in U_i} \min_{j=1,\ldots,k} c^\top x^{(j)} + c_0 \right) \end{aligned}$$

Hence applying the maximum on both sides of the latter inequality like in the proof of Theorem 5.22 we obtain

$$\max_{i=1,\dots,k} \{\mathrm{val}(A(U_i,\varepsilon))\} \leq (1+\varepsilon)\left(\max_{(c,c_0)\in U} \min_{j=1,\dots,k} c^\top x^{(j)} + c_0\right) ,$$

and together with the previous equation it follows

$$\max_{(c,c_0)\in U} \min_{j=1,\dots,k} c^\top \bar{x}^{(j)} + c_0 \leq (1+\varepsilon)\left(\max_{(c,c_0)\in U} \min_{j=1,\dots,k} c^\top x^{(j)} + c_0\right)$$

which proves the result. □

Corollary 5.26. Problem (M^3) admits an FPTAS for the shortest path problem, the minimum spanning tree problem, and the knapsack problem if the number of scenarios is fixed.

Proof. It was shown in [3] that the min-max versions of the problems listed in the theorem admit an FPTAS if the number of scenarios is constant. Since Algorithm (5) calls Algorithm A a polynomial number of times if the number of scenarios is assumed to be constant, replacing A with the algorithms in [3] proves the result. □

In fact in [3] an FPTAS for (M_0^2) for the previous problems with scenario set $U = \{c_1, \dots, c_m\}$ is derived by an FPTAS for the related multicriteria problem

$$\min_{x\in X} \begin{pmatrix} c_1^\top x \\ \vdots \\ c_m^\top x \end{pmatrix} \tag{5.3}$$

as it was shown in Theorem 3.3 for Problem (M^2). By an analogous proof the same result can be proved for Problem (M^3).

Theorem 5.27. If the multicriteria Problem (5.3) has a fully polynomial approximation scheme, then Problem (M^3) with fixed k and

$$U = \{(c_1, (c_1)_0), \dots, (c_m, (c_m)_0)\}$$

has a fully polynomial approximation scheme.

Note that the result of the latter theorem is stronger than the result of Theorem 5.25 since by an FPTAS for the multicriteria problem an FPTAS for Problem (M^2) can be derived by Theorem 3.3. Nevertheless it can be possible that the min-max version of a combinatorial problem admits an FPTAS while the corresponding multicriteria problem does not and hence both results are stated above.

Chapter 6

Experiments

In this section we present practical experiments for the exact Algorithm 2 presented in Section 4.1.2 and for the heuristic algorithm presented in Section 4.2.2. The former algorithm for Problem (M^3) without an uncertain constant was implemented for the knapsack problem and intensively studied on benchmark instances of the vehicle routing problem, which we define later. The heuristic algorithm was implemented for the shortest path problem and tested on the instances of [41].

The different steps in Algorithm 2 were implemented as follows: the deterministic oracle in Step (5) was solved by problem-specific algorithms which we will explain in the related subsection. The dual problem which has to be solved in Step (4) was implemented in CPLEX 12.5 for the computations of the knapsack problem and the shortest path problems while it was implemented in CPLEX 12.4 for the computations of the vehicle routing problem. To obtain the coefficients λ_j of the convex combination in Step (8) of Algorithm 2 we solved the continuous problem

$$\begin{aligned} &\min_{x} \max_{c \in U} c^\top x \\ &s.t.\ x = \sum_{j=0}^{i} \lambda_j x_j^* \\ &\sum_{j=0}^{i} \lambda_j = 1 \\ &\lambda_j \geq 0, \quad j = 0, \ldots, i \end{aligned}$$

by CPLEX where $x_0^*, \ldots, x_i^*$ are the calculated solutions in Step (5). Note that for ellipsoidal and polyhedral uncertainty the latter problem can be reformulated as in Propositions 4.6 and 4.10 and can be implemented in CPLEX.

All implementations regarding the specific problem and the uncertainty sets are explained in the following sections.

6.1 Exact Algorithm

From Theorem 4.7 it follows that we can solve problem (M^3) for any combinatorial problem, given by an oracle which solves the deterministic problem, by a polynomial time algorithm. Since the algorithm given in the proof is mainly based on the results in [40] which are again based on the ellipsoid method, the algorithm is hard to implement and probably not very efficient. In contrast to this we presented Algorithm 2 in Section 4.1.2 which is not provably a polynomial time algorithm but easy to implement and still based on an oracle for the deterministic problem. Additionally it uses a dual oracle which depends on the uncertainty set. In the following we will show computational results for the knapsack problem and the vehicle routing problem each for ellipsoidal and budgeted uncertainty.

6.1.1 Knapsack Problem

In the following we give some evidence that Algorithm 2 performs very well in practice for the knapsack problem. We implemented it to solve (M^3) for

$$X := \{x \in \{0,1\}^n \mid a^\top x \leq b\}$$

where $a \in \mathbb{N}^n$ and $b \in \mathbb{N}$. As uncertainty sets we chose ellipsoidal and budgeted uncertainty for the profits with a certain constant. The ellipsoidal uncertainty sets are given by

$$U^E := \left\{(c,0) \in \mathbb{R}^{n+1} \mid (c-\bar{c})^\top \Sigma^{-1}(c-\bar{c}) \leq \Omega^2\right\} ,$$

where $\Sigma \in \mathbb{Q}^{n\times n}$ is a symmetric positive semidefinite matrix, $\bar{c} \in \mathbb{Q}^n$ a given center point and $\Omega \in \mathbb{N}$. The budgeted uncertainty sets are given by

$$U^\Gamma := \left\{(c,0) \in \mathbb{R}^{n+1} \mid c_i = \bar{c}_i + \delta_i d_i,\ \sum_{i=1}^{n} \delta_i \leq \Gamma\right\} ,$$

where Γ is a given parameter, $\bar{c} \in \mathbb{Q}^n$ the mean vector and $d \in \mathbb{Q}^n$ the deviation vector.

For our experiments, we created instances similar to those used in [20]. For any $n \in \{250, 500, 750\}$ we created 10 random knapsack instances each

with a random ellipsoid. The weights a_i were chosen randomly from the set $\{100, \ldots, 1500\}$ and b was set to $100n$. For the ellipsoidal uncertainty sets the ellipsoid center $\bar{c}$ was chosen randomly with $\bar{c}_i \in \{10000, \ldots, 15000\}$. The extreme rays of the ellipsoid were calculated as random orthonormal bases where the length of the rays were chosen as $\sqrt{\delta_j c_j}$, where δ_j is a random number in $[0, 1]$. Note that the resulting ellipsoids are not axis-parallel in general and therefore more general than the ellipsoids in [20]. For any instance, we scaled the ellipsoid by varying the parameter Ω from 1 to 5.

Additionally, to compare our algorithm to the mixed-integer linear problem formulation given in Theorem 3.15, we implemented our algorithm for the knapsack problem with gamma-uncertainty sets. Again we created 10 random knapsack instances as above, each equipped with a gamma-uncertainty set. As in [55], the mean vector $\bar{c}$ was chosen randomly with

$$\bar{c}_i \in \{10000, \ldots, 15000\},$$

and d_i was set to $0.1\bar{c}_i$ for all $i = 1, \ldots, n$. Each instance has been solved for all values of Γ from the set $\{0.05n, 0.1n, 0.15n, 0.25n, 0.5n\}$, rounded down if fractional.

To solve the deterministic knapsack problem in Step (5) of Algorithm 2 we used the dynamic programming algorithm mentioned in Example 2.23. The dual problem which has to be solved in Step (4) is, in the case of ellipsoidal uncertainty, a continuous quadratically constrained problem of the form

$$\begin{aligned} &\max z \\ &s.t.\ c^\top x_j^* \geq z \ \forall j = 1, \ldots, i \\ &\quad (c - \bar{c})^\top \Sigma^{-1} (c - \bar{c}) \leq \Omega^2 \\ &\quad z \in \mathbb{R}, c \in \mathbb{R}^n \end{aligned}$$

while for budgeted uncertainty it is a continuous linear problem

$$\begin{aligned} &\max z \\ &s.t.\ c^\top x_j^* \geq z \ \forall j = 1, \ldots, i \\ &\quad c_i = \bar{c}_i + \delta_i d_i \\ &\quad \sum_{i=1}^{n} \delta_i \leq \Gamma \\ &\quad z \in \mathbb{R},\ c \in \mathbb{R}^n,\ \delta \in [0, 1]^n. \end{aligned}$$

We used CPLEX 12.5. to solve both problems.

n	Ω	diff	$\lvert X^* \rvert$	iter	t_{dual}	t_{comb}	t_{tot}
250	1	6.3	7.2	9.8	1.1	0.7	2.5
	2	12.4	19.7	27.2	3.7	1.9	6.3
	3	18.3	44.4	54.8	8.3	3.9	13.0
	4	23.9	77.9	89.4	14.8	6.3	22.0
	5	29.2	135.5	154.5	29.8	11.0	41.7
500	1	4.5	10.5	13.9	9.5	3.9	18.5
	2	8.9	26.5	33.8	25.9	9.6	41.2
	3	13.2	72.4	79.1	67.1	22.4	94.5
	4	17.4	123.4	134.7	120.8	38.2	165.1
	5	21.5	147.3	194.9	182.5	55.3	243.1
750	1	3.6	14.9	19.1	42.6	12.4	69.8
	2	7.2	48.7	54.8	139.4	35.3	188.6
	3	10.7	142.1	146.3	383.6	93.1	493.2
	4	14.2	163.2	168.8	457.3	107.5	581.4
	5	17.5	243.0	252.0	808.8	160.7	986.8

Table 6.1: Results for the knapsack problem with ellipsoidal uncertainty.

The results of the computations are listed in Tables 6.1 and 6.2. For each combination of n and Ω or n and Γ, respectively, we show the average over all 10 instances of the following numbers (from left to right): the difference (in percent) of the objective value of Problem (M^3) to the value of the certain problem with the ellipsoid center $\bar{c}$ or the mean vector $\bar{c}$, respectively, as cost function; the number of solutions in the computed set X^*; the number of major iterations; the run-times used by the two oracles (t_{dual} for the dual problem in Step (4) and t_{comb} for solving the certain combinatorial problem M in Step (5)); and the total run-time. All times are given in CPU seconds, on an Intel Xeon processor running at 2.5 GHz.

For ellipsoidal uncertainty the results show that run-times increase with Ω and (of course) n. However, even the hardest instances with $n = 750$ and $\Omega = 5$ could be solved within 16.5 minutes on average. Interestingly, the number of solutions needed in order to solve Problem (M^3) to optimality (and even the number of iterations) usually remains well below n, in particular for small uncertainty sets. The higher Ω, i.e. the larger the size of the uncertainty set is, the more solutions are calculated. This is quite intuitive since if there are more uncertain scenarios there must be more solutions to keep the worst case small. Here the oracle for the dual problem takes most of the run-time which is in contrast to the results in the following section and to the results for budgeted uncertainty in this section.

For budgeted-uncertainty, the results are even more positive, with much shorter run-times, less iterations, and significantly smaller solution sets X^*. Contrarily to the computations with ellipsoidal uncertainty, here the combi-

n	Γ	diff	$\lvert X^* \rvert$	iter	t_{dual}	t_{comb}	t_{tot}
250	12	1.8	1.8	3.0	0.00	0.2	0.3
	25	3.6	3.0	4.6	0.00	0.3	0.4
	37	5.2	4.0	5.9	0.00	0.4	0.5
	62	8.2	8.9	14.7	0.00	1.0	1.1
	125	10.0	1.0	8.2	0.00	0.6	0.7
500	25	1.8	5.5	10.6	0.00	3.0	3.3
	50	3.6	9.3	21.1	0.00	5.9	6.3
	75	5.2	12.6	30.0	0.01	8.5	8.9
	125	8.2	18.5	50.7	0.02	14.3	14.7
	250	10.0	1.0	9.8	0.00	2.7	3.1
750	37	1.8	5.4	9.9	0.00	6.3	7.0
	75	3.6	10.1	28.5	0.01	18.0	18.9
	112	5.3	14.4	40.8	0.02	25.9	26.7
	187	8.3	25.2	82.3	0.07	52.5	53.5
	375	10.0	1.0	7.2	0.00	4.6	5.3

Table 6.2: Results for the knapsack problem with gamma-uncertainty.

natorial oracle takes most of the total run-time, while the dual oracle runs less than a tenth of a second for all instance sizes. This is because instead of the quadratic problem here CPLEX has to solve a linear problem which can be done more efficiently. For comparison, we also performed experiments using the formulation of Theorem 3.15. It turned out however that CPLEX was not able to solve the corresponding problem within hours even for $n = 20$.

To obtain some insight into the typical structure of an optimal solution computed by Algorithm 2, we picked one instance with ellipsoidal uncertainty and counted in how many of the computed solutions a given object is used. The result is shown in Figure 6.1, where objects are sorted (from left to right) by the number of appearances. It turns out that more than half of the objects are never used, about one fifth is used in every computed solution, while only the remaining objects are used in a non-empty proper subset of the solutions. Similar pictures are obtained when considering other instances.

6.1.2 Vehicle Routing Problem

As mentioned in the introduction a typical practical motivation for problem (M^3) is a parcel service which has to deliver parcels to the same customers every day. Depending on the traffic situation every day the company wants to find the best solution to serve all customers with the available fleet of vehicles. This so called vehicle routing problem which we will define properly later is very hard to solve in practice for high dimensions even in the deter-

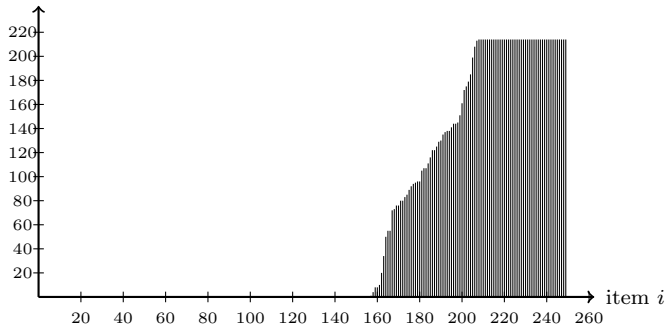


Figure 6.1: One instance for the knapsack problem in dimension 250. The number of calculated solutions is 214; the y-axis shows the number of solutions in which an item i is selected.

ministic case. A set of benchmark instances can be found in [29]. Several of the latter instances have not been solved to optimality yet. One advantage of problem (M^3) is that it can be solved in a preprocessing which can take a long time but afterwards the best of the pre calculated solutions depending on the traffic situation can be chosen every day. This can be done easily by comparing the objective values of the calculated solutions. Moreover Problem (M^3) hedges against uncertainty in a robust way but is not as conservative as the normal min-max problem (M^2). In this section we will present a case study on our computations for Problem (M^3) on some small benchmark instances of the deterministic capacitated vehicle routing problem presented in [29].

In the literature many different variants of the vehicle routing problem were presented [47, 57]. For our computations we will concentrate on the *capacitated vehicle routing problem*, which is defined in the following. Let $G = (V, E)$ be a directed complete graph with nodes $V = \{0, 1, \ldots, n\}$ and $c : E \to \mathbb{R}$ a cost function on the edges of G where the cost can be interpreted as traveling times. Node 0 represents the *depot* and the nodes $V \setminus \{0\}$ represent the *customers*. Each customer $i \in V \setminus \{0\}$ has a positive demand $d_i \in \mathbb{R}_+$ while we set $d_0 = 0$. Furthermore we have a set of *vehicles* $K = \{1, \ldots, m\}$ where each vehicle has the same *capacity* $C \in \mathbb{R}_+$. A *tour* $T \subset E$ in G is a cycle in G which traverses the depot 0. This can be interpreted as a tour of a vehicle which starts in the depot and supplies a subset of customers before it returns to the depot. The cost of a tour is defined by

$$c(T) := \sum_{e \in T} c(e)$$

and the demand of a tour is the sum over all demands of the customers which

are passed by the tour, i.e.

$$d(T) := \sum_{\{v \in V:\ \exists\ e=(v,w) \in T\}} d_v.$$

The capacitated vehicle routing problem is now to find a set of m tours which minimize the total cost such that the sum of all demands on each tour does not exceed the capacity of the vehicle. Formally we define the problem as follows.

Problem 6.1 (Capacitated Vehicle Routing Problem (CVRP))**.** Let $G = (V, E)$ be a complete directed graph and c and d defined like above. Find a set of distinct tours $T_1, \ldots, T_m \subset E$ with $d(T_i) \leq C$ for $i = 1, \ldots, m$ and with minimal cost

$$c(T_1, \ldots, T_m) := c(T_1) + \ldots + c(T_m)$$

such that each customer $i \in V \setminus \{0\}$ is traversed by exactly one of the tours.

Note that in Problem (6.1) every vehicle has to be used. The same problem with variable number of vehicles is also studied in the literature. In this case often extra costs for each vehicle are defined which have to be paid by the company if it makes use of the vehicle. Another assumption we make is that the graph has to be directed. This is due to the fact that in real-world applications the travel time from A to B can be different than the travel time from B to A.

Theorem 6.2 ([47])**.** The capacitated vehicle routing problem is *NP*-hard.

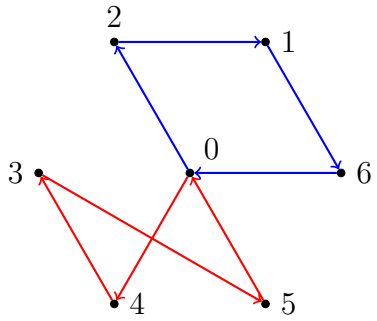


Figure 6.2: A feasible solution of instance E-n7-k2 with $m = 2$.

In the literature several methods and algorithms were presented to solve the capacitated vehicle routing problem. Besides dynamic programming methods

one way to solve the CVRP is to formulate it as an integer program and then solve it by Branch & Bound methods. In Algorithm 2 we use an integer programming formulation called *Miller-Tucker-Zemlin* formulation to solve the deterministic oracle in Step (5) (see [38]). Here we identify each edge $e = (i, j) \in E$ with a variable $x_{ij} \in \{0, 1\}$ which has value one if the edge is contained in any of the m tours and zero otherwise. Additionally we add variables $u_i \geq 0$ for each customer $i = 1, \ldots, n$ which represent the total demand a vehicle supplied up to the point immediately after it lefts i. The Miller-Tucker-Zemlin problem is then of the form

$$\begin{aligned}
\min \quad & c^\top x && (6.1)\\
s.t. \quad & \sum_{i \in V, i \neq j} x_{ij} = \sum_{i \in V, i \neq j} x_{ji} = 1 \ \ \forall\, j = 1, \ldots, n && (6.2)\\
& \sum_{i \in V, i \neq 0} x_{i0} = \sum_{i \in V, i \neq 0} x_{0i} = m && (6.3)\\
& u_j - u_i + C(1 - x_{ij}) \geq q_j \ \ \forall i, j \in V \setminus \{0\},\ i \neq j && (6.4)\\
& q_i \leq u_i \leq C \ \ \forall i \in V \setminus \{0\} && (6.5)\\
& x_{ij} \in \{0, 1\} \ \ i, j = 1, \ldots, n && (6.6)\\
& u_i \in \mathbb{R}_+ \ i = 1, \ldots, n && (6.7)
\end{aligned}$$

Here the constraints (6.2) and (6.3) make sure that each customer is visited exactly once on a tour and that exactly m vehicles leave the depot and return to it. The constraints (6.4) ensure that if the edge (i, j) is passed by a vehicle then $u_j = u_i + q_j$ holds and otherwise constraints (6.5) ensure that u_i does not exceed the capacity and has at least the value q_i. It can be easily proved that each feasible solution of the latter problem induces a feasible solution for Problem (6.1) and that each feasible solution for Problem (6.1) is feasible for the latter problem.

In the following we present our results on Problem (M^3) for the CVRP. As mentioned before this problem is very hard to solve in practice even in its deterministic version. Since Algorithm (2) has to solve the deterministic problem several times, clearly our algorithm is at most as efficient as the algorithm for the underlying deterministic version. Especially the instances which have not been solved to optimality yet in its deterministic version can not be solved by our algorithm as well. To improve the run-time of Algorithm 2 we adjusted the algorithm as follows. In the first loops of the algorithm instead of calculating an exact solution for the deterministic problem in Step (5) we used a heuristic algorithm to find any feasible solution. If this solution does not satisfy the stopping criteria of the loop we go on using the heuristic algorithm. If no such solution is found anymore we switch back

to the exact algorithm. Clearly in the last loop the exact algorithm is used to obtain an exact solution of Problem (M^3). Furthermore we ran a further variant of the algorithm which only uses the heuristic algorithm without ever switching back to the exact algorithm. By this idea we obtain a heuristic algorithm for Problem (M^3). For the computations below we implemented the original Algorithm 2 and the two variants of it which are explained above. As a heuristic for the deterministic problem we used the open source library VRPH (https://projects.coin-or.org/VRPH) while for the exact algorithm we implemented the Miller-Tucker-Zemlin formulation in CPLEX 12.4. The dual problem in Step (4) was also implemented in CPLEX. All computations were calculated on a cluster of 64-bit Intel(R) Xeon(R) E5-2670 processors running at 2.60 GHz with 20MB cache.

For our computations we chose several instances from [29] and similar to the instances in the previous sections we created 10 ellipsoidal and 10 budgeted uncertainty sets for each instance. To this end as the mean cost vector $\bar{c}$ we chose the Euclidean distances between the coordinates of the nodes given by the instances. If no coordinates are defined then we chose the edge-weights which are given by the instance.

The ellipsoidal uncertainty sets are then given by

$$U^E := \left\{(c,0) \in \mathbb{R}^{n+1} \mid (c-\bar{c})^\top \Sigma^{-1}(c-\bar{c}) \leq \Omega^2\right\} ,$$

where $\Sigma \in \mathbb{Q}^{n\times n}$ is a symmetric positive semidefinite matrix, $\bar{c} \in \mathbb{Q}^n$ defined like above and $\Omega \in \mathbb{N}$. The extreme rays of the ellipsoids were calculated as random orthonormal bases where the length of the rays were chosen as $\sqrt{\delta_j c_j}$, where δ_j is a random number in $[0,1]$. Note that the resulting ellipsoids are not axis-parallel in general and therefore more general than the ellipsoids in [20]. For any instance, we scaled the ellipsoid by varying the parameter $\Omega \in \{1,3,5\}$.

The budgeted uncertainty sets are given by

$$U^\Gamma := \left\{(c,0) \in \mathbb{R}^{n+1} \mid c_i = \bar{c}_i + \delta_i d_i,\ \sum_{i=1}^{n} \delta_i \leq \Gamma\right\} ,$$

where Γ is a given parameter, $\bar{c} \in \mathbb{Q}^n$ defined like above and $d \in \mathbb{Q}^n$ is the deviation vector. Here we chose d_i randomly between 0 and $\bar{c}_i$ for all $i = 1,\ldots,n$. Each instance has been solved for all values of Γ from the set

$$\{0.15(n+m), 0.5(n+m), 0.75(n+m)\} ,$$

rounded down if fractional. The latter choice is motivated by the fact that each feasible solution of Problem (M^3) uses exactly $n+m$ edges in the graph.

For $\Gamma \geq n + m$ our computations showed that often only one solution is calculated which is the optimal robust min-max solution.

					exact oracle					exact & heuristic oracle					
Inst.	n	m	Ω	diff	$\lvert X^* \rvert$	iter	t_{tot}	t_{dual}	t_{comb}	$\lvert X^* \rvert$	iter_e	iter_a	t_{tot}	t_{dual}	t_{comb}
E-n7-k2	6	2	1	14.4	11.8	14.5	0.3	0.0	0.3	11.8	8.5	8.4	0.7	0.0	0.6
			3	35.8	13.7	18	0.3	0.0	0.3	13.9	10.1	10.7	0.8	0.0	0.7
			5	56.0	16.3	20.2	0.3	0.0	0.3	17.8	12.4	13.9	1.0	0.1	0.8
E-n13-k4	12	4	1	11.0	17.9	23.5	125.5	1.1	124.3	23.2	14.8	22.2	98.1	1.9	96.0
			3	28.8	28.5	44.6	266.6	2.0	264.4	30.4	27.6	44.4	201.5	3.8	197.5
			5	44.1	34.8	57.7	258.6	2.8	255.7	36.0	35.8	57.0	215.4	5.0	210.2
P-n16-k8	15	8	1	11.0	14.5	16.8	21.8	2.0	19.6	14.7	12.7	8.5	23.1	2.7	20.0
			3	30.3	21.6	26.1	28.4	2.9	25.2	21.2	18.9	14.3	29.3	3.9	25.0
			5	48.5	22.6	29	25.8	3.3	22.2	25.5	23.6	19.6	32.3	5.3	26.7
gr-n17-k3	16	3	1	26.8	18.9	23.9	61.9	3.8	57.7	19.4	23.1	10.0	69.5	5.2	63.8
			3	64.3	33.5	56.7	560.2	9.1	550.7	33.0	43.6	42.3	581.5	16.1	564.9
			5	94.9	40.5	67.4	1347.4	11.8	1335.1	43.1	46.7	67.8	1463.8	23.2	1440.0

Table 6.3: Results of Algorithm 2 for CVRP with ellipsoidal uncertainty.

In the Tables 6.3, 6.4, 6.5 and 6.6 we list the computational results for a selection of instances. For each combination of n and Ω or n and Γ, respectively, we show the average over all 10 instances of the following numbers (from left to right): the difference (in percent) of the objective value of Problem (M^3) to the value of the certain problem with the ellipsoid center $\bar{c}$ or the mean vector $\bar{c}$, respectively, as cost function; the number of solutions in the computed set X^*; the number of major iterations; the run-times used by the two oracles (t_{dual} for the dual problem in Step (4) and t_{comb} for solving the certain combinatorial problem (M) in Step (5)) and the total run-time t_{tot}. Furthermore for the variant of the algorithm where we use both, the exact and the heuristic oracle we show the number of calls of the exact oracle in column iter_e and the number of calls of the heuristic oracle in column iter_h. Table 6.5 and 6.6 which show the results for the second variant of the algorithm, which only uses the heuristic algorithm, includes the column diff_h. In this column the difference (in percent) of the objective value of the calculated heuristic solution and of the optimal value of Problem (M^3) is shown if the latter could be calculated. All times are given in CPU seconds and all numbers are rounded to one decimal.

In Table 6.3 we show the results of Algorithm 2 implemented with the exact oracle and with a combination of the exact oracle together with the heuristic oracle for ellipsoidal uncertainty sets. The number of calculated solutions and the number of iterations clearly grows with increasing Ω. The same holds for the difference of the objective value. Note that in instance gr-n17-k3 the difference is nearly 95% for $\Omega = 5$. Furthermore the total run-time as well as the number of solutions increase with the dimension. Instance P-n16-k8 could be solved very fast compared to the other instances since the deterministic

					exact oracle					exact & heuristic oracle					
Inst.	n	m	Γ	diff	$\lvert X^* \rvert$	iter	t_{tot}	t_{dual}	t_{comb}	$\lvert X^* \rvert$	iter_e	iter_a	t_{tot}	t_{dual}	t_{comb}
E-n13-k4	12	4	2	4.8	11.2	18.4	76.7	0.0	76.7	11.3	11.0	9.7	55.3	0.0	55.3
			8	13.6	17.7	34.4	142.6	0.0	142.6	17.9	23.0	27.5	109.9	0.0	109.8
			12	18.2	18.7	36.7	150.7	0.0	150.7	18.6	27.5	32.0	138.0	0.0	137.9
P-n16-k8	15	8	3	9.0	6.2	8.4	3.3	0.0	3.3	5.5	5.4	6.2	4.0	0.0	3.9
			11	22.5	12.2	22.7	10.4	0.0	10.4	12.5	16.6	13.3	12.5	0.0	12.4
			17	29.0	13.3	23.4	10.0	0.0	10.0	13.4	19.0	14.0	13.5	0.0	13.4
gr-n17-k3	16	3	2	7.1	4.7	6.2	3.3	0.0	3.2	4.8	4.6	4.8	4.7	0.0	4.5
			9	20.2	16.1	25.3	68.4	0.0	68.3	16.1	25.1	13.3	78.3	0.0	78.1
			14	26.2	18.9	33.6	118.6	0.0	118.6	18.9	30.5	17.2	123.0	0.0	122.9
P-n20-k2	19	2	3	6.0	11.9	16.7	389.0	0.0	388.9	11.9	12.5	13.7	358.0	0.0	357.8
			10	14.5	21.1	37.0	2259.8	0.0	2259.7	21.1	24.7	37.2	1860.3	0.0	1860.1
			15	18.8	26.9	53.1	4537.0	0.0	4537.0	26.8	33.6	47.0	3905.4	0.0	3905.1

Table 6.4: Results of Algorithm 2 for CVRP with budgeted uncertainty.

versions were solved faster by the MTZ-formulation. In contrast to the results for the knapsack problem in the previous section here the deterministic oracle takes most of the run-time, while the dual oracle could be solved in a few seconds. Note that the combination of the exact oracle and the heuristic oracle does only improve the run-time in instance E-n13-k4. This is possibly due to the fact that the heuristic solutions do not improve the dual problem significantly in each loop and the resulting higher number of loops increases the run-time. Nevertheless it turned out that for instances with more than 16 customers we could not solve all configurations for the 10 ellipsoidal instances in days.

				heuristic oracle					
Inst.	n	m	Ω	diff_h	$\lvert X^* \rvert$	iter	t_{tot}	t_{dual}	t_{comb}
E-n7-k2	6	2	1	4.4	4.6	5.9	0.2	0.0	0.1
			3	11.7	5.1	6.1	0.2	0.0	0.2
			5	19.6	4.8	5.9	0.2	0.0	0.1
E-n13-k4	12	4	1	1.6	8.2	11.0	1.5	0.4	0.9
			3	4.2	9.8	12.3	1.6	0.4	1.0
			5	8.3	10.3	12.7	1.6	0.5	1.0
P-n16-k8	15	8	1	3.7	4.2	5.3	1.6	0.5	0.8
			3	5.4	5.5	6.5	1.8	0.6	0.9
			5	10.7	3.9	5.1	1.4	0.5	0.6
gr-n17-k3	16	3	1	7.6	5.0	7.2	2.7	0.9	1.3
			3	6.3	14.4	21.7	7.8	3.5	3.9
			5	7.6	15.5	21.4	7.7	3.5	3.7
gr-n21-k3	20	3	1	-	5.8	8.7	6.8	3.4	2.4
			3	-	14.4	17.7	14.4	8.5	4.8
			5	-	18.6	24.8	21.2	13.5	6.5

Table 6.5: Results of Algorithm 2 with heuristic oracle for CVRP with ellipsoidal uncertainty.

The results for budgeted uncertainty in Table 6.4 are very similar but the total run-time is much lower. Again the combination of exact and heuris-

tic algorithm is only faster in instance P-n16-k8. Overall the total run-time is much lower than for ellipsoidal uncertainty sets which is here due to the lower number of iterations which also leads to a lower number of calculated solutions. In contrast to ellipsoidal uncertainty sets we could solve all configurations for instance P-n20-k2. Nevertheless it turned out that for instances with more than 19 customers we could not solve all configurations for the 10 budgeted uncertainty instances in days.

				heuristic oracle					
Inst.	n	m	Γ	diff_h	$\lvert X^* \rvert$	iter	t_{tot}	t_{dual}	t_{comb}
E-n13-k4	12	4	2	4.8	3.5	5.1	0.5	0.0	0.5
			8	7.6	7.1	10.6	1.0	0.0	0.9
			12	8.9	7.0	11.2	1.1	0.0	1.0
P-n16-k8	15	8	3	4.9	2.6	4.0	0.5	0.0	0.5
			11	9.4	3.8	6.1	0.8	0.0	0.7
			17	8.7	4.0	6.0	0.8	0.0	0.7
gr-n17-k3	16	3	2	9.6	2.5	4.5	1.0	0.0	0.8
			9	11.4	5.8	8.3	1.6	0.0	1.5
			14	8.9	8.3	12.5	2.4	0.0	2.3
P-n20-k2	19	2	3	19.1	6.1	9.4	2.1	0.0	1.9
			10	42.6	6.9	9.3	2.0	0.0	1.8
			15	27.8	6.6	9.1	1.9	0.0	1.8
gr-n21-k3	20	3	3	-	3.6	5.7	1.8	0.0	1.6
			11	-	7.7	10.4	3.1	0.0	2.8
			17	-	9.9	13.9	4.0	0.0	3.8

Table 6.6: Results of Algorithm 2 with heuristic oracle for CVRP with budgeted uncertainty.

The results for the heuristic variant of the algorithm are very promising. Both for ellipsoidal and budgeted uncertainty sets the total run-time and the number of calculated solutions is very low compared to the exact versions above. In average the total run-time never exceeded 22 seconds. Furthermore the difference to the exact value is often not higher than 12 % in average. Nevertheless for instance P-n20-k2 it is nearly 43% for $\Gamma = 10$. We could even solve instance gr-n21-k3 in at most 22 seconds in average for ellipsoidal uncertainty sets, while we were not able to solve all the configurations for the exact version to optimality in days.

In the following we present all solutions of an exact optimal solution of Problem (M^3) for a selected instance. Since we assumed a directed graph which can have different costs on edges (i, j) and (j, i) for any customers $i, j \in V \setminus \{0\}$ it can happen that in one optimal solution one set of tours can occur more than once but at least one tour is oriented in a different direction (see solution 8 and 10 in Figure 6.3).

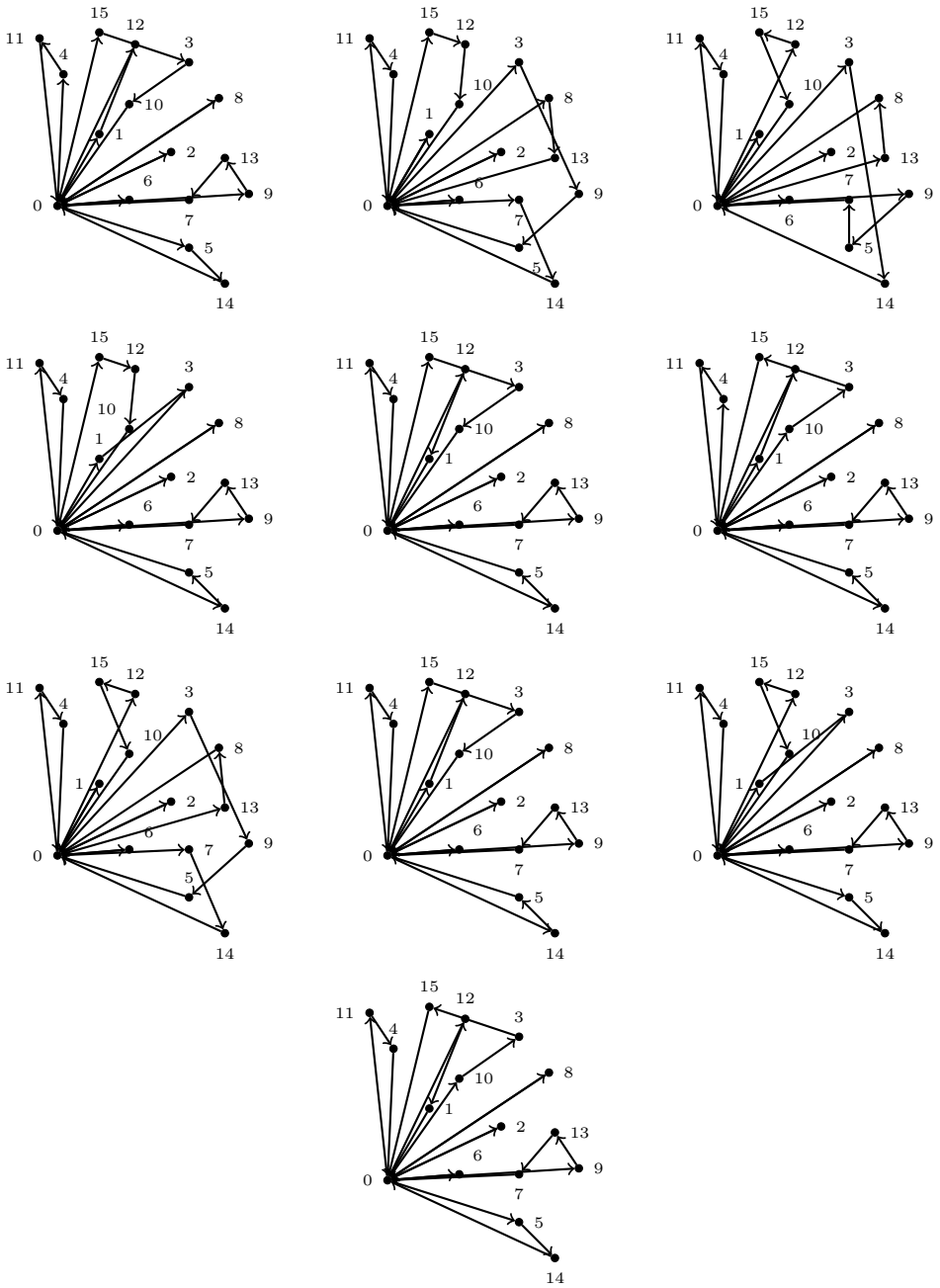


Figure 6.3: The optimal solution of Problem (M^3) for instance P-n16-k8 with a budgeted uncertainty set.

6.2 Heuristic Algorithm

In Section 4.2.2 we presented a heuristic algorithm which is mainly based on the algorithm given in Theorem 4.7. Though we presented an example which shows that the error bound in Theorem 4.20 can be arbitrary large, in this section we provide some computational results showing that the heuristic Algorithm 3 often calculates solutions that are close to optimal even for small k. To this end, we replace the theoretically fast algorithm of Theorem 4.7 by the practically fast Algorithm 2. We applied our heuristic to the instances of the shortest path problem used in [41]. The authors create graphs on $20, 25, \ldots, 50$ nodes, corresponding to points in the Euclidean plane with random coordinates in $[0, 10]$, and choose a budgeted uncertainty set of the form

$$\Xi = \left\{ \xi_{ij} \in [0,1] \mid \textstyle\sum_{ij} \xi_{ij} \leq \Gamma \right\}$$

where the cost on the edges is set to $c_{ij}(\xi) = (1 + \frac{\xi_{ij}}{2})d_{ij}$ and d_{ij} is the Euclidean distance of node i to node j. No uncertain constants are considered in the objective function. The parameter Γ is chosen from $\{3, 6\}$. For more details see Section 4.2 in [41]. Again we solved the linear dual problem in Algorithm 2 by CPLEX 12.5. Here we also implemented the oracle for the deterministic problem using the linear programming formulation for the shortest path problem in CPLEX.

In Table 6.7, the computational results for Algorithm 3 are shown. In columns indicated by #, we find the number of instances (out of 100) for which the problem was solved to optimality by the authors of [41] within a time limit of two hours. For the latter instances, in columns marked Δ_{sol}, we state how far our heuristic solution value is above the exact minimum, on average (in percent). Similarly, for all 100 instances, we denote by Δ_{all} how far our heuristic solution value is above the best solution value found by the authors of [41] within the time limit, i.e., this number includes the instances which could not be solved to proven optimality in [41]. For every type of instance, the run-time for our heuristic was at most one CPU second on average, and is thus not reported in the table. Note that we are able to calculate heuristic solutions for all k up to $n + 1$ at one stroke.

Table 6.7 shows that for every instance size considered, the mean of the differences Δ_{all} with respect to the best known solutions is within 8 %. In Figure 6.4, we illustrate the mean differences in dependence of the problem size and k. Not surprisingly, the gap grows with the number of nodes, whereas a larger number k leads to better solutions. This is also due to the fact that the exact problem (M^3) is much harder to solve for larger k and therefore

		20 nodes			25 nodes			30 nodes			35 nodes		
Γ	k	#	Δ_{sol}	Δ_{all}	#	Δ_{sol}	Δ_{all}	#	Δ_{sol}	Δ_{all}	#	Δ_{sol}	Δ_{all}
3	1	100	4.1	4.1	100	4.6	4.6	100	5.4	5.4	100	6.5	6.5
	2	100	1.9	1.9	99	2.4	2.5	69	2.8	3.0	17	2.6	3.4
	3	97	0.8	0.8	31	0.5	1.2	6	0.1	1.8	0	-	2.2
	4	51	0.2	0.5	6	0.1	0.6	0	-	0.9	0	-	1.3
6	1	100	5.3	5.3	100	4.8	4.8	100	5.2	5.2	100	6.7	6.7
	2	100	3.7	3.7	99	4.6	4.7	67	4.8	5.2	16	7.1	5.8
	3	97	2.3	2.3	38	2.3	3.1	6	0.7	3.5	0	-	4.1
	4	55	1.0	1.4	7	0.5	2.0	0	-	2.3	0	-	3.2

		40 nodes			45 nodes			50 nodes		
Γ	k	#	Δ_{sol}	Δ_{all}	#	Δ_{sol}	Δ_{all}	#	Δ_{sol}	Δ_{all}
3	1	100	7.0	7.0	100	7.3	7.3	100	8.0	8.0
	2	6	3.1	3.9	0	-	3.9	0	-	4.3
	3	0	-	2.5	0	-	2.4	0	-	2.8
	4	0	-	1.6	0	-	1.6	0	-	1.9
6	1	100	7.2	7.2	100	7.8	7.8	100	7.8	7.8
	2	5	5.0	6.6	0	-	7.1	0	-	7.0
	3	0	-	4.5	0	-	4.9	0	-	4.9
	4	0	-	3.3	0	-	3.7	0	-	3.7

Table 6.7: Computational results for Algorithm 3.

the best known solution after the time limit has a worse objective value. In contrast to this our heuristic solution is obtained by the same optimal solution of the Problem ($\mathrm{M}^2[n+1]$) for all k.

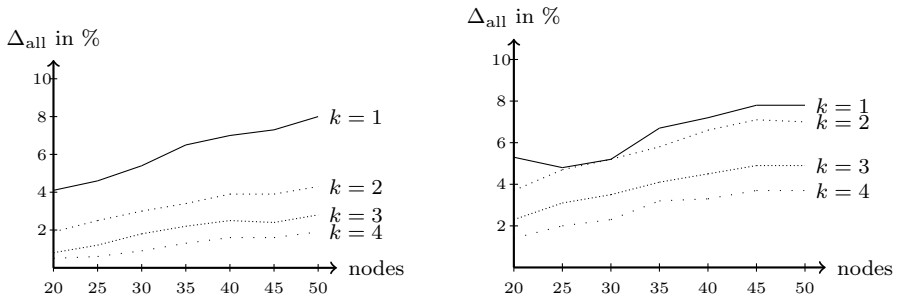


Figure 6.4: Difference to the best known solution in % for $\Gamma = 3$ (left) and $\Gamma = 6$ (right)

Chapter 7

Outlook

In this thesis we intensively studied Problem (M^3) for discrete uncertainty sets and convex uncertainty sets. We proved that for convex uncertainty sets Problem (M^3) is as easy as the underlying problem if the number of calculated solutions is greater than the dimension of the problem. Furthermore we showed that for any fixed number of solutions the problem is *NP*-hard. The *NP*-hardness was also shown for several combinatorial problems with discrete uncertainty sets. Overall most of the relevant complexity questions regarding convex and discrete uncertainty sets have been answered for Problem (M^3) in this thesis.

Nevertheless there are still a lot of interesting open questions regarding Problem (M^3). First of all the algorithm given in Theorem 4.7 heavily depends on the ellipsoid method and the results in [40] and is therefore not efficient to implement. Furthermore the exact algorithm 2 has not provably polynomial run-time. Therefore it would be interesting to find a polynomial time algorithm for Problem (M^3), at least for several combinatorial problems with convex uncertainty sets, which is not based on the ellipsoid method and easier to implement. Another open question is how to practically solve Problem (M^3) for any $k < n + 1$ which is an interesting case for practical applications since the number of calculated solutions can be chosen to be small. Our computations, which make use of the MILP formulation in Theorem 3.15, and the computations in [41] showed that even in small dimension the problem can not be solved in hours.

Further interesting problems could be different variants of problem (M^3) e.g. with uncertainty also occuring in the constraints similar to the problems studied in [41]. One could also apply the idea of Problem (M^3) to other robust approaches presented in Chapter 3, e.g., to regret robustness.

One very promising variant of Problem (M^3) is a distributionally robust version of the problem. The approach of distributional robustness received increasing attention in the last years and has been intensively studied in the literature. The K-adaptability approach which was studied in [41] for two-stage robust problems was also applied to distributionally robust problems [42]. Similar to the two-stage version the K-adaptable approach is used to approximate the distributionally robust problem. Here besides other results, the authors show a very similar result to Theorem 3.14. It is proved that if the number of calculated solutions K is greater or equal to a certain value which depends on the dimension of the problem and other problem parameters then the K-adaptability problem calculates an exact solution. This result can be easily adapted to a distributionally robust version of Problem (M^3) which is the problem

$$\min_{x^{(1)},\dots,x^{(k)} \in X} \sup_{\mathbb{P} \in \mathcal{P}} \mathbb{E}_{\mathbb{P}} \left(\min_{i=1,\dots,k} c^\top x^{(i)} \right)$$

where $\mathcal{P}$ is a given set of probability distributions of the probability vector c. Similar to the result of Corollary 4.3 using the idea of the proof in [42] we can transform the latter problem into a continuous linear program if the number of calculated solutions k is greater or equal a certain value. Therefore under the latter assumption this problem can be solved in pseudopolynomial time.

References

[1] D. Adjiashvili, S. Stiller, and R. Zenklusen. Bulk-robust combinatorial optimization. *Mathematical Programming*, 149(1-2):361–390, 2015.

[2] D. Adjiashvili and R. Zenklusen. An s–t connection problem with adaptability. *Discrete Applied Mathematics*, 159(8):695–705, 2011.

[3] H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation complexity of min-max (regret) versions of shortest path, spanning tree, and knapsack. In *Algorithms – ESA 2005*, volume 3669 of *Lecture Notes in Computer Science*, pages 862–873. Springer, 2005.

[4] H. Aissi, C. Bazgan, and D. Vanderpooten. Complexity of the min–max and min–max regret assignment problems. *Operations research letters*, 33(6):634–640, 2005.

[5] H. Aissi, C. Bazgan, and D. Vanderpooten. Complexity of the min-max (regret) versions of cut problems. In *Algorithms and Computation*, pages 789–798. Springer, 2005.

[6] H. Aissi, C. Bazgan, and D. Vanderpooten. Pseudo-polynomial algorithms for min-max and min-max regret problems. In *5th International Symposium on Operations Research and its Applications (ISORA 2005)*, pages 171–178, 2005.

[7] H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009.

[8] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95(1):3–51, 2003.

[9] A. Armon and U. Zwick. Multicriteria global minimum cuts. *Algorithmica*, 46(1):15–26, 2006.

[10] I. Averbakh and V. Lebedev. Interval data minmax regret network optimization problems. *Discrete Applied Mathematics*, 138(3):289–301, 2004.

[11] I. Averbakh and V. Lebedev. On the complexity of minmax regret linear programming. *European Journal of Operational Research*, 160(1):227 – 231, 2005. Applications of Mathematical Programming Models.

[12] F. Baumann, C. Buchheim, and A. Ilyina. A Lagrangean decomposition approach for robust combinatorial optimization. Technical report, Optimization Online, 2015.

[13] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[14] A. Ben-Tal and A. Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[15] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[16] D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. *Automatic Control, IEEE Transactions on*, 55(12):2751–2766, 2010.

[17] D. Bertsimas and V. Goyal. On the approximability of adjustable robust convex optimization under uncertainty. *Mathematical Methods of Operations Research*, 77(3):323–343, 2013.

[18] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical programming*, 98(1-3):49–71, 2003.

[19] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[20] D. Bertsimas and M. Sim. Robust discrete optimization under ellipsoidal uncertainty sets. 2004.

[21] J. W. Blankenship and J. E. Falk. Infinitely constrained optimization problems. *Journal of Optimization Theory and Applications*, 19(2):261–281, 1976.

[22] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[23] C. Buchheim and J. Kurtz. Min-max-min robust combinatorial optimization. *Optimization Online*, 2016.

[24] C. Buchheim and J. Kurtz. Min-max-min robust combinatorial optimization subject to discrete uncertainty. *Optimization Online*, 2016.

[25] C. Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012.

[26] A. Chassein and M. Goerigk. Min-max regret problems with ellipsoidal uncertainty sets. *arXiv preprint arXiv:1606.01180*, 2016.

[27] G. Cornuejols and R. Tütüncü. *Optimization methods in finance*, volume 5. Cambridge University Press, 2006.

[28] E. Delage and Y. Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations research*, 58(3):595–612, 2010.

[29] B. Diaz. Vrp web, 2012.

[30] M. Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2013.

[31] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460, 2000.

[32] L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997.

[33] L. El Ghaoui, F. Oustry, and H. Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9(1):33–52, 1998.

[34] T. Erlebach, H. Kellerer, and U. Pferschy. Approximating multiobjective knapsack problems. *Management Science*, 48(12):1603–1612, 2002.

[35] M. Fischetti and M. Monaci. Light robustness. In *Robust and online large-scale optimization*, pages 61–84. Springer, 2009.

[36] M. Fischetti, D. Salvagnin, and A. Zanette. Fast approaches to improve the robustness of a railway timetable. *Transportation Science*, 43(3):321–335, 2009.

[37] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co Ltd, 1979.

[38] C. E. Gounaris, W. Wiesemann, and C. A. Floudas. The robust capacitated vehicle routing problem under demand uncertainty. *Operations Research*, 61(3):677–693, 2013.

[39] M. Grötschel, L. Lovász, and A. Schrijver. Geometric methods in combinatorial optimization. In *Proc. Silver Jubilee Conf. on Combinatorics*, pages 167–183, 1984.

[40] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1993.

[41] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

[42] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. K-adaptability in two-stage distributionally robust binary programming. *Operations Research Letters*, 44(1):6–11, 2016.

[43] M. Inuiguchi and M. Sakawa. Minimax regret solution to linear programming problems with an interval objective function. *European Journal of Operational Research*, 86(3):526 – 536, 1995.

[44] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Springer, Berlin, 2004.

[45] B. Korte and J. Vygen. *Combinatorial optimization*. Springer, 2002.

[46] P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Springer, 1996.

[47] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[48] C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In *Robust and online large-scale optimization*, pages 1–27. Springer, 2009.

[49] E. Nikolova. Approximation algorithms for reliable stochastic combinatorial optimization. In *Proceedings of the 13th International Workshop on Approximation and the 14th International Workshop on Randomization and Computation*, pages 338–351. Springer, 2010.

[50] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 86–92. IEEE, 2000.

[51] R. T. Rockafellar. *Convex analysis*. Princeton university press, 2015.

[52] A. P. Ruszczynski and A. Shapiro. *Stochastic programming*, volume 10. Elsevier Amsterdam, 2003.

[53] A. Schöbel. Generalized light robustness and the trade-off between robustness and nominal quality. *Mathematical Methods of Operations Research*, pages 1–31, 2014.

[54] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer Science & Business Media, 2003.

[55] M. Sim. *Robust optimization*. PhD thesis, Massachusetts Institute of Technology, 2004.

[56] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

[57] P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications*, volume 18. Siam, 2014.

[58] W. Wiesemann, D. Kuhn, and M. Sim. Distributionally robust convex optimization. *Operations Research*, 62(6):1358–1376, 2014.