INTRODUCTION

Our life is affected by countless complex interdependent processes forming the backbone of economy including production, trading, logistics, distribution, and communication. Oftentimes, thousands, millions or even more decisions must be taken into account to plan and operate these processes.

Mathematical optimization strives for providing theory, models, and methods to tackle these problems and obtain relevant solutions in practice. As a sub-discipline of mathematics, mathematical optimization investigates the (hidden) problem structure to identify and exploit reoccurring (sub)structures and develop tailor-made strategies.

An example of reoccurring structures are networks. A network describes the dependencies between entities, e. g., logistics networks describe the places from which and where to commodities are sent, energy networks characterize the supply of energy, or telecommunication networks specify the possible ways of information exchange. The problem to plan a network, i. e., determine its layout and the rules how it can be used later on, is called the capacitated network design problem (NDP) in mathematical optimization.

Another crucial aspect are scarce resources and the resulting challenging question of prioritizing their usage. For example, the (scarce) loading capacity of a delivery truck must be utilized in the best way in logistics, the limited monetary budget must be managed in finance, or the available bandwidth of an optical fiber must be shared among several optical data signals in telecommunication. Mathematically, this leads to the so-called knapsack problem (KP).

The understanding of problems such as the NDP or the KP allows a more accurate mathematical modeling of the underlying real-world problem. However, the mathematical model is always a simplification; oftentimes a rather rough one. In particular, the temporal dynamics and uncertainties of real-life processes are hard to take into account, e. g., travel times are not constant but subject to delays like traffic jams in practice, the food production depends on future weather conditions, or telecommunication demands fluctuate significantly by the daytime with peaks during certain hours of the day and lows during the night.

Mathematical optimization offers several paradigms to incorporate uncertainty into the mathematical framework. Robust optimization is one of these. Here, the data uncertainty is modeled implicitly by a so-called uncertainty set. The robust optimization problem asks to find an optimal solution that is feasible for any possible data realization in this uncertainty set. In particular, robust linear optimization offers several advantages over other

approaches. The definition of an uncertainty set does not rely on the knowledge of probability distributions and is thus often better suited to applied problems where only a finite discrete set of historical data is available, if any. In addition, robust solutions are feasible for all realizations in the uncertainty set by definition. Further, the complexity of robust linear programs does not increase compared to the original non-robust linear program under mild conditions. Instead, there often exist compact reformulations, i. e., formulations that are at most polynomially larger than the original non-robust formulations. Thus, robust linear optimization problems are more computationally tractable than other mathematical optimization problems applying different paradigms to handle uncertainties.

In this thesis, we consider robust integer linear optimization problems. In particular, we consider four different robustness concepts and the associated uncertainty sets. For each concept, we investigate the corresponding robust KP presenting integer linear programming formulations, results on the polyhedral structure of the solution sets, and algorithms to solve the occurring separation problems or the robust KP themselves. Moreover, we study the corresponding robust NDP problem for two of the concepts, also presenting several integer linear programming formulations, polyhedral insights, and (separation) algorithms to solve the (separation) problems.

Our theoretical investigations are completed by two extensive computational studies: one for the recoverable robust KP, the other for the Γ-robust NDP. The latter uses real-life uncertain data from an application in telecommunication and is based on our work with the German ROBUKOM project in cooperation with Nokia Siemens Networks GmbH & Co. KG.

Contributions. Some results are partially based on joint work as common in the area of applied mathematical optimization. Whenever this is the case, we state explicitly our coauthors and possible prior published publications of our joint work in the beginning of the corresponding chapters.

The main contributions of the thesis are the following.

- The introduction and study of the concept of submodular robustness.
- A detailed investigation of the recoverable robust KP. In particular with a Γ-robust scenario set and the k-removal recovery rule.
- A detailed investigation of the submodular robust KP introducing the classes of submodular robust $(1, k)$-configuration and weight inequalities.
- A study of the structure of covers and their extendability for each considered robust KP.
- A detailed investigation of the Γ-robust NDP including new classes of valid and facet-defining inequalities (e. g., Γ-robust cutset inequalities, Γ-robust envelope

inequalities, Γ-robust arc residual capacity inequalities, and Γ-robust metric inequalities) and algorithms solving the corresponding separation problems as well as the Γ-robust NDP problem itself.

- A first-time investigation of the multi-band robust NDP including mixed integer linear programming formulations, polyhedral studies yielding new classes of valid inequalities (multi-band robust cutset inequalities and multi-band robust metric inequalities), and algorithms to solve the corresponding separation problems. In particular, we point out by examples how results of the Γ-robust NDP can be generalized to the multi-band robust setting.

- Representative extensive computational studies for two recoverable robust knapsack variants and one robust network design problem (the latter with application to telecommunications).

Outline. This thesis is structured into three parts.

In *Part I - Concepts*, we introduce the relevant mathematical methodology, provide a survey on related work, and introduce the objects of research for this thesis. Therefore, we first recap mathematical requirements focusing on mathematical optimization and introducing the classic knapsack and capacitated network design problems in Chapter 1. A brief primer on relevant applications in telecommunications is given at the end of the same chapter. In Chapter 2, we present a detailed survey on literature related to mathematical optimization under data uncertainty and in particular robust optimization. Next in Chapter 3, we introduce the four robustness concepts which are our main focus of investigation in this thesis: Γ-robustness, multi-band robustness, submodular robustness, and recoverable robustness. Moreover, we address the evaluation of robustness discussing several alternative approaches.

In *Part II - Robust Knapsack Problems*, we consider the robust counterpart of the classic knapsack problem for each of the four robustness concepts. For each resulting robust knapsack problem, we present mathematical formulations, study the corresponding polyhedral solution sets identifying strong classes of valid inequalities, and develop algorithms solving the occurring separation problems as well as the robust knapsack problem itself. Following this structure of investigation, we consider the Γ-robust knapsack problem in Chapter 4, the more general multi-band robust knapsack problem in Chapter 5, and the submodular robust knapsack problem in Chapter 6 which generalizes the multi-band robust knapsack problem even further. In Chapter 7, we consider the recoverable robust knapsack problem which is an integrated two-stage problem. Two special cases are of particular interest for us whereof one generalizes the one-stage Γ-robust knapsack problem. We conclude this part of the thesis in Chapter 8 reporting on the results of extensive computational studies we carried out on recoverable robust knapsack problems. Therefore, we focus on the rather general recoverable robust knapsack problem evaluating

the effect of the robustness parameters, the strength of the derived valid inequalities and finally the overall performance in a cut-and-branch approach to solve this problem.

Part III - Robust Network Design Problems is structured similarly to Part II. Here, we consider the robust counterpart of the classic (capacitated) network design problem for selected robustness concepts. We primarily focus on the Γ-robust network design problem and provide several mathematical formulations for this problem, investigate the corresponding polyhedral structure, derive several classes of valid inequalities, and algorithms. Our investigation is described in great detail in Chapter 9. Afterwards, we consider the more general multi-band robust network design problem in Chapter 10 pointing out how results for the Γ-robust network design problem are generalized to the multi-band robust setting. In Chapter 11, we describe the results of computational studies on robust network design problems and in particular the Γ-robust design of telecommunication networks. We experimentally compare its different formulations, the derived classes of valid inequalities, separation algorithms, and algorithms to solve the Γ-robust network design problem itself. For our experiments, we use historical real-life traffic measurements to define the data uncertainty.

Finally, we give concluding remarks to the contributions of this thesis and discuss potential future research directions.

PART ONE

CONCEPTS

CHAPTER ONE

MATHEMATICAL PRELIMINARIES

The first chapter of this thesis gives a brief survey of the mathematical prerequisites and thereby introduces the reader to the used notation. Furthermore, we present two important optimization problems: the knapsack problem and the capacitated network design problem. In later parts of this thesis we will consider variants of both problems where the input data is subject to some random uncertainty. In this chapter we introduce both problems in their classic deterministic settings, reporting on related work, important results, and polyhedral insights to their solution sets. The last section of this chapter is a primer to the telecommunications application area. There, we give a short introduction to the structure and operation of telecommunication networks and the related mathematical optimization challenges.

1.1 Basics

In the following, we introduce our notation while reminding the reader of some basics of (integer) linear optimization and polyhedral combinatorics. We assume that most results are well-known and therefore give only a brief introduction without making the claim to be complete. For further reading, we refer to some well-established monographs below.

An introduction to graphs, networks, flows, and related algorithms is given in the excellent book by Ahuja et al. [10]. The standard book about complexity theory is written by Garey and Johnson [72]. A well-written introduction to linear optimization with various examples is given by Chvatal [53]. A more formal and more recent survey on linear programming can be found in Dantzig [61].

Algorithmic combinatorial optimization is described by Grötschel et al. [76]. The books by Schrijver [143], Nemhauser and Wolsey [126], and Wolsey [159] consider the theory of integer programming, combinatorial optimization and the related polyhedral theory. The three-volume encyclopedic book by Schrijver [144] gives an excellent survey on state-of-the-art combinatorial optimization theory and techniques referencing thousands of original work. A good survey of the algorithmic aspects of solving mixed integer linear programs and experimental results are described in detail by Achterberg [6].

Linear algebra. We denote by $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$, the sets of *integer*, *rational*, and *real numbers*, respectively. The set of positive natural numbers is denoted by $\mathbb{N}$. Let $K \in \{\mathbb{Z}, \mathbb{Q}, \mathbb{R}\}$. Then, $K_{>0}$, $K_{\geq 0}$, $K_{\leq 0}$, and $K_{<0}$ denotes the positive, nonnegative, nonpositive, and negative subset of K, respectively. For two arbitrary sets A and B, let $A \cup B$ denote the *union*, $A \cap B$ the *intersection*, and $A \triangle B := (A \cup B) \setminus (A \cap B)$ the *symmetric difference* of A and B. The *power set* of A, i. e., the set of all subsets of A, is denoted by 2^A. For a function $x : A \to \mathbb{R}$, we define the notation $x(A) := \sum_{a \in A} x(a)$. A function $f : A \to \mathbb{R}$ is called *submodular* if $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ holds for all $S, T \subseteq A$. A function $f : A \times A \to \mathbb{R}_{\geq 0}$ is called *metric* if the following three conditions hold for all $a, b, c \in A$: (i) $f(a, a) = 0$, (ii) $f(a, b) = f(b, a)$, and $f(a, b) \leq f(a, c) + f(c, b)$.

For $x \in \mathbb{R} \setminus \mathbb{Z}$, the largest integer number smaller than x is denoted by $\lfloor x \rfloor$. Analogously, the smallest integer number larger than x is denoted by $\lceil x \rceil$. For $x \in \mathbb{Z}$, we define $\lfloor x \rfloor = \lceil x \rceil = x$. Further, we define $\operatorname{frac}(x) := x - (\lceil x \rceil - 1)$ as the *fractional part* of $x \in \mathbb{R}$. Note, $\operatorname{frac}(x) = 1$ if $x \in \mathbb{Z}$.

Let $v \in \mathbb{R}^n$, $M \in \mathbb{R}^{m \times n}$ be a vector and a matrix, respectively. If not stated differently, all vectors are column vectors and $v^\top$ denotes the transposed vector of v. Further, v_i is the i-th component or entry of v. Analogously, $M_{i\cdot}$ is the i-th row, $M_{\cdot j}$ the j-th column, and M_{ij} the entry in row i and column j of M. Let e_i denote the i-th *unit vector*, i. e., the vector whose i-th entry is 1 and all others are 0. Let $x_1, x_2, \ldots, x_t \in \mathbb{R}^n$. A vector $x \in \mathbb{R}^n$ is a *linear (affine, conic, or convex) combination* of $x_1, \ldots, x_t$ if there exists a $\lambda \in \mathbb{R}^n$ so that $x = \sum_{i=1}^t \lambda_i x_i$ (and $\sum_{i=1}^t \lambda_i = 1$, $\lambda \geq 0$, or $\sum_{i=1}^t \lambda_i = 1$ and $\lambda \geq 0$, respectively). Considering one type of combination, if $\lambda = 0$ is the only solution, we call x *linearly, affinely, conic, or convex independent* of $x_1, \ldots, x_t$, respectively. Let $X \subseteq \mathbb{R}^n$. Then $\operatorname{lin}(X)$, $\operatorname{aff}(X)$, $\operatorname{cone}(X)$, or $\operatorname{conv}(X)$, denotes the *linear, affine, conic, or convex hull*, i. e., the set of all linear, affine, conic, or convex combinations of vectors in X, respectively. The *dimension* $\dim(X)$ is defined as the maximum number of affinely independent vectors in X minus 1.

Let E be a finite set, and $\mathcal{I} \subset 2^E$. Then, the pair $(E, \mathcal{I})$ is called an *independence system* if $\emptyset \in \mathcal{I}$ and $A \in \mathcal{I} \Rightarrow B \in \mathcal{I}$ for all $B \subseteq A \in \mathcal{I}$. The elements of $\mathcal{I}$ are called *independent sets*.

Complexity theory. Next, we provide a rather informal introduction to mathematical optimization and complexity theory. Based on Garey and Johnson [72] we define a *problem* as a question to be answered. Usually the answer depends on some input *parameters*. A *(problem) instance* of a problem is an assignment of values to all its input parameters. If only "yes" and "no" are feasible answers, we call the problem a *decision problem*. An *optimization problem* is a problem whose answer is the minimum or maximum value of a given *objective function*. For each optimization problem there exists a corresponding decision problem asking if the objective function is less than or greater than a given value (depending on if the optimization problem is a minimization or maximization problem).

An *algorithm* is a procedure which answers the problem for all problem instances. It *solves* the problem if the answers given are always feasible to the problem.

In this thesis, we assume all problem instances are coded binary and the model of computation is a deterministic one-tape Turing machine, see Garey and Johnson [72] for details. The *(time) complexity* of an algorithm is the number of elementary operations executed to solve a problem instance. The big-O notation is used to denote the complexity of an algorithm or problem depending on the size n of its input data. Then $\mathcal{O}(f(n))$ means there exist functions $f, g : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ with $\lim_{n\to\infty} \frac{f(n)}{g(n)} > 1$ and a scalar $a \in \mathbb{R}_{\geq 0}$ such that (there exists an algorithm such that) the number of elementary operations to solve the given instance of size n is bounded by the term $a \cdot f(n) + g(n)$. Notice that the scalar a and the dominated term $g(n)$ are usually dropped in the corresponding big-O notation. If $f(n)$ is a polynomial, we say the algorithm or problem has polynomial (time) complexity. In addition to the time complexity, we can also count the elementary read and write operations accessing the information storage. Analogously, this yields the so-called *memory* or *size complexity*.

We define $\mathcal{P}$ as the class of all decision problems for which a polynomial time algorithm exists. The class $\mathcal{NP}$ consists of all decision problems for which a "yes"-instance can be verified in polynomial time by another algorithm. It holds $\mathcal{P} \subseteq \mathcal{NP}$. The class *$\mathcal{NP}$-hard* consists of all problems as hard as the hardest problems in $\mathcal{NP}$. Although the misleading name, a $\mathcal{NP}$-hard problem may not be in $\mathcal{NP}$. If a problem is both in $\mathcal{NP}$ and $\mathcal{NP}$-hard, then it is called *$\mathcal{NP}$-complete*. We define the class *co-$\mathcal{NP}$* as all decision problems for which a "no"-instance can be verified with polynomial complexity. It holds $\mathcal{P} \subseteq$ co-$\mathcal{NP}$. Analogously, we define the complexity classes co-$\mathcal{NP}$-hard and co-$\mathcal{NP}$-complete. An algorithm has *pseudo-polynomial* complexity if its complexity is polynomial w.r.t. numeric value of the input and not its binary encoding. A $\mathcal{NP}$-hard problem with pseudo-polynomial complexity is called *weakly $\mathcal{NP}$-hard*. A problem in $\mathcal{NP}$-hard is called *strongly $\mathcal{NP}$-hard* if it is proven that no pseudo-polynomial algorithm solving this problem exists (unless $\mathcal{P} = \mathcal{NP}$). There exist polynomial time algorithms to $\mathcal{NP}$-hard optimization problems if and only if $\mathcal{P} = \mathcal{NP}$ holds which is one famous open question in complexity theory; cf. Cook [56]. An extensive list of classical combinatorial problems known to be $\mathcal{NP}$-complete is given by Garey and Johnson [72]. The definition of (fully) polynomial approximation schemes can also be found therein.

Graph theory. An (undirected) *graph* is defined by a set of *nodes* V, a set of *edges* $E \subset V \times V$ and an incidence function $\psi : E \to V^2$ relating each edge $e = \{i, j\} \in E$ to its *end nodes* $i, j \in V$. A graph is denoted by $G = (V, E, \psi)$ or $G = (V, E)$ for short (in which case we assume that the omitted incidence function ψ is implicitly defined by the set of edges). Let $G = (V, E)$ be a graph with n nodes, $U \subset V$, and $F \subset E$. Then, $G' = (U, F)$ is the *subgraph* of G with node set U and edge set F. A subgraph $G' = (U, F)$ is called a *tree* if it is connected and $|F| = |U| - 1$; if $U = V$, then G' is called a *spanning tree* of G. The subset of edge $P = (v_1v_2, v_2v_3, \ldots, v_{i-1}v_i) \subset E$ is called a *path* if $v_j \neq v_k$ for all $j, k \in \{1, \ldots, i\}$, $j \neq k$. A *cycle* C is defined as $C := P \cup \{v_iv_1\} \subset E$. A subset $S \subset V$ of the nodes partitions the graph two parts S and $V \setminus S$ and is called *cut*. The subset of edges with one end node in S and the other in $V \setminus S$ is denoted by $\delta(S)$ and called *cutset*.

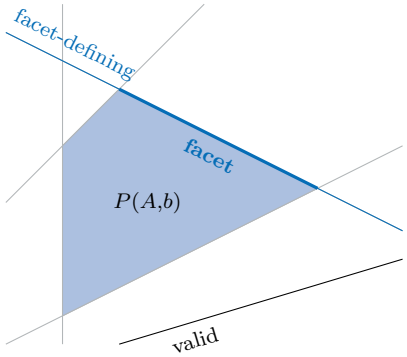


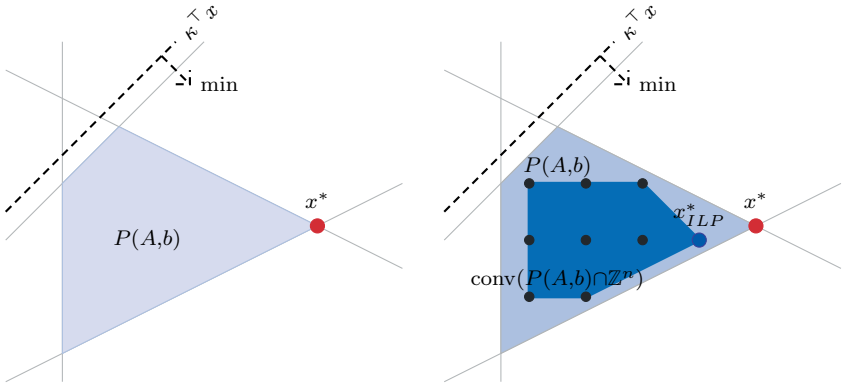
Figure 1.1: Example of a polytope $P(A, b)$ with a valid inequality, a facet-defining inequality, and a facet.

In addition to undirected graphs, there exist *directed graphs* or *digraphs* for short. A digraph D is analogously defined as the triple (V, A, ψ) of a set of nodes V, a set A of directed *arcs*, and an incidence function ψ. An undirected graph $G = (V, E)$ can be *directed* by an orientation $o : E \to V \times V$ assigning each edge $\{i, j\} \in E$ to either the arc (i, j) or the arc (j, i).

Notice, for simplicity, we also write ij for an edge or an arc if it is unambiguous in the current context.

Polyhedral theory. Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $J = \{1, 2, \ldots, m\}$, and $Ax \leq b$ a system of linear inequalities. Then, the set $P(A, b) := \{x \in \mathbb{R}^n \ : \ Ax \leq b\}$ is called a *(convex) polyhedron*. W. l. o. g. we assume $P(a, b)$ to be full-dimensional in the following definitions. If $P(A, b)$ is bounded, it is called a *polytope*. The convex hull of all integer lattice points of a polyhedron, $\text{conv}(P(A, b) \cap \mathbb{Z}^n)$, is called *integer hull* and a polyhedron itself. A vector $x \in P(A, b)$ is called an *extreme point* if it is not a convex combination of any vectors in $P(A, b)$. A set F is called a *face* of $P(A, b)$ if $F := \{x \in P(A, b) \ : \ \exists J' \subset J, A_{J'}.x = b_{J'}\}$ holds. A face $F \notin \{P(A, b), \emptyset\}$ is called *proper*. A proper face F of $P(A, b)$ which is not a subset of another face is called *facet*, i. e., it holds $\dim(F) = \dim(P(A, b)) - 1$. For $v \in \mathbb{R}^n$ and $w \in \mathbb{R}$, we call an inequality $v^\top x \leq w$ *valid* for $P(A, b)$ if $P(A, b) \cap P(v, w) = P(A, b)$ holds. A valid inequality is called *facet-defining* for $P(A, b)$ if there exist a facet F of $P(A, b)$ so that $F \subseteq \left\{x \in \mathbb{R}^n \ : \ v^\top x = w\right\} \neq \emptyset$. Figure 1.1 illustrates a polytope, a valid inequality, a facet-defining inequality, and a facet.

Let $Q(A, b) := \left\{x_1 \in \mathbb{R}^{n_1}, x_2 \in \mathbb{R}^{n_2} \ : \ A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq b\right\}$. Then we define its *projection* onto the space of x_1 by $\text{proj}_{x_1} Q(A, b) := \left\{x_1 \in \mathbb{R}^{n_1} \ : \ \exists x_2 \in \mathbb{R}^{n_2} \text{ so that } A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq b\right\}$. Let $a_1 \in \mathbb{R}^{n_1}$, $b \in \mathbb{R}$, and ${a_1}^\top x_1 \leq b$ for $x_1 \in \mathbb{R}^{n_1}$. Then, we call $\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^\top \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq b$ for $x_2 \in \mathbb{R}^{n_2}$ the *lifted inequality* to the space of the x_1- and x_2-variables. For $x \geq 0$, an inequality $v^\top x \leq w$ *dominates* another inequality ${v'}^\top x \leq w^\top$ if there exists a $\lambda \in \mathbb{R}_{>0}$ so that $\lambda v' \leq v$ and $\lambda w' \geq w$ holds. The case $x \leq 0$ is analog.



(a) LP (b) LP with integer hull as ILP

Figure 1.2: Example of an LP and the ILP defined by the integer hull $\text{conv}(P(A,b)\cap\mathbb{Z}^n)$. The optimal LP solution x^* and integer solution x^*_{ILP} are shown.

Linear programming. Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $\kappa \in \mathbb{R}^n$. Then, we call the optimization problem to maximize a linear function over the polyhedron $P(A,b)$ a *linear programming (LP) problem* (in standard form). It can be written as

$$\max \kappa^\top x \tag{1.1}$$
$$\text{s.t.} Ax \leq b. \tag{1.2}$$

A vector $x \in \mathbb{R}^n$ satisfying the conditions of LP (1.1) is called *feasible*; a feasible vector x^* minimizing the objective value $\kappa^\top x^*$ is called *optimal*. If there exists an optimal solution, then there exists an optimal solution which is an extreme point of $P(A,b)$. In Figure 1.2(a), an LP and its optimal solution x^* is visualized.

Given an LP in standard form (1.1), we call the associated LP

$$\min b^\top y \tag{1.3}$$
$$\text{s.t.} A^\top y = \kappa \tag{1.4}$$
$$y \geq 0 \tag{1.5}$$

its *dual LP*. The original LP (1.1) is called the *primal LP*. Note, the dual LP of the dual LP (1.3) is again the primal LP (1.1).

Theorem 1.1 (Duality of linear programming)**.** *Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $\kappa \in \mathbb{R}^n$. If there exist feasible solutions $\tilde{x}$ and $\tilde{y}$ of the primal LP $\max\{\kappa^\top x : Ax \leq b\}$ and the dual LP $\min\{b^\top y : A^\top y = \kappa, y \geq 0\}$, respectively, then there exist finite optimal solutions x^* and y^* so that $\kappa^\top x^* = b^\top y^*$ holds.*

Notice that the standard form of an LP is no restriction of generality: there exist transformations between maximization and minimization problems, equality and inequality constraints, and unbounded and bounded/nonnegative variables.

In 1951, Dantzig [59] developed the *simplex algorithm*, an iterative algorithm which starts with a feasible solution (corresponding to an extreme point of the polytope) and improves to another feasible solution (also corresponding to an extreme point) which yields a better objective value until an optimal solution is reached. Unfortunately, the simplex algorithm has exponential worst-case complexity since exponentially-many extreme points have to be evaluated in the worst-case (using known pivot rules); cf. Klee and Minty [95]. However, later it has been proven that its complexity is polynomial on average; cf. Borgwardt [43] and Spielman and Teng [148]. Another approach to is the ellipsoid method which has polynomial complexity but is useless in practice due to numerical instabilities; cf. Khachiyan [93] and Grötschel et al. [75]. A different polynomial algorithm to solve an LP has been introduced by Karmarkar [90] and is called *interior point method* or *barrier algorithm*.

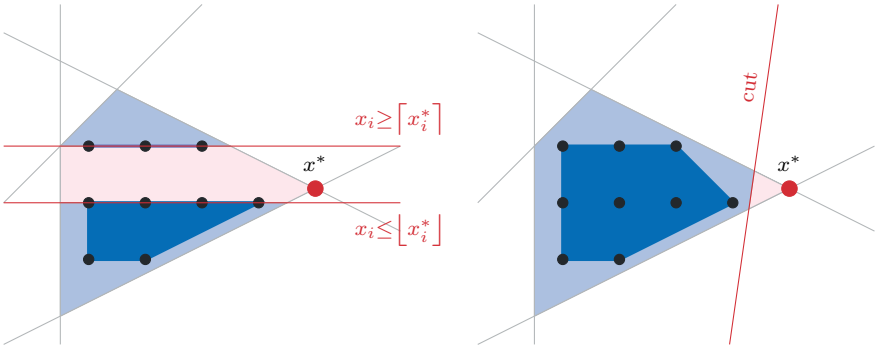
(Mixed) integer linear programming. Given an LP in standard form (1.1) and with rational input data. If we restrict its feasible solutions to integer vectors only, we obtain an *integer linear programming (ILP) problem*. It reads

$$\max \kappa^\top x \tag{1.6}$$
$$\text{s.t.} Ax \leq b. \tag{1.7}$$
$$x \in \mathbb{Z}^n. \tag{1.8}$$

By relaxing the integrality constraint $x \in \mathbb{Z}^n$, we obtain the *linear (programming) relaxation* of ILP (1.6). In Figure 1.2(b), an LP and its integer hull are visualized. In addition, the optimal LP solution x^* and integer solution x^*_{ILP} are shown. Note, if only a subset of the variables is restricted to integrality, a *mixed integer linear program* (MILP) is obtained. The following algorithmic approaches to ILPs are w. l. o. g. also applicable to MILPs. In contrast to LPs, solving ILPs is known to be strongly $\mathcal{NP}$-hard; see Kannan and Monma [88].

In the 1960s, Land and Doig [108] and Dakin [57] introduced the *branch-and-bound* algorithm to solve ILPs. It is an (implicitly) enumerative algorithm following the divide-and-conquer principle used in computer science.

In this algorithm, only the LP relaxation of an ILP is solved, e. g., by using the simplex algorithm. If the LP relaxation is unbounded or has no solution, the ILP is as well or has not one either, respectively. If the LP relaxation has an integer optimal solution, then this solution is also optimal for the original ILP. If the optimal solution vector x^* of the LP relaxation has a fractional valued entry x^*_i, it is not feasible for the ILP. This fractionality is removed by splitting the LP relaxation into two new subproblems where the constraint $x_i \leq \lfloor x^*_i \rfloor$ is added to one of them and the constraint $x_i \geq \lceil x^*_i \rceil$ to the other. This split step is called *branching*. Notice, the union of the set of feasible solutions of both newly created subproblems contains all feasible integer points of the solution set of the original ILP. Figure 1.3(a) illustrates the branching geometrically. After branching, each subproblem is solved individually. If the solution vector of a subproblem has again fractional entries, the algorithm is recursively repeated for this



(a) branching (b) cutting plane

Figure 1.3: Example of branching and cutting in a branch-and-bound or branch-and-cut algorithm, respectively.

subproblem. By succeeding branchings, the so-called (binary) *branch-and-bound tree* is built. Note, the LP relaxation of the original problem is located at the root node of this tree. Its leaf nodes correspond to subproblems which are either infeasible (i.e. have no solution) or have an integer optimal solution.

Clearly, the branch-and-bound tree grows exponentially. Hence, the explicit enumeration of all tree nodes should be avoided. A way to do so is motivated by the following observations: on the one hand, every integer feasible solution found at any node of the branch-and-bound tree is feasible to the original ILP. Thus, its objective value yields an global lower bound (for a maximization problem) on the actual possible currently unknown objective value of the ILP. On the other hand, at each node, the objective value of an optimal solution of the LP relaxation is a local upper bound on the objective value of an optimal solution of the corresponding ILP at the same node. Hence, whenever the local upper bound is lower than the currently best known global upper bound while running the branch-and-bound algorithm, the node corresponding to the local upper bound cannot yield any better integer solution than the one corresponding to the current global upper bound. Thus, this node and its subproblem can be ignore. No branching takes places. It can be removed from the branch-and-bound tree; the node and its potential subtree are called *fathomed*. This overall principle is known as *bounding*.

Another approach to reduce the number of actual solved branch-and-bound nodes is to tighten the LP relaxations by adding additional inequalities which are valid for the ILP but violated for the actual fractional solution of the LP relaxation. Geometrically, these inequalities cut-off some region of the polyhedron associated with the LP relaxation including the fractional LP solution (but no integer feasible solution of the ILP). Hence, these inequalities are called *cutting planes* or *cuts* for short. Figure 1.3(b) shows an example of a cut. The problem to determine a cut given a fractional LP solution or proof that none exist, is called *separation problem*, the procedure itself we call *separation*. In 1981, Grötschel et al. [75] have shown that optimization and separation are polynomially equivalent, i. e., an optimization problem can be solved efficiently if and only if the

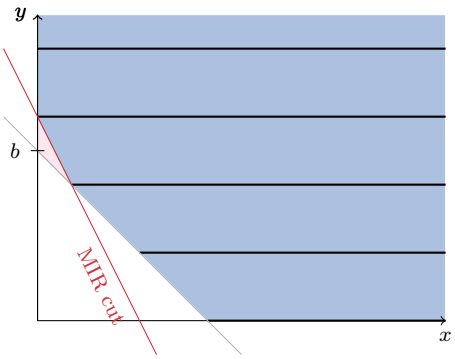


Figure 1.4: Example of a MIR cut in two dimensions.

separation problem to remove infeasible solutions can be solved efficiently. Numerous classes of cutting planes are known. Some of then are general cutting planes exploiting the structure of the coefficient matrix A, the right hand side vector b, or the variable ranges. Others are problem-specific cuts which are only valid for the specific polyhedral structure of selected combinatorial problems. Chvátal-Gomory cuts [73, 74] or mixed integer rounding (MIR) cuts [126] (for MILPs) are examples for the first type of cuts, cover inequalities to the knapsack problem for the latter [18, 80, 158].

We will use MIR to derive valid inequalities for several polyhedra in this thesis. Let us now consider the two-dimensional MIR inequalities as an example.

Lemma 1.2 (Wolsey [159]). *Let* $Q = \{(x, y) \in \mathbb{R} \times \mathbb{Z} \ : \ x + y \geq b, \ x \geq 0\}$. *Then, the* mixed integer rounding inequality

$$x + ry \geq r \lceil b \rceil \tag{1.9}$$

with $r := b - \lfloor b \rfloor$ *is valid for* Q.

In Figure 1.1 a mixed integer set and MIR cut is shown.

By integrating the separation of cutting planes into the branch-and-bound algorithm, we obtain the so-called *branch-and-cut* algorithm. Here, cuts are separated, added to the LP relaxation, and the extended LP is resolved. A problem is only branched into two subproblems if no violated cut has been found or another technically motivated abort criterion for the implemented separation algorithm is met. If the cutting planes are only applied at the root node, the resulting algorithm is called *cut-and-branch*. Achterberg [6] presents an excellent survey on the state-of-the-art algorithms and implementations of branch-and-cut algorithms.

1.2 The knapsack problem

One of the most fundamental problems in mathematical optimization is the well-known knapsack problem. In its general form the (binary or 0-1) knapsack problem asks to select

a subset of valuable items such that their total value is maximized while they have to "fit into" the knapsack, i. e., their total weight must not exceed the given capacity of the knapsack. Although its apparent clearness, the knapsack problem turns out to be a hard problem, in fact weakly co-$\mathcal{NP}$-hard. Nevertheless, it occurs in the mathematical models of many applications. Oftentimes, it is a subproblem or relaxation of more complex real-world problems, e. g., (i) in telecommunications traffic has to be routed within the capacity of cables or bandwidth restrictions of base station antennas, (ii) in logistics the capacity restrictions of trucks, planes, ships have to be met, and (iii) in finance the costs of taken decisions must be within a given budget.

Furthermore, each individual constraint of a general 0-1 integer linear program (0-1 ILP) can be considered as knapsack constraint. Therefore inequalities for the knapsack polytope can be used as general cutting planes to 0-1 ILPs. In fact, many results from the 1950/60s on the polyhedral structure of the knapsack polytope were obtained when considering individual rows of 0-1 ILPs; cf. Martello and Toth [119].

The knapsack problem, its variants, and extensions have been studied for several decades. For example, Karp [91] has investigated the complexity of the knapsack problem showing its $\mathcal{NP}$-hardness. Kolesar [98] and Horowitz and Sahni [82] have considered branch-and-bound approaches to solve the knapsack problem exactly. At the same time, a polynomial time approximation scheme has been presented by Johnson [87]. One year later, a fully polynomial time approximation scheme has been published by Ibarra and Kim [83]. Whereas Salkin and De Kluyver [142] have studied the relation between ILPs and knapsack problems. Dudzinski and Walukiewicz [62] have studied LP and Lagrangian relaxations of the problem obtaining lower/dual bounds. In 1979 Martello and Toth [115] have presented an exact exponential algorithm to solve the binary knapsack problem. An algorithmic survey including dynamic programming approaches is given in Martello and Toth [118] and the books by Martello and Toth [119] and Kellerer et al. [92].

Besides its simplest form, the binary knapsack problem, many variants and extensions of the knapsack problem exist. A very famous one is the *subset-sum problem* where the item values are identical to the item weights, see Karp [91], Martello and Toth [119]. It has applications in complexity theory, cryptography and computer science. The *(un)bounded knapsack problem* allows items to be selected more than once up to an (optional) upper bound, see Martello and Toth [119]. The *multiple knapsack problem* and *multiple-choice knapsack problem* group the items such that at most one item per group may be selected, see Martello and Toth [117], Sinha and Zoltners [146]. In *multi-dimensional knapsack problems*, the selected items have to "fit" into several knapsacks at the same time while the weight of an item may differ between these knapsacks, see Weingartner and Ness [156]. *Multi-objective knapsack problems* and *min-max knapsack problems* attach several values to an item and consider a multi-objective approach to determine the combined total value of items, see Ehrgott [67]. Furthermore, variants exists where items have to be selected with minimum value and a total weight above a certain threshold (*minimum knapsack problem*), or where the total weight of the items to be selected is given (*equality knapsack problem*; if additionally all item values are the same: *change-making problem*, see Martello and Toth [116]). Sometimes certain items have to be selected before other items, leading to *precedence constraint knapsack problems*, see Boyd [45]. In addition,