N.N. Pisaruk

Mixed Integer Programming: Models and Methods

May 4, 2019

+++++++

Preface

Initially, I wrote a short manual for users of the optimization library MIPCL (Mixed Integer Programming Class Library). Later I decided to rework the manual to make it useful for a wider audience. As a result, this book was written, in which almost all the main theoretical results, already implemented in modern commercial mixed-integer programming (**MIP**) software, are presented. All algorithms and techniques described here (and many others) are implemented in **MIPCL**. Having the experience of developing such a complex software product as **MIPCL**, I dared to teach others of using **MIP** in practice.

It is clear that in a book of this volume it is impossible to cover the diversity of researches in **MIP**. In particular, specialized algorithms for solving numerous special cases of mixed integer programs (MIPs). are not considered here since the number of publications on this topic is enormous. Therefore, when selecting materials, I decided to discuss only those results that are already implemented in modern computer **MIP** libraries or those that potentially can be included in these libraries in the near future.

The strength of **MIP** as a modeling tool for practical problems was recognized immediately with its appearance in the 50th and 60th years of the 20th century. Unfortunately, for a long time the computers and software available could not solve those models. As a result, the illusion melted. Even today, many potential users still believe that **MIP** is just a tool for writing models, but with very limited capacity for solving those models. In fact, the situation has changed dramatically over the past twenty years. Today, we can solve many difficult practical MIPs using standard software.

What is the reason for the wide use of **MIP** in practice? A brief answer is that, using *binary* variables that take only two values, 0 or 1, we can model many types of nonlinearities, in particular, almost any logical conditions. And the latter are present in almost every non-trivial practical application. It is also very important that **MIP** models are easily expandable. When developing a decision-making system, be careful of using highly specialized models and software. Even if you do not encounter problems at the development stage, they can appear later, during system operation, when the requirements to it change, and their accounting is impossible in the currently used model.

For many years, the basic approach to solving MIPs has remained unchanged: it was a linear-programming-based branch-and-bound method proposed by Land and Doig back in 1960. And this was despite the fact that at the same time there was significant progress in the theory of linear programming and related areas of combinatorial optimization. Many of the ideas developed there ”passed” through intensive computational experiments, but until recently only a few of them were implemented in commercial software products used by practitioners. Nowadays, the best **MIP** programs include many ideas that accumulate theoretical achievements, for example, the modern **MIP** solvers preprocess and automatically reformulate the problems being solved, generate cuts of various types, use a variety of heuristics in the nodes of the search tree to build feasible solutions. This allowed R.E. Bixby to state that the gap between theory and practice was being closed.

Next, we briefly present the contents of this book. In the introduction, we discuss the specific features of the MIPs that distinguish them from other mathematical programming problems. Next, we give examples of the formulations of various types of nonlinearities in **MIP** models. Then we try to understand why one **MIP** formulation is stronger (better) than another one, and also discuss some ways of strengthening existing formulations. Understanding that not all **MIP** formulations are equally good in practice has come relatively recently. Prior to this, as a rule, preference was given to more compact formulations.

In typical situations, to use **MIP** in practice, one does not need to be an expert in theory. Some skills are only needed to formulate practical problems as MIPs. This can be learned by studying applications and their formulations that have already become classical. Chapter 2 presents a number of such applications and their formulations. Even more applications are considered in the other chapters as examples for demonstrating some of the techniques used in **MIP**. Descriptions of many applications are also found in the exercises that are given after each chapter. Many exercises are specially formulated by asking to justify the validity of the proposed answer. This makes the exercises an additional source of information on the topic under discussion.

Because of its universality, the general MIP is a very difficult computational problem. Many of the most complex problems of combinatorial optimization are very simply formulated as MIPs. A number of results from computational complexity theory indicate that efficient algorithms are unlikely to be proposed for solving such problems. We cannot also expect that in the foreseeable future a computer program will be developed that will be able to solve with equal efficiency all MIPs that arise in practice. Therefore, modern **MIP** libraries are designed so that they allow the users to redefine (reprogram) many of their functions, replacing them with those that take into account specific features of the problem being solved. One cannot effectively use these libraries without knowing the theory on which they are based. The rest of the book is devoted to the study of the algorithms implemented in the modern **MIP** libraries.

Since **MIP** is based on *linear programming* (**LP**), a brief introduction to **LP** is given in Chapter 3. Here we do not intend to discuss all aspects of the theory and practice of **LP**. Our goal is to provide enough information to understand the applications of **LP** methods in **MIP**.

Currently, the main method for solving MIPs is the *branch-and-cut* method, since it is used in all (in all!) modern competitive **MIP** programs. Briefly, the branch-and-cut method is a combination of the branch-and-bound and cutting plane methods. In chapters 4–6 we study both components of the branch-and-cut method, as well as its other important features.

Another efficient methodology widely used for solving specific MIPs is the *branch-and-price method*, which can be viewed as a combination of the branch-and-bound and column generation methods. The advantages and potentialities of the branch-and-price method, as well as the difficulties of its implementation, are discussed in Chapter 7. It is also important that the branch-and-price method perfectly complements the branch-and-cut method: usually the branch-and-price method is used in cases where the branch-and-cut method is not very efficient.

The final chapter discusses the relatively new **MIP** applications for solving optimization problems with uncertain parameters. Such problems often arise in economic applications (e.g., in models of long-term planning), when the decision must be taken today, and its efficiency can be judged only after the completion of the planning horizon. Although the problems of this sort have been studied for a long time, but only the emergence of powerful modern computers has made it possible to apply the results of these studies in practice The material of this chapter should in no case be regarded as a brief introduction to stochastic programming or robust optimization. Here, we consider only those methods that reduce the problem with uncertain parameters to its deterministic equivalent (or counterpart) that is a MIP.

In the comments to each chapter, only some publications on the topic are selectively cited: several important original sources and also several recent reviews and articles that can be used for more in-depth study.

This book will be useful as a primary or auxiliary source in the study of such disciplines as integer programming, operation research, mathematical programming, discrete optimization, decision theory, operations management.

To understand the presented material in full, it is assumed that the reader is familiar with the foundations of linear algebra and linear programming, as well as the basic concepts of convex analysis, graph theory, probability theory, and computational complexity.

Minsk, May 2018 *Nicolai N. Pisaruk*

Contents

Abbreviations and Notations

$\mathbb{R}$: field of real numbers
$\mathbb{R}_+$, $\mathbb{R}_{++}$: sets of non-negative and positive real numbers
$\mathbb{Z}$: ring of integers
$\mathbb{Z}_+$, $\mathbb{Z}_{++}$: sets of non-negative and positive integers
$X \pm Y$: set $\{x \pm y : x \in X,\ y \in Y\}$
$X \times Y$: Cartesian product, $\{(x,y) : x \in X,\ y \in Y\}$, of sets X and Y
$\prod_{i=1}^n X_i$: set $X_1 \times X_2 \times \cdots \times X_n \stackrel{\text{def}}{=} \{(x_1,x_2\dots,x_n) : x_i \in X_i,\ i = 1,\dots,n\}$
X^n: set $\{(x_1,x_2,\dots,x_n) : x_i \in X,\ i = 1,\dots,n\}$
X^A: set $\{(x_{a_1},x_{a_2},\dots,x_{a_n}) : x_{a_i} \in X,\ i = 1,\dots,n\}$ for $A = \{a_1,a_2,\dots,a_n\}$
$\text{conv}(X)$: convex hull of vectors from $X \subseteq \mathbb{R}^n$
$\text{cone}(X)$: convex cone generated by set of vectors $X \subseteq \mathbb{R}^n$
I: identity matrix of required size
$\mathbf{0}$: zero vector or matrix of required size
$\mathbf{e}_i$: i-th unit vector, $\mathbf{e}_i \stackrel{\text{def}}{=} (0,\dots,0,1,0,\dots,1)^T$ with 1 in the i-th position
$\mathbf{e}$: vector $(1,1\dots,1)^T$ of required size
A^T: transposed matrix for matrix A
A^{-1}: inverse matrix for non-degenerate matrix A
$\det(A)$: determinant of square matrix A
A_I^J: submatrix of matrix A formed by elements that are in rows from set I and columns from set J
A_I: submatrix of matrix A formed by elements that are in rows from set I
A^J: submatrix of matrix A formed by elements that are in columns from set J
$\text{rank}(A)$: rank of matrix A
$\|x\| \stackrel{\text{def}}{=} \sqrt{x^T x}$: (Euclidean) norm of vector $x \in \mathbb{R}^n$
χ^X: characteristic function (vector) of subset X of finite set S: $\chi^X(i) = 1$ ($\chi_i^X = 1$) if $i \in X$, and $\chi^X(i) = 0$ ($\chi_i^X = 0$) if $i \in S \setminus X$
$H(a,b)$: hyperplane $\{x \in \mathbb{R}^n : ax = b\}$
$H_{\leq}(a,b)$, $H_{\geq}(a,b)$: half-spaces $\{x \in \mathbb{R}^n : ax \leq b\}$ and $\{x \in \mathbb{R}^n : ax \geq b\}$
$P(A,b)$: polyhedron $\{x \in \mathbb{R}^n : Ax \leq b\}$
$P(A,b;S)$: mixed-integer set $\{x \in \mathbb{R}^n : Ax \leq b,\ x_j \in \mathbb{Z} \text{ for } j \in S\}$

$G = (V, E)$: graph or directed graph (digraph) with vertex set V, and edge (arc) set E

$E(S, T)$: set of edges in graph $G = (V, E)$ with one end in set S, and other one in set T; or set of arcs in digraph $G = (V, E)$ outgoing set S and incoming set T

$f(n) = O(g(n))$ if there exists constant $c > 0$ such that $f(n) \le cg(n)$ for sufficiently large $n \in \mathbb{Z}_+$ (for example, $5n^2 + 7n + 100 = O(n^2)$)

$(\Omega, \mathscr{A}, \mathbb{P})$: probability space, where

- Ω: space of elementary events or sample space,
- $\mathscr{A}$: algebra or σ-algebra of subsets of Ω (elements of $\mathscr{A}$ are called event),
- $\mathbb{P}$: probability measure on $\mathscr{A}$ ($\mathbb{P}(S)$ is probability that randomly chosen $\omega \in \Omega$ belongs to $S \in \mathscr{A}$)

$E(\xi)$: (mathematical) expectation (expected value) of random variable $\xi : \Omega \to \mathbb{R}$, $E(\xi) \overset{\text{def}}{=} \int_\Omega \xi(\omega)\mathbb{P}(d\omega)$

IP: Integer Programming

IP: Integer Program (**IP** problem)

LP: Linear Programming

LP: Linear Program (**LP** problem)

MIP: Mixed Integer Programming

MIP: Mixed Integer Program (**MIP** problem)

NP: class of decision problems (with two answers: ”yes” or ”no”), that can be solved by nondeterministic Turing machine in polynomial time

P: class of decision problems (with two answers: ”yes” or ”no”), that can be solved by deterministic Turing machine in polynomial time

Chapter 1
Introduction

The *mixed integer program* (*MIP*) is the following optimization problem:

$$\max\{c^T x : b^1 \le Ax \le b^2,\ d^1 \le x \le d^2,\ x_j \in \mathbb{Z} \text{ for } j \in S\}, \tag{1.1}$$

where $b^1, b^2 \in \mathbb{R}^m$, $c, d^1, d^2 \in \mathbb{R}^n$, A is a real $m \times n$-matrix, x is an n-vector of variables (unknowns), and $S \subseteq \{1, \dots, n\}$ is the set of integer variables. In the *integer program* (*IP*) all variables are integer ($|S| = n$).

As compared to the *linear program* (*LP*), in which $S = \emptyset$, the MIP has variables taking values from a discrete set. This difference makes the MIP significantly more difficult from the algorithmic point of view. We can say that the MIP is one of the most difficult problems of mathematical programming. And this is not surprising, since many combinatorial optimization problems, including those considered to be the most difficult, are very simply formulated as specific MIPs. One of the most common applications of *mixed integer programming* (**MIP**) in everyday life is the efficient use of limited resources.

1.1 Integerality and Nonlinearity

We will see later many times that the condition "x is integer" can be used to express many nonlinear constraints. But first we note that this restriction itself can be given by means of a single smooth equation:

$$\sin(\pi x) = 0.$$

Another important condition "x is binary" (x can take only one of two values: 0 or 1) is written as one quadratic equation

$$x^2 - x = 0.$$

This representation of binary variables allows us to formulate many combinatorial optimization problems as *quadratic programming* problems that involve quadratic terms in their objective functions and constraints. For example, the **NP**-hard *set partition problem* (see also Sect. 2.1)

$$\max\{c^T x : Ax = \mathbf{e},\ x \in \{0,1\}^n\},$$

where $c \in \mathbb{R}^n$, and A is an $m \times n$-matrix with 0 or 1 elements, is rewritten as the following quadratic programming problem:

$$\begin{aligned} &c^T x \to \max, \\ &Ax = \mathbf{e}, \\ &x_i^2 = x_i\,, \quad i = 1, \dots, n. \end{aligned}$$

Here and below $\mathbf{e}$ denotes a vector of suitable size all components of which are equal to 1.

Suppose now that an integer variable x is non-negative and upper bounded, that is, $0 \le x \le d$, where d is a positive integer. In the binary system, d can be written as a $k = \lfloor \log d \rfloor + 1$ digit number. Therefore, introducing k new continuous variables, $s_0, \dots, s_{k-1}$, we can represent the condition $x \in \{0, 1, \dots, d\}$ by the following system of equations:

$$\begin{aligned} x &= \sum_{i=0}^{k-1} 2^i s_i\,, \\ s_i^2 &= s_i, \quad i = 0, \dots, k-1. \end{aligned}$$

So, we can conclude that any MIP is reduced to a quadratic programming problem and, consequently, the general MIP is not more difficult than the general quadratic programming problem. But a distinctive feature of *integer programming* (**IP**) is that here the integer-valued variables are handled in a very special way at the algorithmic level by branching on integer variables and by generating cuts.

From a practical point of view, it is more important that, introducing additional integer (most often binary) variables, we can model many nonlinearities by linear constraints.

1.1.1 Discrete Variables

A *discrete variable* x can take only a finite number of values $v_1, \dots, v_k$. For example, in the problem of designing a communication network, the capacity of a link can be, say, 1, 2 or 4 gigabytes. This discrete variable x can be represented as an ordinary continuous variable by introducing k binary variables $y_1, \dots, y_k$ and writing down the constraints

$$x - v_1 y_1 - v_2 y_2 - \ldots - v_k y_k = 0, \quad (1.2a)$$
$$y_1 + y_2 + \ldots + y_k = 1, \quad (1.2b)$$
$$y_i \in \mathbb{Z}_+, \quad i = 1, \ldots, k. \quad (1.2c)$$

Let us also note that, instead of declaring that all variables y_i are integer, it is enough to specify that (1.2b) is a *generalized upper bound*, i.e., in such a constraint only one variable can take a non-zero value. The generalized upper bounds in the **MIP** software manuals are often referred to as *special ordered sets of type* 1 (SOS1), and their accounting is carried out by performing a special type of branching (see Sect. 6.3.2).

1.1.2 Fixed and Variable Costs

One of the most significant limitations of linear programming with respect to solving economic problems is that in linear models one cannot take into account fixed costs. In **MIP**, accounting for fixed costs is simple.

Let us assume that the *cost* of producing x units of some product is calculated as follows:

$$c(x) \overset{\text{def}}{=} \begin{cases} f + px & \text{if } 0 < l \le x \le u, \\ 0 & \text{if } x = 0, \end{cases}$$

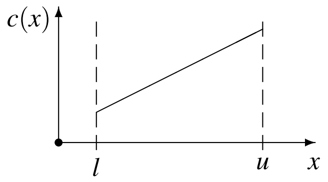

Fig. 1.1 Cost function

where f is a fixed production cost, p is a cost of producing one product unit, l and u are minimum and maximum production capacities.

Introducing a new binary variable y ($y = 1$ if product is produced, and $y = 0$ otherwise) and adding the *variable lower and upper bounds* $ly \le x \le uy$, we transform the nonlinear $c(x)$ into a linear function, $\bar{c}(x,y) = ax + by$, of two variables x and y.

1.1.3 Approximation of Nonlinear Functions

Let a nonlinear function $y = f(x)$ be given on an interval $[a,b]$, and let us choose a partition of this interval:

$$a = \bar{x}_1 < \bar{x}_2 < \cdots < \bar{x}_r = b$$

Connecting the neighboring break-points $(\bar{x}_k, \bar{y}_k = f(\bar{x}_k))$ and $(\bar{x}_{k+1}, \bar{y}_{k+1} = f(\bar{x}_{k+1}))$ by the line segments, we obtain a piecewise-linear approximation, $\tilde{f}(x)$, of the function $f(x)$ (Fig. 1.2). Now we can write down the following system to describe the set of points (x,y) lying on the graph of $\tilde{f}$:

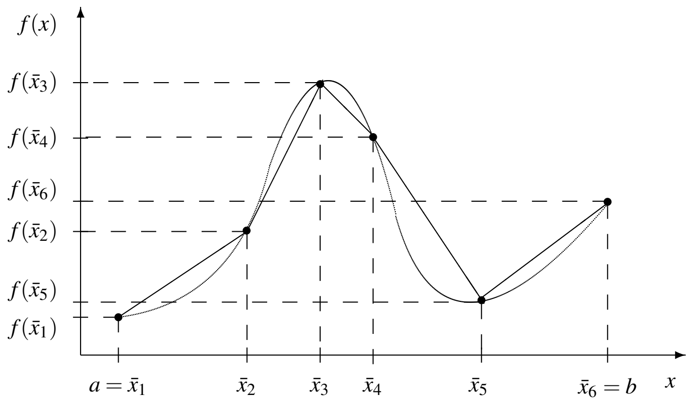

Fig. 1.2 Piecewise-linear approximation of a nonlinear function

$$x = \sum_{k=1}^{r} \lambda_k \bar{x}_k, \tag{1.3a}$$

$$y = \sum_{k=1}^{r} \lambda_k \bar{y}_k, \tag{1.3b}$$

$$\sum_{k=1}^{r} \lambda_k = 1, \tag{1.3c}$$

$$\lambda_k \le \delta_k, \quad k = 1, \dots, r, \tag{1.3d}$$

$$\delta_i + \delta_j \le 1, \quad j = 3, \dots, r, \ i = 1, \dots, j-2, \tag{1.3e}$$

$$\lambda_k \ge 0, \ \delta_k \in \{0,1\}, \quad k = 1, \dots, r. \tag{1.3f}$$

Equations (1.3a)–(1.3c) ensure that the point (x,y) belongs to the convex hull of the points $(\bar{x}_1,\bar{y}_1),\dots,\bar{x}_1,\bar{y}_1)$. The other relations, (1.3d)–(1.3f), require that no more than two variables λ_k take nonzero values, and the indices of these non-zero variables be consecutive numbers. These conditions reflect the requirement that the point (x,y) must lie on some line segment connecting two neighboring break-points.

It should be noted that almost all modern commercial **MIP** solvers take into account Ineqs. (1.3d) and (1.3e) algorithmically, organizing branching in a special way (see Sect. 6.3.2). In this case, it is not necessary to explicitly specify these inequalities, it suffices to indicate that Eq. (1.3c) is of type SOS2 (Special Ordered Set of Type 2).

1.1.4 Approximation of Convex Functions

If $f(x)$ is a convex function, then in many cases we can represent the relation $y = f(x)$ without introducing integer variables. As before, given a partition $a = \bar{x}_1 < \bar{x}_2 < \cdots < \bar{x}_r = b$ of the interval $[a,b]$, we need to approximate $f(x)$ with a piecewise linear function $\tilde{f}$ (see Fig. 1.3).

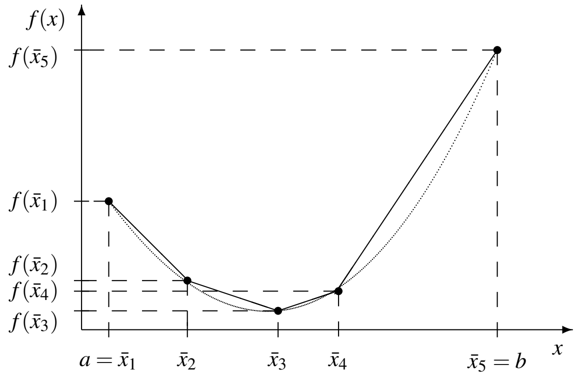

Fig. 1.3 Approximation of a convex function

Let us define the numbers

$$d_k = \bar{x}_{k+1} - \bar{x}_k, \quad q_k = \frac{f(\bar{x}_{k+1}) - f(\bar{x}_k)}{d_k}, \quad k = 1,\dots,r-1.$$

As f is convex, we have $q_1 \le q_2 \le \cdots \le q_{r-1}$. Introducing auxiliary real variables x_k $(k = 1,\dots,r-1)$, we can write down the following representation for the relation $y = \tilde{f}(x)$:

$$\begin{aligned} &x = \sum_{k=1}^{r-1} x_k, \\ &y = f(a) + \sum_{k=1}^{r-1} q_k x_k, \\ &0 \le x_k \le d_k, \quad k = 1,\dots,r-1. \end{aligned} \tag{1.4}$$

It is not difficult to justify the following statement.

Proposition 1.1. *If in a MIP containing System* (1.4) *all coefficients of the variable y are positive in all constraints with the sign "$\le$", and negative in all constraints with the sign "$\ge$" and in the objective function* (*assuming that the objective function is maximized*)*, then* (1.4) *is sufficient to represent the relation* $y = \tilde{f}(x)$.

1.1.5 Logical Conditions

Formally, we write down logical conditions using boolean variables and formulas. Any *boolean variable* can take only two values: **true** and **false**. From boolean variables, using binary logical operations $\vee$ (*or*), $\wedge$ (*and*), and a unary operation $\neg$ ($\neg x$ means *not* x), we can make up *boolean formulas* in much the same way that we can make up algebraic expressions using arithmetic operations over real variables. For example,

$$(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \tag{1.5}$$

is a boolean formula. Substituting values for boolean variables, we can calculate the value of the boolean formula using the rules presented in Table 1.1.

Table 1.1 Logical operations $\neg$, $\wedge$ and $\vee$

x	$\neg x$
false	**true**
true	**false**

x_1	x_2	$x_1 \wedge x_2$	$x_1 \vee x_2$
false	**false**	**false**	**false**
true	**false**	**false**	**true**
false	**true**	**false**	**true**
true	**true**	**true**	**true**

For example, for a truth set $(x_1, x_2, x_3) =$ (**true**,**false**,**false**), (1.5) takes the value of **false**.

Any boolean formula of n boolean variables can be represented in a *conjunctive normal form* (CNF):

$$\bigwedge_{i=1}^{m} \left(\bigvee_{j \in S_i} x_j^{\sigma_j^i} \right), \tag{1.6}$$

where $S_i \subseteq \{1,\dots,n\}$ ($i = 1,\dots,m$) and all $\sigma_j^i \in \{0,1\}$. Here we use the following notation: $x^1 \overset{\text{def}}{=} x$ and $x^0 \overset{\text{def}}{=} \neg x$. Note that (1.5) is already represented as a CNF.

CNF (1.6) takes the value of **true** only if every clause $\left(\bigvee_{j \in S_i} x_j^{\sigma_j^i}\right)$ contains at least one literal (a literal is a variable or its negation) with the value of **true**. If we identify **false** with 0, and **true** with 1, then the negation operation $\neg$ converts x into $1 - x$. In view of what has been said, the truth sets on which (1.6) takes the value of **true**, are the solutions to the following system of inequalities:

$$\begin{aligned} \sum_{j \in S_i^1} x_j + \sum_{j \in S_i^0} (1 - x_j) \geq 1, \quad i = 1,\dots,m, \\ x_j \in \{0,1\}, \quad j = 1,\dots,n. \end{aligned} \tag{1.7}$$

Here, for $\delta \in \{0,1\}$, we use the notation $S_i^\delta \overset{\text{def}}{=} \{j \in S_i : \sigma_j^i = \delta\}$. For example, the CNF

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1)$$

takes the value of **true** on the sets that are solutions to the system

$$\begin{aligned} x_1+x_2+x_3 &\ge 1,\\ x_1+(1-x_2) &\ge 1,\\ x_2+(1-x_3) &\ge 1,\\ x_3+(1-x_1) &\ge 1,\\ x_1,x_2,x_3 &\in \{0,1\}. \end{aligned}$$

1.2 Multiple Alternatives and Disjunctions

It is required that out of given m inequalities

$$A_i x \le b_i\,, \quad i=1,\dots,m,$$

at least any q inequalities be satisfied. For example, if two jobs i and j are executed on the same machine, then we must require the validity of the following disjunction:

$$e_i - s_j \le 0 \quad \text{or} \quad e_j - s_i \le 0,$$

where s_i and e_i are, respectively, the start and end times of job i.

Introducing binary variables y_i for $i=1,\dots,m$, with $y_i=0$ if $A_i x \le b_i$ is valid, and $y_i=0$ otherwise, we can take into account the required condition as follows:

$$\begin{aligned} A_i x &\le b_i + M(1-y_i), \quad i=1,\dots,m,\\ \sum_{i=1}^{m} y_i &\ge q,\\ y_i &\in \{0,1\}, \quad i=1,\dots,m. \end{aligned}$$

Here M is a sufficiently large number such that the inequalities $A_i x \le b_i + M$ are satisfied automatically for all feasible solutions x of the problem being solved.

Now let us consider the case when at least one of the two conditions must hold:

$$x_1 \ge a \quad \text{or} \quad x_2 \ge b.$$

For example, we want to have a workstation with $x_1 \ge a$ processors or a single-processor system with a processor frequency $x_2 \ge b$.

If both variables x_1 and x_2 are nonnegative, then, introducing an auxiliary binary variable y, we can express the required disjunction by two inequalities:

$$x_1 \ge ay, \quad x_2 \ge b(1-y).$$

Next, we demonstrate the use of multiple alternatives and disjunctions on three examples.

1.2.1 Floor Planning

On a rectangular chip of width W and height H, n rectangular modules must be placed, module i has a width w_i and height h_i.

We choose a coordinate system with the origin O in the lower left corner of the chip, the Ox axis directed to the right, and the Oy axis directed upwards. Let the pair of real variables x_i, y_i determine the coordinate of the lower-left corner of module i, $i = 1,\dots,n$. Obviously, the following inequalities must hold:

$$0 \le x_i \le W - w_i,\ \ 0 \le y_i \le H - h_i, \quad i = 1,\dots,n. \tag{1.8}$$

To ensure that two modules, i and j, do not intersect, at least one of the following four inequalities must be valid:

$$\begin{aligned}
x_i + w_i &\le x_j \quad (i \text{ lies to the left of } j),\\
x_j + w_j &\le x_i \quad (i \text{ lies to the right of } j),\\
y_i + h_i &\le y_j \quad (i \text{ lies below } j),\\
y_j + h_j &\le y_i \quad (i \text{ lies above } j).
\end{aligned}$$

Introducing four binary variables z^l_{ij}, z^r_{ij}, z^b_{ij}, and z^a_{ij}, we can represent this disjunction by the following system of inequalities

$$\begin{aligned}
x_i + w_i &\le x_j + W(1 - z^l_{ij}),\\
x_j + w_j &\le x_i + W(1 - z^r_{ij}),\\
y_i + h_i &\le y_j + H(1 - z^b_{ij}),\\
y_j + h_j &\le y_i + H(1 - z^a_{ij}),\\
z^l_{ij} + z^r_{ij} &+ z^b_{ij} + z^a_{ij} \ge 1.
\end{aligned} \tag{1.9}$$

Usually the modules can be rotated by 90°. For each module $i = 1,\dots,n$, we introduce an additional binary variable δ_i, which takes the value of 1 if the module is rotated. Now the width and height of module i are respectively equal

$$(1-\delta_i)w_i + \delta_i h_i \quad \text{and} \quad (1-\delta_i)h_i + \delta_i w_i.$$

In view of this, (1.8) and (1.9) are rewritten as follows:

$$\begin{aligned}
0 \le x_i &\le W - ((1-\delta_i)w_i + \delta_i h_i), && i = 1,\dots,n,\\
0 \le y_i &\le H - ((1-\delta_i)h_i + \delta_i w_i), && i = 1,\dots,n,\\
x_i + (1-\delta_i)w_i + \delta_i h_i &\le x_j + W(1 - z^l_{ij}), && i = 1,\dots,n-1;\ j = i+1,\dots,n,\\
x_j + (1-\delta_j)w_j + \delta_j h_j &\le x_i + W(1 - z^r_{ij}), && i = 1,\dots,n-1;\ j = i+1,\dots,n,\\
y_i + (1-\delta_i)h_i + \delta_i w_i &\le y_j + H(1 - z^b_{ij}), && i = 1,\dots,n-1;\ j = i+1,\dots,n,\\
y_j + (1-\delta_j)h_j + \delta_j w_j &\le y_i + H(1 - z^a_{ij}), && i = 1,\dots,n-1;\ j = i+1,\dots,n,
\end{aligned}$$

$$z_{ij}^l + z_{ij}^r + z_{ij}^b + z_{ij}^a \geq 1, \quad i = 1,\dots,n-1;\ j = i+1,\dots,n,$$
$$z_{ij}^l, z_{ij}^r, z_{ij}^b, z_{ij}^a \in \{0,1\}, \quad i = 1,\dots,n-1;\ j = i+1,\dots,n,$$
$$\delta_i \in \{0,1\}, \quad i = 1,\dots,n.$$

1.2.2 Linear Complementarity Problem

It is necessary to solve the following system

$$Ax + y = b, \tag{1.10a}$$
$$x^T y = 0, \tag{1.10b}$$
$$x, y \geq 0, \tag{1.10c}$$

where A is a nonsingular $n \times n$-matrix, $b \in \mathbb{R}^n$, and x, y are n-vectors of continuous variables. This problem is known as the *linear complementarity problem*. Its important special cases are the linear programming problem, the quadratic programming problem under linear constraints (see Sect. 1.2.3), and the problem of finding a Nash equilibrium in a bimatrix game (see Exercise 1.7).

Despite its name, (1.10) is not a linear problem because (1.10b) is not a linear equation, which, due to the non-negativity of the vectors x and y, is equivalent to the system

$$x_i y_i = 0, \quad i = 1,\dots,n, \tag{1.11}$$

in which each equality $x_i y_i = 0$ expresses the disjunction: $x_i = 0$ or $y_i = 0$.

Assuming that we know upper bounds $x_i \leq g_i$ and $y_i \leq h_i$[1] for all variables x_i and y_i, we can represent (1.11) by the following system:

$$x_i \leq g_i z_i\,,\ \ y_i \leq h_i(1 - z_i)\,,\ z_i \in \{0,1\}, \quad i = 1,\dots,n.$$

1.2.3 Quadratic Programming Under Linear Constraints

The *quadratic programming problem* under linear constraints is formulated as follows:

$$\begin{aligned} c^T x + \frac{1}{2} x^T D x &\to \min, \\ Ax &\geq b, \\ x &\geq 0, \end{aligned} \tag{1.12}$$

[1] For an integer matrix A, we can estimate the values of x_i and y_i from Cramer's rule (do this as an exercise). But from a practical point of view, such estimates are too rough.

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, A is a real $m \times n$-matrix, D is a real symmetric $n \times n$-matrix, x is a vector of n variables.

If a point x is an optimal solution to (1.12), then, by the first-order necessary optimality conditions (also known as the Karush-Kuhn-Tucker (KKT) conditions), there exists a vector $y \in \mathbb{R}^m$ such that the following constraints hold:

$$\begin{aligned} & x \geq 0,\ y \geq 0, \\ & Ax \geq b, \\ & c + Dx - A^T y \geq 0, \\ & y^T(Ax - b) = 0, \\ & \left(c + Dx - A^T y\right)^T x = 0. \end{aligned} \tag{1.13}$$

If (x, y) is a solution to (1.13), then the point x is called a *stationary point* (or *KKT-point*) for (1.12). If D is a positive semi-definite matrix, then the objective function is convex, and, consequently, every stationary point is an optimal solution to (1.12).

Let us consider the next MIP:

$$\begin{aligned} z &\to \max, \\ 0 \leq Au - bz &\leq \mathbf{e} - \alpha, \\ 0 \leq Du - A^T v + cz &\leq \mathbf{e} - \beta, \\ 0 \leq u &\leq \beta, \\ 0 \leq v &\leq \alpha, \\ 0 \leq z &\leq 1, \\ \alpha &\in \{0,1\}^m, \\ \beta &\in \{0,1\}^n. \end{aligned} \tag{1.14}$$

We denote by z^* the optimal objective value in (1.14). It is not difficult to verify the following statements.

1. If $z^* = 0$, then problem (1.12) does not have stationary points.
2. If $z^* > 0$ and (u^*, v^*, z^*) is an optimal solution to (1.14), then the vectors $x^* = (1/z^*)u^*$, $y^* = (1/z^*)v^*$) make up a solution to (1.13), and hence x^* is a stationary point for (1.12).

1.3 How an LP May Turn Into a MIP?

When solving a practical problem, it is very important to foresee possible modifications of the initial model from the very beginning. Such modifications can easily bring the model out of the class of problems that can be solved by your program. In particular, it is very often the case with **LP** models, when addition of some simple and natural constraints transforms an LP into a MIP. Let us demonstrate this on an example of the *transportation problem*, which is one of the most famous **LP** models.

There are m suppliers and n consumers of some product. Supplier i has available a_i product units, and consumer j wants to get b_j product units. The unit transportation cost from supplier i to consumer j is c_{ij}. Let x_{ij} denote the quantity of the product delivered by supplier i to consumer j. It is necessary to determine a supply plan, $X = [x_{ij}]$, for which the total transportation cost is minimum.

This transport problem is formulated as the following LP:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min, \tag{1.15a}$$

$$\sum_{j=1}^{n} x_{ij} \le a_i, \quad i = 1,\dots,m, \tag{1.15b}$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1,\dots,n, \tag{1.15c}$$

$$x_{ij} \ge 0, \quad i = 1,\dots,m;\ j = 1,\dots,n. \tag{1.15d}$$

Objective (1.15a) is to minimize the total transportation cost. Inequalities (1.15b) require that the total amount of all deliveries of each supplier does not exceed the available quantity. Equations (1.15c) guarantee that each consumer will receive as much as is needed.

Below, we list several reasons why an optimal solution to (1.15) may be unacceptable in practice.

1. In many cases, especially when small quantities are supplied, *fixed costs* constitute a significant part of the transportation costs. In this case, the cost of supplying $x_{ij} > 0$ product units from supplier i to consumer j is $f_{ij} + c_{ij}x_{ij}$, where f_{ij} is a fixed cost.

2. It may turn out that, according to an optimal plan, some consumers have only one supplier, and they may require to *diversify* their supplies. Therefore, we have a new condition that each consumer j must receive the product from at least k_j suppliers.

3. It may well be that a particular consumer receives from some supplier, say, less than one percent of the demand, and such *splitting of supplies* may also not satisfy this consumer. Therefore, we have another requirement that the volume of any delivery to each consumer j be not less than u_j. Obviously, too small deliveries are unprofitable also for suppliers, and we suppose that it is required that the volume of any delivery from each supplier i be not less than v_i.

To take into account all the above requirements, we introduce an additional family of binary variables y_{ij}, where $y_{ij} = 1$ only if $x_{ij} > 0$. An extended formulation of the transportation problem is written as follows:

$$\sum_{i=1}^{m}\sum_{j=1}^{n}(f_{ij}y_{ij}+c_{ij}x_{ij}) \to \min, \tag{1.16a}$$

$$\sum_{j=1}^{n} x_{ij} \le a_i, \quad i=1,\dots,m, \tag{1.16b}$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j=1,\dots,n, \tag{1.16c}$$

$$\sum_{i=1}^{m} y_{ij} \ge k_j, \quad j=1,\dots,n, \tag{1.16d}$$

$$x_{ij} \le \min\{a_i,b_j\}y_{ij}, \quad i=1,\dots,m;\ j=1,\dots,n, \tag{1.16e}$$

$$x_{ij} \ge \max\{v_i,u_j\}y_{ij}, \quad i=1,\dots,m;\ j=1,\dots,n, \tag{1.16f}$$

$$y_{ij} \in \{0,1\}, \quad i=1,\dots,m;\ j=1,\dots,n. \tag{1.16g}$$

Objective (1.15a) is to minimize the sum of the fixed and variable transportation costs. Inequalities (1.16d) ensure that each consumer j will have at least k_j suppliers. Together (1.16e) and (1.16f) imply that $y_{ij}=0$ only if $x_{ij}=0$. Inequalities (1.16f) require that the volume of each delivery be not less than the minimum delivery volumes of the involved supplier and consumer.

1.4 Polyhedra

A *polyhedron* is the set of solutions of a system of linear inequalities. Bounded polyhedra are called *polytopes* in order to distinguish them from unbounded polyhedra. An alternative definition of a polyhedron is given by the following theorem.

Theorem 1.1 (Weyl). *A set $P \subseteq \mathbb{R}^n$ is a polyhedron if and only if $P = \mathrm{conv}(S) + \mathrm{cone}(T)$ for some finite subsets of vectors $S,T \subset \mathbb{R}^n$.*

The following notations were used in the above theorem:

- $\mathrm{conv}(S)$: *convex hull* of a set $S \subseteq \mathbb{R}^n$ that is the minimum (by inclusion) convex set containing S. Recall that a set is called *convex* if, with any two its points x^1 and x^2, it contains the segment $\{(1-\lambda)x^1+\lambda x^2:\ 0\le\lambda\le 1\}$ joining them.
- $\mathrm{cone}(T)$: *convex cone* generated by a set of vectors $T \subseteq \mathbb{R}^n$. If the set T is finite, $T=\{y^1,\dots,y^q\}$, then

$$\mathrm{cone}(T) \stackrel{\mathrm{def}}{=} \{x:\ x=\sum_{i=1}^{q}\lambda_i y^i,\ \lambda\in\mathbb{R}^q_+\}.$$

Theorem 1.1 also gives an alternative definition of a polytope as a convex hull of a finite set of points.

An affine subspace of $\mathbb{R}^n$ is the result of a parallel translation of a linear subspace. More precisely, for any point a of the affine subspace $\mathscr{A} \subseteq \mathbb{R}^n$ the set $\mathscr{L}_a = \{x - a : x \in \mathscr{A}\}$ is a linear subspace, and $\mathscr{A} = a + \mathscr{L}_a$.

The dimension of a set $X \subseteq \mathbb{R}^n$ is the dimension of the minimum affine subspace containing X. An affine subspace of dimension $n-1$ is called a *hyperplane*. We can also define a hyperplane as the set $H(a,\beta)$ of the solutions to a linear equation $a^T x = \beta$, where $a \in \mathbb{R}^n$ ($a \neq 0$), $\beta \in \mathbb{R}$. The hyperplane $H(a,\beta)$ divides the vector space $\mathbb{R}^n$ into two *half-spaces*

$$H_{\leq}(a,\beta) \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : \ a^T x \leq \beta\} \quad \text{and} \quad H_{\geq}(a,\beta) \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : \ a^T x \geq \beta\}.$$

We can say that the polyhedron is the intersection of a finite number of half-spaces.

Let $P \in \mathbb{R}^n$ be a polyhedron of size d, and $H(a,\beta)$ be a hyperplane. If P completely belongs to one of the half-spaces $H_{\leq}(a,\beta)$ or $H_{\geq}(a,\beta)$ and touches the hyperplane $H(a,\beta)$ ($P \cap H(a,\beta) \neq \emptyset$), then $P \cap H(a,\beta)$ and $H(a,\beta)$ are called a *face* and a *supporting hyperplane* of the polyhedron P. We specifically distinguish three types of faces:

- *facet*: face of size $d-1$;
- *vertex*: face of size 0 (dot);
- *edge*: face of size 1 (segment).

Two vertices of a polyhedron are called *adjacent* if they are connected by an edge (lie on one edge).

Any supporting hyperplane that is tangent to a facet is also called a *facet defining hyperplane*. For a *full-dimensional* polyhedron (of dimension n), the facet defining hyperplanes are uniquely defined (up to multiplication by a positive scalar). If in a system of inequalities $Ax \leq b$ a hyperplane $H(A_i, b_i)$ is not facet defining for the polyhedron $P(A,b) \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : Ax \leq b\}$, then it can be excluded from the system of inequalities without expanding the set of its solutions. In practice, we can recognize facet defining hyperplanes based on the following statement.

Proposition 1.2. *A hyperplane is facet defining for a polyhedron P of dimension d if and only if it contains at least d vertices of P.*

A three-dimensional polytope P, shown in Fig. 1.4, is formed by the intersection of half-spaces, which are given by the inequalities:

$$\begin{array}{rrrl}
x_1 & +x_2 & +x_3 & \leq 4, \\
 & x_2 & & \leq 2, \\
 & & x_3 & \leq 3, \\
3x_1 & & +\,x_3 & \leq 6, \\
x_1 & & & \geq 0, \\
 & x_2 & & \geq 0, \\
 & & x_3 & \geq 0.
\end{array}$$

It has 7 facets, 8 vertices (depicted as bold dots), and 13 edges (segments that connect the vertices).

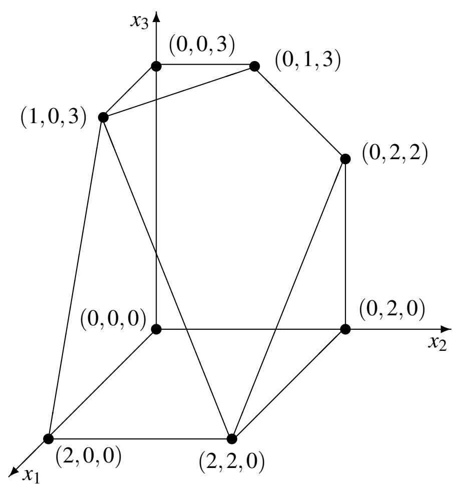

Fig. 1.4 Example of a polytope

1.5 Good and Ideal Formulations, Reformulation

In many cases, the same problem can be formulated in several different ways. Not all formulations are equivalent. In this section we will try to understand why one formulation is better than another.

Let us consider a mixed integer set

$$P(A,b;S) \overset{\text{def}}{=} \{x \in \mathbb{R}^n : Ax \le b,\ x_j \in \mathbb{Z} \text{ for } j \in S\}, \tag{1.17}$$

where A is a real $m \times n$-matrix, $b \in \mathbb{R}^m$, $S \subseteq \{1,\dots,n\}$. If in the definition of the set $P(A,b;S)$ we drop the requirement about integrality of the variables, we obtain the *relaxation polyhedron*, $P(A,b)$, for this set.

Let $P(A,b;S)$ and $P(A',b';S)$ with $P(A,b;S) = P(A',b';S)$ be two different formulations for the feasible domain of some MIP. We say that $P(A,b;S)$ is *stronger* than $P(A',b';S)$ if the *relaxation polyhedron* $P(A,b)$ is contained in the relaxation polyhedron $P(A',b')$.

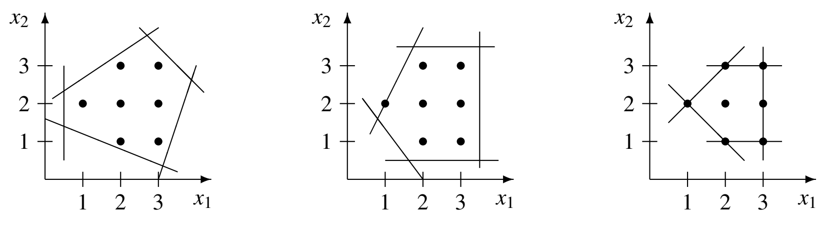

Fig. 1.5 Three formulations for a set in the plane

Figure 1.5 presents three different formulations for a set of seven points in the plane. The rightmost figure shows the *ideal formulation*, the polyhedron of which coincides with the convex hull of the given set of points. If it is possible to write down an ideal formulation for some MIP, then this MIP can be considered as an LP.

If a MIP is solved by the branch-and-bound method (see Sect. 6.1), as a rule, the use of a stronger formulation leads to a reduction in the solution time due to the decrease in the number of branchings. An obvious way to strengthen an existing formulation $P(A,b;S)$ is to add new inequalities valid for $P(A,b;S)$, but not valid for the relaxation polyhedron $P(A,b)$. We say that an inequality is *valid* for some set if all points from this set satisfy this inequality. Let inequalities $a^T x \le u$ and $\alpha^T x \le v$ hold for all points in $P(A,b;S)$. It is said that the inequality $a^T x \le u$ is *stronger* than the inequality $\alpha^T x \le v$ (or the inequality $a^T x \le u$ *dominates* the inequality $\alpha^T x \le v$) if

$$P(A,b) \cap H_{\le}(a,u) \subset P(A,b) \cap H_{\le}(\alpha,v).$$

Example 1.1 *We need to compare two formulations*

$$\left\{ x \in \{0,1\}^{n+1}, \quad \sum_{j=1}^{n} x_j \le n x_{n+1} \right\} \tag{1.18}$$

and

$$\left\{ x \in \{0,1\}^{n+1}, \quad x_j \le x_{n+1}, \quad j = 1,\dots,n \right\} \tag{1.19}$$

for the set of points

$$X = \{x \in \{0,1\}^n : \, x_{n+1} = 0 \Rightarrow x_1 = x_2 = \dots = x_n = 0\}.$$

Solution. Let

$$P_1 = \{x \in \mathbb{R}^{n+1} : \, \sum_{j=1}^{n} x_j \le n x_{n+1}, \; 0 \le x_j \le 1, \; j = 1,\dots,n+1\},$$
$$P_2 = \left\{x \in \mathbb{R}^{n+1} : \, x_j \le x_{n+1}, \; 0 \le x_j \le 1, \; j = 1,\dots,n+1\right\}$$

be the relaxation polytopes for (1.18) and (1.19), respectively. Summing together the inequalities

$$x_j \le x_{n+1}, \quad j = 1,\dots,n,$$

we obtain the inequality

$$\sum_{j=1}^{n} x_j \le n x_{n+1}.$$

Thus, we have shown that $P_2 \subseteq P_1$.

To show that P_2 is a proper subset of P_1 ($P_2 \subset P_1$), it is enough to specify a point from P_1 that does not belong to P_2. There are many such points. In particular, the vertices $(\mathbf{e}_j, 1/n)$ $(j = 1,\dots,n)$ of P_1 do not belong to P_2. Here $\mathbf{e}_j$ is the j-th unit vector in $\mathbb{R}^n$. Since $P_2 \subset P_1$, (1.19) is stronger than (1.18).

We now show that (1.19) is an ideal formulation. For this it suffices to show that the polytope P_2 is *integer*, that is, all its vertices are integer. Let $\bar{x} \in P_2$. If $0 < \bar{x}_{n+1} < 1$, then, taking into account the inequalities $\bar{x}_j \le \bar{x}_{n+1}$, we conclude that the point $(1/\bar{x}_{n+1}) \cdot \mathbf{e}$ also belongs to P_2. But since $0 \in P_2$ and

$$\bar{x} = (1 - \bar{x}_{n+1}) \cdot \mathbf{0} + \bar{x}_{n+1} \frac{1}{\bar{x}_{n+1}} \cdot \mathbf{e},$$

then $\bar{x}$ lies on the segment joining two points of the polytope P_2. Hence, $\bar{x}$ is not a vertex of P_2.

So, if $\bar{x}$ is a vertex of P_2, then its component $\bar{x}_{n+1}$ is 0 or 1. If $\bar{x}_{n+1} = 0$, then all other components $\bar{x}_j$ are also equal to zero. If $\bar{x}_{n+1} = 1$, then the inequalities $x_j \le x_{n+1}$ become the inequalities $x_j \le 1$. Therefore, the point $(\bar{x}_1,\dots,\bar{x}_n)$ must be one of the vertices of the cube $[0,1]^n$, which are all integer. $\square$

Let us note, that the *disaggregation* of inequalities, i.e., replacing one inequality with a system of inequalities that is stronger than the initial inequality, is a powerful preprocessing technique (see Sect. 6.5).

1.6 Strong Inequalities

Let A be a real $m \times n$ -matrix, and $b \in \mathbb{R}^m$. We say that an inequality $\alpha^T x \le \beta$ is a *consequence* of the system of inequalities $Ax \le b$ if $\alpha^T x \le \beta$ holds for all solutions to $Ax \le b$.

Proposition 1.3. *An inequality $\alpha^T x \le \beta$ is a consequence of a consistent system of inequalities $Ax \le b$ if and only if there exists a vector $u \in \mathbb{R}^m_+$ such that $\alpha^T = u^T A$ and $\beta \ge u^T b$.*

Proof. If $\alpha^T = u^T A$ and $\beta \ge u^T b$ for some $u \in \mathbb{R}^m_+$, then, for any solution $\bar{x}$ of the system $Ax \le b$, the inequality

$$\alpha^T \bar{x} = u^T A \bar{x} \le u^T b \le \beta$$

holds, and, therefore, $\alpha^T x \le \beta$ is a consequence of $Ax \le b$.

Now we assume that an inequality $\alpha^T x \le \beta$ is a consequence of a system $Ax \le b$. By Theorem 3.1 (of LP duality), we have

$$\beta \ge \max\{\alpha^T x : \ Ax \le b\} = \min\{b^T u : \ A^T u = \alpha, \ u \ge 0\}.$$

Let $\bar{u} \in \mathbb{R}^m_+$ be an optimal solution to the right-hand LP. Then $\alpha = A^T \bar{u}$ and $\beta \ge \bar{u}^T b$. $\square$

An inequality $A_i x \le b_i$ is *redundant* in a system of inequalities $Ax \le b$ if $P(A,b) = P(A_{M\setminus\{i\}}, b_{M\setminus\{i\}})$, where $M = \{1,\dots,m\}$. Proposition 1.3 allows us to give an algebraic equivalent to this geometric definition: the inequality $A_i x \le b_i$ is redundant in $Ax \le b$ if it is a consequence of the subsystem of all other inequalities $A_k x \le b_k$, $k \ne i$. Note that this algebraic definition is most often used to identify redundant inequalities. We also note that the presence of redundant inequalities in the constraint system of an LP is highly undesirable, since this can significantly slow down the process of solving this LP.

Now we give a characterization of *non-redundant* inequalities of a system $Ax \le b$. Since a non-redundant inequality cannot be removed from the system without expanding the polyhedron $P(A,b)$, each such inequality must touch the polyhedron $P(A,b)$ over some its facet (see Sect. 1.4). Let us remember that such inequalities are called *facet defining*.

Proposition 1.4. *If the dimension of a polyhedron $P \subseteq \mathbb{R}^n$ is d, then a valid for P inequality $\alpha^T x \le \beta$ is facet defining for P if and only if P contains d affinely independent points[2] lying on the hyperplane $H(\alpha,\beta)$.*

Now let us consider a mixed integer set $P(A,b;S) \subseteq \mathbb{R}^n$. We say that out of two valid for $P(A,b;S)$ inequalities, $\alpha^T x \le \beta$ and $\pi^T x \le \gamma$, the inequality $\alpha^T x \le \beta$ is stronger than (dominates) the inequality $\pi^T x \le \gamma$, if the inclusion

$$P(A,b) \cap H_{\le}(\alpha,\beta) \subset P(A,b) \cap H_{\le}(\pi,\gamma)$$

is valid. An inequality is called *strong* if there exists no other inequality that is stronger than it.

Let us call a set from $\mathbb{R}^n$ *polyhedral* if its convex hull is a polyhedron. Suppose that the set $P(A,b;S)$ is bounded, and all elements of the matrix A and vector b are rational numbers. Under this assumption, the set $X = \text{conv}(P(A,b;S))$ is a polyhedron (see Exercise 4.1). Therefore, the inequalities that are facet defining for X are strong for the set $P(A,b;S)$. Since all vertices of the polyhedron X belong to the set $P(A,b;S)$, we can reformulate Proposition 1.4 as follows.

Proposition 1.5. *If the dimension of a polyhedral set $P(A,b;S)$ is equal to d, then a valid for $P(A,b;S)$ inequality $\alpha^T x \le \beta$ is strong for $P(A,b;S)$ if and only if $P(A,b;S)$ contains d affinely independent points lying on the hyperplane $H(\alpha,\beta)$.*

1.7 Extended Formulations

We can strengthen a formulation of a MIP by adding to it new constraints. Alternatively, we can try to develop an *extended* formulation by adding to the existing formulation new variables and constraints. In this section we consider two examples

[2] A set of points $x^1,\dots,x^d$ from $\mathbb{R}^n$ is affinely independent if all these points do not lie in an affine subspace of dimension less than d. Equivalently, the points $x^1,\dots,x^d$ are affinely independent if the points $x^2 - x^1,\dots,x^d - x^1$ are linearly independent.

of extended formulations trying to understand why some formulations are stronger than others.

1.7.1 Single-Product Lot-Sizing Problem

Consider a single-product version of the lot-sizing problem. A firm is producing some product, production and storage capacities are unlimited (in comparison to the demands). The planning horizon consists of T periods. For each period $t = 1,\dots,T$, we know

- d_t: demand for product;
- f_t: fixed production cost;
- c_t: unit production cost;
- h_t: unit storage cost.

Inventory of the product in the warehouse before the start of the planning horizon is s_0.

It is necessary to determine how many units of the product to produce in each period in order to fully meet the demands and so that the total production and storage cost over all T periods is minimum.

For $t = 1,\dots,T$, we introduce the following variables:

- x_t: amount of product produced in period t;
- s_t: amount of product stored in the warehouse at the end of period t;
- $y_t = 1$, if the product is produced in period t, and $y_t = 0$ otherwise.

Having determined $D_t = \sum_{\tau=t}^{T} d_\tau$, we can write the following MIP:

$$\sum_{t=1}^{T} (f_t y_t + c_t x_t + h_t s_t) \to \min, \tag{1.20a}$$

$$s_{t-1} + x_t = d_t + s_t, \quad t = 1,\dots,T, \tag{1.20b}$$

$$0 \le x_t \le D_t y_t, \quad t = 1,\dots,T, \tag{1.20c}$$

$$y_t \in \{0,1\}, \quad t = 1,\dots,T. \tag{1.20d}$$

Objective (1.20a) is to minimize total expenses over all T periods. Each balance equation in (1.20b) relates two neighboring periods: the amount of product, s_{t-1}, in the warehouse at the end of period $t-1$ plus the amount, x_t, produced in period t equals the demand, d_t, in period t plus the amount, s_t, stored in the warehouse at the end of period t. The inequalities in (1.20c) impose the implications: $y_t = 0 \Rightarrow x_t = 0$.

If all fixed costs, f_t, are positive, then for an optimal solution (x^*, y^*) of the relaxation LP for (1.20), we have

$$y_t^* = x_t^*/D_t, \quad t = 1,\dots,T.$$

Consequently, for all producing periods t (when $x_t^* > 0$), except for the last one, y_t^* is a fractional number, since $y_t^* = x_t^*/D_t \le d_t/D_t < 1$. Many integer variables taking fractional values is a clear indicator that the used formulation is weak.

To obtain an ideal formulation for our lot-sizing problem, we need to add to (1.20) the system of (l,S)-inequalities:

$$\sum_{t\in S} x_t + \sum_{t\in \bar{S}} d_{tl}\, y_t \ge d_{1l}, \quad S \subseteq \{1,\dots,l\},\ l=1,\dots,T, \tag{1.21}$$

where $\bar{S} = \{1,\dots,l\} \setminus S$ and $d_{ij} = \sum_{t=i}^{j} d_t$. This inequalities reflect the following simple observation: the amount of product produced in periods $t \in S$ $(\sum_{t\in S} x_t)$ and the maximum amount of product that can be produced in periods $t \in \bar{S}$ for use in the first l periods $(\sum_{t\in\bar{S}} d_{tl}\, y_t)$ must be no less than the product demand in these first l periods (d_{1l}).

We see that the ideal formulation for the set of feasible solutions to (1.20) contains an exponentially many inequalities. Although this is not a big obstacle to using this formulation in practice, it is still impossible to solve such a MIP using the standard software, and therefore, to represent any exponentially large family of inequalities, we need to implement a specialized separation procedure (for example see Sect. 6.6).

We can strengthen (1.20) in another way by *disaggregating* the decision variables: $x_t = \sum_{\tau=t}^{T} x_{t\tau}$, where, for $t = 1,\dots,T$, and $\tau = t,\dots,T$, the new variable $x_{t\tau}$ represents the amount of the product produced in period t for period τ.

First, we exclude from (1.20) the variables s_t. Adding together the balance equations $s_{k-1} + x_k = d_k + s_k$ for $k = 1,\dots,t$, we obtain

$$s_t = s_0 + \sum_{k=1}^{t} x_k - \sum_{k=1}^{t} d_k.$$

Using these equalities, we rewrite the objective function in the following way

$$\begin{aligned}&\sum_{t=1}^{T}(f_t y_t + c_t x_t + h_t \left(s_0 + \sum_{k=1}^{t} x_k - \sum_{k=1}^{t} d_k\right)\\ &= \sum_{t=1}^{T}(f_t y_t + w_t x_t) + K = \sum_{t=1}^{T} f_t y_t + \sum_{t=1}^{T}\sum_{\tau=t}^{T} w_t x_{t\tau} + K,\end{aligned}$$

where $w_t = c_t + h_t + \dots + h_T$ and $K = \sum_{t=1}^{T} h_t\,(s_0 - \sum_{k=1}^{t} d_k)$.

In the new variables $x_{t\tau}$, (1.20) can be reformulated as follows:

$$\begin{aligned}
&\sum_{t=1}^{T} f_t y_t + \sum_{t=1}^{T}\sum_{\tau=t}^{T} w_t x_{t\tau} \to \min,\\
&\sum_{t=1}^{\tau} x_{t\tau} = d_\tau, \quad \tau = 1,\dots,T, \qquad (1.22)\\
&0 \le x_{t\tau} \le d_\tau y_t, \quad t = 1,\dots,T;\ \tau = t,\dots,T,\\
&y_t \in \{0,1\}, \quad t = 1,\dots,T.
\end{aligned}$$

It is not difficult to show that among the solutions of the relaxation LP for (1.22) there are solutions (x^*,y^*) for which all components of the vector y^* are integer and from $x^*_{t\tau} > 0$ it follows that $x^*_{t\tau} = d_\tau$, i.e., the whole demand for product in any period τ is fully produced only in one period. For this reason, (1.22) can be considered as an "almost" ideal formulation.

The main drawback of many extended formulations is their large size. In our case, we replaced Formulation (1.20) having $3T$ variables and $2T$ nontrivial[3] constraints with Formulation (1.22) having $T(T+1)/2$ variables and $2T$ constraints. For example, if $T = 100$, we have only 300 variables in the first case, and 5050 in the second case. The difference is huge! Sometimes, it is more efficient to use in practice so-called *approximate extended formulations*, such a formulation is obtained by adding to the basic compact formulation only a part of the "most important" variables and constraints of the extended formulation.

1.7.2 Fixed Charge Network Flows

A transportation network is given by a *directed graph* (*digraph*) $G = (V,E)$. For each node $v \in V$, we know the *demand* d_v for some product. If $d_v > 0$, then v is a demand node; if $d_v < 0$, then v is a supply node; $d_v = 0$ for transit nodes. It is assumed that supply and demand are balanced: $\sum_{v\in V} d_v = 0$. The *capacity* of an arc $e \in E$ is $u_e > 0$, and the *cost* of shipping $x_e > 0$ units of product along this arc is $f_e + c_e x_e$. Naturally, if the product is not moved through the arc ($x_e = 0$), then nothing is paid. The *fixed charge network flow problem* (*FCNF*) is to decide on how to transport the product from the supply to the demand nodes so that the transportation expenses are minimum.

The FCNF problem appears as a subproblem in many practical applications such as designing transportation and telecommunication networks, or optimizing supply chains.

Introducing the variables

- x_e: *flow* (quantity of shipping product) through arc $e \in E$,
- $y_e = 1$ if product is shipped ($x_e > 0$) through arc e, and $y_e = 0$ otherwise,

we formulate the FCNF problem as follows:

[3] Normally, the lower and upper bounds for variables are called trivial constraints

$$\sum_{e\in E}(f_e y_e + c_e x_e) \to \min, \tag{1.23a}$$

$$\sum_{e\in E(V,v)} x_e - \sum_{e\in E(v,V)} x_e = d_v, \quad v \in V, \tag{1.23b}$$

$$0 \le x_e \le u_e y_e, \quad e \in E, \tag{1.23c}$$

$$y_e \in \{0,1\}, \quad e \in E. \tag{1.23d}$$

Here $E(V,v)$ (resp., $E(v,V)$) denote the sets of arcs from E that are entering (resp., leaving) a node $v \in V$.

Objective (1.23a) is to minimize the total transportation expenses. Each *balance equation* in (1.23b) requires that the number of flow units entering a particular node be equal to the number of flow units leaving this node. The *variable upper bounds* (1.23c) are *capacity restrictions* with the following meaning:

- the flow through any arc cannot exceed the arc capacity;
- if some arc is not used for shipping product ($y_e = 0$), then the flow through this arc is zero ($x_e = 0$).

Since, for any optimal solution of the relaxation LP, $y_e = x_e/u_e$, then (1.23) cannot be a strong formulation if the capacities, u_e, of many arcs are greater than the flows, x_e, along these arcs. This, for example, happens in problems without capacity limitations, when the numbers u_e are some rough upper estimates for the values of the arc flows x_e.

We can strengthen (1.23) by disaggregating the flow variables x_e. In what follows we assume that all the values of f_e and c_e are non-negative. In a new formulation for each flow unit along any arc, we will indicate its origin supply node and its destination demand node. Let us denote by S and T, respectively, the set of *supply* ($d_v < 0$) and *demand* ($d_v > 0$) nodes. We introduce two new families of variables:

- q_{st}: number of product units supplied from node $s \in S$ to node $t \in T$;
- z_e^{st}: (s,t)-flow along arc $e \in E$ that is a part of flow sending from $s \in S$ to $t \in T$ and going through arc e.

With these new variables, we rewrite (1.23) as follows:

$$\sum_{e\in E}(f_e y_e + c_e x_e) \to \min, \tag{1.24a}$$

$$\sum_{e\in E(V,s)} z_e^{st} - \sum_{e\in E(s,V)} z_e^{st} = -q_{st}, \quad s \in S,\ t \in T, \tag{1.24b}$$

$$\sum_{e\in E(V,v)} z_e^{st} - \sum_{e\in E(v,V)} z_e^{st} = 0, \quad v \in V \setminus \{s,t\}),\ s \in S,\ t \in T, \tag{1.24c}$$

$$\sum_{t\in T} q_{st} = -d_s, \quad s \in S, \tag{1.24d}$$

$$\sum_{s\in S} q_{st} = d_t, \quad t \in T, \tag{1.24e}$$

$$\sum_{(s,t)\in S\times T} z_e^{st} = x_e, \quad e \in E, \tag{1.24f}$$

$$0 \le z_e^{st} \le \min\{u_e, -d_s, d_t\} y_e, \quad e \in E,\ s \in S,\ t \in T, \tag{1.24g}$$
$$0 \le x_e \le u_e y_e, \quad e \in E, \tag{1.24h}$$
$$q_{st} \ge 0, \quad s \in S,\ t \in T, \tag{1.24i}$$
$$y_e \in \{0,1\}, \quad e \in E. \tag{1.24j}$$

In this formulation, (1.24b) and (1.24c) are flow conservation constraints: for $s \in S$ and $t \in T$, $z^{st} \in \mathbb{R}^E$ is a flow from s to t of value q_{st}, i.e., q_{st} flow units are sent from s to t, and, for all nodes other than s or t, the incoming and outgoing flows are equal. Equations (1.24d) and (1.24e) ensure that each supply nodes sends and each demand node receives the required quantity of product. Equations (1.24f) determine the total flow along any arc by summing up all the flows going from supply to demand nodes. The variable upper bounds (1.24g) impose the capacity limitations for the arc flows: the flow z_e^{st} along arc e sent from s to t cannot exceed the capacity u_e of arc e, the supply $-d_s$ at node s and the demand d_t at node t. In fact, these more precise variable bounds make (1.24) stronger than (1.23).

1.8 Alternative Formulations for Scheduling Problems

We have already seen that a formulation of a MIP can be strengthened by adding to it new constraints and variables. When some existing formulation do not allow us to solve problems of required sizes (from practical point of view), sometimes, it is possible to completely change the modeling concept and develop an *alternative* formulation. In this section we consider two different modeling concepts for a rather general *scheduling problem*, which subsumes as special cases a great deal of scheduling problems studied in the literature. We have to fulfill a set of jobs on a number of processors under certain constraints such as restrictions on the job completion times, priorities between jobs (one job cannot start until another one is finished), and etc. The goal is to optimize some criterion, e.g, to minimize the total processing time, which is the completion time of the last job (assuming that the first job starts at time 0); or to maximize the number of processed jobs.

Formally, we are given n jobs to be processed on m processors (machines). Let $P_j \subseteq \{1,\dots,m\}$ denote the subset of processors that can fulfill job $j \in J \overset{\text{def}}{=} \{1,\dots,n\}$.

Each job j is characterized by the following parameters:

- w_j: weight;
- r_j, d_j: release and due dates (the job must be processed during the time interval $[r_j, d_j]$);
- p_{ij}: processing time on processor $i \in P_j$.

Precedence relations between jobs are given by an acyclic digraph $G = (J, E)$ defined on the set J of jobs: for any arc $(j_1, j_2) \in J$, job j_2 cannot start until job j_1 is finished.

In general not all jobs can be processed. For a given schedule, let $U_j = 0$ if job j is processed, and $U_j = 1$ otherwise. Then the problem is to find such a job schedule for which the weighted number of not processed jobs, $\sum_{j=1}^n w_j U_j$, is minimum. Alternatively, we can say that our goal is to maximize the weighted sum of processed jobs, which is $\sum_{j=1}^n w_j(1-U_j)$.

1.8.1 Continuous Time Model

In models with continuous time, the main variables correspond to events that are defined as the moments when individual jobs begin or end. In our model, we use the following variables:

- s_j: start time of job j;
- $y_{ij} = 1$ if job j is accomplished by processor i, and $y_{ij} = 0$ otherwise;
- $x_{i,j_1,j_2} = 1$ if both jobs, j_1 and j_2, are carried out on processor i, and j_1 is finished before j_2 starts, and $x_{i,j_1,j_2} = 0$ otherwise.

In these variables we formulate our scheduling problem as follows:

$$
\begin{aligned}
\sum_{j=1}^{n} w_j \sum_{i \in P_j} y_{ij} &\to \max, && \text{(1.25a)}\\
\sum_{i \in P_j} y_{ij} &\le 1, \quad j = 1 = 1,\dots,n, && \text{(1.25b)}\\
t_j &= \sum_{i \in P_j} p_{ij} y_{ij}, \quad j = 1,\dots,n, && \text{(1.25c)}\\
r_j \le s_j &\le d_j - t_j, \quad j = 1,\dots,n, && \text{(1.25d)}\\
s_{j_2} - s_{j_1} + M(1 - x_{i,j_1,j_2}) &\ge p_{i,j_1}, \quad j_1, j_2 = 1,\dots,n,\ j_1 \ne j_2,\\
&\qquad i \in P_{j_1} \cap P_{j_2}, && \text{(1.25e)}\\
x_{i,j_1,j_2} + x_{i,j_2,j_1} &\le y_{i,j_2}, \quad j_1, j_2 = 1,\dots,n,\ j_1 \ne j_2,\\
&\qquad i \in P_{j_1} \cap P_{j_2}, && \text{(1.25f)}\\
y_{i,j_1} + y_{i,j_2} - x_{i,j_1,j_2} - x_{i,j_2,j_1} &\le 1, \quad j_1 = 1,\dots,n-1,\\
&\qquad j_2 = j_1 + 1,\dots,n,\ i \in P_{j_1} \cap P_{j_2}, && \text{(1.25g)}\\
s_{j_1} + t_{j_1} &\le s_{j_2}, \quad (j_1, j_2) \in E, && \text{(1.25h)}\\
y_{ij} &\in \{0,1\}, \quad j = 1,\dots,n,\ i \in P_j, && \text{(1.25i)}\\
x_{i,j_1,j_2} &\in \{0,1\}, \quad j_1, j_2 = 1,\dots,n,\ j_1 \ne j_2,\\
&\qquad i \in P_{j_1} \cap P_{j_2}. && \text{(1.25j)}
\end{aligned}
$$

Here M is a sufficiently large number, for example,

$$M = \max_{1 \le j \le n} d_j - \min_{1 \le j \le n} r_j.$$

Objective (1.25a) is to maximize the weighted sum of accomplished jobs. Here the sum $\sum_{i \in P_j} y_{ij}$ take the value of 1 only if job j is carried out by some processor. Inequalities (1.25b) ensure that any job will be assigned to at most one processor. Equations (1.25c) determine the actual processing times, t_j, of all jobs j, where t_j is the processing time of job j on the processor it is assigned to. Inequalities (1.25d) require that all jobs be processed within the given time intervals. Since M is sufficiently large, any particular inequality in (1.25e) is a real restriction only if $x_{i,j_1,j_2} = 1$. In this case, both jobs, j_1 and j_2, are assigned to processor i, and the inequality

$$s_{j_2} - s_{j_1} \geq p_{i,j_1}$$

means that job j_1 is finished when job j_2 starts; the latter agrees well with the equality $x_{i,j_1,j_2} = 1$. Two inequalities, one from (1.25f) and the other from (1.25g), written for particular i, j_1 and j_2, imply that if both jobs, j_1 and j_2, are accomplished by processor i ($y_{i,j_1} = y_{i,j_2} = 1$), then either $x_{i,j_1,j_2} = 1$ (j_1 precedes j_2) or $x_{i,j_2,j_1} = 1$ (j_2 precedes j_1) but not both. Inequalities (1.25h) reflect the precedence relations.

As a rule, formulations with big M are weak. Our formulation (1.25) is not an exception. For simplicity, suppose that there is only one processor ($m = 1$, $P_1 = \{1,\dots,n\}$), and there are no precedence relations ($E = \emptyset$). For sufficiently large M, the relaxation LP for (1.25) has an optimal solution (s^*, x^*, y^*) with $y^*_{1j} = 1$, $s^*_j = r_j$ for $j = 1,\dots,n$, and all $x^*_{1,j_1,j_2} = 1/2$. Obviously, such a solution can be very far from the problem feasible domain. As a consequence, (1.25) cannot be used for solving scheduling problems of practical importance.

1.8.2 Time-Index Formulation

A time-index formulation is based on time-discretization, i.e., the *planning horizon*, from time $R = \min_{1 \leq j \leq n} r_j$ to time $D = \max_{1 \leq j \leq n} d_j$, is divided into periods, and period t starts at time $t-1$ and ends at time t. Now a *schedule* is represented by a family of *decision* binary variables $\{x_{jit}\}$, where $x_{jit} = 1$ if job j starts in period t on processor i, and $x_{jit} = 0$ otherwise. To formulate precedence relations we need three families of *auxiliary* variables, which are uniquely defined by the decision variables x_{jit}:

- $y_j = 1$ if job j is processed, and $y_j = 0$ otherwise;
- s_j: start time of job j;
- t_j: processing time of job j.

We consider the following time-index formulation:

$$\sum_{j=1}^{n} w_j y_j \to \max, \tag{1.26a}$$

$$\sum_{\substack{1\le j\le n:\\ r_j\le t\le d_j-p_{ij}}} \sum_{\tau=\max\{t-p_{ij},r_j\}}^{\min\{t,d_j-p_{ij}\}} x_{ji\tau} \le 1, \quad t=R,\dots,D,\ i=1,\dots,m, \tag{1.26b}$$

$$y_j = \sum_{i\in P_j} \sum_{t=r_j}^{d_j-p_{ij}} x_{jit}, \quad j=1,\dots,n, \tag{1.26c}$$

$$s_j = \sum_{i\in P_j} \sum_{t=r_j}^{d_j-p_{ij}} t\cdot x_{jit}, \quad j=1,\dots,n, \tag{1.26d}$$

$$t_j = \sum_{i\in P_j} \sum_{t=r_j}^{d_j-p_{ij}} p_{ij}\cdot x_{jit}, \quad j=1,\dots,n, \tag{1.26e}$$

$$y_{j_1} - y_{j_2} \ge 0, \quad (j_1,j_2)\in E, \tag{1.26f}$$

$$s_{j_2} - s_{j_1} \ge t_{j_1}, \quad (j_1,j_2)\in E, \tag{1.26g}$$

$$x_{jit}\in\{0,1\}, \quad i\in P_j,\ t=r_j,\dots,d_j-p_{ij},\ j=1,\dots,n, \tag{1.26h}$$

$$y_j\in\{0,1\}, \quad j=1,\dots,n. \tag{1.26i}$$

Objective (1.26a) is to maximize the weighted number of processed jobs. Inequalities (1.26b) ensure that any processor will perform at most one job in any period. Equations (1.26c), (1.26d), and (1.26e) determine the values of all auxiliary variables, y_j, s_j, and t_j. Simultaneously, Eqs. (1.26c) imply that each job can start only once (because $y_j \in \{0,1\}$). The precedence relations are expressed by Ineqs. (1.26f) and (1.26g). For each pair of related jobs $(j_1, j_2) \in E$, (1.26f) requires that j_2 be processed only if j_1 is processed, while (1.26g) requires that, if both jobs, j_1 and j_2, are processed, then j_1 must be finished when j_2 starts.

The time-index formulation can be easily modified to model many other types of scheduling problems, and this is its important advantage. For example, if we set $y_j = 1$ for all j, and redefine the objective as follows

$$\sum_{j=1}^{n} w_j \sum_{i\in P_j} \sum_{t=l_j}^{u_j-p_{ij}} (t+p_{ij})x_{jit} \to \min,$$

then our goal is to minimize the weighted completion time.

In addition, the optimal objective value of the relaxation LP for (1.26) — this LP is obtained from (1.26) after dropping the requirement about integrality of the variables — provides a strong bound on the objective value of our scheduling problem, and it dominates the bounds provided by the other known IP formulations. This is because any solution to the relaxation LP of the time-index formulation can be interpreted as some *non-preemptive* relaxation schedule. More precisely, such a relaxation schedule is obtained by slicing jobs into pieces and then each piece is processed without interruption.

The main disadvantage of the time-index formulation is its size: even for one machine problems, there are $n+T$ constraints and there may be up to nT variables. As a consequence, for instances with many jobs and long processing intervals $[r_j, d_j]$, the relaxation LPs will be very big in size, and their solution times will be large. Nevertheless, the time-index formulation can be used in practice for solving scheduling problems with relatively short planning horizons.

1.9 Knapsack Problems

There are two basic variations of the *knapsack problems*:

integer knapsack:

$$\max\{c^T x : \ a^T x \le b, \ x \in \mathbb{Z}^n_+\}, \tag{1.27}$$

0,1-knapsack:

$$\max\{c^T x : \ a^T x \le b, \ x \in \{0,1\}^n\}. \tag{1.28}$$

Here $c \in \mathbb{R}^n_{++}$, $a \in \mathbb{Z}^n_{++}$, and b is a positive integer.

The problem was named due to the following not very serious interpretation. You won a prize that allows you to fill your knapsack with any items present in a supermarket, provided that the total weight of items should not exceed the weight limit, b, of your knapsack. Suppose that there are n types of items in the supermarket, and the cost of one item j is c_j. You will most likely want to fill your knapsack so that the total cost of items in it is maximum. To do this, you have to solve an integer knapsack problem. If you are allowed to take not more than one item of each type, then you need to solve a 0,1-knapsack problem.

Of course, we could also give more important examples of the application of the knapsack problems in practice. But in fact, the real world is not so simple, and there are not many real situations in it that can be modeled only by one linear constraint, even with integer variables[4]. Here we study the knapsack problems for another reason: the separation problems for many classes of inequalities, and the estimation problems for a number of column generation algorithms are formulated as knapsack problems.

The knapsack problems are the simplest IPs, but even they, (1.27) and (1.28)), are **NP**-hard[5]. Despite this fact, in practice we can solve knapsack problems with not very big coefficients relatively quickly using dynamic programming algorithms.

[4] Sometimes, objecting to this statement, one recalls the aggregation of equations (see Exercise 1.10) that allows us to express several constraints with just one equation. But aggregation always weakens the formulation, and in practice it should be avoided.

[5] An **NP**-*complete problem* is a recognition problems (with the answer ”yes” or ”no”), solvable on a non-deterministic Turing machine in polynomial time. A problem $\mathscr{P}$ is called **NP**-hard if any **NP**-complete problem can be solved in polynomial time using a procedure, $\mathscr{A}$, for solving $\mathscr{P}$; it is assumed that one call to $\mathscr{A}$ takes constant time.

1.9.1 Integer Knapsack

Let us consider the integer knapsack problem (1.27). For $\beta = 0,\dots,b$, we define

$$F(\beta) \overset{\text{def}}{=} \max\{c^T x :\ a^T x = \beta,\ x \in \mathbb{Z}^n_+\}.$$

It is easy to verify that the following recurrence formula holds:

$$\begin{aligned} &F(0) = 0,\\ &F(\beta) = \max_{j:\, a_j \le \beta} F(\beta - a_j) + c_j, \quad \beta = 1,\dots,b. \end{aligned} \tag{1.29}$$

As usual, we assume that the maximum over the empty set of alternatives is equal to $-\infty$.

Calculating the values of $F(\beta)$ using (1.29) is called a *direct step* of dynamic programming. When all values $F(\beta)$ are calculated, an optimal solution, x^*, to (1.27) can be found by performing the following *reverse step*.

```
Start with $\beta \in \arg\max_{0\le q\le b} F(q)$, and set $x^*_j = 0$ for $j = 1,\dots,n$.
While $\beta > 0$, do the following computations:
    find an index $j$ such that $F(\beta) = F(\beta - a_j) + c_j$,
    and set $x^*_j := x^*_j + 1$, $\beta := \beta - a_j$.
```

Example 1.2 *We need to solve the problem*

$$\begin{aligned} 4x_1 + 5x_2 + \ \ x_3 + 2x_4 &\to \max,\\ 5x_1 + 4x_2 + 2x_3 + 3x_4 &\le 7,\\ x_1, x_2, x_3, x_4 &\in \mathbb{Z}_+. \end{aligned}$$

Solution. First, we compute

$$\begin{aligned} &F(0) = 0,\\ &F(1) = -\infty,\\ &F(2) = F(0) + 1 = 1,\\ &F(3) = \max\{F(1)+1, F(0)+2\} = \max\{-\infty, 2\} = 2,\\ &F(4) = \max\{F(0)+5, F(2)+1, F(1)+2\} = \max\{5, 2, -\infty\} = 5,\\ &F(5) = \max\{F(0)+4, F(1)+5, F(3)+1, F(2)+2\}\\ &\qquad = \max\{4, -\infty, 3, 3\} = 4,\\ &F(6) = \max\{F(1)+4, F(2)+5, F(4)+1, F(3)+2\}\\ &\qquad = \max\{-\infty, 6, 6, 4\} = 6,\\ &F(7) = \max\{F(2)+4, F(3)+5, F(5)+1, F(4)+2\} = \max\{5, 7, 5, 7\} = 7. \end{aligned}$$

Now we can find an optimal solution x^*. As $F(7) = \max\limits_{0\le q\le 7} F(q)$, we start with $\beta = 7$ and $x^* = (0,0,0,0)^T$. Since $F(7) = F(7-a_2)+c_2$, we set $x_2^* = 0+1 = 1$ and $\beta = 7-a_2 = 3$. Next, as $F(3) = F(3-a_4)+c_4$, we set $x_4^* = 0+1 = 1$ and $\beta = 3-a_4 = 0$.

Therefore, the point $x^* = (0,1,0,1)^T$ is a solution to the knapsack problem of Example 1.2. □

1.9.2 0,1-Knapsack

Now let us consider the 0,1-knapsack problem (1.28). For $k = 1,\dots,n$ and $\beta = 0,\dots,b$, let us define

$$F_k(\beta) \overset{\text{def}}{=} \max\left\{\sum_{j=1}^k c_j x_j \,:\, \sum_{j=1}^k a_j x_j = \beta,\ x_j \in \{0,1\},\ j=1,\dots,k\right\}.$$

According to this definition, the optimal objective value in (1.28) is $\max\limits_{0\le\beta\le b} F_n(\beta)$.

For $k = 1,\dots,n$, the following recurrence formula holds:

$$F_k(\beta) = \begin{cases} F_{k-1}(\beta), & \beta = 0,\dots,a_k-1,\\ \max\{F_{k-1}(\beta), F_{k-1}(\beta-a_k)+c_k\}, & \beta = a_k,\dots,b, \end{cases} \tag{1.30}$$

under the initial conditions

$$F_0(0) = 0, \quad F_0(\beta) = -\infty \text{ for } \beta = 1,\dots,b.$$

Since we need to compute $n\times b$ values $F_k(\beta)$, and computing each of these values, we perform one comparison and one assignment, therefore the calculations by Formula (1.30) can be performed in $O(nb)$ time using $O(nb)$ memory cells.

Having calculated all values $F_k(\beta)$, we can find an optimal solution, x^*, to (1.28) by executing the following *reverse step*.

```
Start with $\beta \in \arg\max_{0\le q\le b} F_n(q)$.
For $k = n,\dots,1$,
    set $x_k^* = 0$ if $F_k(\beta) = F_{k-1}(\beta)$, otherwise, set $x_k^* = 1$ and $\beta := \beta - a_k$.
```

Formula (1.30) is not the only possible one. Suppose that all values c_j are integer. Let $C \in \mathbb{Z}$ be an upper bound for the optimal objective value in (1.28). For $k = 1,\dots,n$ and $z = 0,\dots,C$, we define

$$G_k(z) \overset{\text{def}}{=} \min\left\{\sum_{j=1}^k a_j x_j \,:\, \sum_{j=1}^k c_j x_j = z,\ x_j \in \{0,1\},\ j=1,\dots,k\right\}.$$

For $k = 1,\dots,n$, the following recurrence formula holds:

$$G_k(z) = \begin{cases} G_{k-1}(z), & z = 0,\dots,c_k - 1, \\ \min\{G_{k-1}(z), G_{k-1}(z-c_k)+a_k\}, & z = c_k,\dots,C, \end{cases} \tag{1.31}$$

under the initial conditions

$$G_0(0) = 0, \quad G_0(z) = \infty \text{ for } z = 1,\dots,C.$$

The optimal objective value in (1.28) is equal to

$$\max\{z : G_n(z) \le b\}.$$

After calculating all values $G_k(z)$, we can find an optimal solution, x^*, to (1.28) by performing the following *reverse step*.

```
Start with $z \in \arg\max\{q : G_n(q) \le b\}$.
For $k = n,\dots,1$,
    if $G_k(z) = G_{k-1}(z)$, set $x_k^* = 0$, otherwise, set $x_k^* = 1$ and $z := z - c_k$.
```

The calculations by Formula (1.31) can be performed in $O(nC)$ time using $O(nC)$ memory cells.

Comparing the computational complexity of both recurrence formulas, (1.30) and (1.31), we can conclude that (1.30) should be used if $b < C$, otherwise we need to use (1.31).

Example 1.3 *We need to solve the problem*

$$\begin{aligned} 10x_1 + 7x_2 + 25x_3 + 24x_4 &\to \max, \\ 2x_1 + 1x_2 + 6x_3 + 5x_4 &\le 7, \\ x_1, x_2, x_3, x_4 &\in \{0,1\}. \end{aligned}$$

Solution. Obviously, the optimal objective value in this problem is greater than $b = 7$. Therefore, we use Formula (1.30). The calculations are presented in Table 1.2.

Table 1.2 Calculations by Formula (1.30) for Example 1.3

β	F_0	F_1	F_2	F_3	F_4
0	0	0	0	0	0
1	$-\infty$	$-\infty$	7	7	7
2	$-\infty$	10	10	10	10
3	$-\infty$	$-\infty$	17	17	17
4	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
5	$-\infty$	$-\infty$	$-\infty$	$-\infty$	24
6	$-\infty$	$-\infty$	$-\infty$	25	31
7	$-\infty$	$-\infty$	$-\infty$	32	34

The optimal objective value for our example problem is

$$\max_{0\le q\le 7} F_4(q) = F_4(7) = 34.$$

To find an optimal solution x^*, we need to execute the reverse step starting with $\beta = 7$:

$$\begin{aligned}
&F_4(7) = \max\{F_3(7), F_3(7-5)+24\} = \max\{32, 10+24\} = 34\\
&\qquad \Rightarrow x_4^* = 1 \text{ and } \beta = 7-5 = 2;\\
&F_3(2) = F_2(2) = 10\\
&\qquad \Rightarrow x_3^* = 0 \text{ and } \beta = 2;\\
&F_2(2) = \max\{F_1(2), F_1(2-1)+7\} = \max\{10, 0+7\} = 10\\
&\qquad \Rightarrow x_2^* = 0 \text{ and } \beta = 2;\\
&F_1(2) = \max\{F_0(2), F_0(2-2)+10\} = \max\{-\infty, 0+10\} = 10\\
&\qquad \Rightarrow x_1^* = 1 \text{ and } \beta = 0.
\end{aligned}$$

Hence, the point $x^* = (1,0,0,1)^T$ is an optimal solution to the $0,1$-knapsack problem from Example 1.3. □

Example 1.4 *We need to solve the problem*

$$\begin{aligned}
2x_1 + x_2 + 3x_3 + 4x_4 &\to \max,\\
35x_1 + 24x_2 + 69x_3 + 75x_4 &\le 100,\\
x_1, x_2, x_3, x_4 &\in \{0,1\}.
\end{aligned}$$

Solution. First, we estimate from above the optimal objective value. To do this, we solve the *relaxation LP* that is obtained from the original problem by allowing all binary variables to take values from the interval $[0,1]$. The algorithm for solving LPs with only one constraint is very simple (see Exercise 3.4).

1. First we sort the ratios c_j/a_j by non-increasing:

$$\frac{c_1}{a_1} = \frac{2}{35} \ge \frac{c_4}{a_4} = \frac{4}{75} \ge \frac{c_3}{a_3} = \frac{3}{69} \ge \frac{c_2}{a_2} = \frac{1}{24}.$$

2. Then we compute the solution $\hat{x}$:

$$\hat{x}_1 = 1,\ \hat{x}_4 = \frac{100-35}{75} = \frac{13}{15},\ \hat{x}_3 = \hat{x}_2 = 0.$$

As an upper bound, we take the number $C = \lfloor c^T\hat{x} \rfloor = \lfloor 2 + 4\cdot(13/15) \rfloor = 5$. Since $C = 5 < 100 = b$, we will use Formula (1.31). The calculations are presented in Table 1.3.

Since $G_4(5) = 99 < 100 = b$, the optimal objective value to our example problem is 5. To find an optimal solution x^*, we need to execute the reverse step starting from

Table 1.3 Computations by Formula (1.31) for Example 1.4

z	G_0	G_1	G_2	G_3	G_4
0	0	0	0	0	0
1	∞	∞	24	24	24
2	∞	35	35	35	35
3	∞	∞	59	59	59
4	∞	∞	∞	93	75
5	∞	∞	∞	104	99

$z = 5$:

$$\begin{aligned}
G_4(5) &= \min\{G_3(5), G_3(5-4)+75\} = \min\{104, 24+75\} = 99\\
&\Rightarrow x_4^* = 1 \text{ and } z = 5-4 = 1;\\
G_3(1) &= G_2(1) = 24\\
&\Rightarrow x_3^* = 0;\\
G_2(1) &= \max\{G_1(1), G_1(1-1)+24\} = \max\{\infty, 0+24\} = 24\\
&\Rightarrow x_2^* = 1 \text{ and } z = 0;\\
G_1(0) &= G_0(0) = 0\\
&\Rightarrow x_1^* = 0.
\end{aligned}$$

Thus, the point $x^* = (0,1,0,1)^T$ is an optimal solution to the 0,1-knapsack problem from Example 1.4. □

1.10 Notes

***Sect.* 1.2.** The general linear complementarity problem is **NP**-hard, but a number of special cases of this problem can be solved by simplex-like algorithms (see [95]). Nevertheless, the formulation of the linear complementarity problem as a MIP allows us to seek a solution with additional properties by assigning an appropriate objective function.

The Karush-Kuhn-Tucker (KKT) optimality conditions for the non-linear constrained optimization problems can be found in almost any book on non-linear programming, for example, in [29, 94].

***Sect.* 1.4.** Fundamental works on systems of linear inequalities and polyhedral theory are [66, 34]. Many aspects of polyhedral theory related to optimization are also presented in [122, 102].

***Sect.* 1.5.** The importance of strong **MIP** formulations was not recognized immediately. In the literature, say, thirty years ago, we can find **MIP** formulations that are considered weak (bad) today. The principles, on which strong formulations are build, are discussed in the reviews [141, 144].

***Sect.* 1.7.1.** A single-product lot-sizing model with unlimited production capacities, as well as a dynamic programming algorithm for its solution are described in [135]. Approximate and extended formulations for various lot-sizing problems are discussed in [133]. Inequalities (1.21), complementing Formulation (1.20) to the ideal one, were obtained in [17].

***Sect.* 1.8.** The **MIP** formulations are known for a wide variety of scheduling problems [114, 2]. The formulations with indexing by time were first introduced in [47]. The handbook [111] provides an up-to-date coverage of theoretical models and practical applications of modern scheduling theory.

***Sect.* 1.9.** The classic work on using dynamic programming to solve the knapsack problems is [56]. The idea to change the roles of the objective and the constraint, which allowed us to write Formula (1.31), was proposed in [75]. For the complexity of the knapsack problems, see, for example, [110, 109].

***Sect.* 1.11.** Theorem 1.4 was proved in [73] (see also more accessible sources [110, 122]). The result of Exercise 5.8 was obtained in [145].

1.11 Exercises

1.1. Describe the following sets by systems of linear inequalities, introducing, where necessary, binary variables:

a) $X_1 = \{x \in \mathbb{R} : \ |x| \ge a\}$;
b) $X_2 = P(A,b) \setminus \{\bar{x}\}$, where $\bar{x} \in P(A,b)$;
c) $X_3 = \left\{x \in \mathbb{R}^3 : \ x_3 = \min\{x_1,x_2\},\ 0 \le x_1,x_2 \le d\right\}$;
d) $X_4 = \left\{(x,y) \in \{0,1\}^n \times \{0,1\}^m : \ \sum_{j=1}^n x_j \ge a \Rightarrow \sum_{i=1}^m y_i \ge b\right\}$.

1.2. *The sum of the largest components of a vector*. Consider the set

$$X_{r,\alpha} \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : \ \sum_{i=1}^r x_{\pi^x(i)} \le \alpha\},$$

where r is an integer ($1 \le r \le n$), α is a real number, and the permutation π^x orders the components of a real n-vector x by non-increasing: $x_{\pi^x(1)} \ge x_{\pi^x(2)} \ge \cdots \ge x_{\pi^x(n)}$.

For example, if a vector x represents an investment portfolio (x_i is the share of asset i), then the requirement to *diversify* the investments so that not more than 80% of the budget can be invested in any 10% of assets is written as the inclusion $x \in X_{\lceil 0.1n \rceil, 0.8}$.

Prove that the set $X_{r,\alpha}$ is described by the system

$$x_{i_1} + \cdots + x_{i_r} \le \alpha, \quad 1 \le i_1 < i_2 < \cdots < i_r \le n,$$

of $n!/(r!(n-r)!)$ inequalities.

Let us also note that a compact extended formulation for the set $X_{r,\alpha}$ is given in Exercise 3.3.

1.3. Sudoku is a popular logic puzzle. An $n \times n$-matrix A is formed from an $m \times m$-matrix by replacing its elements for $m \times m$-matrices, which we call *blocks*. So, $n = m^2$. Some positions in A are filled with some numbers:

$$a_{ij} = \bar{a}_{ij} \quad \text{for} \quad (i,j) \in E \subseteq \{1,\dots,n\}^2.$$

It is necessary to fill all empty positions with numbers from 1 to n so that each row, column and block does not contain the same numbers. An example of such a puzzle for $m = 3$ is presented below, where a puzzle is on the left, and its solution is on the right.

		7	8		5	2		
8			6		4			5
	1			9			8	
4			2	8	9			7
5			7	6	1			2
	7			3			6	
3			1		6			4
		2	5		8	1		

6	4	**7**	**8**	1	**5**	**2**	3	9
8	9	3	**6**	2	**4**	7	1	**5**
2	**1**	5	3	**9**	7	4	**8**	6
4	3	1	**2**	**8**	**9**	6	5	**7**
7	2	6	4	5	3	8	9	1
5	8	9	**7**	**6**	**1**	3	4	**2**
1	**7**	4	9	**3**	2	5	**6**	8
3	5	8	**1**	7	**6**	9	2	**4**
9	6	**2**	**5**	4	**8**	**1**	7	3

Formulate an IP to find a solution to the Sudoku puzzle so that the sum of the diagonal elements of the resulting matrix A is maximum.

Hint. Use binary variables x_{ijk} with $x_{ijk} = 1$ if k is written into position (i,j).

1.4. Show that the IP

$$\max\{c^T x : \; Ax \le b, \; x \in \{0,1\}^n\}$$

is equivalent to the following quadratic programming problem

$$\max\left\{c^T x - M x^T(\mathbf{e} - x) : \; Ax \le b, \; x \in [0,1]^n\right\},$$

where M is a sufficiently large number. Estimate the value of M for given integer-valued A, b and c?

1.5. Consider the *quadratic knapsack problem*

$$\sum_{j=1}^{n} c_j x_j + \sum_{j=2}^{n} \sum_{i=1}^{j-1} c_{ij} x_i x_j \to \min,$$

$$\sum_{j=1}^{n} a_j x_j \ge b, \quad x \in \{0,1\}^n.$$

Introducing binary variables y_{ij} to represent the products $x_i x_j$, formulate this problem as an IP.

1.6. The *binary classification problem* is among the most important problems in machine learning. We are given a set $\{x^1,\dots,x^k\}$ of "*positive*" points from $\mathbb{R}^n$, and a set $\{y^1,\dots,\tilde{x}^l\}$ of "*negative*" points also from $\mathbb{R}^n$. Ideally, we would like to find a hyperplane $H(a,1)$, which is called a *linear classifier*, that separates positive and negative points:

$$\begin{aligned} a^T x^i &\le 1, \quad i=1,\dots,k,\\ a^T y^j &> 1, \quad j=1,\dots,l. \end{aligned} \tag{1.32}$$

In most practical cases this is impossible, and therefore we are looking for a hyperplane that minimizes some "classification error". Intuitively, the most natural measure for the classification error is the number of points, both positive and negative, that are on the "wrong" side of the hyperplane $H(a,1)$. With this measure, we need to find a vector $a \in \mathbb{R}^n$ that violates the minimum number of inequalities in (1.32)[6]. Formulate this problem as a MIP.

1.7. Consider a *bimatrix game* in which the gains of the first and second players are given by two matrices $A=[a_{ij}]_{m\times n}$ and $B=[b_{ij}]_{m\times n}$. A pair of vectors (*mixed strategies*) $(p,q)\in\mathbb{R}^m\times\mathbb{R}^n$ is a *Nash equilibrium* if it satisfies the following constraints:

$$\begin{aligned} \sum_{j=1}^n a_{ij}q_j &\le \sum_{i=1}^m\sum_{j=1}^n a_{ij}p_iq_j, \quad i=1,\dots,m,\\ \sum_{i=1}^m a_{ij}p_i &\le \sum_{i=1}^m\sum_{j=1}^n a_{ij}p_iq_j, \quad j=1,\dots,n,\\ \sum_{i=1}^m p_i = 1,&\quad \sum_{j=1}^n q_j = 1,\\ p_i &\ge 0, \quad i=1,\dots,m,\\ q_j &\ge 0, \quad j=1,\dots,n. \end{aligned} \tag{1.33}$$

a) Prove that for any solution (p,q) to (1.33) the following *complementary slackness conditions* are valid:

$$\begin{aligned} p_i\left(\sum_{i=1}^m\sum_{j=1}^n a_{ij}p_iq_j - \sum_{j=1}^n a_{ij}q_j\right) &= 0, \quad i=1,\dots,m,\\ q_j\left(\sum_{i=1}^m\sum_{j=1}^n a_{ij}p_iq_j - \sum_{i=1}^m b_{ij}p_i\right) &= 0, \quad j=1,\dots,n. \end{aligned}$$

b) Defining

$$U_1 = \max_{\substack{1\le i\le m,\\ 1\le j\le n}} a_{ij} - \min_{\substack{1\le i\le m,\\ 1\le j\le n}} a_{ij} \quad\text{and}\quad U_2 = \max_{\substack{1\le i\le m,\\ 1\le j\le n}} b_{ij} - \min_{\substack{1\le i\le m,\\ 1\le j\le n}} b_{ij},$$

we formulate the following MIP:

[6] Since this optimization problem is **NP**-hard, in practice other measures of the classification error are used, which are less adequate but easier to optimize.

$$
\begin{aligned}
w &\to \max, && && (1.34a)\\
p_i + x_i &\le 1, && i = 1,\dots,m, && (1.34b)\\
-U_1 x_i \le \sum_{j=1}^{n} a_{ij} q_j - v_1 &\le 0, && i = 1,\dots,m, && (1.34c)\\
q_j + y_j &\le 1, && j = 1,\dots,n, && (1.34d)\\
-U_2 y_j \le \sum_{i=1}^{m} a_{ij} p_i - v_2 &\le 0, && j = 1,\dots,n, && (1.34e)\\
\sum_{i=1}^{m} p_i = 1, \quad \sum_{j=1}^{n} q_j &= 1, && && (1.34f)\\
p_i \ge 0,\ x_i &\in \{0,1\}, && i = 1,\dots,m, && (1.34g)\\
q_j \ge 0,\ y_j &\in \{0,1\}, && j = 1,\dots,n, && (1.34h)\\
w \le v_1,\ w &\le v_2. && && (1.34i)
\end{aligned}
$$

Let $(p^*, q^*, x^*, y^*, v_1^*, v_2^*, w^*)$ be an optimal solution to (1.34). Using the statement of item a), prove that (p^*, q^*) is a Nash equilibrium such that the minimum, $w^* = \min\{v_1^*, v_2^*\}$, of the gains of both players is maximum.

1.8. Solve the following knapsack problems:

a)
$$
\begin{aligned}
&15x_1 + 19x_2 + 24x_3 + 27x_4 \to \max,\\
&2x_1 + 3x_2 + 4x_3 + 5x_4 \le 7,\\
&x_1, x_2, x_3, x_4 \in \mathbb{Z}_+;
\end{aligned}
$$

b)
$$
\begin{aligned}
&12x_1 + 5x_2 + 17x_3 + 9x_4 + 7x_5 \to \max,\\
&3x_1 + 5x_2 + 2x_3 + 4x_4 + 6x_5 \le 9,\\
&x_1, x_2, x_3, x_4, x_5 \in \{0,1\};
\end{aligned}
$$

c)
$$
\begin{aligned}
&x_1 + 2x_2 + 3x_3 + 2x_4 + 3x_5 \to \max,\\
&12x_1 + 9x_2 + 15x_3 + 11x_4 + 6x_5 \le 29,\\
&x_1, x_2, x_3, x_4, x_5 \in \{0,1\}.
\end{aligned}
$$

1.9. Consider the single-product lot-sizing problem from Sect. 1.7.1. Let $H(t)$ denote the cost of the optimal solution to the subproblem in which the number of periods in the planning horizon is t ($0 \le t \le T$). Let us introduce the notations

$$
w_t \overset{\text{def}}{=} c_t + \sum_{\tau=t}^{T} h_\tau, \quad d_{\tau t} \overset{\text{def}}{=} \sum_{k=\tau}^{t} d_k, \quad \hat{H}(t) \overset{\text{def}}{=} H(t) + \sum_{\tau=1}^{t} h_\tau d_{1\tau}.
$$

a) Prove the validity of the following recurrence formula:

$$
\begin{aligned}
&\hat{H}(0) = 0,\\
&\hat{H}(t) = \min_{1 \le \tau \le t} \{\hat{H}(\tau - 1) + f_\tau + w_\tau d_{\tau t}\}, \quad t = 1,\dots,T.
\end{aligned}
\qquad (1.35)
$$

b) Using (1.35), solve the example of the lot-sizing problem with the following parameters: $T = 4$, $d = (2,4,4,2)^T$, $c = (3,2,2,3)^T$, $h = (1,2,1,1)^T$ and $f = (10,20,16,10)^T$.

1.10. *Aggregation of systems of linear equations with integer coefficients.* Consider the system of equations

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1,2, \tag{1.36}$$

with non-negative integer coefficients a_{ij}, b_1, and b_2. Prove the following theorem.

Theorem 1.2. *Let λ_1 and λ_2 be coprime integers, λ_1 does not divide b_2, and λ_2 does not divide b_1. If $\lambda_1 > b_2 - a^{min}$ and $\lambda_2 > b_1 - a^{\min}$, where $a^{\min}$ is the minimum nonzero number among the coefficients a_{ij}, then the set of non-negative integer solutions to* (1.36) *coincides with the set of non-negative integer solutions to their linear combination*

$$\sum_{j=1}^{n} (\lambda_1 a_{1j} + \lambda_2 a_{2j})x_j = \lambda_1 b_1 + \lambda_2 b_2.$$

1.11. A matrix is said to be *totally unimodular* if all its minors (determinants of square submatrices) are 0 or ± 1. Prove the following statement.

Theorem 1.3. *The polyhedron $P(A,b)$ is integer (all its vertices are integer vectors) for any integer vector b if and only if the matrix A is totally unimodular.*

1.12. Justify the following criterion.

Theorem 1.4. *An integer matrix with elements $0, \pm 1$ is totally unimodular if each of its columns contains at most two non-zero elements, and its rows can be partitioned into two subsets in such a way that:* (1) *if both non-zero elements of some column are of the same sign, then these elements are in the rows from different subsets;* (2) *if non-zero elements of some column have different signs, then these elements are in the rows of the same subset.*

Chapter 2
MIP Models

The most reliable way to learn how to formulate complex practical problems as MIPs is to study those MIP models which now are regarded as classical. In this chapter, we consider examples of **MIP** formulations for various practical applications. Not all of our formulations are the strongest because in some cases to elaborate a strong formulation, we must conduct an in-depth analysis of the structure of the problem being solved in order to take into account its specific features. Nevertheless, each of the models studied here can be a starting point for developing a solution tool for the corresponding practical application. Let us also note that many practical **MIP** applications are also discussed in the other chapters. In addition, a lot of applications are presented in the exercises.

2.1 Set Packing, Partitioning, and Covering Problems

Given a finite set S and a family $\mathscr{E} = \{S_1, \dots, S_n\}$ of its subsets. Often the pair $H = (S, \mathscr{E})$ is called a *hypergraph*. By analogy with graphs, the elements of the set S are called *vertices*, and the subsets in $\mathscr{E}$ are called *hyperedges*. A subset of hyperedges $\mathscr{J} \in \mathscr{E}$ is called a *packing* if every vertex from S belongs to at most one hyperedge from $\mathscr{J}$. If every vertex of S belongs to exactly one hyperedge from $\mathscr{J}$, then $\mathscr{J}$ is called a *partition*. Finally, if every vertex in S belongs to at least one hyperedge from $\mathscr{J}$, then $\mathscr{J}$ is called a *cover*.

Let us assign to each hyperedge S_j its *cost* c_j. The *set packing problem* is to find a packing with the maximum total cost of its hyperedges. The *set covering* (resp., *set partitioning*) problem is to find a cover (resp., partition) with the minimum total cost of its hyperedges. Note that if H is a graph (all sets S_j contain two elements), then the set packing problem is known as the *weighted matching problem*.

To simplify the exposition, we assume that $S = \{1, \dots, m\}$. The *incidence matrix* A of the hypergraph $H = (S, \mathscr{E})$ has m rows, n columns, and its element a_{ij} equals 1 if $i \in S_j$, and $a_{ij} = 0$ otherwise. For $j = 1, \dots, n$, we introduce a binary variable x_j that takes the value of 1 if hyperedge j is in the packing, partitioning or covering.

Now all three problems are very simply formulated as IPs:

$$
\begin{aligned}
&\text{(set packing problem)} && \max\{c^T x : Ax \le \mathbf{e},\ x \in \{0,1\}^n\}, && (2.1)\\
&\text{(set partitioning problem)} && \min\{c^T x : Ax = \mathbf{e},\ x \in \{0,1\}^n\}, && (2.2)\\
&\text{(set covering problem)} && \min\{c^T x : Ax \ge \mathbf{e},\ x \in \{0,1\}^n\}. && (2.3)
\end{aligned}
$$

Here **e** is the vector of size m with all components equal to 1.

Forming a Team of Performers

There are n candidates to participate in some project consisting of m jobs. Candidate j can execute a subset of jobs, S_j, and wants to get c_j for this. A *team* is a set of workers that, for each project job, has at least one member able to perform this job. We need to form a team from existing candidates spending the minimum amount of money.

Obviously, this is a set partitioning problem that is given on the hypergraph $H = (S, \{S_1, \dots, S_n\})$ with $S = \{1, \dots, m\}$.

Crew Scheduling

A number of flight legs are given (taken from the time table of some company). A *leg* is a flight taking off from its departure airport at some time and landing later at its destination airport. The problem is to partition these legs into *routes*, and then assign exactly one crew to each route. A *route* is a sequence of flight legs such that the destination of one leg is the departure point of the next, and the destination of the last leg is the departure point of the first leg. For example, there might be the following short rout:

- leg 1: from Paris to Berlin departing at 9:20 and arriving at 10:50;
- leg 2: from Berlin to Rom departing at 12:30 and arriving at 14:00;
- leg 3: from Rom to Paris departing at 16:30 and arriving at 18:00.

A schedule is good if the crews spend flying as much of the elapsed time as it is possible subject to the safety regulation and contract terms are satisfied. These terms regulate the maximum number of hours a pilot can fly in a day, the maximum number of days before returning to the base, and minimum overnight rest times. The *cost* of a schedule depends on several of its attributes, and the wasted times of all crews is the main one.

In practice, the crew scheduling problem is solved in two stages. Let all the legs be numbered from 1 to m. First, a collection of n reasonable routes $S_j \subset \{1, \dots, m\}$ (that meet all the constraints mentioned above) are selected, and the cost c_j of each route j is calculated. This route-selection problem is far from being trivial, but it is not our main interest here.

Given the set of potential routes, $\mathscr{E} = \{S_1, \dots, S_n\}$, the second stage is to identify a subset of them, $\mathscr{J} \subseteq E$, so that each leg is covered by exactly one route, and the total cost of all routes in $\mathscr{J}$ is minimum. Of course, this second stage problem is a *set partitioning problem*

Combinatorial Auctions

At an auction, m objects are put up for sale. Suppose that in a certain round of trades, the auctioneer received n bids from the buyers. The difference between the combinatorial auction and the usual one is that any buyer in its bid is allowed to value not only a single object, but also any group of objects. Therefore, each bid j is described by a pair (S_j, c_j), where S_j is a subset of objects for which the buyer who submitted the bid agrees to pay c_j. Naturally, no object can be sold twice. Therefore, two bids, (S_{j_1}, c_{j_1}) and (S_{j_2}, c_{j_2}) such that $S_{j_1} \cap S_{j_2} \neq \emptyset$ cannot be satisfied simultaneously. The auctioneer must decide which of the bids to satisfy so that the seller's profit is maximum.

Clearly, here we have a set packing problem.

2.2 Service Facility Location

The problem of facility location is critical to a company's eventual success. Companie's decisions on locating its services are guided by variety of criteria. Locating service centers close to the customers is especially important because this enables faster delivery goods and services.

Given a set of customer locations $N = \{1, \dots, n\}$ with b_j customers at location $j \in N$, a set of potential sites $M = \{1, \dots, m\}$ for locating service centers. For each $i \in M$, we know a fixed cost f_i of locating a center at site i, a capacity u_i of the center at site i, and a cost c_{ij} of serving customer j from site i during some planning horizon. The *facility location problem* (*FLP*) is to decide where to locate service centers so that to minimize the total cost of locating centers and serving customers.

Choosing the following decision variables

- $y_i = 1$ if a service center is located at site i, and $y_i = 0$ otherwise,
- x_{ij}: number of customers at location j served from the service center established at site i,

we formulate the FLP as follows:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i y_i \to \min, \tag{2.4a}$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1,\dots,n, \tag{2.4b}$$

$$\sum_{j=1}^{n} x_{ij} \le u_i y_i, \quad i = 1,\dots,m. \tag{2.4c}$$

$$x_{ij} \le \min\{u_i, b_j\} y_i, \quad i = 1,\dots,m, \ \ j = 1,\dots,n, \tag{2.4d}$$

$$y_i \in \{0,1\}, \quad i = 1,\dots,m, \tag{2.4e}$$

$$x_{ij} \in \mathbb{Z}_+, \quad i = 1,\dots,m, \ \ j = 1,\dots,n. \tag{2.4f}$$

Objective (2.4a) is to minimize the total cost of locating centers and serving customers. Let us note, that if all $c_{ij} = 0$ and all $f_i = 1$, then the objective is to minimize the number of service centers needed to serve all customers. Equations (2.4b) insure that each customer is served. Inequalities (2.4c) reflect the capacity limitations: if a service center is established at site i ($y_i = 1$), then at most u_i customers can be served from this site i. Inequalities (2.4d), which are logically implied by (2.4c) and therefore are redundant, are introduced to strengthen the formulation.

2.3 Portfolio Management: Index Fund

Portfolio optimization is the problem of investing a given capital in a number of securities in order to maximize the return with a limited "risk."

There are two orthogonal portfolio management strategies: active and passive. When an *active strategy* is used, the methods of analysis and forecasting are used to achieve the required level of efficiency. In contrast, any *passive strategy* advises not to rely on forecasts, but to diversify investments to minimize the risk. The goal is to create and maintain a portfolio that reflects changes in a broad market population (or market index). Such a portfolio is called an *index fund*.

Formation of the index fund begins with the choice of a broad market index as an approximation of the entire market, for example, the Standard and Poor's List of 500 stocks (S&P 500). In its pure form, the index approach consists in buying all assets in the same proportions as they are present in the index. In practice, this is difficult or even impossible to accomplish. Therefore, the market index is aggregated by a relatively small *index fund* of shares of not more than q types, where q is substantially smaller than the number of all types of shares in the index. Such an approach does not necessarily lead to the formation of an optimal portfolio relative to the return/risk ratio.

The input data for the model are given by an $n \times n$-matrix $[\rho_{ij}]$, whose element ρ_{ij} estimates the "similarity" between the shares i and j (ρ_{ij} is smaller for more similar shares). For example, we can estimate the coefficients ρ_{ij} by the returns of shares for T previous periods. Let $R_i(t)$ be the return (per one enclosed dollar) of share i in period t. Then we can calculate

$$\rho_{ij} = \sum_{t=1}^{T} p^{T-t} \left(R_i(t) - R_j(t)\right)^2,$$

where $p \in (0,1]$ is a discount factor, which is introduced to increase the significance of recent periods in comparison with the early ones.

It is necessary to determine what stocks and in what proportions should be present in the portfolio. The first step in solving this problem is to select shares for including in the index fund. We introduce the following variables:

- $y_i = 1$ if share i is in the index fund, and $y_i = 0$ otherwise;
- $x_{ij} = 1$ if share i represent share j in the index fund, and $x_{ij} = 0$ otherwise.

Now the problem of forming the index fund is written as follows:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \rho_{ij} x_{ij} \to \min, \tag{2.5a}$$

$$\sum_{i=1}^{n} y_i \le q, \tag{2.5b}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1,\dots,n, \tag{2.5c}$$

$$x_{ij} \le y_i, \quad i,j = 1,\dots,n, \tag{2.5d}$$

$$x_{ij} \in \{0,1\}, \quad i,j = 1,\dots,n, \tag{2.5e}$$

$$y_i \in \{0,1\}, \quad i = 1,\dots,n. \tag{2.5f}$$

Objective (2.5a) is to build an index fund that most accurately represents the market index. Inequality (2.5b) do not allow the index fund to contain more than q shares. Equalities (2.5c) require that each share of the market index be represented by a share from the index fund. Inequalities (2.5d) do not allow the shares that are not in the index fund to represent other shares.

This may seem strange, but it turns out that (2.5) is a special case of (2.4), which is an IP formulation of the problem of locating service centers, when $m = n$, $c_{ij} = \rho_{ij}$, $b_i = n$, $f_i = 0$. It worth noting that such a coincidence of the formulations of seemingly completely different problems are encountered quite often.

When (2.5) is solved and the set of shares in the index fund, $I = \{i : y_i = 1\}$, is known, we can proceed with the formation of the portfolio. First we calculate the weights $w_i \overset{\text{def}}{=} \sum_{j=1}^{n} V_j x_{ij}$ of all shares $i \in I$, where V_j is the total value of all shares of type j in the market index. In other words, w_i is the total market value of all shares represented by share i in the index fund. Therefore, the proportion of capital invested in each share i from the index fund ($i \in I$) must be equal to $w_i / (\sum_{k \in I} w_k)$.

2.4 Multiproduct Lot-Sizing

We need to work out an aggregate production plan for n different products processed on a number of machines of m types for a planning horizon that extends over T periods.

Inputs parameters:

- l_t: duration (length) of period t;
- m_{it}: number of machines of type i available in period t;
- f_{it}: fixed cost of producing on one machine of type i in period t;
- $T_i^{\min}, T_i^{\max}$: minimum and maximum working time of one machine of type i;
- c_{jt}: per unit production cost of product j in period t;
- h_{jt}: inventory holding cost per unit of product j in period t;
- d_{jt}: demand for product j in period t;
- ρ_{jk}: number of units of product j used for producing one unit of product k;
- τ_{ij}: per unit production time of product j on machine of type i;
- s_j^i: initial stock of product j at the beginning of the planning horizon.
- s_j^f: final stock of product j at the end of the planning horizon.

A *production plan* specifies on which machines and in which quantities each product is produced in each of the periods. The goal is to determine a production plan that can be implemented on existing equipment and the total production and inventory cost is minimum.

Let us introduce the following variables:

- x_{jt}: amount of product j produced in period t;
- s_{jt}: amount of product j in stock at the end of period t;
- y_{it}: number of machines of type i working in period t.

Now we formulate the problem as follows:

$$\sum_{t=1}^{T}\sum_{j=1}^{n}(h_{jt}s_{jt}+c_{jt}x_{jt})+\sum_{t=1}^{T}\sum_{i=1}^{m}f_{it}y_{it}\to \min, \tag{2.6a}$$

$$s_j^i+x_{j1}=d_{j1}+s_{j1}+\sum_{k=1}^{n}\rho_{jk}x_{k,1},\quad j=1,\dots,n, \tag{2.6b}$$

$$s_{j,t-1}+x_{jt}=d_{jt}+s_{jt}+\sum_{k=1}^{n}\rho_{jk}x_{k,t},\quad j=1,\dots,n,\ t=2,\dots,T, \tag{2.6c}$$

$$\sum_{j=1}^{n}\tau_{ij}x_{jt}\le l_t y_{it},\quad i=1,\dots,m,\ t=1,\dots,T, \tag{2.6d}$$

$$s_{jT}=s_j^f,\quad j=1,\dots,n, \tag{2.6e}$$

$$0\le s_{jt}\le u_j,\ x_{jt}\ge 0,\quad j=1,\dots,n,\ t=1,\dots,T, \tag{2.6f}$$

$$0\le y_{it}\le m_{it},\ y_{it}\in\mathbb{Z},\quad i=1,\dots,m,\ t=1,\dots,T. \tag{2.6g}$$

Objective (2.6a) is to minimize the total production and inventory expenses. Each of the balance equations in (2.6b) and (2.6c) joins two adjacent periods for each of the products: the stock in period $t-1$ plus the amount of product produced in period t equals the demand in period t plus the amount of product used when producing other products, and plus the stock in period t. Inequalities (2.6d) require that the working times of all machines be withing given limits; besides, if machine i does not work in period t ($y_{it} = 0$), then no product is produced by this machine (all $x_{jt} = 0$).

2.5 Balancing Assembly Lines

Assembly lines are special product-layout production systems that are typical for the industrial production of high quantity standardized commodities. An assembly line consists of a number of work stations arranged along a conveyor belt. The work pieces are consecutively launched down the conveyor belt and are moved from one station to the next. At each station, one or several operations, which are necessary to manufacture the product, are performed. The operations in an assembly process usually are interdependent, i.e., there may be precedence relations that must be enforced. The problem of distributing the operations among the stations with respect to some objective function is called the *assembly line balancing problem* (ALBP). We will consider here the simple assembly line balancing problem which is the core of many other ALBPs.

The manufacturing of some product consists of a set of *operations* $\mathscr{O} = \{1,\dots,n\}$. We denote by t_o the processing time of operation $o \in \mathscr{O}$. The precedence relations between the operations are represented by a digraph $G = (\mathscr{O}, E)$, where $(o_1, o_2) \in E$ means that operation o_1 must be finished before operation o_2 starts. Suppose that the demand for the product is such that the assembly line must have a cycle time C, which means that the running time of each station on one product unit must not exceed C.

The *simple assembly line balancing problem* (SALBP) is to decide what is the minimum number of stations that is enough for the line running with the given cycle time to fulfill all the operations in an order consistent with the precedence relations.

An example of SALBP is presented in Fig. 2.1. Here we have $n = 11$ operations that correspond to the vertices of the digraph representing precedence relations between these operations. The numbers over vertices are the processing times of operations.

To formulate SALBP as an IP, we need to know an upper bound, m, on the number of needed stations. In particular, we can set m to be the the number of station in a solution build by one of the heuristics developed for solving SALBPs.

For example, let us consider the heuristic that assigns operations, respecting precedence relations, first to Station 1, then to Station 2, and so on until all the operations are assigned to the stations. If we apply this heuristic to the example of Fig. 2.1 when the cycling time is $C = 45$, we get the following assignment:

- operations 1 and 2 are accomplished by station 1,

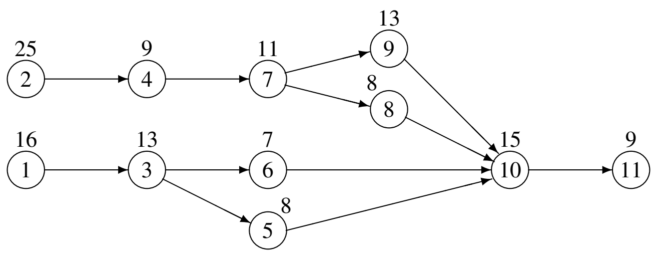

Fig. 2.1 Example of SALBP

- operations 3, 4, 5, and 6 by station 2,
- operations 7, 8, and 9 by station 3,
- operations 10 and 11 by station 4.

So in this example we can set $m = 4$. Let us also note that this heuristic solution is not optimal as there exists an assignment that uses only three stations:

- operations 1, 3, 5, and 6 are accomplished by station 1,
- operations 2, 4, and 7 by station 2,
- operations 8, 9, 10, and 11 by station 3.

To write an IP, we introduce the following variables:

- $y_s = 1$ if station s is open (in use), $y_s = 0$ otherwise;
- x_{so} if operation o is assigned to station s, $x_{so} = 0$ otherwise;
- $z_o = s$ if operation o is assigned to station s.

With these variables the formulation of SALBP is as follows:

$$\sum_{s=1}^{m} y_s \to \min \tag{2.7a}$$

$$\sum_{s=1}^{m} x_{so} = 1, \quad o = 1,\dots,n, \tag{2.7b}$$

$$\sum_{o=1}^{n} t_o x_{so} \le C y_s, \quad s = 1,\dots,m, \tag{2.7c}$$

$$\sum_{s=1}^{m} s x_{so} = z_o, \quad o = 1,\dots,n, \tag{2.7d}$$

$$z_{o_1} \le z_{o_2}, \quad (o_1,o_2) \in E, \tag{2.7e}$$

$$y_{s-1} \ge y_s, \quad s = 2,\dots,m, \tag{2.7f}$$

$$x_{so} \le y_s, \quad o = 1,\dots,n, \; s = 1,\dots,m, \tag{2.7g}$$

$$x_{so} \in \{0,1\}, \quad s = 1,\dots,m, \; o = 1,\dots,n, \tag{2.7h}$$

$$y_s \in \{0,1\}, \quad s = 1,\dots,m, \tag{2.7i}$$

$$z_o \in \mathbb{R}_+, \quad o = 1,\dots,n. \tag{2.7j}$$

Objective (2.7a) is to minimize the number of open stations. Equations (2.7b) require that each operation be assign to exactly one station. Inequalities (2.7c) reflect the capacity restrictions inducing that the total running time of each open stations does not exceed the cycle time. Equations (2.7d) establish the relation between the assignment variables, binary x and integer z. Each precedence relation constraint in (2.7e) require that, for a pair $(o_1, o_2) \in E$ of related operations, operation o_1 be assigned to the same or an earlier station than operation o_2; this guaranties that operation o_1 is finished before operation o_2 starts. Inequalities (2.7f) and (2.7g) ensure that earlier stations are opened first.

2.6 Electricity Generation Planning

The *unit commitment problem* is to develop an hourly (or half-hourly) electricity production schedule spanning some planning horizon (a day or a week) so as to decide which generators will be producing and at what levels. The very essence of this problem is to appropriately balance of using generators with different capacities: it is cheaper to produce electricity on more powerful generators while less powerful and smaller generators take less time to switch on or off in case of necessity.

Let T be the number of periods in the planning horizon. Period 1 follows period T. We know the *demand* d_t for electricity in each period t. It is required that, in each period, the total capacity of all active generators be at least q times more than the demand (q is a level of reliability).

Let n be the number of generators, and let generator i have the following characteristics:

- l_i, u_i: minimum and maximum per period production levels (capacities);
- r_i^1, r_i^2: ramping parameters (when a generator is on in two successive periods, its output cannot decrease by more than r_i^1, and increase by more than r_i^2);
- g_i: start-up cost (if a generator is off in some period, it costs g_i to start it in the next period);
- f_i, p_i: fixed and variable costs (if in some period a generator is producing at level v, it costs $f_i + p_i v$).

With a natural choice of variables

- $x_{it} = 1$ if generator i produces in period t, and $x_{it} = 0$ otherwise,
- $z_{it} = 1$ if generator i is switched on in period t, and $z_{it} = 0$ otherwise,
- y_{it}: amount of electricity produced by generator i in period t,

we write down the following formulation:

$$\sum_{i=1}^{n} \sum_{t=1}^{T} (g_i z_{it} + f_i x_{it} + p_i y_{it}) \to \min \tag{2.8a}$$

$$\sum_{i=1}^{n} y_{it} = d_t, \quad t = 1, \dots, T, \tag{2.8b}$$

$$\sum_{i=1}^{n} u_i x_{it} \ge q\, d_t, \quad t = 1,\dots,T, \tag{2.8c}$$

$$l_i x_{it} \le y_{it} \le u_i x_{it}, \quad i = 1,\dots,n,\ t = 1,\dots,T, \tag{2.8d}$$

$$-r_i^1 \le y_{it} - y_{i,((t-2+T) \bmod T)+1} \le r_i^2, \quad i = 1,\dots,n,\ t = 1,\dots,T, \tag{2.8e}$$

$$x_{it} - x_{i,((t-2+T) \bmod T)+1} \le z_{it}, \quad i = 1,\dots,n;\ t = 1,\dots,T, \tag{2.8f}$$

$$z_{it} \le x_{it},, \quad i = 1,\dots,n,\ t = 1,\dots,T, \tag{2.8g}$$

$$x_{it}, z_{it} \in \{0,1\}, \quad i = 1,\dots,n,\ t = 1,\dots,T, \tag{2.8h}$$

$$y_{it} \in \mathbb{R}_+, \quad i = 1,\dots,n,\ t = 1,\dots,T. \tag{2.8i}$$

Objective (2.8a) is to minimize the total (for all n generators over all T periods) cost of producing electricity plus the sum of start-up costs. Equations (2.8b) guarantee that, for each period, the total amount of electricity produced by all working generators meets the demand at that period. Inequalities (2.8c) require that, in any period, the total capacity of all working generators be at least q times more than the demand at that period. The lower and upper bounds in (2.8d) impose the capacity restrictions for each generator in each period; simultaneously, these constrains ensure that non-working generators do not produce electricity. Two-sided inequalities (2.8e) guarantee that generators cannot increase (ramp up) or decrease (ramp down) their outputs by more than the values of their ramping parameters. Let us note that period $((t-2+T) \bmod T)+1$ is immediately followed by period t. Inequalities (2.8f) and (2.8g) reflect the fact that any generator is working in a given period only if it has been switched on in this period or it was working in the preceding period.

2.7 Designing Telecommunication Networks

Let us denote by V the set of nodes (subscribers) of a telecommunications network to be designed, and let E denote the set of all pairs of nodes that exchange information. The pair $G = (V,E)$ is called a *demand graph*. Note that this logical graph is not a representation of the structure of the designed network. For each demand $e = (i, j) \in E$, we know the intensity of communications, d_e, between nodes i and j. In the designed telecommunications network, one can set an unlimited number of identical *rings* with a capacity (bandwidth) of U. Every demand must be assigned entirely to one ring. The sum of the intensities of all demands assigned to one ring must not exceed the capacity of the ring. If a demand $e = (i, j)$ is assigned to a ring, then two *multiplexers*, each of cost α, must be installed on this ring, one at node i and the other at node j. Each multiplexer has S slots for installing *special cards*. There are T different types of cards, a card of type t has a bandwidth of c_t and it costs β_t. Our goal is to assign demands to the rings in such a way that all the above requirements are met, and the total cost of all installed multiplexers and cards is minimum.

Let R be an upper bound on the number of rings to be established. We use the following variables:

- $y_{er} = 1$ if demand e is assigned to ring r, and $y_{er} = 0$ otherwise;
- $x_{ir} = 1$ if a multiplexer is installed at node i on ring r, and $x_{ir} = 0$ otherwise;
- z_{irt}: number of cards of type t used at node i on ring r.

Now we can write the model:

$$\alpha \sum_{r=1}^{R} \sum_{i\in V} x_{ir} + \sum_{r=1}^{R} \sum_{i\in V} \sum_{t=1}^{T} \beta_t z_{irt} \to \min, \tag{2.9a}$$

$$\sum_{r=1}^{R} y_{er} = 1, \quad e \in E, \tag{2.9b}$$

$$\sum_{e\in E} d_e y_{er} \le U, \quad r = 1,\dots,R, \tag{2.9c}$$

$$\sum_{e\in E(i,V)} d_e y_{er} \le \sum_{t=1}^{T} c_t z_{irt}, \quad i \in V,\ r = 1,\dots,R, \tag{2.9d}$$

$$\sum_{t=1}^{T} z_{irt} \le S x_{ir}, \quad i \in V,\ r = 1,\dots,R, \tag{2.9e}$$

$$y_{er} \le x_{ir},\ y_{er} \le x_{jr}, \quad e = (i,j) \in E,\ r = 1,\dots,R, \tag{2.9f}$$

$$x_{ir} \in \{0,1\}, \quad i \in V,\ r = 1,\dots,R, \tag{2.9g}$$

$$y_{er} \in \{0,1\}, \quad e \in E,\ r = 1,\dots,R, \tag{2.9h}$$

$$z_{irt} \in \mathbb{Z}_+, \quad i \in V,\ r = 1,\dots,R,\ t = 1,\dots,T. \tag{2.9i}$$

Objective (2.9a) is to minimize the total cost of installed multiplexers and cards. Equations (2.9b) assign every demand to exactly one ring. Inequalities (2.9c) guarantee that the bandwidth of any ring is not exceeded. Similar inequalities (2.9d) require that the bandwidth of any multiplexer, which is the sum of the capacities of all cards installed on the multiplexer, be not exceeded. Here $E(i,V)$ stands for the set of edges $e = (i,j) \in E$ incident to node i. Inequalities (2.9e) do not allow us to insert more cards into any multiplexer than there are slots, and they also prevent the insertion of cards into slots of not used multiplexers (if $x_{ir} = 0$, then $z_{irt} = 0$ for all $t = 1,\dots,T$). Finally, each pair of inequalities in (2.9f) implies that, if a demand $e = (i,j)$ is assigned to ring r ($y_{er} = 1$), then the multiplexers must be installed on this ring at both nodes i and j ($x_{ir} = x_{jr} = 1$).

2.8 Placement of Logic Elements on the Surface of a Crystal

We have a set $\mathscr{C}$ of *logic elements* (*gates*) that implement basic boolean functions. Geometrically the surface of a crystal of size $k \times q$ can be considered as a rectangular area with the uniform rectangular grid on it (like a sheet of a school notebook in a box). The cells on the crystal are numbered from 1 to kq. For simplicity of exposition we will assume that each gate can be placed into one cell of the crystal, i.e., each gate can be considered as a unit square[1]. The gates are connected to each other by *signal circuits* (hereinafter simply "*circuits*"). Any circuit $n \in \mathscr{N}$ is given as a subset $\mathscr{N}(n) \subseteq \mathscr{C}$ of gates, which it connects. Our goal is to place the set of gates, $\mathscr{C}$, into a subset of cells, $\mathscr{I} \subseteq \{1,\dots,kq\}$ ($|\mathscr{I}| \geq |\mathscr{C}|$), so that to minimize the sum of the semiperimeters of the minimal rectangles bounding the circuits.

The problem of placing a set of gates on the surface of a crystal is very complex and usually it is solved in two stages. At the stage of *global placement*, the crystal surface is divided into a set of disjoint rectangles, each of which is assigned a subset of gates (without specifying specific positions). At the stage of *detailed* (*local*) *placement*, for each of these rectangles, it is necessary to solve the placement problem with an indication of exact positions of all the gates. Here we will consider only the problem of detailed placement.

Input data:

- $\mathscr{C}$: set of gates;
- $\mathscr{I}$: subset of crystal cells;
- $\mathscr{N}$: set of circuits;
- $\mathscr{N}(n) \subseteq \mathscr{C}$: set of gates that are connected by circuit $n \in \mathscr{N}$;
- a_i: distance from the center of cell $i \in \mathscr{I}$ to the crystal left side;
- b_i: distance from the center of cell $i \in \mathscr{I}$ to the crystal top side.

Let us introduce the following variables:

- $z_{ci} = 1$ if gate $c \in \mathscr{C}$ is placed into cell i, and $z_{ci} = 0$ otherwise;
- $x_n^{\min}$, $x_n^{\max}$, $y_n^{\min}$, $y_n^{\max}$: the pairs $(x_n^{min}, y_n^{\min})$ and $(x_n^{\max}, y_n^{\max})$ are the coordinates of the left-bottom and right-top corners of the minimal rectangle that contains all gates $c \in \mathscr{N}(n)$.

In these variables our **IP** formulation is as follows:

$$\sum_{n\in\mathscr{N}} (x_n^{\max} - x_n^{\min} + y_n^{\max} - y_n^{\min}) \to \min, \tag{2.10a}$$

$$\sum_{c\in\mathscr{C}} z_{ci} \leq 1, \quad i \in \mathscr{I}, \tag{2.10b}$$

$$\sum_{i\in\mathscr{I}} z_{ci} = 1, \quad c \in \mathscr{C}, \tag{2.10c}$$

$$\sum_{i\in\mathscr{I}} a_i z_{ci} \geq x_n^{\min}, \quad c \in \mathscr{N}(n),\ n \in \mathscr{N}, \tag{2.10d}$$

$$\sum_{i\in\mathscr{I}} a_i z_{ci} \leq x_n^{\max}, \quad c \in \mathscr{N}(n),\ n \in \mathscr{N}, \tag{2.10e}$$

$$\sum_{i\in\mathscr{I}} b_i z_{ci} \geq y_n^{\min}, \quad c \in \mathscr{N}(n),\ n \in \mathscr{N}, \tag{2.10f}$$

[1] In rare cases, when the gate occupies several cells, it can be represented as several gates that are unit squares which should be placed in an adjacent manner.

$$\sum_{i\in\mathscr{I}} b_i z_{ci} \ge y_n^{\max}, \quad c \in \mathscr{N}(n),\ n \in \mathscr{N}, \tag{2.10g}$$

$$z_{ci} \in \{0,1\}, \quad c \in \mathscr{C},\ i \in \mathscr{I}, \tag{2.10h}$$

$$x_n^{\min}, x_n^{\max}, y_n^{\min}, y_n^{\max} \ge 0, \quad n \in \mathscr{N}. \tag{2.10i}$$

Objective (2.10a) is to minimize the sum of the semiperimeters of the rectangles that frame the circuits. Inequalities (2.10b) do not allow us to place two gates into the same cell. Equations (2.10c) ensure that each gate will be placed exactly into one cell. For each circuit $n \in \mathscr{N}$, four inequalities in (2.10d)–(2.10g) determine the coordinates of the left-bottom, $(x_n^{min}, y_n^{\min})$, and the right-top, $(x_n^{\max}, y_n^{\max})$, corners of the rectangle containing all gates of circuit n.

2.9 Assigning Aircrafts to Flights

The flight schedule of even an average airline is huge and usually is stored in a database in which the information is presented in a form similar to that shown in Table. 2.1. In this particular example, we see that there is a flight from XYZ airport

Table 2.1 Flight Schedule

Flight	Departure		Arrival		Type airplane	Cost
	airport	time	airport	time		
201	XYZ	6:55	ZYX	9:45	734	8570
201	XYZ	6:55	ZYX	9:45	757	12085
201	XYZ	6:55	ZYX	9:45	767	13095
202	ZYX	11:05	XYZ	14:00	734	8570
202	ZYX	11:05	ZYZ	14:00	757	12085
202	ZYX	11:05	XYZ	14:00	767	13095

to ZYX airport departing at 6:55 and arriving at 9:45. This flight can be performed by Boeing-734, Boeing-757, or Boeing-767 aircrafts with the flight costs of \$8570, \$12085, or \$13095, respectively.

Any *feasible assignment* of aircraft to flights obeys the following requirements:

- no more aircrafts of each type can be used than there is in stock;
- aircrafts arriving at the airport must either fly away or remain on the ground;
- aircrafts must depart from the airports where they landed earlier.

The *problem of assigning aircrafts to flights* is to find a feasible assignment of minimum cost.

Input data:

- n: number of flights;

- m: number of aircraft types;
- q_i: number of planes of type i;
- T_j: set of aircrafts suitable for flight j;
- c_{ij}: cost of flight j performed by a plane of type i;
- l: number of airports;
- r_k: number of events at airport k; any *event* e corresponds to the time moment $t(k,e)$ when at least one company‘s plane is landing or taking off at the airport; we assume that the events are numbered from 0 to $r_k - 1$, and they follow according to the order of their occurrence starting from event 0;
- *list of all n flights*, in which flight j is described by the four-tuple $(a_j^d, e_j^d; a_j^a, e_j^a)$ which means that the flight departures from airport a_j^d at time $t(a_j^d, e_j^d)$ when event e_j^d occurs, and later arrives at destination airport a_j^a at time $t(a_j^a, e_j^a)$ when event e_j^a occurs.

Let us introduce two families of variables:

- $x_{ij} = 1$ if a plane of type i is assigned to flight j, and $x_{ij} = 0$ otherwise;
- f_{ike}: number of planes of type i at airport k at time $t(k,e)$.

In these variable our problem is formulated as follows:

$$\sum_{j=1}^{n} \sum_{i \in T_j} c_{ij} x_{ij} \to \min, \tag{2.11a}$$

$$\sum_{i \in T_j} x_{ij} = 1, \quad j = 1, \dots, n, \tag{2.11b}$$

$$f_{i,k,(e+1) \bmod r_k} = f_{ike} + \sum_{j:\, i \in T_j,\, a_j^a = k,\, e_j^a = e} x_{ij} - \sum_{j:\, i \in T_j,\, a_j^d = k,\, e_j^d = e} x_{ij}, \tag{2.11c}$$

$$i = 1, \dots, m, \;\; k = 1, \dots, l, \;\; e = 0, \dots, r_k - 1,$$

$$\sum_{\substack{j:\, i \in T_j, \\ t(a_j^d, e_j^d) > t(a_j^a, e_j^a)}} x_{ij} + \sum_{k=1}^{l} f_{i,k,r_k - 1} \le q_i, \quad i = 1, \dots, m, \tag{2.11d}$$

$$x_{ij} \in \{0,1\}, \quad j = 1, \dots, n, \;\; i \in T_j, \tag{2.11e}$$

$$f_{ike} \in \mathbb{Z}_+, \quad i = 1, \dots, m, \;\; k = 1, \dots, l, \;\; e = 0, \dots, r_k - 1. \tag{2.11f}$$

Objective (2.11a) of this IP is to minimize the total cost of all flights. Equations (2.11b) ensure that each flight will be assigned to exactly one aircraft type. According to the balance equations (2.11c), for each airport k and every event e that occurs there, the number of aircraft of any type i at the airport until the next event is equal to their number at time $t(k,e)$ plus the number of aircrafts landing at time $t(k,e)$ and minus the number of aircrafts taking off at time $t(k,e)$. Since (2.11c) is valid, the number of aircraft of each type remains constant during a day. Inequalities (2.11d) require that at midnight the total number of aircrafts of each type in the air and on the ground be not more than their number.

2.10 Optimizing the Performance of a Hybrid Car

A *hybrid car* among many other things has an internal combustion engine, a motor/generator connected to a battery, and a braking system. We will consider an extremely simple *parallel car* model in which the motor/generator and the internal combustion engine are directly connected to the driving wheels. The internal combustion engine transfers mechanical energy to the wheels, and the braking system takes away this energy from the wheels turning it into heat. The motor/generator can work as an electric motor using the energy of the battery and feeding it to the wheels, or as a generator when it consumes mechanical energy from the wheels or directly from the internal combustion engine, and converts this mechanical energy into electricity charging the battery. When the generator consumes mechanical energy of the wheels and charges the battery, it is called a *regenerative brake*.

A diagram illustrating energy flows in a hybrid car is presented in Fig. 2.2. Here the arrows indicate positive directions of energy transmission. The engine power p^{eng} is always positive and is transmitted in the direction from the engine to the wheels. The power of the braking system, p^{br}, is always non-negative, and it is positive when the car brakes. The energy consumption of the wheels, p^{req}, is positive when it is spent on driving the car (when the car accelerates, goes uphill or evenly moves along the road), and p^{req} is negative when the car brakes or goes down the hill. We consider the motor/generator as two devices operating in turn. When the motor is running, the energy p^{m} is fed from it to the wheels, and when the generator is running, it receives the energy p^{g} from the wheels.

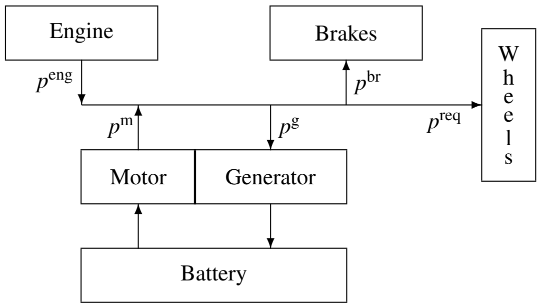

Fig. 2.2 Energy flows in a hybrid car

The car is tested on a track with fixed characteristics. The speed of the car on each section of the route is predefined. Therefore, the time of passing the track is also known and is equal to T seconds. We will build a discrete-time model with T time intervals, each lasting one second. Because the profile of the route is known and the speed on all route sections is also set, then it is possible to calculate the power P_t^{req} required for feeding to the wheels. We also know the following parameters:

- $P_{\max}^{\text{eng}}$: maximum engine power;

- $P^{\text{g}}_{\max}$: maximum generator power;
- $P^{\text{m}}_{\max}$: maximum motor power;
- $E^{\text{batt}}_{\max}$: maximum battery energy (charge);
- η: fraction of energy lost when converting mechanical energy into electricity, and then into battery charge and vice versa;
- t_1, t_2: for any time interval of continuous t_2 seconds, the electric motor should run no more than t_1 seconds.

If the engine runs at a power of p, then per unit of time it consumes $F(p)$ fuel units. We assume that $F : \mathbb{R}_+ \to \mathbb{R}_+$ is an increasing convex function.

For $t = 1, \dots, T$, we introduce the following variables:

- p^{eng}_t: engine power in period t;
- p^{m}_t: motor power in period t;
- p^{g}_t: generator power in period t;
- p^{br}_t: braking system power in period t;
- E_t: battery charge (energy) in period t;
- $y_t = 1$ if the motor/generator works as a motor, and $y_t = 0$ if the motor/generator works as a generator.

We can determine an optimal operation mode of a hybrid car solving the following program:

$$\sum_{t=1}^{T} F(p^{\text{eng}}_t) \to \min, \tag{2.12a}$$

$$p^{\text{eng}}_t + p^{\text{m}}_t - p^{\text{g}}_t - p^{\text{br}}_t = P^{\text{req}}_t, \quad t = 1, \dots, T, \tag{2.12b}$$

$$E_t - (1+\eta)p^{\text{m}}_t + (1-\eta)p^{\text{g}}_t = E_{t+1}, \quad t = 1, \dots, T, \tag{2.12c}$$

$$\sum_{\tau=t-t_2+1}^{t} y_\tau \le t_1, \quad t = t_2, \dots, T, \tag{2.12d}$$

$$E_{T+1} \ge E_1, \tag{2.12e}$$

$$0 \le E_t \le E^{\text{batt}}_{\max}, \quad t = 1, \dots, T, \tag{2.12f}$$

$$0 \le p^{\text{eng}}_t \le P^{\text{eng}}_{\max}, \quad t = 1, \dots, T, \tag{2.12g}$$

$$0 \le p^{\text{m}}_t \le P^{\text{m}}_{\max} y_t, \quad t = 1, \dots, T, \tag{2.12h}$$

$$0 \le p^{\text{g}}_t \le P^{\text{g}}_{\max}, \quad t = 1, \dots, T, \tag{2.12i}$$

$$p^{\text{br}}_t \ge 0, \quad t = 1, \dots, T, \tag{2.12j}$$

$$y_t \in \{0,1\}, \quad t = 1, \dots, T. \tag{2.12k}$$

Objective (2.12a) is to minimize fuel consumption. Equations (2.12b) ensure that at any time the right amount of energy will be supplied to the wheels. Each balance equation in (2.12c) relates the battery charges for two neighboring periods. Inequalities (2.12d) do not allow the electric motor to run more than t_1 seconds during any continuous time interval of t_2 seconds. Inequality (2.12e) is introduced for a fair comparison of a hybrid car with a non-hybrid one: at the finish, the battery charge should be no more than the battery charge at the start.

Since the objective function is nonlinear, (2.12) is not a MIP. But we can approximate the convex function F with a piecewise linear function, and then, using the method described in Sect. 1.1.4, we can represent this piecewise linear function as linear by introducing new variables and constraints.

2.11 Short-Term Financial Management

Short-term financial management is one of the tasks of the accounting of a large firm. If the management of finances is inefficient, the incomes will be received by the banks, in which the funds are stored, and not by their owner. Free money should also work. Profit can be significantly increased if the firm works actively on the securities market.

Suppose that the planning horizon is divided into T periods of usually varying duration, and let period $T+1$ represents the end of the horizon. There are n types of securities on the market. The company's portfolio at the beginning of the planning horizon is represented by a vector s of size n, where $s_i \ge 0$ is the number of securities of type i in the portfolio. The costs of selling and buying a security of type i in period t are c_{it}^s and c_{it}^b, respectively. Note that the values of c_{it}^s and c_{it}^b can be less than or greater than the nominal value of a security of type i.

Short-term financial sources (other than selling securities from the portfolio) are represented by k open credit lines. The maximum amount of borrowing along line l is u_l. Loans can be obtained at the beginning of each period, and they are to be returned after the completion of the planning horizon. To assess the effectiveness of all borrowing, the costs, f_{lt}, are calculated, where f_{lt} is the monthly rate of interest along credit line l multiplied by the time (in months) remaining from the beginning of period t to the end of the planning horizon.

Exogenous (external) cash flows are given by the values $d_t, t = 1, \dots, T$. If $d_t > 0$ (resp., $d_t < 0$), then the firm will receive d_t (resp., pay $-d_t$) at the beginning of period t. We assume that the cash reserve at the beginning of the planning horizon is taken into account when calculating d_1. For each period $t = 1, \dots, T$, the minimum cash requirement, q_t, is also specified.

It is necessary to balance the cash budget in such a way to maximize the firm ”wealth” (cash plus the sale values of all securities minus the total amount of all borrowings, taking into account interest) at the end of the planning horizon.

Let us introduce the following variables:

- x_{it}: number of securities of type i at the end of period t;
- x_{it}^s: number of securities of type i sold in period t;
- x_{it}^b: number of securities of type i purchased in period t;
- y_t: amount of cash at the end of period t;
- z_{lt}: amount of money borrowed from credit line l in period t.

In these variables our problem is formulated as follows:

$$y_T + \sum_{i=1}^{n} c^s_{i,T+1} x_{i,T} - \sum_{t=1}^{T}\sum_{l=1}^{k}(1+f_{lt})z_{lt} \to \max, \tag{2.13a}$$

$$d_1 + \sum_{i=1}^{n} c^s_{i1} x^s_{i1} + \sum_{l=1}^{k} z_{l1} = y_1 + \sum_{i=1}^{n} c^b_{i1} x^b_{i1}, \tag{2.13b}$$

$$d_t + y_{t-1} + \sum_{i=1}^{n} c^s_{it} x^s_{it} + \sum_{l=1}^{k} z_{lt} = y_t + \sum_{i=1}^{n} c^b_{it} x^b_{it}, \quad t = 2,\dots,T, \tag{2.13c}$$

$$s_i + x^b_{i1} - x^s_{i1} = x_{i1}, \quad i = 1,\dots,n, \tag{2.13d}$$

$$x_{i,t-1} + x^b_{it} - x^s_{it} = x_{it}, \quad i = 1,\dots,n,\ t = 2,\dots,T, \tag{2.13e}$$

$$\sum_{t=1}^{T} z_{lt} \le u_l, \quad l = 1,\dots,k, \tag{2.13f}$$

$$y_t \ge q_t, \quad t = 0,\dots,T, \tag{2.13g}$$

$$x_{it},\ x^s_{it},\ x^b_{it} \in \mathbb{Z}_+, \quad i = 1,\dots,n,\ t = 1,\dots,T, \tag{2.13h}$$

$$y_t \in \mathbb{R}_+, \quad t = 1,\dots,T, \tag{2.13i}$$

$$z_{lt} \in \mathbb{R}_+, \quad l = 1,\dots,k,\ t = 1,\dots,T. \tag{2.13j}$$

Objective (2.13a) is to maximize the firm "wealth" at the end of the planning horizon. Equations (2.13c) and (2.13e) balance the budgets of, respectively, cash and securities in periods $2,\dots,T$. The similar balance constraints, (2.13b) and (2.13d), are applied only for period 1. Inequalities (2.13f) ensure that the total borrowing from any credit line does not exceed the volume of this credit line. Inequalities (2.13g) require that the necessary minimum of cash be available in any period.

2.12 Planning Treatment of Cancerous Tumors

In the past, oncologists used the apparatus with two or three comparatively large beams (10×10 centimeters in cross section) of a fixed orientation. The number of beams in modern devices is constantly increasing, and each beam is divided into several smaller *rays*, the size and intensity of which can vary within certain limits. The intensity of a ray at the points along its path (and to a much lesser extent at the points closest to it) is measured in *doses*. The unit of the dose is *Gray* (Gy) defined to be the amount of energy per unit of mass received from the beam in the ionization process around a given point. Treatment of cancerous tumors with such devices is known as *intensive modulated radiation therapy* (IMRT).

Treatment with the IMRT method begins with elaborating a treatment plan. The aim of the planning is to guarantee that, as a result of the treatment, the cancer cells receive the required dose, and the dose received by healthy tissues must be safe (there must be no irreversible damage). Planning begins with the definition of the critical area around the tumor. The critical region is covered by a three-dimensional uniform rectangular lattice. Let us assume that the lattice points are numbered from 1 to l, and let $L = \{1,\dots,l\}$. Let $T \subset L$ denote the set of lattice points inside the tumor. The rest of the critical area is broken (by type of tissue) into subdomains. Let K denote the number of such subdomains, and $H_k \subset L$ be the set of lattice points inside subdomain k. Table 2.2 presents the parameters that characterize an example of planning treatment for prostate cancer. Here $|T| = 2438$ points belong to the tumor, $|H_1| = 1566$ points belong to the region immediately surrounding the tumor ("collar"), additional unclassified $|H_2| = 1569$ points (other) lie near the tumor. The bladder ($|H_3| = 1292$ points) and the rectum ($|H_4| = 1250$ points) require special attention.

Table 2.2 Example of planning prostate cancer treatment

Tissue	Number of points	Limiting dose b_k (in cGy)	Threshold d_k (in cGy)	% to threshold	Uni-formity
Collar	1566	15000			
Other	1569	15000			
Bladder	1292	10000	8000	80 %	
Rectum	1250	10000	7500	75 %	
Tumor	2438				0.9

Suppose that each beam can be directed under one of n possible angles. A *sample* is a particular way of breaking the beam into rays, indicating their dimensions and intensities. Let P_j denote the set of possible samples for a beam directed at an angle j. Using special software, one can calculate the dose a_{ijp} obtained at point i from a beam of unit intensity directed at angle j if sample p $(p \in P_j)$ is used. It is assumed that the dose at any point is approximately equal to the sum of the doses received from all the beams. It is necessary to determine the intensities x_{jp} of all the beams in order to satisfy a number of conditions for doses at points of the critical regions. We will introduce these conditions when we explain the constraints of the following model:

$$t \to \max, \tag{2.14a}$$

$$t \le \sum_{j=1}^{n} \sum_{p \in P_j} a_{ijp} x_{jp} \le \frac{1}{\alpha} t, \quad i \in T, \tag{2.14b}$$

$$\sum_{j=1}^{n} \sum_{p \in P_j} a_{ijp} x_{jp} \ge s, \quad i \in T', \tag{2.14c}$$

$$\sum_{j=1}^{n} \sum_{p \in P_j} a_{ijp} x_{jp} \le b_k, \quad i \in H_k,\ k = 1,\dots,K, \tag{2.14d}$$

$$\sum_{j=1}^{n} \sum_{p \in P_j} a_{ijp} x_{jp} \le d_k + (b_k - d_k) y_i, \quad i \in H_k,\ k \in \hat{K}, \tag{2.14e}$$

$$\sum_{i \in H_k} y_i \le \lfloor (1 - f_k) |H_k| \rfloor, \quad k \in \hat{K}, \tag{2.14f}$$

$$x_{jp} \ge 0, \quad p \in P_j,\ j = 1, \dots, n, \tag{2.14g}$$

$$y_i \in \{0, 1\}, \quad i \in \cup_{k \in \hat{K}} H_k. \tag{2.14h}$$

In this model, the variable t represents the minimum dose at the tumor points. The goal (2.14a) is to maximize this minimum dose. The *uniformity coefficient*, α, in (2.14b) sets the lower bound for the ratio of the minimum and maximum doses at the tumor points. In the example from Table 2.2 this coefficient is equal to 0.9.

In order not to miss the microscopic areas of the affected tissues in the immediate vicinity of the tumor, a set of points, T', surrounding the tumor is selected, and it is required that each of these points receive at least the minimum dose of s. This condition is expressed by (2.14c).

Inequalities (2.14d) limit the doses received by healthy tissues. Here b_k is the limiting dose for subdomain k. In the example from Table 2.2 the dose limits are set for the ”collar”, bladder, rectum and other tissues.

To prevent irreversible damage to certain tissue types $k \in \hat{K} \subseteq \{1, \dots, K\}$, it is required that the fraction of points with a dose exceeding a given threshold d_k ($d_k < b_k$) be not greater than f_k ($0 < f_k < 1$). In the example from Table 2.2 such proportions are given for the bladder and rectum. In our model, this condition is expressed by Ineqs. (2.14e) and (2.14f), where, for each point $i \in H_k$ ($k \in \hat{K}$), we introduce an auxiliary binary variable y_i taking the value of 1 only if the dose at point i exceeds the threshold d_k.

Concluding the discussion of MIP (2.14), it should be noted that we cannot solve such MIPs using standard software. Since each of the sets P_j consists of a very large number of samples, the number of variables x_{jp} is usually huge. To solve such problems, it is necessary to develop a branch-and-price algorithm that is based on the technique of column generation, which is discussed in Chap. 7.

2.13 Project Scheduling

Let us consider a project that consists of n jobs. There are q^r *renewable*[2] (*nonperishable*) resources, with, respectively, R_i^r units of resource i available per unit of time. There are also q^n *nonrenewable*[3] (*perishable*) resources, with, respectively, R_i^n

[2] Renewable resources are available in the same quantities in any period. Manpower, machines, storage spaces are renewable resources.

[3] In contrast to a renewable resource, which consumption is limited in each period, overall consumption of a nonrenewable resource is limited for the entire project. Money, energy, and raw materials are nonrenewable resources.

units of resource i available for the entire project. It is assumed that all resources are available when the project starts.

The jobs can be processed in different modes. Any job cannot be interrupted, thus, if a job once started in some mode, it has to be completed in the same mode. If job j is processed in mode m ($m = 1,\dots,M_j$), then

- it takes p_j^m units of time to process the job,
- ρ^r_{jmi} units of renewable resource i ($i = 1,\dots,q^r$) are used in each period when job j is processed,
- and ρ^n_{jmi} units of nonrenewable resource i ($i = 1,\dots,q^n$) are totally consumed.

Precedence relations between jobs are given by an acyclic digraph $G = (\mathscr{J},\mathscr{R})$ defined on the set of jobs $\mathscr{J} \overset{\text{def}}{=} \{1,\dots,n\}$: for any arc $(j_1, j_2) \in \mathscr{R}$, job j_2 cannot start until job j_1 is finished.

A project schedule specifies when each job starts and in which mode it is processed. The goal is to find a schedule with the minimum makespan that is defined to be the completion time of the last job.

Let us assume that we know an upper bound H on the optimal makespan value. We can take as H the makespan of a schedule produced by one of numerous project scheduling heuristics. The planning horizon is divided into H periods numbered from 1 to H, and period t starts at time $t-1$ and ends at time t.

To tighten our formulation, we can estimate (say, by the *critical path method*) the earliest, es_j, and latest, ls_j, start times of all jobs j.

First, we define the family of decision binary variables:

- $x_{jmt} = 1$ if job j is processed in mode m and starts in period t (at time $t-1$), and $x_{jmt} = 0$ otherwise.

For modeling purposes, we also need the following families of auxiliary variables:

- T: schedule makespan;
- $d_j \in \mathbb{R}$: duration of job j (d_j depends on the mode in which job j is processed);
- $s_j \in \mathbb{R}$: start time of job j.

In these variables the model is written as follows:

$$T \to \min, \tag{2.15a}$$

$$\sum_{m=1}^{M_j} \sum_{t=es_j}^{ls_j} x_{jmt} = 1, \quad j = 1,\dots,n, \tag{2.15b}$$

$$s_j = \sum_{m=1}^{M_j} \sum_{t=es_j}^{ls_j} t x_{jmt}, \quad j = 1,\dots,n, \tag{2.15c}$$

$$d_j = \sum_{m=1}^{M_j} \sum_{t=es_j}^{ls_j} p_j^m x_{jmt}, \quad j = 1,\dots,n, \tag{2.15d}$$

$$T \ge s_j + d_j, \quad j = 1,\dots,n, \tag{2.15e}$$

$$\sum_{j=1}^{n}\sum_{m=1}^{M_j}\sum_{t=\max(\tau-p_j^m+1,es_j)}^{\min(\tau,ls_j)} \rho^r_{jmi} x_{jmt} \le R^r_i, \quad i=1,\dots,q^r,\ \tau=1,\dots,H, \tag{2.15f}$$

$$\sum_{j=1}^{n}\sum_{m=1}^{M_j}\sum_{t=es_j}^{ls_j} \rho^n_{jmi} x_{jmt} \le R^n_i, \quad i=1,\dots,q^n, \tag{2.15g}$$

$$s_{j_2} - s_{j_1} \ge d_{j_2}, \quad (j_1,j_2)\in\mathscr{R}, \tag{2.15h}$$

$$x_{jmt}\in\{0,1\}, \quad t=es_j,\dots,ls_j,\ m=1,\dots,M_j,\ j=1,\dots,n, \tag{2.15i}$$

$$d_j, s_j \in \mathbb{R}, \quad j=1,\dots,n. \tag{2.15j}$$

Objective (2.15a) is to minimize the makespan. Equations (2.15b) ensure that each job is processed only in one mode, and it starts only once. For the schedule given by the values of x_{jmt} variables, (2.15c) and (2.15d) calculate the start times, s_j, and durations, d_j, of all jobs j. Inequalities (2.15e) imply that, being minimized, T is the completion time of the job finished the last, i.e., T is the makespan. The limitations on the renewable resources are imposed by (2.15f): for each period τ and each resource i, the total consumption of this resource by all jobs that are processed in this period cannot exceed the given limit R^r_i. Due to (2.15b), each inequality in (2.15g) restricts the usage of a particular nonrenewable resource. The precedence relations between jobs are given by (2.15h).

2.14 Short-Term Scheduling in Chemical Industry

It is easier to start with an example. Two products, 1 and 2, are produced from three different raw products A,B, and C according to the following technological process.

- *Heating*. Heat A for 1 h.
- *Reaction 1*. Mix 50% feed B and 50% feed C and let them for 2 h to form intermediate BC.
- *Reaction 2*. Mix 40% hot A and 60% intermediate BC and let them react for 2 h to form intermediate AB (60%) and product 1 (40%).
- *Reaction 3*. Mix 20% feed C and 80% intermediate AB and let them react for 1 h to form impure E.
- *Separation*. Distill impure E to separate pure product 2 (90%, after 1 h) and pure intermediate AB (10% after 2 h). Discard the small amount of residue remaining at the end of the distillation. Recycle the intermediate AB.

The above technological process is represented by the *State-Task-Network* (*STN*) shown in Fig. 2.3.

The following processing equipment and storage capacities are available.

- **Equipment:**
  - *Heater*: capacity 100 kg, suitable for task 1;

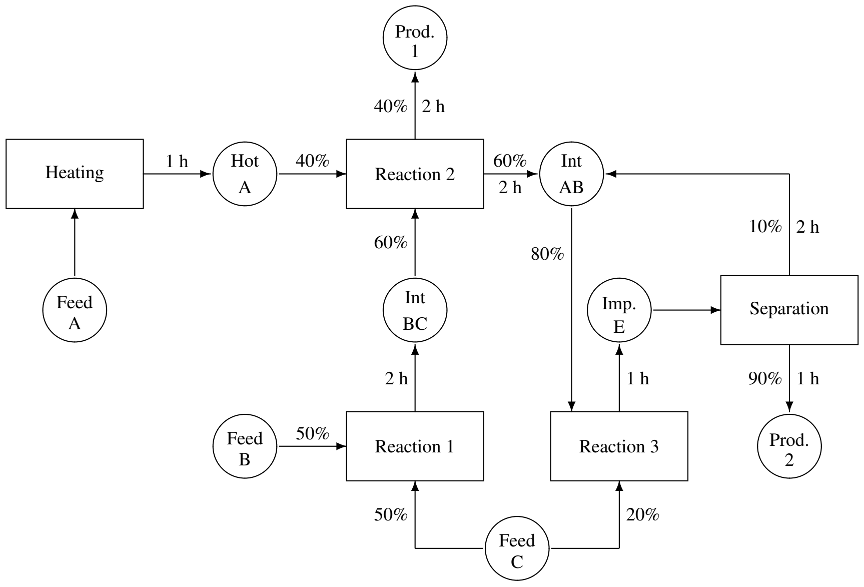

Fig. 2.3 State-task network for example process

- *Reactor 1*: capacity 80 kg, suitable for tasks 2,3,4;
- *Reactor 2*: capacity 50 kg, suitable for tasks 2,3,4;
- *Still*: capacity 200 kg, suitable for task 5.

- **Storage capacity for**
  - *feeds A,B,C*: unlimited;
  - *hot A*: 100 kg;
  - *intermediate AB*: 200 kg;
  - *intermediate BC*: 150 kg;
  - *intermediate E*: 100 kg;
  - *products 1,2*: unlimited.

A number of parameters are associated with the tasks and the states defining the STN, and with the available equipment items.

- **Task i is defined by:**

U_i: set of units capable of performing task i;
S_i^{in}: set of states that feed task i;
S_i^{out}: set of states to which task i outputs its products;
ρ_{is}^{in}: proportion of input of task i from state $s \in S_i^{\text{in}}$, $\sum_{s\in S_i^{\text{in}}} \rho_{is}^{\text{in}} = 1$;
ρ_{is}^{out}: proportion of output of task i to state $s \in S_i^{\text{out}}$, $\sum_{s\in S_i^{\text{out}}} \rho_{is}^{\text{out}} = 1$;
p_{is}: processing time of output of task i sent to state $s \in S_i^{\text{out}}$;

d_i: duration of task i, $d_i \overset{\text{def}}{=} \max_{s \in S_i^{\text{out}}} p_{is}$.

- **State s is defined by:**

  T_s^{out}: set of tasks receiving the product from state s;
  T_s^{in}: set of tasks producing the product for state s;
  z_s^0: initial stock in state s;
  u_s: storage capacity for the product in state s;
  c_s: unit cost (price) of the product in state s;
  h_s: cost of storing a product unit in state s.

- **Unit j is characterized by:**

  I_j: set of tasks that can be performed by unit j;
  $V_{ij}^{\max}, V_{ij}^{\min}$: respectively, maximum and minimum loading of unit j when used for performing task i.

Let n, q, m denote, respectively, the number of tasks, states, and units. The scheduling problem for batch processing system is stated as follows.

Given: STN of a batch process and all the information associated with it, as well as a planning horizon of interest.
Determine: schedule for each equipment unit (i.e. which task, if any, the unit performs at any time during the planning horizon), as well as product flows inside the STN.
Goal: maximize the total cost of the products produced by the end of the planning horizon minus the total storage cost during the planning horizon.

MIP Formulation

Our formulation is based on the discrete representation of time. The planning horizon is divided into a number of periods of equal duration. We number these periods from 1 to T, and assume that period t starts at time $t-1$ and ends at time t. Events of any type — such as the start or end of processing any batch of a task, changes in the availability of equipment units and etc. — are only happen at the beginning or end of the periods.

Preemptive operations are not allowed and materials are transferred instantaneously from states to tasks and vice versa.

We introduce the following variables:

- $x_{ijt} = 1$ if unit j starts processing task i at the beginning of period t, and $x_{ijt} = 0$ otherwise;
- y_{ijt}: total amount of products (batch size) used to start a batch of task i in unit j at the beginning of period t;
- z_{st}: amount of material stored in state s at the beginning of period t.

Now the **MIP** model is written as follows:

$$\sum_{s=1}^{q} c_s z_{s,T} - \sum_{s=1}^{q}\sum_{t=1}^{T} h_s z_{st} \to \max, \tag{2.16a}$$

$$\sum_{i\in I_j}\sum_{\tau=\max\{0,t-d_i+1\}}^{\min\{t,T-d_i\}} x_{ij\tau} \le 1, \quad j=1,\dots,m,\ t=1,\dots,T, \tag{2.16b}$$

$$V_{ij}^{\min} x_{ijt} \le y_{ijt} \le V_{ij}^{\min} x_{ijt}, \quad j=1,\dots,m,\ i\in I_j,\ t=1,\dots,T, \tag{2.16c}$$

$$0 \le z_{st} \le u_s, \quad s=1,\dots,q,\ t=1,\dots,T, \tag{2.16d}$$

$$z_s^0 = z_{s1} + \sum_{i\in T_s^{\text{in}}} \rho_{is}^{\text{in}} \sum_{j\in U_i} y_{ij1}, \quad s=1,\dots,q, \tag{2.16e}$$

$$z_{s,t-1} + \sum_{i\in T_s^{\text{out}}:\, t>p_{is}} \rho_{is}^{\text{out}} \sum_{j\in U_i} y_{ij,t-p_{is}} = z_{st} + \sum_{i\in T_s^{\text{in}}} \rho_{is}^{\text{in}} \sum_{j\in U_i} y_{ijt},$$
$$s=1,\dots,q,\ t=1,\dots,T, \tag{2.16f}$$

$$x_{ijt}=0, \quad t>T-d_i,\ j=1,\dots,m,\ i\in I_j, \tag{2.16g}$$

$$x_{ijt}\in\{0,1\},\ y_{ijt}\in\mathbb{R}_+, \quad j=1,\dots,m,\ i\in I_j,\ t=1,\dots,T, \tag{2.16h}$$

$$z_{st}\in\mathbb{R}_+, \quad s=1,\dots,q,\ t=1,\dots,T. \tag{2.16i}$$

Objective (2.16a) is to maximize the total profit that equals the total cost of the products in all states at the end of the planning horizon minus the expenses for storing products during the planning horizon. Inequalities (2.16b) ensure that at any time any unit cannot process more than one task. The variable bounds in (2.16c) restrict the batch size of any task to be within the minimum and maximum capacities of the unit performing the task. The stock limitations are imposed by Ineqs. (2.16d): the amount of product stored in any state s must not exceed the storage capacity for this state. The product balance relations from (2.16f) ensure that, for any state s in each period $t > 1$, the amount of product entering the state (the stock from the previous period plus the input from the tasks ending in period $t-1$) equals the amount of product leaving the state (the stock at the end of period t plus the amount of product consumed by the tasks that started in period t). Equations (2.16e) are specializations of the balance relations for period 1, which has no preceding period. In (2.16g) we set to zero the values of some variables to enforce all the tasks to finish within the planning horizon.

2.15 Multidimensional Orthogonal Packing

In an *m-dimensional* (*orthogonal*) *packing problem* we need to pack a number of small m-dimensional rectangular boxes (*items*) into a large m-dimensional rectangular box (*container*) so that no two items overlap, and all item edges are parallel to the container edges.

Two-dimensional ($m = 2$) packing problems arise in different industries, where steel, wood, glass, or textile materials are cut. In such cases these packing problems are also known as the *two-dimensional cutting stock problems*. The problem to optimize the layout of advertisements in a newspaper is also formulated as a two-dimensional packing problem. Three-dimensional ($m = 3$) packing problems — also known as the *container loading problems* — appear as important subproblems in logistics and supply chain applications.

Formally, we have a large m-dimensional rectangular box (*container*) which sizes are given by a vector $L = (L_1, \dots, L_m)^T \in \mathbb{Z}^m$, and we also have a set of n small m-dimensional rectangular boxes (*items*), item r sizes are given by a vector $l^r = (l^r_1, \dots, l^r_m)^T \in \mathbb{Z}^m$.

In the (*orthogonal*) *m-dimensional knapsack problem* (*m-KP*), for each item r, we also know its *cost* c_r. The goal is to pack into the container — which is also called a *knapsack* — a subset of items of maximum total cost so that no two items overlap, and all item edges are parallel to the knapsack edges. If the items cannot be rotated (say, when cutting decorated materials) and every edge of each item must be parallel to the corresponding knapsack edge, we have an m-KP without rotation. Unless otherwise stated, in what follows we shall consider the m-KP without rotation. This restriction will be removed in Sect. 2.15.3, where we consider some extensions of our basic IP formulation.

In the *m-dimensional strip packing problem* (*m-SPP*), it is assumed that one edge (let us call it the *height*) of the container — which now is called a *strip* — is sufficiently big so that all the items can be packed into the strip, and the objective is to minimize the height of the occupied part of the strip.

In the *m-dimensional bin packing problem* (*m-BPP*) there are many large boxes of equal sizes — which are called *bins* — and the objective is to pack all n items into a minimum number of bins. We can translate any m-BPP instance into an instance of $(m+1)$-SPP, where the additional $(m+1)$-st direction (the height of the strip) is used to count bins: L_{m+1} is an upper bound on the number of needed bins, and $l^r_{m+1} = 1$ for all items r.

The main requirement in each of the above packing problems is that any two items put into the container do not overlap. There are basically two ideas on how this non-overlapping requirement is formulated in IPs. The first one originates from the floor-planning applications, and it is considered in Sect. 1.2.1 for the two-dimensional floor planning. We can easily extend the disjunctive approach used there to formulate the m-KP as a MIP. But this disjunctive approach does not result in a tight formulation because of using large coefficients to represent the disjunctions by linear inequalities. Therefore such disjunctive MIP formulations are almost useless in practice (using the best **MIP** solvers one cannot hope to solve m-KPs even of moderate size).

An alternative approach for representing the non-overlapping requirement is to consider the container as an m-dimensional grid which cells are m-dimensional unit cubes. As usual, we associate each cell with its origin defined to be its corner that is nearest to the container origin. Let $\mathscr{L}$ denotes the set $\prod_{k=1}^{m} \{0, 1, \dots, L_k - 1\}$ of grid cell origins. We say that an item is placed at a point $p \in \mathscr{L}$ if its corner that is nearest to the container origin is placed at p. Instead of requiring that no pair of items overlap, now we require that any cell be covered by at most one item (see Exercise 2.12). Although this approach leads to a much tighter **IP**-formulation, the size of this formulation is huge, and its constraint matrix is very dense. Therefore, this formulation cannot be used for solving packing problems even of moderate sizes.

2.15.1 Basic IP Formulation

Here we consider a modeling approach that combines the disjunctive approach with the discrete representation of the container. The latter will allow us to formulate the non-overlapping disjunctions by linear inequalities with small coefficients.

First, let us define two families of decision binary variables:

- $z_r = 1$ if item r is placed into the knapsack, and $z_r = 0$ otherwise;
- $y_{rij} = 1$ if item r is placed at a point $p \in \mathscr{L}$ with $p_i = j$, and $y_{rij} = 0$ otherwise.

For modeling purposes, we also need two families of auxiliary variables:

- $x_{rij} = 1$ if the open unit strip $U^i_j \overset{\text{def}}{=} \{w \in \mathbb{R}^m : j < w_i < j+1\}$ intersects item r, and $x_{rij} = 0$ otherwise;
- $s_{r_1,r_2,i} = 1$ if items r_1 and r_2 are separated by a hyperplane that is orthogonal to axis i, and $s_{r_1,r_2,i} = 0$ otherwise.

In these variables the m-KP is written as follows:

$$\sum_{r=1}^{n} c_r z_r \to \max, \tag{2.17a}$$

$$\sum_{j=0}^{L_i - l_i^r} y_{rij} = z_r, \quad i = 1,\dots,m, \;\; r = 1,\dots,n, \tag{2.17b}$$

$$x_{rij} = \sum_{j_1=\max\{0,j-l_i^r+1\}}^{\min\{j,L_i-l_i^r\}} y_{r,i,j_1}, \quad j = 0,\dots,L_i - 1, \;\; i = 1,\dots,m, \;\; r = 1,\dots,n, \tag{2.17c}$$

$$\sum_{i=1}^{m} s_{r_1,r_2,i} \ge z_{r_1} + z_{r_2} - 1, \quad r_2 = r_1+1,\dots,n, \;\; r_1 = 1,\dots,n-1, \tag{2.17d}$$

$$x_{r_1,i,j} + x_{r_2,i,j} + s_{r_1,r_2,i} \le z_{r_1} + z_{r_2}, \quad j = 1,\dots,L_i,\; i = 1,\dots,m, \;\; r_2 = r_1+1,\dots,n,\; r_1 = 1,\dots,n-1, \tag{2.17e}$$

$$z_r \in \{0,1\}, \quad r = 1,\dots,n, \tag{2.17f}$$

$$y_{rij} \in \{0,1\}, \quad j = 0,\dots,L_i - l_i^r, \;\; i = 1,\dots,m, \;\; r = 1,\dots,n, \tag{2.17g}$$

$$x_{rij} \in \{0,1\}, \quad j = 0,\dots,L_i - 1,\ i = 1,\dots,m, \\ r = 1,\dots,n, \tag{2.17h}$$

$$s_{r_1,r_2,i} \in \{0,1\}, \quad i = 1,\dots,m,\ r_2 = r_1 + 1,\dots,n, \\ r_1 = 1,\dots,n-1. \tag{2.17i}$$

Objective (2.17a) is to maximize the total cost of items placed into the knapsack. Equations (2.17b) ensure that the values of y-variables uniquely determine the positions of all items placed into the knapsack. Simultaneously, these equations set to zero the values of those y-variables that correspond to the items not placed into the knapsack ($z_r = 0$). Equations (2.17c) reflect the relationship between the x and y-variables: a strip U^i_j crosses item r only if coordinate i of its nearest to the origin corner is between $\max\{0, j - l^r_i + 1\}$ and $\min\{j, L_i - l^r_i\}$. These equations together with (2.17b) also impose the restrictions on the item sizes, namely, if item r is placed into the knapsack, then, for any $i = 1,\dots,m$, the number of strips U^i_j crossing r is l^r_i, and these strips are sequential. Two families of inequalities, (2.17d) and (2.17e), imply that each pair of items placed into the knapsack will be separated by at least one hyperplane that is orthogonal to a coordinate axis. The other relations, (2.17f)–(2.17i), declare that all variables are binary.

2.15.2 Tightening Basic Model

Here, we introduce a family of knapsack inequalities that can significantly strengthen our IP (2.17). Let us denote by Vol the volume, $\prod_{i=1}^m L_i$, of the knapsack, and by vol_r the volume, $\prod_{i=1}^m l^r_i$, of item r. With this notations, we introduce the following knapsack inequalities:

$$\sum_{r=1}^{n} vol_r z_r \le Vol, \tag{2.18}$$

$$\sum_{r=1}^{n} \frac{vol_r}{l^r_i} x_{rij} \le \frac{Vol}{L_i}, \quad j = 1,\dots,L_i,\ i = 1,\dots,m. \tag{2.19}$$

Inequality (2.18) imposes a natural restriction that the sum of item volumes cannot exceed the knapsack volume. Inequalities (2.19) reflect the fact that the sum of the volumes of the intersections of all the items with any unit strip orthogonal to some coordinate axis cannot exceed the volume of the intersection of the knapsack with that strip.

Computational experiments showed that these knapsack inequalities may greatly tighten our basic IP formulation.

2.15.3 Rotations and Complex Packing Items

In this section we show how to extend our IP formulation of m-KP to cover the cases when the rotation of items is allowed, and when the items are the unions of rectangular boxes.

To model the first case, let us consider a version of m-KP when all n items are partitioned into k groups, $I_1, \dots, I_k$, and from each group no more than one item can be placed into the container. To take into account this additional restriction, we need to add to (2.17) the following inequalities:

$$\sum_{i \in I_q} z_i \le 1, \quad q = 1, \dots, k.$$

If it is allowed to rotate the items, we put into one group all the items resulting from all possible rotations of a particular item.

To model the case when some items are the unions of two or more rectangular boxes, let us assume that our input n items are divided into groups of one or more items, and all items from any group are together put or not put into the knapsack. In each group of items we choose one item, $\bar{r}$, called the *base*; any non-base item r in this group is assigned a reference, $ref(r) = \bar{r}$, to the base item, $ref(\bar{r}) = -1$. Each non-base item r is also assigned a vector $v^r \in \mathbb{R}^m$: if $o^{\bar{r}}$ is the nearest to the origin corner of the base item $\bar{r} = ref(r)$, then $o^{\bar{r}} + v^r$ is the nearest to the origin corner of item r. In other words, the shape of any group is determined by its base $\bar{r}$ and the vectors v^r assigned to all non-base items r of this group.

The following equations model the above defined group restrictions;

$$\begin{aligned} z_r &= z_{ref(r)}, \quad r = 1, \dots, n,\ ref(r) \ge 0, \\ o_i^r &= o_i^{ref(r)} + v_i^r z_r, \quad i = 1, \dots, m,\ r = 1, \dots, n,\ ref(r) \ge 0. \end{aligned}$$

2.16 Single Depot Vehicle Routing Problem

There is a depot that supplies customers with some goods. This depot has a fleet of vehicles of K different types. There are q_k vehicles of type k, and each such a vehicle is of *capacity* u_k (maximum weight to carry). We also know the *fixed cost*, f_k, of using one vehicle of type k during a day.

At a particular day, n customers have ordered some goods to be delivered from the depot to their places: customer i is expecting to get goods of total weight d_i. To simplify the notations, we will also consider the depot as a customer with zero demand. For a vehicle of type k, it costs c_{ij}^k to travel from customer i to customer j.

A *route* for a vehicle is given by a list of customers, $(i_0 = 0, i_1, \dots, i_r, i_{r+1} = 0)$, in which none of the customers, except for customer 0 (depot), is met twice. This list determines the order of visiting customers. The route is *feasible* for vehicles of type k if the total demand of the customers on this route does not exceed the vehicle capacity, $\sum_{s=1}^{r} d_{i_s} \le u_k$. The cost of assigning a vehicle of type k on this route is $f_k + \sum_{s=1}^{r+1} c^k_{i_{s-1},i_s}$.

The *vehicle routing problem* (*VRP*) is to select a subset of routes such that each customer is just on one route, and then assign a vehicle of sufficient capacity (from the depot fleet) to each selected rout so that the total cost of assigning vehicles to the routes is minimum.

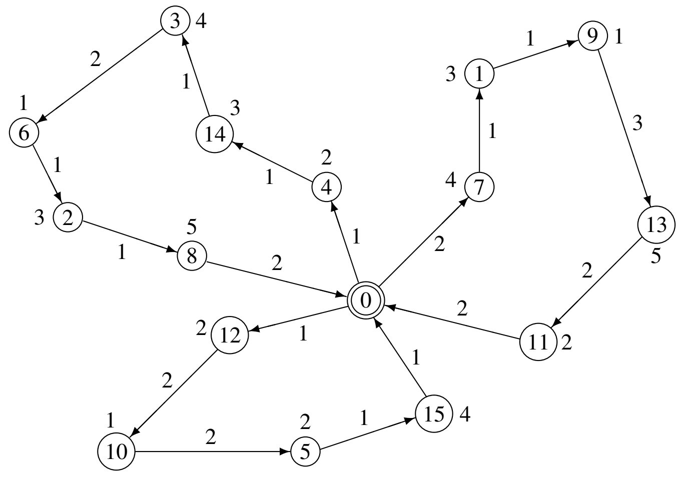

Fig. 2.4 Three rout solution for a VRP

Figure 2.4 displays a solution to some example VRP. Here we have 15 customers (represented by the nodes numbered from 1 to 15) serving from one depot (node 0), the numbers next to the nodes are customer demands, and the number adjacent to each arc is the traveling cost for the car assigned to the route that contains this arc. In our example we have three routes:

- $0 \to 4 \to 14 \to 3 \to 6 \to 2 \to 8 \to 0$ of total demand 18 and cost 9,
- $0 \to 7 \to 1 \to 9 \to 13 \to 11 \to 0$ of total demand 15 and cost 11,
- $0 \to 12 \to 10 \to 5 \to 15 \to 0$ of total demand 9 and cost 7.

To formulate the VRP as an IP, we need the following family of decision binary variables:

- $x^k_{ij} = 1$ if some vehicle of type k travels directly from customer i to customer j, and $x^k_{ij} = 0$ otherwise.

In addition, we also need one family of auxiliary variables:

- y_{ij}: weight of goods that are carried by the vehicle assigned to the route from customer i to customer j.

In these variables the model is written as follows:

$$\sum_{k=1}^{K}\sum_{j=1}^{n}(f_k + c_{0,j}^k)x_{0,j}^k + \sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j\in\{0,\dots,n\}\setminus\{i\}} c_{ij}^k x_{ij}^k \to \min, \tag{2.20a}$$

$$\sum_{j=1}^{n} x_{0,j}^k \le q_k, \quad k = 1,\dots,K, \tag{2.20b}$$

$$\sum_{k=1}^{K}\sum_{i=0}^{n} x_{ij}^k = 1, \quad j = 1,\dots,n, \tag{2.20c}$$

$$\sum_{i=0}^{n} x_{ij}^k - \sum_{i=0}^{n} x_{ji}^k = 0, \quad j = 1,\dots,n,\ k = 1,\dots,K, \tag{2.20d}$$

$$\sum_{i=0}^{n} y_{ij} - \sum_{i=0}^{n} y_{ji} = d_j, \quad j = 1,\dots,n, \tag{2.20e}$$

$$0 \le y_{ij} \le \sum_{k=1}^{K}(u_k - d_i)x_{ij}^k, \quad i,j = 0,\dots,n, \tag{2.20f}$$

$$x_{ii}^k = 0, \quad i = 0,\dots,n,\ k = 1,\dots,K, \tag{2.20g}$$

$$x_{ij}^k \in \{0,1\}, \quad i,j = 0,\dots,n,\ k = 1,\dots,K. \tag{2.20h}$$

Objective (2.20a) is to minimize the total fixed cost of using vehicles plus the traveling cost of all used vehicles along their routes. Inequalities (2.20b) ensure that the number of vehicles of any particular type that leave the depot does not exceed the number of such vehicles in the depot fleet. Equations (2.20c) guarantee that each customer will be visited by just one vehicle, while (2.20d) guarantee that any vehicle that arrives at a customer will leave that customer. The next family of equations, (2.20e), reflects the fact that any vehicle before leaving a customer must upload the goods ordered by that customer. Each inequality from (2.20f) imposes the capacity restriction: if a vehicle of type k travels directly from customer i to customer j, then the total cargo weight on its board cannot exceed $u_k - d_i$; moreover, if none of vehicles (of any type) travels directly from customer i to customer j, then $y_{ij} = 0$. One can easily argue that these capacity restrictions guarantee that each used vehicle will never carry goods of total weight greater than the vehicle capacity.

A note of precaution is appropriate here. Formulation (2.20) may be very *weak* because the variable bound constraints in (2.20f) are usually not tight. Let us assume that some inequality $y_{ij} \le \sum_{k=1}^{k}(u_k - d_i)x_{ij}^k$ holds as equality. If the capacity u_k is big, and the demand d_i is small, then $u_k - d_i$ is big. If we further assume that the sum of demands of those customers that are on the same route with customer i and are visited after customer i is small, then y_{ij} is also small and, therefore, x_{ij}^k takes a small fractional value. Many binary variables taking fractional values is usually an indicator of a difficult to solve problem. Therefore, one can hardly expect that (2.20) can be used for solving to optimality even VRPs of moderate size. Nevertheless, if implemented properly, an application based on this formulation can produce rather good approximate solutions for VRPs of practical importance.

2.16.1 Classical Vehicle Routing Problem

Here we consider a special case of the VRP that is more "uniformly" structured. Let us assume that there are m vehicles in the depot fleet, all of them are of the same type ($K = 1$) and the same capacity U. Fixed costs of using vehicles are not taken into account. This special VRP is known as the *classical vehicle routing problem* (*CVRP*).

Let $N \overset{\text{def}}{=} \{0,1,\dots,n\}$, and let $r(S)$ denote the minimum number of vehicles needed to serve a subset $S \subseteq N \setminus \{0\}$ of customers. The value of $r(S)$ can be computed by solving the 1-BPP (see Sect. 2.15) with all bins of capacity U, and the item set S, where the length of item i is $l_1^i = d_i$. Since 1-BPP is **NP**-hard in the strong sense, in practice, $r(S)$ is approximated from below by the value $\lfloor (\sum_{i\in S} d_i)/U \rfloor$.

To formulate the CVRP as an IP, we use the following family of decision binary variables:

- $x_{ij} = 1$ if some vehicles directly travels from customer i to customer j, and $x_{ij} = 0$ otherwise.

In these variables the model is written as follows:

$$\sum_{i\in N} \sum_{j\in N\setminus\{i\}} c_{ij}x_{ij} \to \min, \tag{2.21a}$$

$$\sum_{j\in N\setminus\{0\}} x_{0,j} \le m, \tag{2.21b}$$

$$\sum_{j\in N\setminus\{i\}} x_{ji} = 1, \quad i \in N \setminus \{0\}, \tag{2.21c}$$

$$\sum_{j\in N\setminus S} \sum_{i\in S} x_{ji} \ge r(S), \quad S \subseteq N \setminus \{0\},\ S \ne \emptyset, \tag{2.21d}$$

$$x_{ij} \in \{0,1\}, \quad i \in N,\ j \in N \setminus \{i\}. \tag{2.21e}$$

Objective (2.21a) is to minimize the total delivery cost. Inequality (2.21b) ensure that no more than m vehicles may leave the depot, and, therefore, there may be no more than m routes. Equations (2.21c) guarantee that each customer will be visited exactly once. Inequalities (2.21d) guarantee that the routs defined by the values of x_{ij}-variables are feasible, i.e., each of them leaves the depot, and, due to definition of $r(S)$, the total weight of all customer demands on this route does not exceed the vehicle capacity.

Since (2.21d) contains exponentially many inequalities, IP (2.21) can be solved only by a cutting plane algorithm (see Sect 4), and this is possible only if the separation problem (see Sect 4.7) for (2.21d) can be solved very quickly. Unfortunately, in general this is not the case. However, when x is an integer vector that satisfied all constraints from (2.21) but (2.21d), then the problem of finding in (2.21d) an inequality that is violated at x is trivial, and we leave it to the reader to elaborate such a separating procedure.

2.17 Notes

***Sect.* 2.1.** In more detail, the set packing, set partitioning and set covering problems are discussed in [98, 123]. The polyhedral structure of the set packing problem is also considered in Sect. 5.5.

The problems similar to the problem of crew scheduling have always been a fruitful area for applications of the set covering and set partitioning problems (see [80]). If you are interested in how trades are organized at combinatorial auctions, see [100].

***Sect.* 2.2.** The problem of locating service centers was first formulated as an IP in [15]. The polyhedral structure of this problem is studied in [98, 142].

***Sect.* 2.3.** Exact and approximate methods of solving the problem of the formation of the index fund are studied in [40].

***Sect.* 2.4.** A classification of multi-product lot-sizing models is given in [143]. The polyhedral structures of some of these models are studied in [98, 142].

***Sect.* 2.5.** The assembly line balancing problems and the algorithms for their solution are discussed in [120].

***Sect.* 2.6.** In the literature on the optimization of the performance of energy systems, the unit commitment problem is one of the highest priority [126].

***Sect.* 2.7.** From the sources [21, 131], you can learn more about the problems of designing telecommunication networks and the methods for solving them.

***Sect.* 2.8.** A good source about the detailed placement problem is the survey [125].

***Sect.* 2.9.** The problem of assigning aircrafts to flights and its **IP** formulation is studied in [1].

***Sect.* 2.10.** The model for determining an optimal operation mode of a hybrid car is an extended version of the model from [29].

***Sect.* 2.11.** The problem of short-term financial management was formulated back in 1969 [101], and its essence has not changed since then.

***Sect.* 2.12.** A historical reference on the application of **MIP** for the treatment of cancer tumors by the method of intensive modulated radiation therapy is given in [112]. A description of the column generation algorithm for solving (2.14) is also proposed there.

***Sect.* 2.13.** A good survey on the resource-constrained project scheduling is given in [128].

***Sect.* 2.14.** Using STNs for describing technological processes was proposed in [82]. A **MIP** formulation for the problem of optimizing technological processes was also presented there.

***Sect.* 2.15.** The disjunctive approach for modeling multidimentional packing problems originates from the floor-planning applications [130]. The first **IP** formulation of a two-dimensional packing problem (namely, the cutting stock problem) based on the discretization of the container space was given by Beasley [20] (see also Exercise 2.12). Formulation (2.17) is published in this book for the first time.

***Sect.* 2.16.** Many **IP** alternative formulations have been proposed for different variations of the vehicle routing problem. The single-commodity flow formulation (2.20) was first presented in [53]. Formulation (2.21) was given in [84] as an extension of the **IP** formulation for the traveling salesman problem proposed in [43] (see also (6.13)). For a survey on vehicle routing, see [38].

2.18 Exercises

2.1. *Frequency assignment*. Frequency ranges numbered from 1 to k must be assigned to n radio stations. For each pair of station, i, j, we know the value p_{ij} of the *interference parameter*, which means that the modulus of the difference of the frequency ranges assigned to the stations i and j must be at least p_{ij}. The goal is to minimize the maximum assigned range number.

Formulate this problem as an IP.

2.2. *ATM allocation*. A bank wants to install a certain number of automated teller machines (ATMs) in a rural area in which there are n communities. ATMs can be placed in any of these locations. It is known that k_i bank's customers reside in community i, and it takes t_{ij} minutes to drive from community i to community j.

1) What is the minimum number of ATMs to be installed and at what locations must they be installed so that the travel time from any community to the nearest ATM does not exceed T minutes?
2) Where to place no more than q ATMs to maximize the number of customers that can get to the nearest ATM for no more than T minutes?

Formulate both problems as MIPs. Compare two formulations. Which one is more appropriate for using in practice?

2.3. Assume that in the scheduling problem considered in Sect. 1.8 it is additionally assumed that, if both jobs j_1 and j_2 are assigned to the same machine k, to start processing job j_2 immediately after job j_1 , $\tau^k_{j_1,j_2}$ units of time are spent preparing the machine.

Extend the time-index formulation (1.26) to take into account this new problem feature.

2.4. *Scheduling multiple machines*. The enterprise has orders for batches of parts, each batch contains n different parts. Each part must be successively processed by machines $1,2,\dots,m$. The processing time of part i on machine k is p_{ik}. It is necessary to determine the order of processing parts on each of the machines so that the processing time of one batch is minimum.

Write two formulations for this problems: in one use a continuous time model, and in the other a discrete one.

2.5. *Control of fuel consumption*. Consider a linear dynamical system that is described by the following linear recursion

$$x(t) = Ax(t-1) + bu(t-1), \quad t = 1, \dots, T,$$

where an $m \times n$-matrix A and a vector $b \in \mathbb{R}^n$ are given parameters, $x(t) \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}$ are, respectively, the state of the system and the control (signal) in period t. We need to define the controls $u(0), \dots, u(T-1)$ in order to transfer the system from an initial state $x(0) = x^0$ to a final state $x(T) = x^{\text{dest}}$ consuming the minimum amount of fuel that is determined by the formula $\sum_{t=0}^{T-1} f(u(t))$.

Assuming that f is a piecewise linear function with the break-points

$$(u_1, f_1 = f(u_1)), \dots, (u_k, f_k = f(u_k)),$$

formulate this problem as a) an LP if f is convex, or b) a MIP in the general case.

2.6. *Single-product lot-sizing with backlogging*. Let us consider the single-product lot-sizing problem studied in Sect. 1.7.1. But now it is assumed that a part of the demand in some periods can be satisfied by deliveries in later periods. It is allowed that no more than $\bar{d}_{jt}$ unit of the product of the total demand in d_{jt} units can be delivered not in time but later in periods $t+1, \dots, T$. Each unit of product j, supplied not in time, is sold at a discount equal to r_j.

Modify (2.6) to take into account these new features.

2.7. Prove that we can strengthen IP (2.7), which is a formulation of the simple assembly line balancing problem, using in place of (2.7e) the following alternative formulation of the precedence relations:

$$\sum_{i=1}^{k} x_{i,j_1} \le \sum_{i=1}^{k} x_{i,j_2}, \quad k = 1, \dots, n-1, \ (j_1, j_2) \in E.$$

2.8. Modify IP (2.7), which is a formulation of the simple assembly line balancing problem, to take into account the following requirements to uniformly load the stations: a) any open station work time (on one product) must be at least q_1 percent of the cycle time, b) the work times of the maximum and minimum loaded stations must not differ by more than q_2 percent.

2.9. *Clearing problem*. There are m banks in a country. The current balance of bank i is b_i. Several times a day, the Interbank Settlement Center receives a list of payments $P_k = (i_k, j_k, A_k^1, A_k^2, S_k)$, $k = 1, \dots, n$. The fields of the tuple P_k are interpreted as follows: it is necessary to transfer the sum S_k from the account A_k^1 in bank i_k to the account A_k^2 in bank j_k. The goal is to accept as many payments as possible, provided that the new balance (calculated taking into account the payments made) of each of the banks will be non-negative. Note that for this optimization problem the fields A_k^1 and A_k^2 of the payment records are insignificant.

Formulate this clearing problem as an IP.

2.10. *Sport tournament scheduling*. The teams participating in the basketball championship are divided into two conferences: Western and Eastern. There are n_1 teams in the western conference, and n_2 teams in the eastern conference. One round of the championship lasts T weeks. Each team must play no more than once a week and must play $2k$ times with each team in its division and $2q$ times with each team in the other division. For each pair of opponents, half of the games must be played on the site of each team. Of all the round-robin schedules, the best is that for which the minimum interval between games of the same pair of teams is maximum.

Formulate the problem of finding the best tournament schedule as an IP.

2.11. *Nearest substring problem*[4]. Let $\mathscr{A}$ be some finite set of symbols (alphabet). The sequence of characters $s =$ "$s_1 s_2 \dots s_k$" ($s_i \in \mathscr{A}$) is called a *string* of length $|s| = k$ in the alphabet $\mathscr{A}$. A *substring* of string s is a string "$s_{i_1} s_{i_2} \dots s_{i_m}$" composed of the characters of string s and written in the same order in which they are present in s, i.e., $1 \le i_1 < i_2 < \dots < i_m \le |s|$. The distance $d(s^1, s^2)$ between two strings s^1 and s^2 of the same length is defined as the number of positions in which these strings differ. For example, if $s^1 =$ "ACT" and $s^2 =$ "CCA", then $d(s^1, s^2) = 2$. If $|s^1| < |s^2|$, then the distance $d(s^1, s^2)$ is defined to be the maximum distance $d(s^1, \bar{s}^2)$ for all substrings $\bar{s}^2$ of length $|s^1|$ in string s^2.

Given a list $(s^1, s^2, \dots, s^n)$ of strings in some alphabet $\mathscr{A}$, the length of each string s^i is at least m. We need to find a string s of length m such that the maximum distance $d(s, s^i)$ $(i = 1, \dots, n)$ is minimum. Formulate this problem as an IP.

2.12. Formulate the 2-KP as an IP using only the following binary variables: y_{rij}, $j = 0, \dots, L_i - l_i^r$, $i = 1, 2$, $r = 1, \dots, n$, where $y_{r,1,s} = y_{r,2,t} = 1$ only if (s,t) is the item r corner that is nearest to the knapsack origin.

2.13. *Balanced airplane loading*. The cargo plane has three cargo compartments: front (1st), central (2nd) and tail (3rd). The base of compartment i is a rectangle of width W_i and length L_i, the total weight of cargo in compartment i must not exceed G_i tone, $i = 1, 2, 3$.

We need to load n containers into the plane, the containers cannot be stacked on top of each other. The weight of container j is g_j, and its base is of width w_j and length l_j. The goal is to load the containers into the plane so that the weights of cargo in different compartments are balanced: the difference between the largest and smallest ratios of the total weight of the cargo in the compartment to the maximum allowable weight of cargo in this compartment must be minimum.

Formulate the problem of balanced airplane loading as an IP.

[4] The problems of comparing strings in different statements are often encountered in many applications of computational biology.

Chapter 3
Linear Programming

After discovering the *interior point methods*, it seemed that the new methods would completely displace the simplex algorithms from the practical use. New methods proved to be very efficient in practice, especially when solving large-scale LPs. A crucial requirement for an **LP** algorithm to be used in **MIP** is its ability to quickly perform reoptimization, i.e., having found an optimal solution of some LP, the method must be able to quickly find an optimal solution of a ”slightly” modified version of the just solved LP. None of the interior point algorithms is able to do reoptimization quickly. The wide use of MIPs in practice and the ability of the dual-simplex method to quickly perform reoptimization enabled the latter to survive in the competition with the interior points methods. Moreover, the acute practical need for efficient **MIP** algorithms stimulated researches that resulted in a significant increase in the efficiency of the simplex algorithms in practice. And today the best implementations of the simplex algorithms are quite competitive with the best implementations of the best interior point methods. Therefore, here we will study only the simplex algorithms, and the main attention will be paid to the dual simplex method.

3.1 Basic Solutions

Let us consider the LP in *canonical form*:

$$z_P = \max\{c^T x : Ax \le b\}, \tag{3.1}$$

where A is a real $m \times n$-matrix, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $x = (x_1, \dots, x_n)^T$ is a vector of *variables* (*unknowns*).

In what follows we will assume that the constraint matrix A in (3.1) is of full column rank, i.e., $\text{rank}(A) = n$. Let $M = \{1, \dots, m\}$ and $N = \{1, \dots, n\}$ be, respectively, the sets of rows and columns of A. We denote by A_I^J the submatrix of A with rows from a set $I \subseteq M$ and columns from a set $J \subseteq N$. If $J = N$ (resp., $I = M$), then instead of A_I^N (resp., A_M^J) we write A_I (resp., A^J).

A subset I of n linearly independent rows of A is called a (*row*) *basic set*, the matrix A_I is called a *basic matrix*, and the only solution $\bar{x} = A_I^{-1} b_I$ of the linear system $A_I x = b_I$ is a *basic solution*. If in addition $\bar{x}$ is *feasible*, i.e., it satisfies the system of inequalities $Ax \le b$, then $\bar{x}$ is called a *feasible basic solution*, and I is called a *feasible basic set*. Note also that the feasible basic solutions are nothing else as the *vertices* of the polyhedron $P(A,b) \overset{\text{def}}{=} \{x \in \mathbb{R}^n : Ax \le b\}$.

To clarify the above definitions, let us consider the feasible region of an LP with $n = 3$ variables and $m = 7$ constraints

$$\begin{array}{rrrll}
-x_1 & & & \le 0, & H_1\\
& -\ x_2 & & \le 0, & H_2\\
& & -x_3 & \le 0, & H_3\\
x_1 + & x_2 + & x_3 & \le 4, & H_4\\
2x_1 & & & \le 5, & H_5\\
& 3x_2 & & \le 7. & H_6
\end{array}$$

These constraints are shown in Fig. 3.1. The basic solutions are depicted as bold dots, and feasible basic solutions in addition are circled.

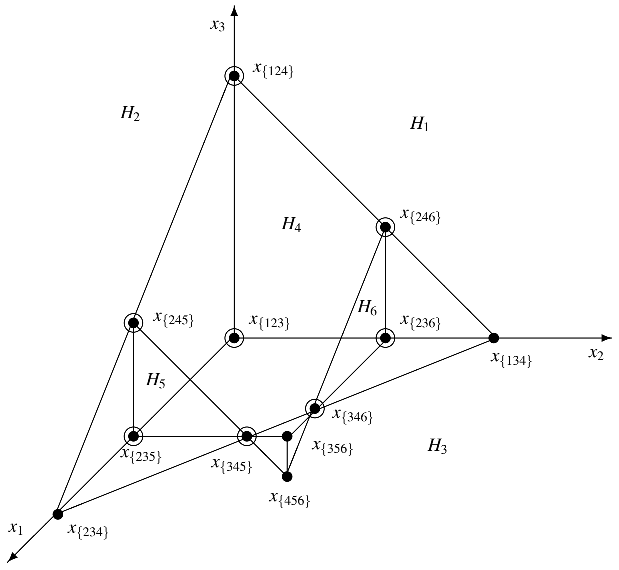

Fig. 3.1 Vertices and basic solutions

A feasible basic solution (vertex) $\bar{x}$ is said to be *degenerate* if it corresponds to two different basic sets I_1 and I_2, i.e., $A_{I_1}\bar{x} = b_{I_1}$, $A_{I_2}\bar{x} = b_{I_2}$, $|I_1| = |I_2| = n$, $I_1 \ne I_2$. Geometrically this means that the vertex $\bar{x}$ lies on more than n facets of the polyhedron $P(A,b)$. For example, two vertices $(1,0,3)$ and $(2,2,0)^T$ of the polytope from Fig. 1.4 are degenerate, and all the others are nondegenerate. A polyhedron $P(A,b)$ that has degenerate vertices is called *degenerate*. Similarly, an LP having degenerate feasible basic solutions is called *degenerate*.

A basic set I and the corresponding to it basic solution $\bar{x} = A_I^{-1} b_I$ are called *dual feasible* if the vector of *potentials* $\pi^T = c^T A_I^{-1}$ is non-negative. In this case, the point $\bar{y} = (\bar{y}_I = \pi, \bar{y}_{M\setminus I} = 0)$ is a feasible solution to the *dual LP*[1]

$$z_D = \min\{b^T y : A^T y \geq c,\ y \geq 0\}. \tag{3.2}$$

We have the equality

$$c^T \bar{x} = c^T A_I^{-1} b_I = \pi^T b_I + b_{M\setminus I} 0 = b^T \bar{y}.$$

On the other hand, for any feasible solution x of the primal LP (3.1) and any feasible solution y of the dual LP (3.2), we have

$$c^T x \leq y^T A x \leq y^T b. \tag{3.3}$$

It follows that $\bar{x}$ and $\bar{y}$ are *optimal solutions* to the primal, (3.1), and dual, (3.2), LPs if the basic set I is simultaneously primal and dual feasible. In the context of duality, feasible basic sets and solutions are also called *primal feasible*, and the solutions to the dual LP are called *dual solutions* (for the primal LP). We also note that the components of an optimal dual solution are also called *shadow prices* (for the primal LP).

3.2 Primal Simplex Method

Let I be a feasible basic set, $B = A_I$ and $\bar{x} = B^{-1} b_I$ be the corresponding basic matrix and feasible basic solution. Now and in what follows, we assume that the order of the elements in the basic set is fixed, i.e., I is a list, and we denote the i-th element of this list by $I[i]$.

Let us remove from the basic set some row index $I[t]$. The solution set of the linear system $A_{I\setminus I[t]} x = b_{I\setminus I[t]}$ is the line $\{x(\lambda) \overset{\text{def}}{=} \bar{x} - \lambda B^{-1} \mathbf{e}_t :\ \lambda \in \mathbb{R}\}$. Let us imagine that we put an n-dimensional chip into the point $\bar{x} = x(0)$, and then, increasing λ, move this chip within the feasible polyhedron $P(A,b)$ along the ray $\{x(\lambda) :\ \lambda \geq 0\}$ until it rests against some hyperplane given by $A_s x = b_s$ for $s \notin I$. Let $\hat{x} = x(\hat{\lambda}_t)$ be the intersection point of our ray with his hyperplane. Notice that

$$\hat{\lambda}_t = \frac{\|\hat{x} - \bar{x}\|}{\|B^{-1}\mathbf{e}_t\|} = \min_{\substack{i \notin I,\\ A_i B^{-1} \mathbf{e}_t > 0}} \frac{b_i - A_i \bar{x}}{A_i B^{-1} \mathbf{e}_t} = \frac{b_s - A_s \bar{x}}{A_s B^{-1} \mathbf{e}_t}$$

[1] Duality in linear programming is discussed in Sect. 3.5. The variables y_i of the dual LP (3.2) are *dual variables* for the primal LP (3.1).

and $\hat{x} = \hat{B}^{-1}b_{\hat{I}}$, where $\hat{B} = A_{\hat{I}}$ and the new basic row set, $\hat{I}$, is defined by the rule

$$\hat{I}[i] = \begin{cases} s, & i = t, \\ I[i], & i \neq t. \end{cases} \tag{3.4}$$

It is not difficult to see that the following equality holds

$$\hat{B}^{-1} = B^{-1}I(t,u)^{-1}, \tag{3.5}$$

where $u^T = A_s B^{-1}$, and the matrix $I(t,u)$ is obtained from the identity matrix I by substituting the row vector u^T for the row t. Notice that

$$I(t,u)^{-1} = I - \frac{1}{u_t}\mathbf{e}_t\left(u^T + \mathbf{e}_t^T\right)$$
$$= \begin{bmatrix} 1 & 0 \dots & 0 & 0 & 0 \dots & 0 \\ 0 & 1 \dots & 0 & 0 & 0 \dots & 0 \\ \vdots & \vdots \ddots & \vdots & \vdots & \vdots \ddots & \vdots \\ 0 & 0 \dots & 1 & 0 & 0 \dots & 0 \\ -\frac{u_1}{u_t} & -\frac{u_2}{u_t} \dots & -\frac{u_{t-1}}{u_t} & \frac{1}{u_t} & -\frac{u_{t+1}}{u_t} \dots & -\frac{u_n}{u_t} \\ 0 & 0 \dots & 0 & 0 & 1 \dots & 0 \\ \vdots & \vdots \ddots & \vdots & \vdots & \vdots \ddots & \vdots \\ 0 & 0 \dots & 0 & 0 & 0 \dots & 1 \end{bmatrix}. \tag{3.6}$$

In **LP**, such a change in the basis is called a *pivot operation*. Column t is called a *pivot column*, row s is a *pivot row*, and the element $A_s B^{-1}\mathbf{e}_t$, which in the matrix AB^{-1} is in row s and column $I[t]$, is called a *pivot element*.

Let us illustrate the pivot operation using the example polytope from Fig. 3.1. Let $I = \{2,4,6\}$ and $t = 1$. Then $\bar{x} = x_{\{246\}}$ and the ray $\{x(\lambda) : \lambda \geq 0\}$ is directed from the point $x_{\{246\}}$ along the edge $[x_{\{246\}}, x_{\{346\}}]$ to the point $x_{\{346\}}$, which lies on the hyperplane H_3. Therefore, $\hat{I} = \{3,4,6\}$ and $\hat{x} = x_{\{346\}}$ are the new feasible basic set and solution.

To ensure that, after performing the pivot operation, the objective function will increase,

$$c^T\hat{x} - c^T\bar{x} = -\frac{\|\hat{x}-\bar{x}\|}{\|B^{-1}\mathbf{e}_t\|}c^T B^{-1}\mathbf{e}_t = -\frac{\|\hat{x}-\bar{x}\|}{\|B^{-1}\mathbf{e}_t\|}\pi_t > 0, \tag{3.7}$$

the index t is chosen so that the directing vector $-B^{-1}\mathbf{e}_t$ forms an acute angle with the gradient, c, of the objective function: $c^T B^{-1}\mathbf{e}_t = \pi_t > 0$.

It may happen that all components of the vector $AB^{-1}\mathbf{e}_t$ are nonpositive. Then $\hat{\lambda}_t = \infty$ (we assume that the minimum over the empty set of alternatives is infinite), and this means that we can move our chip along the ray $\{x(\lambda) : \lambda \geq 0\}$ infinitely long, remaining within the polyhedron $P(A,b)$. If, in addition, $\pi_t < 0$, then the objective function will increase to infinity.

```
simplex(c,A,b,I) // I is a feasible basic set
{
  $B^{-1} = A_I^{-1}$; $x := B^{-1}b_I$; $\pi^T := c^T B^{-1}$;
  while ($\pi \notin \mathbb{R}^n_+$) { // otherwise, x is an optimal solution
    choose an index t such that $\pi_t < 0$;
    $v := -B^{-1}e_t$. // $-v$ is column t of the inverse basic matrix $B^{-1}$
    if ($Av \le 0$) return (false,x,v); // objective function is unbounded
    $\lambda := \min\left\{\frac{b_i - A_i x}{A_i v} : i \notin I,\ A_i v > 0\right\}$;
    chose an index s, for which the value $\lambda$ is attained;
    $I[t] := s$; $u^T := A_s B^{-1}$; $B^{-1} := B^{-1} I(t,u)^{-1}$;
    $x := x + \lambda v$;   // $x = A_I^{-1} b_I$
    $\pi^T := \pi^T I(t,u)^{-1}$;   // $\pi^T = c^T B^{-1}$
  }
  return (true,$x,y = (y_I = \pi, y_{M\setminus I} = 0)$);
}
```

Listing 3.1. Primal simplex method

A detailed description of the (*primal*) *simplex method* is presented in Listing 3.1. The parameters of the LP being solved, A, b and c, as well as an initial feasible basic set I constitute the input to the *simplex* procedure that implements the primal simplex method. The method terminates for two reasons.

1) The current feasible basic set I becomes also dual feasible. In this case, the output is a triple
$$(\mathbf{true}, x, y = (y_I = \pi, y_{M\setminus I} = 0)),$$
where x is an optimal solution to the primal LP (3.1), and y is an optimal solution of the dual LP (3.2).
2) If $Av \le 0$, then the point $x(\lambda) = x + \lambda v$ is a feasible solution to (3.1) for all $\lambda \ge 0$ and $\lim_{\lambda\to\infty} c^T x(\lambda) = \infty$, i.e., the objective function of the primal LP is unbounded. In this case, the procedure returns the triple $(\mathbf{false}, x, v)$, where the pair (x, v) forms a *certificate of unboundedness* (see Sect. 3.4).

Example 3.1 *We need to solve the next LP:*

$$\begin{array}{rcrcrl}
x_1 & & & + & 2x_3 & \to \max,\\
x_1 & + & 2x_2 & + & x_3 & \le 4,\\
x_1 & & & + & x_3 & \le 3,\\
 & - & x_2 & + & x_3 & \le 1,\\
-x_1 & & & & & \le 0,\\
 & - & x_2 & & & \le 0,\\
 & & & - & x_3 & \le 0.
\end{array}$$

Solution. The polyhedron of feasible solutions for this LP is depicted in Fig. 3.2. When the simplex method starts working with the feasible basic set $I = (4,5,6)$, its iterations are as follows.

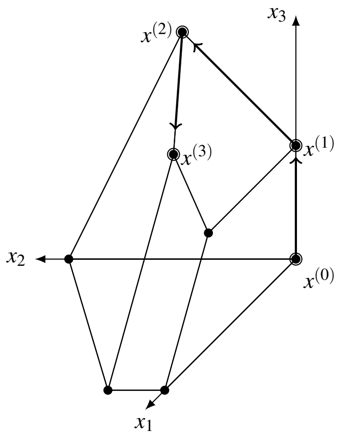

Fig. 3.2 Polyhedron of the LP from Exercise 3.1

0. $I = (4,5,6)$, $B^{-1} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$, $x^{(0)} = (0,0,0)^T$, $\pi = (-1,0,-2)^T$.

1. $t = 3$, $v = (0,0,1)^T$, and since $A_1 v = 1$, $b_1 - A_1 x^{(0)} = 4$, $A_2 v = 1$, $b_2 - A_2 x^{(0)} = 3$, $b_3 - A_3 x^{(0)} = 1$, $A_3 v = 1$, we have

$$\lambda = \min\left\{\frac{4}{1}, \frac{3}{1}, \frac{1}{1}\right\} = 1. \quad s = 3,$$
$$I = (4,5,3), \quad B^{-1} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & -1 & 1 \end{bmatrix},$$
$$x^{(1)} = x^{(0)} + \lambda v = (0,0,1)^T, \quad \pi = c^T B^{-1} = (-1,-2,2)^T.$$

2. $t = 2$, $v = (0,1,1)^T$, and since $A_1 v = 3$, $b_1 - A_1 x^{(1)} = 3$, $A_2 v = 1$, $b_2 - A_2 x^{(1)} = 2$, $A_6 v = -1$, we have

$$\lambda = \min\left\{\frac{3}{3}, \frac{2}{1}\right\} = 1, \quad s = 1,$$
$$I = (4,1,3), \quad B^{-1} = \begin{bmatrix} -1 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{2}{3} \end{bmatrix},$$
$$x^{(2)} = x^{(1)} + \lambda v = (0,1,2)^T, \quad \pi = \left(-\frac{1}{3}, \frac{2}{3}, \frac{4}{3}\right)^T.$$

3. $t = 1$, $v = \left(1, -\frac{1}{3}, -\frac{1}{3}\right)^T$, and since $A_2 v = \frac{2}{3}$, $b_2 - A_2 x^{(2)} = 1$, $A_5 v = \frac{1}{3}$, $b_5 - A_5 x^{(2)} = 1$, $A_6 v = \frac{1}{3}$, $b_6 - A_6 x^{(2)} = 2$, we have

$$\lambda = \min\left\{\frac{1}{2/3}, \frac{1}{1/3}, \frac{2}{1/3}\right\} = \frac{3}{2}, \quad s = 2,$$

$$I = (2,1,3), \quad B^{-1} = \begin{bmatrix} \frac{3}{2} & -\frac{1}{2} & -1 \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix},$$

$$x^{(3)} = x^{(2)} + \lambda v = \left(\frac{3}{2}, \frac{1}{2}, \frac{3}{2}\right)^T, \quad \pi = \left(\frac{1}{2}, \frac{1}{2}, 1\right)^T.$$

Since all potentials π_i are non-negative, then $x^* = x^{(3)} = \left(\frac{3}{2}, \frac{1}{2}, \frac{3}{2}\right)^T$ is an optimal solution to our LP, and $y^* = \left(\frac{1}{2}, \frac{1}{2}, 1, 0, 0, 0\right)^T$ is an optimal solution to the dual LP. □

3.2.1 How to Find a Feasible Basic Solution

A feasible basic set is one of the inputs to the *simplex* procedure. But how to find such a set? It is possible to transform the initial LP in order to obtain an equivalent LP for which a feasible basic set can be simply identified. There are several ways to perform such a transformation, but we will consider only one of them.

Again, let us consider LP (3.1). First, say, by the method of Gaussian elimination, we find a set I of n linearly independent rows of the constraint matrix A. Let $B = A_I$ and $\bar{x} = B^{-1} b_I$. If the vector of the residuals $b - A\bar{x}$ is non-negative, then we are finished: $\bar{x}$ is a feasible solution ($A\bar{x} \le b$), and I is a feasible basic set. Otherwise, we solve the following LP:

$$\begin{aligned} -x_{n+1} &\to \max, \\ Ax + a x_{n+1} &\le b, \\ 0 \le x_{n+1} &\le 1, \end{aligned} \tag{3.8}$$

where the components of the vector $a \in \mathbb{R}^m$ are defined by the rule: $a_i = 0$ if $A_i \bar{x} \le b_i$ and $a_i = b_i - A_i \bar{x} - 1$ otherwise. We obtain a feasible basic set $\hat{I}$ for (3.8) by adding to the set I the index, $m + 1$, of the inequality $x_{n+1} \le 1$. We also note that the basic set $\hat{I}$ determines the feasible basic solution $(\bar{x}, 1)$. If $x_{n+1} \ne 0$ in an optimal solution to (3.8), then (3.1) does not have feasible solutions. Otherwise, removing $m + 2$ (the index of $-x_{n+1} \le 0$) from an optimal basic set for (3.8), we obtain a feasible basic set for (3.1).

So, we can solve (3.1) in two steps (or phases): in the first step, we solve (3.8) to find a feasible basic set for (3.1), and in the second step, we solve (3.1). These two steps can be combined into one step, and we have to solve not two, but only one LP:

$$
\begin{aligned}
c^T x - M x_{n+1} &\to \max, \\
A x + a x_{n+1} &\le b, \\
0 \le x_{n+1} &\le 1,
\end{aligned}
\tag{3.9}
$$

where M is a sufficiently large number. We could estimate the value of M, but any theoretical estimate is usually too large, and it is not easy (for reasons of numerical stability) to use it in practice. Therefore in practice, the solution of (3.9) begins with a moderately large value of M. If it turns out that in the obtained optimal solution the component x_{n+1} is non-zero, then M is doubled and the solution of this LP is continued by the simplex method. This is repeated until either a solution with $x_{n+1} = 0$ is found or the value of M exceeds a certain threshold value. In the latter case, it is concluded that (3.1) does not have feasible solutions.

3.2.2 Pricing Rules

The calculation of the potential vector π in the simplex procedure is called *pricing*, or a *pricing operation*. Obviously, there can be many negative components π_i and, therefore, we need a rule for an unambiguous choice of index t. The following rules (strategies) are best known:

"first negative": $t = \min\{i:\ 1 \le i \le n,\ \pi_i < 0\}$;
"most negative": $t \in \arg\min\limits_{1\le i\le n} \pi_i$;
"steepest edge": $t \in \arg\min\limits_{1\le i\le n} \dfrac{\pi_i}{\|B^{-1}\mathbf{e}_i\|}$;
"maximum increase": $t \in \arg\min\limits_{1\le i\le n} \hat{\lambda}_i \pi_i$.

The meaning of these rules, with the exception for the steepest edge rule, must be clear from their names. When the steepest edge rule is used, the simplex-method moves from the current vertex to the next one along an edge which directing vector, $-B^{-1}\mathbf{e}_t$, forms the most acute angle, of value ϕ_t, with the objective (gradient) vector c:

$$\cos(\phi_t) = \frac{-c^T B^{-1}\mathbf{e}_t}{\|c\| \cdot \|B^{-1}\mathbf{e}_t\|} = \frac{1}{\|c\|} \cdot \frac{-\pi_t}{\|B^{-1}\mathbf{e}_t\|}.$$

The larger $\cos(\phi_t)$, the sharper the angle between the vectors $-B^{-1}\mathbf{e}_t$ and c is.

In practice, for many years the most negative rule (also known as Danzig's rule) prevailed. The first negative rule is the easiest to implement, but in comparison, say, with the most negative rule the number of iterations of the simplex method can increase substantially. The maximum increase rule requires computations that take too much time at each iteration and, therefore, this rule is not practical. The same could be said about the steepest edge rule until the formulas were found for recalculating the squares of the column norms.

Lemma 3.1. *Let I be a feasible basic set, $\hat{I}$ be the basic set determined by* (3.4) *for some $t \in \{1,\dots,n\}$, and let $B = A_I$, $\hat{B} = A_{\hat{I}}$ and*

$$\gamma_i \overset{\text{def}}{=} \|B^{-1}\mathbf{e}_i\|^2 = \mathbf{e}_i^T \left(B^{-1}\right)^T B^{-1}\mathbf{e}_i, \quad i = 1,\dots,n.$$

Then

$$\hat{\gamma}_i \overset{\text{def}}{=} \|\hat{B}^{-1}\mathbf{e}_i\|^2 = \begin{cases} \dfrac{1}{u_t^2}\gamma_t, & i = t, \\ \gamma_i - 2\dfrac{u_i}{u_t}\alpha_i + \dfrac{u_i^2}{u_t^2}\gamma_t, & i \neq t, \end{cases} \tag{3.10}$$

where $u^T = A_s B^{-1}$, $v = B^{-1}\mathbf{e}_t$, $\alpha^T = v^T B^{-1}$.

Proof. In view of (3.5), (3.6), and since

$$\left(I - \frac{1}{u_t}\mathbf{e}_t(u^T + \mathbf{e}_t^T)\right)\mathbf{e}_i = \begin{cases} \mathbf{e}_i - \dfrac{u_i}{u_t}\mathbf{e}_t, & i \neq t, \\ -\dfrac{1}{u_t}\mathbf{e}_t, & i = t, \end{cases}$$

we obtain

$$\begin{aligned} \hat{\gamma}_t &= (1/u_t^2)\gamma_t, \\ \hat{\gamma}_i &= \left(\mathbf{e}_i - \frac{u_i}{u_t}\mathbf{e}_t\right)^T \left(B^{-1}\right)^T B^{-1}\left(\mathbf{e}_i - \frac{u_i}{u_t}\mathbf{e}_t\right) \\ &= \mathbf{e}_i^T \left(B^{-1}\right)^T B^{-1}\mathbf{e}_i - 2\frac{u_i}{u_t}\mathbf{e}_i^T \left(B^{-1}\right)^T B^{-1}\mathbf{e}_t + \frac{u_i^2}{u_t^2}\mathbf{e}_t^T \left(B^{-1}\right)^T B^{-1}\mathbf{e}_t \\ &= \gamma_i - 2\frac{u_i}{u_t}\alpha_i + \frac{u_i^2}{u_t^2}\gamma_t, \quad i \neq t. \end{aligned}$$

□

3.3 Dual Simplex Method

In this section we consider another version of the simplex method known as the *dual simplex method*. This simplex method is called dual, because, solving an LP, it essentially repeats the work of the primal simplex method applied to the dual LP. It should also be noted that the dual simplex method is the main **LP** method in **MIP**.

Again, we consider an LP of the form (3.1). Let I be a dual feasible basic set and let $B = A_I$, $\bar{x} = B^{-1}b_I$, $\pi^T = c^T B^{-1} \geq 0$ and $\bar{y} = (\bar{y}_I = \pi, \bar{y}_{N\setminus I} = 0)$. Let us recall that $\bar{y}$ is a feasible solution to the dual LP (3.2). If a basic solution $\bar{x}$ is feasible, then it is optimal. Otherwise, there is an inequality $A_s x \leq b_s$, $s \notin I$, that is violated at $\bar{x}$. The dual objective function $b^T y$ decreases if we move from the point $\bar{y}$ along the ray $y(\theta) = \bar{y} - \theta v$ ($\theta \geq 0$), where, for $u^T = A_s B^{-1}$, the components of the vector v are determined by the rule

```
dual-simplex($c,A,b,I$); // $I$ is a dual feasible basic set
{
    $B^{-1} := A_I^{-1}$; $\pi^T := c^T B^{-1}$; $x := B^{-1} b_I$;
    while ($x \notin P(A,b)$) {
        chose an inequality $s$ violated at $x$: $A_s x > b_s$;
        $u^T := A_s B^{-1}$; // $u^T$ is the row $A_s$ written in the basis $A_I$
        if ($u \le 0$) {
            $y_s = 1$;
            $y_{I[j]} = -u_j, \quad j = 1,\dots,n$;
            $y_i = 0, \quad i \in \{1,\dots,m\} \setminus (I \cup \{s\})$;
            return (false,$y$); // there are no feasible solutions
        }
        $\lambda := \min\left\{\frac{\pi_i}{u_i} : i = 1,\dots,n;\ u_i > 0\right\}$;
        choose an index $t$ for which the value $\lambda$ is attained;
        $I[t] := s$;
        $\pi := \pi - \lambda u$; $\pi_t := \lambda$; // compute $\pi^T = c^T A_I^{-1}$
        $B^{-1} := B^{-1} I(t,u)^{-1}$; // compute $A_I^{-1}$
        $x := x + (b_s - A_s x) B^{-1} \mathbf{e}_t$; // compute $x = B^{-1} b_I$
    }
    return (true,$x, y = (y_I = \pi, y_{M \setminus I} = 0)$);
}
```

Listing 3.2. Dual Simplex Method

$$v_i = \begin{cases} -1, & i = s, \\ u_j, & i = I[j],\ j = 1,\dots,n, \\ 0, & i \in \{1,\dots,m\} \setminus (I \cup \{s\}). \end{cases}$$

Let $\hat{\theta}_s$ denote the maximum value of θ such that $y(\theta)$ is still a feasible solution to the dual LP, and let $\hat{y} = y(\hat{\theta}_s)$. Notice, that

$$\hat{\theta}_s = \frac{\|\hat{y} - \bar{y}\|}{\|v\|} = \frac{\|\hat{y} - \bar{y}\|}{1 + \|A_s B^{-1}\|} = \min_{\substack{1 \le i \le n, \\ u_i > 0}} \frac{\pi_i}{u_i},$$

and if

$$t \in \arg\min_{\substack{1 \le i \le n, \\ u_i > 0}} \frac{\pi_i}{u_i},$$

then $\hat{y} = (\hat{y}_{\hat{I}} = \hat{\pi}, \hat{y}_{N \setminus \hat{I}})$, where $\hat{\pi} = c^T A_{\hat{I}}^{-1}$, and the new dual feasible basic set $\hat{I}$ is defined by (3.4). In this case, the dual objective function decreases by the amount

$$\begin{aligned} b^T \hat{y} - b^T \bar{y} &= -\hat{\theta}_s (u^T b_I - b_s) = -\hat{\theta}_s (A_s B^{-1} b_I - b_s) \\ &= \hat{\theta}_s (b_s - A_s \bar{x}) = \frac{\|\hat{y} - \bar{y}\| (b_s - A_s \bar{x})}{\sqrt{1 + \|A_s B^{-1}\|^2}}. \end{aligned} \tag{3.11}$$

If all components of u are non-positive, then $\hat{\theta}_s = \infty$ and the dual objective function indefinitely decreases along the ray $\{y(\theta) : \theta \ge 0\}$. If (3.1) had a feasible solution x, then by (3.3) we would have $c^T x \le b^T y(\theta)$ for any positive θ. But since $c^T x$ is finite and $\lim_{\theta\to\infty} b^T y(\theta) = -\infty$, we conclude that (3.1) does not have feasible solutions.

A detailed description of the dual simplex method is presented in Listing 3.2. The input of the *dual-simplex* procedure is composed of a triple (c, A, b) describing an LP, and a dual feasible basic set I. The method terminates for two reasons.

1) The current dual feasible basic set I also becomes feasible. In this case, the output is a triple
$$(\textbf{true}, x, y = (y_I = \pi, y_{M\setminus I} = 0)),$$
where x is an optimal solutions to the primal LP (3.1), and y is an optimal solution to the dual LP (3.2).
2) If $u \le 0$, then (3.1) does not have feasible solutions. In this case, the procedure returns a pair (**false**,y), where y is a "certificate of infeasibility" (see Sect. 3.4).

3.3.1 Adding New Constraints and Changing Bounds

The dual simplex method has a specific feature that predetermined its wide use in **MIP**.

Suppose that we have already solved an instance of (3.1). And now we want to solve one of the following modifications of the just solved LP:

$$\max\{c^T x : \; Ax \le b, \; Hx \le \beta\}, \tag{3.12}$$

$$\max\{c^T x : \; Ax \le \tilde{b}\}. \tag{3.13}$$

If I is an optimal basic set for (3.1), then I will be a dual feasible basic set for both new LPs, (3.12) and (3.13), which allows us to use I as an initial basic set in the *dual-simplex* procedure. If the changes are small (just a few inequalities in $Hx \le \beta$, or $\|b - \tilde{b}\|$ is small enough), we can expect that the dual simplex method will need to perform relatively few iterations to build a solution to the modified program.

In a very similar way, the primal simplex method can be used for reoptimization when the objective function is changed or new columns (variables) are added. This is due to the fact that with such changes the primal feasibility of a feasible basic solution is preserved.

3.3.2 How to Find a Dual Feasible Basic Solution?

In practice, very often LPs are appear in the following most general two-sided form:

$$\max\{c^T x : \; b^1 \le Ax \le b^2, \; d^1 \le x \le d^2\}, \tag{3.14}$$

where $c, d^1, d^2 \in \mathbb{R}^n$, $b^1, b^2 \in \mathbb{R}^m$, A is a real $m \times n$-matrix, and x is an n-vector of variables. The *dual-simplex* procedure can be easily modified for solving the LPs in with two-sided constraints. To do this, let us number the inequalities in (3.14) as follows:

$$\begin{aligned} i: &\quad A_i x \le b_i^2,\\ -i: &\quad -A_i x \le -b_i^1,\\ m+j: &\quad x_j \le d_j^2,\\ -m-j: &\quad -x_j \le -d_j^1. \end{aligned}$$

Then we define $\bar{A}_i = A_i$, $\bar{b}_i = b_i^2$, $\bar{A}_{-i} = -A_i$, $\bar{b}_{-i} = -b_i^1$, and $\bar{A}_{m+j} = \mathbf{e}_j$, $\bar{b}_{m+j} = d_j^2$, $\bar{A}_{-m-j} = -\mathbf{e}_j$, $\bar{b}_{-m-j} = -d_j^1$ to rewrite the constraints of (3.14) in the form $\bar{A}x \le \bar{b}$. Since two row vectors $\bar{A}_i$ and $\bar{A}_{-i}$ are linearly dependent, the indices i and $-i$ cannot simultaneously be in any basic set.

The point x^0, with the coordinates

$$x_j^0 = \begin{cases} d_j^2, & c_j \ge 0,\\ d_j^1, & c_j < 0, \end{cases}$$

is an optimal solution to the following trivial LP:

$$\max\{c^T x : d^1 \le x \le d^2\}.$$

From what was said in Sect. 3.3.1, it follows that x^0 is a dual feasible basic solution for (3.14). The dual feasible basic set corresponding to x^0 is $I = \{m+j : c_j \ge 0\} \cup \{-m-j : c_j < 0\}$.

3.3.3 The Dual Simplex Method Is a Cutting Plane Algorithm

The dual simplex method for solving LP (3.1) can be considered as a *cutting plane algorithm*. Let us illustrate this with an example.

Example 3.2 *We need to solve the LP*

$$\begin{aligned} x_1 + 2x_2 &\to \max,\\ x_1 + \ x_2 &\le 4,\\ -x_1 + \ x_2 &\le 1,\\ -2x_1 - \ x_2 &\le -2,\\ 0 \le x_1 &\le 3,\\ 0 \le x_2 &\le 3. \end{aligned} \tag{3.15}$$

Solution. We begin with the dual feasible basic solution $x^{(0)} = (3,3)^T$, at which the objective function attains its maximum over the parallelepiped

$$P_0 = \{x \in \mathbb{R}^2 : \ 0 \le x_1 \le 3,\ 0 \le x_2 \le 3\} \quad \text{(see Fig. 3.3.a)}.$$

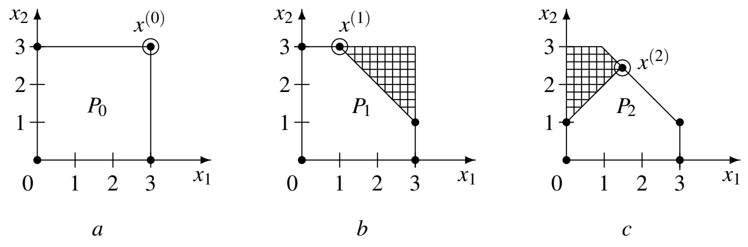

Fig. 3.3 Interpretation of the dual simplex method as a cutting plane algorithm

1. Since the point $x^{(0)}$ does not satisfy the first inequality from (3.15), we cut it off by the hyperplane $x_1 + x_2 = 4$ (Fig. 3.3.b). After this, we perform the iteration of the dual simplex method:

$$s = 1, \quad u = (1,1)^T, \quad \lambda = \min\{1,2\} = 1, \quad t = 1, \quad I = (1,5),$$

$$B^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}, \quad x^{(1)} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \quad \pi = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Note that $x^{(1)}$ is the maximizer of the objective function over the polytope

$$P_1 = \{x \in \mathbb{R}^2 : 0 \le x_1 \le 3,\ 0 \le x_2 \le 3,\ x_1 + x_2 \le 4\}.$$

2. Since $x^{(1)}$ violates the second inequality in (3.15), we cut it off using the hyperplane $-x_1 + x_2 = 1$ (Fig. 3.3.c). Then we perform the iteration of the dual simplex method:

$$s = 2, \quad u = (-1,2)^T, \quad \lambda = \frac{1}{2}, \quad t = 2, \quad I = (1,2),$$

$$B^{-1} = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \quad x^{(2)} = \frac{1}{2}\begin{pmatrix} 3 \\ 5 \end{pmatrix}, \quad \pi = \frac{1}{2}\begin{pmatrix} 3 \\ 1 \end{pmatrix}.$$

Note that $x^{(2)}$ is the maximizer of the objective function over the polytope

$$P_2 = \{x \in \mathbb{R}^2 : 0 \le x_1 \le 3,\ 0 \le x_2 \le 3,\ x_1 + x_2 \le 4,\ -x_1 + x_2 \le 1\}.$$

Since the point $x^{(2)}$ satisfies all constraints in (3.15), then it is an optimal solution to (3.15). □

3.3.4 Separation Rules

The search for a violated inequality $A_s x \le b_s$ in the dual simplex method is called *separation*. It is clear that a point x can violate many inequalities and, therefore, we need a rule for an unambiguous choice of index s. The following rules (strategies) are best known:

”first violated”: $s = \min\{i : A_i x > b_i,\ i \notin I\}$;
”most violated”: $s \in \arg\max_{i \notin I} (A_i x - b_i)$;
”steepest edge”: $s \in \arg\max_{i \notin I} \frac{A_i x - b_i}{\sqrt{1 + \|A_i B^{-1}\|^2}}$;
”maximum decrease”: $s \in \arg\max_{i \notin I} \hat{\theta}_i (A_i x - b_i)$.

We can say about all these rules almost the same as was said in Sect 3.2.2 about the corresponding pricing rules. In practice, various variations of the ”most violated” and ”steepest edge” rules are used. To make the separation based on the steepest edge rule practical, the formulas were obtained for recalculating the row norms of the matrices AB^{-1}.

Lemma 3.2. *Let I be a basic set, and let the basic set $\hat{I}$ be constructed according to (3.4) for some $t \in \{1,\dots,n\}$, $B = A_I$, $\hat{B} = A_{\hat{I}}$ and*

$$\eta_i \overset{\text{def}}{=} 1 + \|A_i B^{-1}\|^2 = 1 + A_i \left(B^{-1}\right)^T B^{-1} A_i^T, \quad i = 1,\dots,m.$$

Then

$$\hat{\eta}_i \overset{\text{def}}{=} 1 + \|A_i \hat{B}^{-1}\|^2 = \begin{cases} 2, & i \in \hat{I}, \\ \frac{1}{u_t^2} \eta_s, & i = I[t], \\ \eta_i - \frac{2}{u_t} (A_i \alpha)(A_i v) + \frac{(A_i v)^2}{u_t^2} \eta_s, & i \notin \hat{I} \cup I[t], \end{cases} \tag{3.16}$$

where $v = B^{-1} \mathbf{e}_t$, $u^T = A_s B^{-1}$, $\alpha = B^{-1} u$.

Proof. Since

$$\hat{B}^{-1} = B^{-1} \left(I - \frac{1}{u_t} \mathbf{e}_t (u^T + \mathbf{e}_t^T) \right) = B^{-1} - \frac{1}{u_t} v u^T - \frac{1}{u_t} v \mathbf{e}_t^T,$$
$$\eta_s = 1 + A_s B^{-1} \left(B^{-1}\right)^T A_s^T = 1 + u^T u = 1 + \|u\|^2,$$

we have

$$\begin{aligned} \hat{\eta}_i &= 1 + A_i \hat{B}^{-1} \left(\hat{B}^{-1}\right)^T A_i^T \\ &= 1 + A_i \left(B^{-1} - \frac{1}{u_t} v u^T - \frac{1}{u_t} v \mathbf{e}_t^T \right) \left(B^{-1} - \frac{1}{u_t} v u^T - \frac{1}{u_t} v \mathbf{e}_t^T \right)^T A_i^T \\ &= 1 + A_i B^{-1} \left(B^{-1}\right)^T A_i^T - \frac{2}{u_t} A_i B^{-1} u v^T A_i^T - \frac{2}{u_t} A_i B^{-1} \mathbf{e}_t v^T A_i^T + \\ &\quad \frac{1}{u_t^2} A_i v u^T u v^T A_i^T + \frac{2}{u_t^2} A_i v u^T \mathbf{e}_t v^T A_i^T + \frac{1}{u_t^2} A_i v \mathbf{e}_t^T \mathbf{e}_t v^T A_i^T \end{aligned}$$

$$
\begin{aligned}
&= \eta_i - \frac{2}{u_t}(A_i\alpha)(A_iv) - \frac{2}{u_t}(A_iv)^2 + \frac{\|u\|^2}{u_t^2}(A_iv)^2 + \frac{2}{u_t}(A_iv)^2 + \frac{1}{u_t^2}(A_iv)^2 \\
&= \eta_i - \frac{2}{u_t}(A_i\alpha)(A_iv) + \frac{(A_iv)^2}{u_t^2}\eta_s\,.
\end{aligned}
$$

To complete the proof, it suffices to note that for $i = I[t]$

$$\eta_i = 2, \quad A_i = \mathbf{e}_t B, \quad A_iv = \mathbf{e}_t BB^{-1}\mathbf{e}_t = 1, \quad A_i\alpha = \mathbf{e}_t BB^{-1}u = u_t,$$

and therefore $\hat{\eta}_i = (1/u_t^2)\eta_s$. □

3.4 Why an LP Does Not Have a Solution?

Having solved an LP on the computer and received a message that the problem did not have a solution, we would probably want to know the reason for this, in particular, in order to try to correct possible errors in our formulation.

It is said that an LP has no solution if: 1) its constraint system is infeasible (there are no feasible solutions) or 2) the objective value is unbounded over the set of feasible solutions.

To understand the reason for the inconsistency of a system of linear inequalities, let us consider a simple example:

$$
\begin{aligned}
2x_1 + 5x_2 + x_3 &\le 5, \\
x_1 + 2x_2 \phantom{{}+x_3} &\ge 3, \\
x_2 \phantom{{}+x_3} &\ge 0, \\
x_3 &\ge 0.
\end{aligned}
$$

Summing together the first inequality, the second multiplied by -2, and the third and the fourth, multiplied by -1, we obtain the false inequality $0 \le -1$. Hence, we can conclude that the system of inequalities in question is incompatible.

Strange as it may seem, but in the general case, a system of linear inequalities is incompatible if and only if the false inequality $0 \le -1$ can be derived from it. Let us give a more precise formulation of this criterion known as *Farkas' lemma*.

Lemma 3.3 (Farkas). *A system of inequalities $Ax \le b$ has no solutions if and only if there exists a vector $y \ge 0$ such that $y^T A = 0$ and $y^T b < 0$.*

Proof. We call a vector y that satisfies the conditions of Lemma 3.3 a *certificate of infeasibility* for the system of inequalities $Ax \le b$.

The necessity of the assertion of Lemma 3.3 is obvious. Let us prove the sufficiency. First we recall that the dual simplex method decides, that the system of inequalities of LP (3.1) is infeasible if, performing an iteration with a basic set I, it turns out that the vector $u = A_sB^{-1}$ is non-positive. At this point, we can determine a *certificate of infeasibility*, $y \in \mathbb{R}^m$, by the rule:

$$\begin{aligned} y_s &= 1, \\ y_{I[j]} &= -u_j, \quad j = 1,\dots,n, \\ y_i &= 0, \quad i \in \{1,\dots,m\} \setminus (I \cup \{s\}). \end{aligned}$$

Indeed,

$$\begin{aligned} y^T A &= -u^T B + A_s = -A_s B^{-1} B + A_s = 0, \\ y^T b &= -u^T b_I + b_s = -A_s B^{-1} b_I + b_s = -A_s x + b_s < 0. \end{aligned} \qquad \square$$

The objective function of LP (3.1) is not bounded if and only if there exists a feasible ray

$$\{x(\lambda) = x^0 + \lambda v :\ \lambda \ge 0\},$$

along which the objective function strictly increases, i.e., $c^T v > 0$, $Ax^0 \le b$ and $Av \le 0$. Such a pair (x^0, v) is called a *certificate of unboundedness* for LP (3.1). Note that the *simplex* procedure from Listing 3.1, after detecting that the objective function is unbounded, returns a certificate of unboundedness.

3.5 Duality in Linear Programming

Justifying the correctness of the primal and dual simplex methods, we established that there is a close relationship between the dual LPs (3.1) and (3.2). This relationship is expressed in the following theorems.

Theorem 3.1 (duality). *For the pair of dual LPs* (3.1) *and* (3.2) *the following alternatives take place:*

1) *both LPs have solutions and then* $z_P = z_D$*;*
2) *if one of the LPs,* (3.1) *or* (3.2)*, has a solution, and the other has not, the objective function of the LP that has a solution is unbounded;*
3) *both LPs have no solutions.*

Proof. We have actually proved assertions 1) and 2) when justifying the correctness of the primal and dual simplex methods. To prove the validity of assertion 3), it suffices to give an example of a pair of dual LPs for which this assertion holds:

$$\max\{-x :\ 0x \le -1\}, \quad \min\{-y :\ 0y = -1,\ y \ge 0\}. \qquad \square$$

Theorem 3.2. *Let $\bar{x}$ and $\bar{y}$ be feasible solutions of the primal,* (3.1)*, and dual,* (3.2)*, LPs respectively. Then the following conditions are equivalent:*

a) *$\bar{x}$ and $\bar{y}$ are optimal solutions to the primal and dual LPs;*
b) $c^T \bar{x} = b^T \bar{y}$*;*
c) ***(complementary slackness condition)***

$$\bar{y}^T (b - A\bar{x}) = 0 \quad \textit{and} \quad \bar{x}^T (c - A^T \bar{y}) = 0.$$

Proof. The equivalence of conditions a) and b) follows from the duality theorem. Let us prove the equivalence of conditions b) and c). Taking into account the inequalities $A\bar{x} \le b$ and $\bar{y}^T A \ge c^T$, we have

$$c^T \bar{x} \le \bar{y}^T A\bar{x} \le \bar{y}^T b = c^T \bar{x}.$$

Therefore

$$c^T \bar{x} = \bar{y}^T A\bar{x} = \bar{y}^T A\bar{x} = \bar{y}^T b,$$

hence

$$\bar{x}^T (c - A^T \bar{y}) = 0 \quad \text{and} \quad \bar{y}^T (b - A\bar{x}) = 0. \qquad \square$$

Informally, we say that two LPs are dual to each other if all the statements of Theorems 3.1 and 3.2 hold for them. A formal rule for writing the dual LP for a given LP is presented in Exercise 3.1.

3.6 Linear Programs With Two-Sided Constraints

As we noted in Sect. 3.3.2, in practice it is most convenient to write LPs in Form (3.14) with two-sided constraints. It is not at all necessary to reduce such an LP to the canonical form (3.1). We can easily modify both, primal and dual, simplex methods for solving LPs with two-sided constraints.

Now a basis is defined by a quadruple $(I, b; J, d)$, where

- $I \subseteq M \stackrel{\text{def}}{=} \{1, \dots, m\}$ is a *row basic set*;
- $b \in \mathbb{R}^m$, b_i is equal to b_i^1 or b_i^2 for $i \in I$;
- $J \subseteq N \stackrel{\text{def}}{=} \{1, \dots, n\}$ is a *column basic set*;
- $d \in \mathbb{R}^n$, d_j is equal to d_j^1 or d_j^2 for $j \in N \setminus J$.

The basis $(I, b; J, d)$ uniquely determines

- basic matrix $B = A_I^J$,
- basic solution $\bar{x} = (\bar{x}_J, \bar{x}_{N\setminus J}) = \left(B^{-1}\left(b_I - A_I^{N\setminus J} d_{N\setminus J}\right), d_{N\setminus J}\right)$,
- vectors of shadow prices, $\bar{y}^T = (\bar{y}_I, \bar{y}_{M\setminus I}) = (c_J^T B^{-1}, 0)$, and reduced costs, $\bar{c} = c - \bar{y}^T A$.

The basic solution $\bar{x}$ is feasible if

$$d_j^1 \le \bar{x}_j \le d_j^2, \;\; j \in J,$$
$$b_i^1 \le A_i \bar{x} \le b_i^2, \;\; i \in M \setminus I.$$

The feasible basic solution $\bar{x}$ is optimal if the following conditions are satisfied:

- for $i \in I$, if $b_i = b_i^2 > b_i^1$, then $\bar{y}_i \ge 0$, and if $b_i = b_i^1 < b_i^2$, then $\bar{y}_i \le 0$;
- for $j \in N \setminus J$, if $d_j = d_j^2 > d_j^1$, then $\bar{c}_j \ge 0$, and if $d_j = d_j^1 < d_j^2$, then $\bar{c}_j \le 0$.

For the LP in canonical form, the pivot operation is to substitute a non-basic row for a basic one. For the LP with two-sided constraints, the pivot operation is more complicated. Recalling the indexing of two-sided inequalities introduced in Sect. 3.3.2, which made it possible to transform the LP with two-sided constraints into an LP in canonical form, it is not difficult to imagine that the following options are possible:

- substitution of a non-basic row for a basic one if $|s| \le m$ and $|I[t]| \le m$;
- substitution of a non-basic column for a basic one if $|s| \ge m$ and $|I[t]| \ge m$;
- addition of a non-basic row and a non-basic column if $|s| \le m$ and $|I[t]| \ge m$;
- deletion of a basic row and a basic column if $|s| \ge m$ and $|I[t]| \le m$.

In particular, for the *LP in standard form*

$$\max\{c^T x : \ Ax = b, \ x \ge 0\},$$

considered in most **LP** manuals, the row basic set I contains all m rows, and each pivot operation always consists in substituting a non-basic column for a basic one.

3.7 Notes

The first who proposed an algorithm for solving a general LP was L.V. Kantorovich — Nobel Prize winner in Economics in 1975 — (see Exercise 3.11). But still J. Danzig is deservedly considered to be the father of linear programming for his invention of the simplex method. A book of J. Danzig [42] remains a classic book on linear programming. The author's views on linear programming have changed significantly after studying an excellent brochure of L.G. Khachijan [81].

Those who wants to learn more about linear programming we refer to one of the sources [37, 102, 110, 117, 122, 134].

***Sects.* 3.2.2 and 3.3.4.** The rules for recalculating column and row norms were obtained in [57]. The computational efficiency of using the steepest edge rules was proved in [51].

***Sect.* 3.8.** The statements of Exercises 3.11 and 3.14 were taken respectively from [122] and [78]. The DEA method described in Exercise 3.13 was proposed in [33].

3.8 Exercises

3.1. The rules for writing the dual LP for a given LP are presented in Table 3.1. Prove that the statements of Theorems 3.1 and 3.2 are also true for the pair of LPs from this table.

Hint. Convert the primal LP to the canonical form, write down the dual to the obtained LP, compare the new pair of dual LPs with the original pair from the table.

Table 3.1 Dual LPs

Primal LP	Dual LP
$z_P = \max c^T x$	$z_D = \min b^T y$
$A_i x \le b_i,\ i \in M_1$	$y_i \ge 0,\ i \in M_1$
$A_i x = b_i,\ i \in M_2$	$y_i \in \mathbb{R},\ i \in M_2$
$A_i x \ge b_i,\ i \in M_3$	$y_i \le 0,\ i \in M_3$
$x_j \ge 0,\ j \in N_1$	$y^T A^j \ge c_j,\ j \in N_1$
$x_j \in \mathbb{R},\ j \in N_2$	$y^T A^j = c_j,\ j \in N_2$
$x_j \le 0,\ j \in N_3$	$y^T A^j \le c_j,\ j \in N_3$

3.2. Using the rules from Table 3.1, write down the dual LPs for the following LPs:

a)
$$\begin{aligned} 2x_1 - 4x_2 + 3x_3 &\to \max,\\ x_1 + x_2 - x_3 &= 9,\\ -2x_1 + x_2 &\le 5,\\ x_1 - 3x_3 &\ge 4,\\ x_1 &\ge 0,\\ x_3 &\le 0; \end{aligned}$$

b)
$$\begin{aligned} 5x_1 - x_2 + 4x_3 &\to \max,\\ x_1 + x_2 + x_3 &= 12,\\ 3x_1 - 2x_3 &\ge 1,\\ x_2 - x_3 &\le 2,\\ x_1, x_3 &\ge 0; \end{aligned}$$

c) Formulation (1.15) of the transportation problem.

3.3. Prove that the set $X_{r,\alpha}$ from Exercise 1.2 is the projection, onto the space of x-variables, of the solution set of the system of inequalities

$$\begin{aligned} r\lambda + \sum_{i=1}^{n} y_i &\le \alpha,\\ x_i - y_i - \lambda &\le 0, \quad i = 1,\dots,n,\\ y_i &\ge 0, \quad i = 1,\dots,n, \end{aligned}$$

with $2n+1$ variables ($\lambda \in \mathbb{R}$, $x, y \in \mathbb{R}^n$) and $2n+1$ constraints.

Hint. First, show that $x \in X_{r,\alpha}$ if and only if

$$\alpha \ge \max\left\{ \sum_{i=1}^{n} x_i v_i : \sum_{i=1}^{n} v_i = r,\ 0 \le v_i \le 1,\ i = 1,\dots,n \right\}.$$

Then write down the dual to the LP in the right-hand side of this inequality.

3.4. Consider the LP

$$\max\left\{ \sum_{j=1}^{n} c_j x_j \,:\, \sum_{j=1}^{n} a_j x_j \le b,\ 0 \le x_j \le u_j,\ j = 1,\dots,n \right\} \tag{3.17}$$

with $c_j, a_j > 0$ for $j = 1,\dots,n$. Let a permutation $\pi : \{1,\dots,n\} \to \{1,\dots,n\}$ be such that

$$\frac{c_{\pi(1)}}{a_{\pi(1)}} \ge \frac{c_{\pi(2)}}{a_{\pi(2)}} \ge \dots \ge \frac{c_{\pi(n)}}{a_{\pi(n)}},$$

and let an integer r be chosen to satisfy

$$\sum_{j=1}^{r-1} a_{\pi(j)}\, u_{\pi(j)} \le b \quad \text{and} \quad \sum_{j=1}^{r} a_{\pi(j)}\, u_{\pi(j)} > b.$$

Show that the components of an optimal solution to (3.17) are defined by the rule:

$$\begin{aligned} x^*_{\pi(j)} &= u_{\pi(j)}, \quad j = 1,\dots,r-1, \\ x^*_{\pi(r)} &= \frac{b - \sum_{j=1}^{r-1} a_{\pi(j)}\, u_{\pi(j)}}{a_{\pi(r)}}, \\ x^*_{\pi(j)} &= 0, \quad j = r+1,\dots,n. \end{aligned}$$

3.5. Can we solve LPs using a computer program that solves the systems of linear inequalities $Ax \le b$?

3.6. Use Theorem 3.1 to prove the following important result from game theory.

Theorem 3.3 (von Neumann). *For every real $m \times n$-matrix A,*

$$\max_{x\in\Sigma_m} \min_{1\le j\le n} \sum_{i=1}^m a_{ij} x_i = \min_{y\in\Sigma_n} \max_{1\le i\le m} \sum_{j=1}^n a_{ij} y_j\,,$$

where Σ_m denotes an m-dimensional simplex $\left\{x \in \mathbb{R}^m_+ :\ \sum_{i=1}^m x_i = 1\right\}$.

3.7. *Arbitrage*. We have at our disposal n financial assets, the price of j-th of them at the beginning of the investment period is p_j. At the end of the investment period, the price of asset j is a random variable v_j. Suppose that m scenarios (outcomes) are possible at the end of the investment period, and then v_j is a discrete random variable. Let v_{ij} be the value of v_j when scenario i occurs. From the elements v_{ij}, we compose the $m \times n$-matrix $V = [v_{ij}]$.

A *trading strategy* is represented by a vector $x = (x_1,\dots,x_n)^T$: if $x_j > 0$, then we buy x_j units of asset j, and if $x_j < 0$, then $-x_j$ units of asset j are sold. A trading strategy is called an *arbitrage*, if it allows us to earn today without any risk of losses at the end of the investment period:

$$p^T x < 0, \tag{3.18}$$

$$Vx \ge 0. \tag{3.19}$$

The strict inequality (3.18) means that at the beginning of the investment period we get more than spend. And the validity of all inequalities $\sum_{j=1}^n v_{ij} x_j \ge 0$ from (3.19) means that the reverse trading strategy, $-x$, will not be loss-making at the end of the period in any of m possible scenarios.

Since prices adjust very quickly on the market, an opportunity to earn from an arbitrage also disappears very quickly. Therefore, in mathematical financial models, it is often assumed that arbitrage does not exist. Prove the following statements.

a) There exists no arbitrage if and only if the system $V^T y = p,\ y \geq 0$ is consistent.
b) If for fixed prices $p_1, \dots, p_{j-1}, p_{j+1}, \dots, p_n$ both LPs

$$\begin{aligned} p_j^{\min} &= \min\{p_j : V^T y = p,\ y \geq 0,\ p_j \geq 0\}, \\ p_j^{\max} &= \max\{p_j : V^T y = p,\ y \geq 0,\ p_j \geq 0\} \end{aligned}$$

have solutions, then there exists no arbitrage if and only if $p_j^{\min} \leq p_j \leq p_j^{\max}$.

3.8. *Cycling* in a simplex method is a situation where a sequence of basic sets repeats cyclically. Cycling can occur only when solving degenerate LPs (see Sect. 3.1 for the definition of degeneracy). In practice cycling occurs often when solving combinatorial optimization problems.

Solve the following LP

$$\begin{array}{rrrrrl} & \frac{3}{4}x_1 - & 20x_2 + & \frac{1}{2}x_3 - & 6x_4 & \to \max, \\ 1: & \frac{1}{4}x_1 - & 8x_2 - & x_3 + & 9x_4 & \leq 0, \\ 2: & \frac{1}{2}x_1 - & 12x_2 - & \frac{1}{2}x_3 + & 3x_4 & \leq 0, \\ 3: & & & x_3 & & \leq 1, \\ 4: & -x_1 & & & & \leq 0, \\ 5: & & -x_2 & & & \leq 0, \\ 6: & & & -x_3 & & \leq 0, \\ 7: & & & & -x_4 & \leq 0 \end{array}$$

by the primal simplex method starting with the basic set $I = (4,5,6,7)$ and using the following rules for resolving ambiguities when selecting rows for entering and leaving the basic set:

a) the entering row is the row $I[s]$ for $s \in \arg\min_{1 \leq i \leq n} \pi_i$;
b) the leaving row has the minimum index, t, among the row indices on which the value of λ is attained (see Listing 3.1).

3.9. In the literature, several rules were proposed for eliminating cycling in the simplex algorithms. Perhaps the most useful in practice is the *lexicographic rule*. A nonzero vector $x \in \mathbb{R}^n$ is said to be *lexicographically positive* if its first non-zero component is positive. A vector x is *lexicographically greater* than a vector $y \in \mathbb{R}^n$ if the vector $x - y$ is lexicographically positive. Prove the following theorem, which conveys the essence of the lexicographic rule as applied to the dual simplex method.

Theorem 3.4. *Suppose that when the dual simplex method starts all columns of the matrix*

$$A(I) \stackrel{\text{def}}{=} \begin{bmatrix} c^T \\ A \end{bmatrix} A_I^{-1}$$

are lexicographically positive. Then, during the execution of the algorithm, the columns of the matrix $A(I)$ remain lexicographically positive, the vector of residuals

$$r(x) \stackrel{\text{def}}{=} \begin{pmatrix} -c^T x \\ b - Ax \end{pmatrix}$$

strictly lexicographically increases from iteration to iteration, and the method terminates after a finite number of iterations if the row $I[t]$ leaving the basic set is selected according to the following rule:

$$t \in \arg \operatorname{lexmin} \left\{ \frac{A(I)^j}{u_j} : u_j > 0,\ j = 1, \dots, n \right\}.$$

Here the operator lexmin *means the choice of the lexicographically minimal vector.*

3.10. *Overdetermined systems of linear equations*. Given a real $m \times n$-matrix A and a vector $b \in \mathbb{R}^m$. If $m > n$, then the system $Ax = b$ may have no solutions. In such cases, as a solution to the system, we seek a vector x, for which the vector of residuals, $Ax - b$, has the minimum norm. In practice, three norms, l_2, l_1 and l_∞, are most often used. Depending on the type of norm, one of the following unconditional optimization problem must be solved:

$$\|Ax - b\|^2 = \sum_{i=1}^{m} (A_i x - b_i)^2 \to \min, \tag{3.20}$$

$$\|Ax - b\|_1 = \sum_{i=1}^{m} |A_i x - b_i| \to \min, \tag{3.21}$$

$$\|Ax - b\|_\infty = \max_{1 \le i \le m} |A_i x - b_i| \to \min. \tag{3.22}$$

Problem (3.20) is solved simply: its solutions are solutions to the system of linear equations $A^T A x = A^T b$. Two other problems, which are not smooth optimization problems, are more difficult to solve. Formulate (3.21) and (3.22) as LPs.

3.11. Show that each LP can be reduced to the LP

$$\begin{aligned} \lambda &\to \max, \\ \sum_{i=1}^{m} t_{ij} &= 1, \quad i = 1, \dots, n, \\ \sum_{j=1}^{m} \sum_{i=1}^{n} a_{ijk} t_{ij} &= \lambda, \quad k = 1 \dots, q, \\ t_{ij} &\ge 0, \quad i = 1, \dots, m;\ j = 1, \dots, n, \end{aligned}$$

which was investigated by L.V. Kantorovich. This LP admits the following interpretation. To produce a unit of some final product, one unit of each of q intermediate products is used. There are n machines that can perform m tasks. When machine i performs task j then a_{ijk} units of intermediate product k are produced per one shift. If t_{ij} is the fraction of time when machine i performs task j, then λ is the number of final product units produced.

3.12. *Hypothesis testing*. Let X be a discrete random variable taking values from the set $\{1,\dots,n\}$ and having a probability distribution that depends on the value of a parameter $\theta \in \{1,\dots,m\}$. The probability distributions of X for m possible values of θ are given by an $n \times m$-matrix P with elements $p_{kj} = \mathbb{P}(X = k|\theta = i)$, i.e., the i-th column of P defines the probability distribution for X provided that $\theta = i$.

We consider the problem of estimating the value of parameter θ by observing (sampling) values of X. In other words, a value of X is generated for one of the m possible distributions (values of θ), and we need to determine which distribution (value of θ) was used in this case. The values of θ are called *hypotheses*, and guessing, which of m hypotheses is true, is called *hypothesis testing*.

A *probabilistic classifier* for θ is a discrete random variable $\hat{\theta}$, which depends on the observed value of X and takes values from $\{1,\dots,m\}$. Such a classifier can be represented by an $m \times n$-matrix T with the elements $t_{ik} = \mathbb{P}(\hat{\theta} = i|X = k)$. If we observe the value $X = k$, then the classifier with probability t_{ik} chooses the value $\hat{\theta} = i$ as an estimate of the parameter θ. The quality of the classifier can be determined by the $m \times m$-matrix $D = TP$ with the elements $d_{ij} = \mathbb{P}(\hat{\theta} = i|\theta = j)$, i.e., d_{ij} is the probability of predicting $\hat{\theta} = i$ when $\theta = j$.

It is necessary to determine a probabilistic classifier for which the maximum of the probabilities of classification errors,

$$1 - d_{ii} = \sum_{j \neq i} d_{ij} = \mathbb{P}(\hat{\theta} \neq i|\theta = i), \quad i = 1,\dots,m,$$

is minimum. Formulate this problem as an LP.

3.13. *Data Envelopment Analysis* (*DEA*) is used to compare the performance of a number of similar service units (bank branches, restaurants, educational institutions, health care and many others). DEA does not require a cost valuation of the services provided. Suppose that there are $n+1$ departments that are numbered from 0 to n. For some test period, department i ($i = 0,1,\dots,n$) used r_{ij} units of resource j ($j = 1,\dots,m$), and rendered s_{ik} services of type k ($k = 1,\dots,l$). The efficiency of department i is estimated by the ratio

$$E_i(u,v) \stackrel{\text{def}}{=} \frac{\sum_{k=1}^{l} s_{ik} u_k}{\sum_{j=1}^{m} r_{ij} v_j},$$

where u_k and v_j are weights that need to be determined.

To compare department 0 with the other departments, the following problem of fractional linear programming is solved:

$$\gamma_0 = \max\{E_0(u,v) : E_i(u,v) \le 1 \text{ for } i = 1,\dots,n,\ u \in \mathbb{R}^l_+,\ v \in \mathbb{R}^m_+\}. \tag{3.23}$$

If $\gamma_0 < 1$, then department 0 does not work efficiently, and it should adopt the experience of other departments i for which $E_i(u^*,v^*) = 1$, where (u^*,v^*) is an optimal solution to (3.23).

Formulate (3.23) as an LP.

3.14. In a *Markov decision process* (with a finite number of states and discrete time), if at a given period of time the system is in state i from a finite set of states S, we choose an action a from a finite set of actions $A(i)$, which provides profit r_{ia}; then the system passes to a new state $j \in S$ with a given probability p_{iaj} that depends on the action a undertaken in state i. Our goal is to find a decision strategy that maximizes the expected average profit for one period during an infinite time horizon. It is known that an optimal strategy can be sought among stationary strategies. A *stationary strategy* (or *policy*) π, every time the system is in state i, prescribes to undertake the same action $\pi(i) \in A(i)$. We also note that there is a stationary strategy, which is optimal for all initial states of the system, and therefore, such a strategy is called *uniform*.

Let (x^*, y^*) be an optimal solution to the following LP

$$\begin{aligned}
&\sum_{i\in S}\sum_{a\in A(i)} r_{ia}x_{ia} \to \max,\\
&\sum_{i\in S}\sum_{a\in A(i)} (\delta_{ij} - p_{iaj})x_{ia} = 0, \quad j \in S,\\
&\sum_{a\in A(j)} x_{ja} + \sum_{i\in S}\sum_{a\in A(i)} (\delta_{ij} - p_{iaj})y_{ia} = \frac{1}{|S|}, \quad j \in S,\\
&x_{ia},\, y_{ia} \ge 0, \quad a \in A(i),\ i \in S.
\end{aligned} \tag{3.24}$$

Here δ_{ij} is the Kronecker delta ($\delta_{ij} = 1$ if $i = j$, and $\delta_{ij} = 0$ if $i \neq j$). Let us define

$$A^*(i) \overset{\text{def}}{=} \begin{cases} \{a:\ x^*_{ia} > 0\}, & \text{if } \sum_{a\in A(i)} x^*_{ia} > 0,\\ \{a:\ y^*_{ia} > 0\}, & \text{if } \sum_{a\in A(i)} x^*_{ia} = 0. \end{cases}$$

Show that any stationary strategy π^* such that $\pi^*(i) \in A^*(i)$ is optimum.

3.15. A firm produces some products using a number of identical machine. At the beginning of each week, each machine is in one of the following states: excellent, good, average, or bad. Working a week, the machine generates the following income depending on its state: \$100 in excellent state, \$80 in good state, \$50 in average state, and \$10 in bad state. After inspecting each of the machines at the end of the week, the firm can decide to replace it with a new one in excellent state. A new machine costs \$200. The state of any machine deteriorates over time as shown in the table below.

State	Excellent	Good	Average	Bad
Excellent	0.7	0.3	0.0	0.0
Good	0.0	0.7	0.3	0.0
Average	0.0	0.0	0.6	0.4
Bad	0.0	0.0	0.0	1.0

It is necessary to determine a strategy of replacing machines that generates the maximum per week profit in the long run. Write down LP (3.24) for this example, and solve that LP using your favorite **LP** solver.

Chapter 4
Cutting Planes

As noted in Sect. 1.5, we can strengthen a **MIP** formulation by adding to it new inequalities valid for all feasible solutions, but invalid for the relaxation polyhedron. Such inequalities are called *cuts*. We know that cuts can be added when formulating (reformulating) the problem. New inequalities can also be added in the process of solving the problem. The *cutting plane method* for solving MIPs can be viewed as an extension of the dual simplex method in which the separation procedure (for searching violated inequalities) is not limited to verifying the constrains of the current formulation, but it can also generate new cuts.

In this chapter, we study cuts that are used for solving general IPs and MIPs. Here we also demonstrate the use of these cuts in the cutting plane algorithms. Any cuts are useful in practice only if they can be generated (computed) by very fast separation procedures. In the last section of this chapter we discuss the relationship between the optimization and separation problems.

4.1 Cutting Plane Algorithms

Let us demonstrate how a cutting plane algorithm works on the following simple example:

$$\begin{aligned} x_1 + 2x_2 &\to \max,\\ 3x_1 + 2x_2 &\le 9,\\ x_2 &\le 2,\\ x_1, x_2 &\in \mathbb{Z}_+ . \end{aligned} \tag{4.1}$$

First, we solve the relaxation LP for IP (4.1), which is obtained from (4.1) by allowing integer variables to take also real values. The feasible polytope, P_0, of this relaxation LP and its optimal solution, $x^{(0)} = (\frac{5}{3}, 2)^T$, are depicted in Fig. 4.1.a. Since $x^{(0)}$ is not an integer point, it is not a solution to (4.1).

A *cutting plane* or simply *cut* is an inequality that ”cuts off” the point $x^{(0)}$ from the set

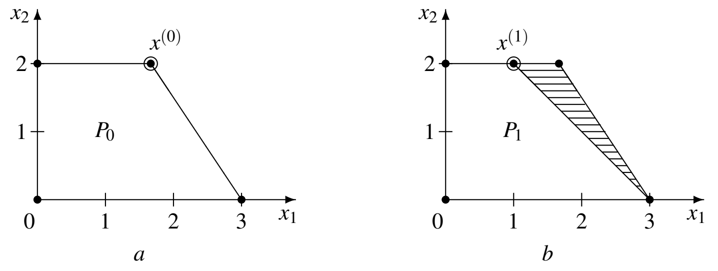

Fig. 4.1 Geometrical interpretation of cutting planes

$$X = \{x \in \mathbb{Z}^2_+ : \; 3x_1 + 2x_2 \le 9, \; x_2 \le 2\}$$

of feasible solutions to (4.1). There are several ways to build (generate) cuts. Some of these methods will be considered later in this and the next chapters.

Here we cut off the point $x^{(0)}$, starting from a very simple observation that both inequalities, $3x_1 + 2x_2 \le 9$ and $x_2 \le 2$, cannot simultaneously be satisfied as equalities at the points from X. Therefore, the inequality

$$3x_1 + 2x_2 + x_2 \le 9 + 2 - 1, \quad \text{or} \quad 3x_1 + 3x_2 \le 10$$

holds for X, but not for $x^{(0)}$ ($3 \cdot \frac{5}{3} + 3 \cdot 2 = 11 > 10$). We can strengthen this inequality if we first divide it by 3, and then round down the right-hand side:

$$x_1 + x_2 \le \left\lfloor \frac{10}{3} \right\rfloor = 3.$$

So, we have found the cut $x_1 + x_2 \le 3$, and now we add it to (4.1) as an additional constraint. As a result, a piece of the relaxation polytope P_0 is cut off. Let us note that this cut off area (the shadow area in Fig. 4.1.b) does not contain integer points, and, therefore, no feasible point from X has been cut off, and X is contained in the feasible polytope P_1 of the new relaxation LP (with added cut). An optimal solution to this LP is the point $x^{(1)} = (1,2)^T$, which is integer and, therefore, $x^{(1)}$ is an optimal solution to (4.1).

4.2 Chvátal-Gomory Cuts

Let A be a real $m \times n$-matrix, $b \in \mathbb{R}^n$, and let us suppose that the polyhedron $P(A,b)$ belongs to $\mathbb{R}^n_+$. Surprisingly, but we can build all inequalities that define the convex hull of the set $P(A,b) \cap \mathbb{Z}^n$ using a procedure based on the following very simple observation.

Rounding principle: the inequality $x \le \lfloor b \rfloor$ is valid for the set $\{x \in \mathbb{Z} : \; x \le b\}$.

Proceeding from this principle, we present the following very general procedure for constructing cuts for the set $X = P(A,b) \cap \mathbb{Z}^n$.

Chvátal-Gomory procedure:

1) choose $u \in \mathbb{R}^m_+$;
2) since $u \geq 0$, the inequality $u^T A x \leq u^T b$ is valid for X;
3) since $x \geq 0$, the inequality $\sum_{j=1}^n \lfloor u^T A^j \rfloor x \leq u^T b$ is valid for X;
4) since x is integer, the inequality

$$\sum_{j=1}^{n} \lfloor u^T A^j \rfloor x \leq \lfloor u^T b \rfloor \tag{4.2}$$

is valid for X.

Theorem 4.1. *If $P(A,b) \subseteq \mathbb{R}^n_+$, then each valid for $P(A,b) \cap \mathbb{Z}^n$ inequality can be obtained by applying the Chvátal-Gomory procedure a finite number of times.*

Theorem 4.1 motivates the following definition. For a given set $X = P(A,b) \cap \mathbb{Z}^n$ the *Chvátal rank* of a valid for X inequality $\alpha^T x \leq \beta$ is the minimum number of applications of the Chvátal-Gomory procedure necessary to obtain an inequality that is not weaker than $\alpha^T x \leq \beta$. The *Chvátal rank* of the set X is the maximum Chvátal rank of an inequality in the description of its convex hull.

For example, consider an integer set

$$X = \{x \in \mathbb{Z}^4_+ : \ 2x_1 + x_2 + x_3 + x_4 \leq 4,\ x_2 + x_3 \leq 1\ x_3 + x_4 \leq 1,\ x_2 + x_4 \leq 1\}.$$

The inequality $x_2 + x_3 + x_4 \leq 1$ is a Chvátal-Gomory cut (of Chvátal rank 1) with the vector of multipliers $\frac{1}{2}(0,1,1,1)^T$. Adding this cut to the system of inequalities describing X as the fifth inequality and using the multiplier vector $\frac{1}{2}(1,0,0,0,1)$, we obtain the cut $x_1 + x_2 + x_3 + x_4 \leq 2$, which Chatal rank is 2.

It should also be noted that many classes of cuts for well-known combinatorial sets are Chvátal-Gomory cuts of Chvátal rank 1.

Example 4.1 *Given a graph $G = (V,E)$, each edge $e \in E$ is assigned a costs c_e. A subset of edges $M \subseteq E$ is called a matching if no two edges from M have a common vertex. The maximum weighted matching problem is to find a matching M of maximum cost $c(M) \overset{\text{def}}{=} \sum_{e \in M} c_e$ (see also Sect.* 2.1*). Introducing binary variables x_e for $e \in E$ with $x_e = 1$ only if $e \in M$, we formulate this combinatorial optimization problem as the following IP:*

$$\sum_{e \in E} c_e x_e \to \max, \tag{4.3a}$$

$$\sum_{e \in E(v,V)} x_e \leq 1, \quad v \in V, \tag{4.3b}$$

$$x_e \in \{0,1\}, \quad e \in E, \tag{4.3c}$$

where $E(S,T)$ denotes the set of edges with one end in S, and the other in T. The convex hull of vectors x satisfying (4.3b) and (4.3c) is called a matching polytope. We need to show that, for each subset $S \subseteq V$ of odd cardinality, the inequality

$$\sum_{e\in E(S,S)} x_e \le \frac{|S|-1}{2} \tag{4.4}$$

is valid for the matching polytope.

Solution. Let $S \subseteq V$ and $|S|$ be odd. Summing up the inequalities

$$\sum_{e\in E(v,V)} x_e \le 1, \quad v \in S,$$

we have

$$\sum_{e\in E(S,V)} x_e = 2\sum_{e\in E(S,S)} x_e + \sum_{e\in E(S,V\setminus S)} x_e \le |S|.$$

Dividing the result by 2, we obtain the inequality

$$\sum_{e\in E(S,S)} x_e + \frac{1}{2}\sum_{e\in E(S,V\setminus S)} x_e \le \frac{|S|}{2}.$$

Rounding down first the right and then the left-hand sides of this inequality, we derive (4.4). □

From a practical point of view, a class of cuts is useful only if one can efficiently solve the separation problem for this class. With respect to Ineqs. (4.2), the *separation problem* is formulated as follows:

Given a point $\tilde{x} \in \mathbb{R}^n_+$; find $u \in \mathbb{R}^n_+$ such that the corresponding inequality in (4.2) *is violated at $\tilde{x}$, or prove that $\tilde{x}$ satisfies all inequalities from* (4.2).

It is known that this separation problem is **NP**-hard. But, on the other hand, there are a number of important special cases when it is solved efficiently. We consider only one of such special cases when $\tilde{x}$ is a vertex of the relaxation polyhedron $P(A,b) \subseteq \mathbb{R}^n_+$.

Theorem 4.2. *Let $\tilde{x} = A_I^{-1} b_I$ be a feasible basic solution to the system of inequalities $Ax \le b$ (a vertex of $P(A,b)$), where $I \subseteq \{1,\dots,m\}$ is a feasible basic set. Suppose that $\tilde{x}_i \notin \mathbb{Z}$, and denote by $v^T = \mathbf{e}_i^T A_I^{-1}$ the i-th row of the inverse basic matrix. Then the inequality (Gomory cut)*

$$\sum_{j=1}^{n} \left\lfloor (v - \lfloor v \rfloor)^T A_I^j \right\rfloor x_j \le \left\lfloor (v - \lfloor v \rfloor)^T b_I \right\rfloor \tag{4.5}$$

is valid for $P(A,b) \cap \mathbb{Z}^n$ but is violated at $\tilde{x}$.

Proof. Since (4.5) is a Chvátal-Gomory cut, it is valid for X. Let us prove that this inequality is violated at $\tilde{x}$:

$$\sum_{j=1}^{n} \left\lfloor (v - \lfloor v \rfloor)^T A_I^j \right\rfloor \tilde{x}_j - \lfloor (v - \lfloor v \rfloor) b_I \rfloor =$$

$$\tilde{x}_i - \sum_{j=1}^{n} \lfloor v \rfloor^T A_I^j \tilde{x}_j - \left\lfloor \tilde{x}_i - \lfloor v \rfloor^T b_I \right\rfloor =$$

$$\tilde{x}_i - \lfloor v \rfloor^T b_I - \left\lfloor \tilde{x}_i - \lfloor v \rfloor^T b_I \right\rfloor > 0. \qquad \square$$

Example 4.2 *We need to solve the IP*

$$\begin{array}{rr} & x_1 + 2x_2 \to \max, \\ 1: & 3x_1 + 2x_2 \le 6, \\ 2: & -3x_1 + 2x_2 \le 0, \\ 3: & 0 \le x_1 \le 2, \\ 4: & 0 \le x_2 \le 2, \\ & x_1, x_2 \in \mathbb{Z} \end{array}$$

by the cutting plane algorithm that generates only Gomory cuts.

Solution. First, we solve the relaxation LP by the dual simplex method.

0. We start with the dual feasible basic solution $x^{(0)} = (2,2)^T$ that corresponds to the basic set $I = (3,4)$. The inverse basic matrix B^{-1} and the potential vector π are the following:

$$B^{-1} = B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \pi = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

1. The first inequality $3x_1 + 2x_2 \le 6$ ($s = 1$) is violated at $x^{(0)}$ by $\beta = 6 - 3 \cdot 2 - 2 \cdot 2 = -4$. Therefore, we calculate

$$u^T = (3,2) \cdot B^{-1} = (3,2),$$

$$\lambda = \min\left\{\frac{1}{3}, \frac{2}{2}\right\} = \frac{1}{3} \quad \Rightarrow \quad t = 1 \quad \Rightarrow \quad I = (1,4)$$

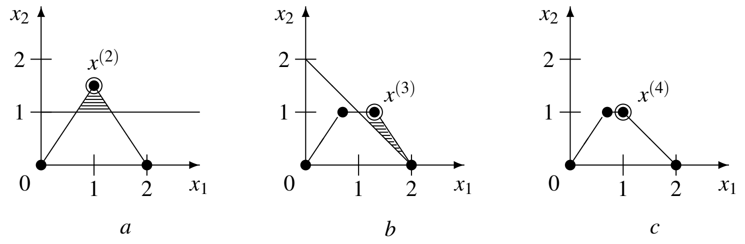

Fig. 4.2 Illustration for example 4.2

and then we perform the pivot operation:

$$B^{-1} := B^{-1} \cdot I(1,u)^{-1} = \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} \\ 0 & 1 \end{bmatrix},$$

$$\pi = \begin{pmatrix} \frac{1}{3} \\ 2 - \frac{1}{3} \cdot 2 \end{pmatrix} = \begin{pmatrix} \frac{1}{3} \\ \frac{4}{3} \end{pmatrix},$$

$$x^{(1)} = x^{(0)} + \beta B^{-1}\mathbf{e}_1 = \begin{pmatrix} 2 \\ 2 \end{pmatrix} - 4 \begin{pmatrix} \frac{1}{3} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ 2 \end{pmatrix}.$$

2. Now the second inequality $-3x_1 + 2x_2 \le 0$ ($s = 2$) is violated at $x^{(1)}$ by $\beta = -2$. Therefore, we calculate

$$u^T = (-3,2) \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} \\ 0 & 1 \end{bmatrix} = (-1,4),$$

$$\lambda = \min\{-, (4/3)/4\} = \frac{1}{3} \quad \Rightarrow \quad t = 2 \quad \Rightarrow \quad I = (1,2),$$

and then we perform the pivot operation:

$$B^{-1} := B^{-1} \cdot I(2,u)^{-1} = \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & -\frac{1}{6} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix},$$

$$\pi = \begin{pmatrix} \frac{1}{3} - \frac{1}{3} \cdot (-1) \\ \frac{1}{3} \end{pmatrix} = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \end{pmatrix},$$

$$x^{(2)} = x^{(1)} + \beta B^{-1}\mathbf{e}_2 = \begin{pmatrix} \frac{2}{3} \\ 2 \end{pmatrix} - 2 \begin{pmatrix} -\frac{1}{6} \\ \frac{1}{4} \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{3}{2} \end{pmatrix}.$$

3. The point $x^{(2)}$ satisfies all constraints of the relaxation LP. We have only one fractional component x_2, which will be used to build the Gomory cut. For

$$v^T = \left(\frac{1}{4}, \frac{1}{4}\right), \quad (v - \lfloor v \rfloor)^T = \left(\frac{1}{4}, \frac{1}{4}\right),$$

taking into account that $B = \begin{bmatrix} 3 & 2 \\ -3 & 2 \end{bmatrix}$, we write down the cut

$$\left\lfloor \left(\frac{1}{4}, \frac{1}{4}\right) \begin{pmatrix} 3 \\ -3 \end{pmatrix} \right\rfloor x_1 + \left\lfloor \left(\frac{1}{4}, \frac{1}{4}\right) \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right\rfloor x_2 \le \left\lfloor \left(\frac{1}{4}, \frac{1}{4}\right) \begin{pmatrix} 6 \\ 0 \end{pmatrix} \right\rfloor.$$

Having carried out the calculations, we obtain the inequality $x_2 \le 1$ (Fig. 4.2.a), which is added to our program as the 5-th constraint. This inequality is violated at $x^{(2)}$ by $\beta = 1 - \frac{3}{2} = -\frac{1}{2}$.

Having the violated inequality ($s = 5$), we perform the next iteration of the dual simplex method. First we calculate

$$u^T = (0,1)\begin{bmatrix} \frac{1}{6} & -\frac{1}{6} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix} = \left(\frac{1}{4}, \frac{1}{4}\right),$$
$$\lambda = \min\left\{3, \frac{4}{3}\right\} = \frac{4}{3} \quad \Rightarrow \quad t = 2 \quad \Rightarrow \quad I = (1,5),$$

and then we perform the pivot operation:

$$B^{-1} := B^{-1} \cdot I(2,u)^{-1} = \begin{bmatrix} \frac{1}{6} & -\frac{1}{6} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ -1 & 4 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} \\ 0 & 1 \end{bmatrix},$$
$$\pi = \begin{pmatrix} \frac{2}{3} - \frac{4}{3} \cdot \frac{1}{4} \\ \frac{4}{3} \end{pmatrix} = \begin{pmatrix} \frac{1}{3} \\ \frac{4}{3} \end{pmatrix},$$
$$x^{(3)} = x^{(2)} + \beta B^{-1}\mathbf{e}_2 = \begin{pmatrix} 1 \\ \frac{3}{2} \end{pmatrix} - \frac{1}{2}\begin{pmatrix} -\frac{2}{3} \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{4}{3} \\ 1 \end{pmatrix}.$$

4. The point $x^{(3)}$ satisfies all constraints of the relaxation LP (including one cut) but is not integer. We have only one fractional component x_1, which we will use to build the Gomory cut. For

$$v^T = \left(\frac{1}{3}, -\frac{2}{3}\right), \quad (v - \lfloor v \rfloor)^T = \left(\frac{1}{3}, \frac{1}{3}\right),$$

taking into account that $B = \begin{bmatrix} 3 & 2 \\ 0 & 1 \end{bmatrix}$, we write down the cut

$$\left\lfloor \left(\frac{1}{3}, \frac{1}{3}\right)\begin{pmatrix} 3 \\ 0 \end{pmatrix} \right\rfloor x_1 + \left\lfloor \left(\frac{1}{3}, \frac{1}{3}\right)\begin{pmatrix} 2 \\ 1 \end{pmatrix} \right\rfloor x_2 \le \left\lfloor \left(\frac{1}{3}, \frac{1}{3}\right)\begin{pmatrix} 6 \\ 1 \end{pmatrix} \right\rfloor.$$

After the calculations, we obtain the inequality $x_1 + x_2 \le 2$ (Fig. 4.2.b), which will be the 6-th in our IP. This new inequality is violated at $x^{(3)}$ by $\beta = 2 - \frac{4}{3} - 1 = -\frac{1}{3}$.

Having the violated inequality ($s = 6$), we perform the next iteration of the dual simplex method. First we calculate

$$u^T = (1,1)\begin{bmatrix} \frac{1}{3} & -\frac{2}{3} \\ 0 & 1 \end{bmatrix} = \left(\frac{1}{3}, \frac{1}{3}\right),$$
$$\lambda = \min\{1, 4\} = 1 \quad \Rightarrow \quad t = 1 \quad \Rightarrow \quad I = (6,5),$$

and then we perform the pivot operation:

$$B^{-1} := B^{-1} \cdot I(1,u)^{-1} = \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix},$$
$$\pi = \begin{pmatrix} 1 \\ \frac{4}{3} - \frac{1}{3} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

$$x^{(4)} = x^{(3)} + \beta B^{-1}\mathbf{e}_1 = \begin{pmatrix} \frac{4}{3} \\ 1 \end{pmatrix} - \frac{1}{3}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The point $x^{(4)}$ is integer and it satisfies all constraints (including cuts) of our IP (Fig. 4.2.c). Therefore, $x^{(4)}$ is an optimal solution of our example IP. □

4.3 Mixed Integer Rounding

It is clear that the rounding principle is not applicable when deriving cuts for mixed integer sets, i.e., when there are variables of both types, integer and continuous. Another simple observation will be useful here.

Disjunctive principle: if an inequality is valid for both sets X_1 and X_2, then it is also valid for their union $X_1 \cup X_2$.

Lemma 4.1. *Let* $X = \{(x,y) \in \mathbb{Z} \times \mathbb{R}_+ :\ x - y \le b\}$*,* $f = b - \lfloor b \rfloor$*. The inequality*

$$x - \frac{y}{1-f} \le \lfloor b \rfloor \tag{4.6}$$

is valid for X*.*

Proof. Let

$$X_1 = X \cap \{(x,y) :\ x \le \lfloor b \rfloor\} \quad \text{and} \quad X_2 = X \cap \{(x,y) :\ x \ge \lfloor b \rfloor + 1\}.$$

For $(x,y) \in X_1$, summing up the inequalities $(1-f)\cdot(x - \lfloor b \rfloor) \le 0$ and $0 \le y$, we obtain $(1-f)\cdot(x - \lfloor b \rfloor) \le y$.

For $(x,y) \in X_2$, summing up the inequalities $-f \cdot (x - \lfloor b \rfloor) \le -f$ and $x - \lfloor b \rfloor \le f + y$, we again obtain $(1-f)\cdot(x - \lfloor b \rfloor) \le y$.

Now, by virtue of the disjunctive principle, (4.6) is valid for the union $X = X_1 \cup X_2$. □

The statement of Lemma 4.1 is illustrated in Fig. 4.3, where the set X is represented by the straight horizontal lines. Inequality (4.6) cuts off from the relaxation polyhedron, $\{(x,y) \in \mathbb{R} \times \mathbb{R}_+ :\ x - y \le b\}$, the shaded triangle.

Theorem 4.3. *Let* $Y = \{(x,y) \in \mathbb{Z}^n_+ \times \mathbb{R}^2_+ :\ \sum_{j=1}^n a_j x_j + y_1 - y_2 \le b\}$*,* $f = b - \lfloor b \rfloor$ *and* $f_j = a_j - \lfloor a_j \rfloor$ *for* $j = 1, \dots, n$*. The inequality*

$$\sum_{j=1}^n \left(\lfloor a_j \rfloor + \frac{\max\{0, f_j - f\}}{1-f} \right) x_j - \frac{1}{1-f} y_2 \le \lfloor b \rfloor \tag{4.7}$$

is valid for Y*.*

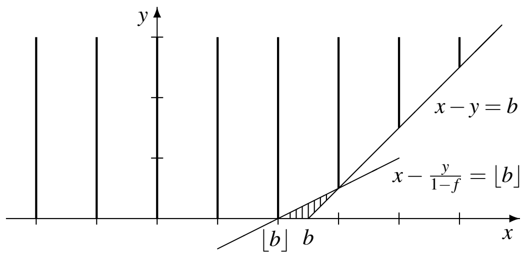

Fig. 4.3 Mixed integer rounding

Proof. Let us weaken the inequality $\sum_{j=1}^n a_jx_j + y_1 - y_2 \le b$ to

$$\sum_{j\in N_1}\lfloor a_j\rfloor x_j + \sum_{j\in N_2} a_jx_j - y_2 \le b,$$

where $N_1 = \{j:\ 1\le j\le n,\ f_j \le f\}$ and $N_2 = \{1,\dots,n\}\setminus N_1$. Applying Lemma 4.1 to the inequality $w - z \le b$ with

$$w = \sum_{j\in N_1}\lfloor a_j\rfloor x_j + \sum_{j\in N_2}\lceil a_j\rceil x_j \in \mathbb{Z} \quad \text{and} \quad z = y_2 + \sum_{j\in N_2}(1-f_j)x_j \ge 0,$$

we get the inequality

$$w - \frac{z}{1-f} \le \lfloor b\rfloor.$$

Substituting the expressions for w and z into this inequality, we obtain (4.7). □

Inequality (4.7) is also known as the *mixed integer rounding* of the inequality $\sum_{j=1}^n a_jx_j + y_1 - y_2 \le b$.

Note that, for an inequality $\sum_{j=1}^n a_jx_j \le b$ with non-negative integer variables, its mixed integer rounding

$$\sum_{j=1}^n\left(\lfloor a_j\rfloor + \frac{\max\{0, f_j - f\}}{1-f}\right)x_j \le \lfloor b\rfloor$$

is stronger than the integer Chvatal-Gomory cut,

$$\sum_{j=1}^n \lfloor a_j\rfloor x_j \le \lfloor b\rfloor,$$

if at least one of the numbers $f_j - f$ is positive.

Example 4.3 *We need to separate the point* $(\tilde{x},\tilde{y}) = \left(\frac{3}{2},\frac{7}{2}\right)$ *from the mixed integer set*

$$X = \{(x,y)\in\mathbb{Z}_+\times\mathbb{R}_+ :\ x+y\le 5,\ y-x\le 2\}.$$

Solution. In practice, mixed integer rounding is used in conjunction with the technique of mixing constraints. Introducing two non-negative slack variables, s_1 and s_2, we rewrite two inequalities defining X as the equations

$$\begin{aligned} x+y+s_1 &= 5, \\ -x+y+s_2 &= 2. \end{aligned}$$

In the extended space, the point $(\tilde{x},\tilde{y})$ corresponds to the point

$$(\tilde{x},\tilde{y},\tilde{s}_1,\tilde{s}_2) = \left(\frac{3}{2},\frac{7}{2},0,0\right).$$

Of three non-integer variables, y, s_1 and s_2, we choose the variable y whose current value is farthest from its nearest bound (in case of y, this is its lower bound). Then we mix our two equations to exclude y, i.e., we subtract the second equation from the first one to get the equation

$$2x+s_1-s_2 = 3,$$

which is divided by 2:

$$x+\frac{1}{2}s_1-\frac{1}{2}s_2 = \frac{3}{2}.$$

Let us note that in general, the equation is divided by the coefficient of an integer variable taking a fractional value. If there are several such variables, then as a divisor we can try each of their coefficients.

Applying Theorem 4.3 to the inequality

$$x+\frac{1}{2}s_1-\frac{1}{2}s_2 \le \frac{3}{2},$$

we obtain its mixed integer rounding

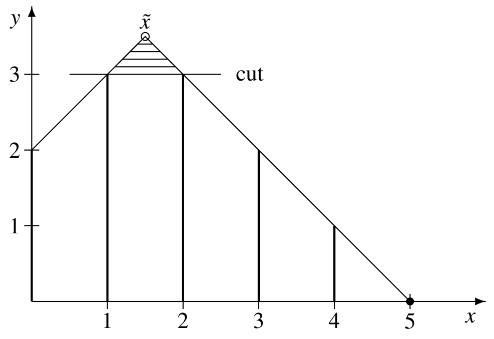

Fig. 4.4 Illustration for Example 4.3

$$x - \frac{\frac{1}{2}s_2}{1-\frac{1}{2}} \le 1 \quad \text{or} \quad x - s_2 \le 1.$$

Substituting $s_2 = 2 + x - y$ into the last inequality, we obtain the cut $y \le 3$. In Fig. 4.4 the set X is represented by five bold lines and the point $(5,0)$; the region that is cut off (by $y \le 3$) from the relaxation polyhedron is shaded. □

4.4 Fractional Gomory Cuts

In this section we will learn how to construct fractional Gomory cuts for a mixed integer set $P(A,b;S) \subseteq \mathbb{R}^n_+$. Let $\tilde{x} = A_I^{-1} b_I$ be a feasible basic solution, where I is a feasible basic set of rows for the system of linear inequalities $Ax \le b$. Introducing a vector of slack variables, $s \in \mathbb{R}^n_+$, we rewrite the subsystem of *basic* inequalities, $A_I x \le b_I$ in the equality form $A_I x + s = b_I$, or

$$x + A_I^{-1} s = A_I^{-1} b_I = \tilde{x}. \tag{4.8}$$

We will denote the elements of the inverse basic matrix A_I^{-1} by $\bar{a}_{ij}$.

Let us pick up an integer variable x_i which current value $\tilde{x}_i$ is not integer. Let $N_1(i)$ and $N_2(i)$ be, respectively, the index sets of the integer and non-integer variables s_j in the i-th equation from (4.8). We consider a slack variable s_j to be integer if all variables and coefficients in both parts of the inequality $A_{I[j]} x \le b_{I[j]}$ are integers. Now, let us rewrite the i-th equation from (4.8) in the form:

$$x_i + \sum_{j \in N_1(i)} \bar{a}_{ij} s_j + \sum_{j \in N_2(i)} \bar{a}_{ij} s_j = \tilde{x}_i. \tag{4.9}$$

Theorem 4.4. *If x_i is an integer variable and its value $\tilde{x}_i$ is not integer, $f_0 = \tilde{x}_i - \lfloor \tilde{x}_i \rfloor$ and $f_j = \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor$ for $j \in N_1(i) \cup N_2(i)$, then the following fractional Gomory cut*

$$\begin{aligned} &\sum_{j \in N_1(i):\, f_j \le f_0} f_j s_j + \sum_{j \in N_1(i):\, f_j > f_0} \frac{f_0(1-f_j)}{1-f_0} s_j + \\ &\sum_{j \in N_2(i):\, \bar{a}_{ij} > 0} \bar{a}_{ij} s_j - \sum_{j \in N_2(i):\, \bar{a}_{ij} < 0} \frac{f_0}{1-f_0} \bar{a}_{ij} s_j \ge f_0 \end{aligned} \tag{4.10}$$

is valid for all $(x_i, s) \in \mathbb{Z}_+ \times \mathbb{R}^n_+$ such that (4.9) *is satisfied and $s_j \in \mathbb{Z}$ for all $j \in N_1(i)$.*

Proof. Let us write down the mixed integer rounding of (4.9):

$$x_i + \sum_{j\in N_1(i):\, f_j\le f_0} \lfloor \bar{a}_{ij}\rfloor s_j + \sum_{j\in N_1(i):\, f_j> f_0} \left(\lfloor \bar{a}_{ij}\rfloor + \frac{f_j - f_0}{1-f_0}\right) + \sum_{j\in N_2(i):\, \bar{a}_{ij}<0} \frac{1}{1-f_0}\bar{a}_{ij}s_j \le \lfloor \tilde{x}_i \rfloor. \tag{4.11}$$

Using (4.9), we express the variable x_i in terms of the variables s_j, and then substitute the resulting expression into (4.11); as a result, we obtain (4.10). □

Gomory fractional cuts (4.10) are written in terms of the slack variables s_j. To return to the original variables x_i, we need to substitute $s = b_I - A_I x$ into (4.10) to get a cut for the set $P(A,b;S)$.

Example 4.4 *We need to solve the IP*

$$\begin{array}{rl} & x_1 + x_2 \to \max \\ 1: & 2x_1 + 2x_2 + x_3 \le 9, \\ 2: & x_2 - x_3 \le 0, \\ 3: & 0 \le x_1 \le 4, \\ 4: & 0 \le x_2 \le 3, \\ 5: & 0 \le x_3 \le 5, \\ & x_1, x_2 \in \mathbb{Z} \end{array} \tag{4.12}$$

by the cutting plane algorithm that generates only fractional Gomory cuts.

Solution. This time, we will not practice the dual simplex method to solve the relaxation LP for (4.12). We just start with its optimal basic solution:

$$I = (1,2,3),\ B^{-1} = \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \\ \frac{1}{3} & -\frac{2}{3} & -\frac{2}{3} \end{bmatrix},\ x^{(1)} = \begin{pmatrix} 4 \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix},\ \pi = \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix}.$$

Let us choose the variable x_2, which is integer and which current value is not integer, to start computing the first cut:

$$x_2 + \frac{1}{3}s_1 + \frac{1}{3}s_2 - \frac{2}{3}s_3 = \frac{1}{3}.$$

Here s_3 is the only integer slack variable. Next we compute the coefficients

$$f_0 = \frac{1}{3}, \quad f_3 = -\frac{2}{3} - (-1) = \frac{1}{3},$$

and then we write down the cut

$$\frac{1}{3}s_1 + \frac{1}{3}s_2 + \frac{1}{3}s_3 \ge \frac{1}{3},$$

or

$$s_1 + s_2 + s_3 \geq 1. \tag{4.13}$$

Substituting the expressions

$$s_1 = 9 - 2x_1 - 2x_2 - x_3, \quad s_2 = 0 - x_2 + x_3, \quad s_3 = 4 - x_1,$$

into (4.13), after simplification, we obtain the cut in the original variables x_1, x_2, x_3:

$$6: \quad x_1 + x_2 \leq 4,$$

which is violated at $x^{(1)}$ by $\beta = 4 - 4 - \frac{1}{3} = -\frac{1}{3}$.

Next we apply the dual simplex method to carry out reoptimization. Compute

$$u^T = (1,1,0)\begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \\ \frac{1}{3} & -\frac{2}{3} & -\frac{2}{3} \end{bmatrix} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right),$$
$$\lambda = \min\{1,1,1\} = 1 \quad \Rightarrow \quad t = 1 \quad \Rightarrow \quad I = (6,2,3)$$

and perform the pivot operation:

$$B^{-1} := B^{-1} \cdot I(1,u)^{-1} = \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \\ \frac{1}{3} & -\frac{2}{3} & -\frac{2}{3} \end{bmatrix} \cdot \begin{bmatrix} 3 & -1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & -1 \\ 1 & -1 & -1 \end{bmatrix},$$
$$x^{(2)} = x^{(1)} + \beta B^{-1}\mathbf{e}_1 = \begin{pmatrix} 4 \\ \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} - \frac{1}{3}\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \\ 0 \end{pmatrix},$$
$$\pi = \begin{pmatrix} 1 \\ \frac{1}{3} - \frac{1}{3} \\ \frac{1}{3} - \frac{1}{3} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

The point $x^{(2)}$ is integer and satisfies all constraints in (4.12). Therefore, $x^{(2)}$ is an optimal solution to (4.12). □

4.5 Disjunctive Inequalities

The union $X_1 \cup X_2$ of two sets X_1 and X_2 is also called the disjunction of X_1 and X_2, since the condition $x \in X_1 \cup X_2$ is also written as the disjunction $x \in X_1$ or $x \in X_2$. The disjunctive principle was introduced in Sect. 4.3 where we studied the mixed integer rounding cuts. Let us recall the essence of this simple principle: if an inequality holds for both sets, X_1 and X_2, then it is also valid for $X_1 \cup X_2$. In this section proceeding from this disjunctive principle, we will develop another version of the disjunctive cuts.

From the algorithmic point of view, the disjunction of polyhedra is of special interest.

Theorem 4.5. *Let $P_i = \{x \in \mathbb{R}^n_+ :\ A^i x \le b^i\}$ for $i = 1,2$. If both polyhedra, P_1 and P_2, are non-empty, then an inequality $\alpha^T x \le \beta$ is valid for the union $P_1 \cup P_2$ if and only if there exists a pair of vectors $y^1, y^2 \ge 0$ such that $(A^i)^T y^i \ge \alpha$ and $(y^i)^T b^i \le \beta$ for $i = 1,2$.*

Proof. The sufficiency is verified simply. Suppose that there exists a pair of vectors $y^1, y^2 \ge 0$ such that

$$(A^i)^T y^i \ge \alpha \quad \text{and} \quad (y^i)^T b^i \le \beta \quad \text{for } i = 1,2.$$

If $x \in P_1 \cup P_2$, then $A^i x \le b^i$ for some $i \in \{1,2\}$. Therefore

$$\alpha^T x \le (y^i)^T A^i x \le (y^i)^T b \le \beta.$$

Now we prove the necessity. For $i = 1,2$, let an inequality $\alpha^T x \le \beta$ be valid for P_i and let x^i and y^i be optimal solutions for the pair of dual LPs

$$\max\{\alpha^T x :\ A^i x \le b,\ x \ge 0\},$$
$$\min\{b^T y :\ (A^i)^T y \ge \alpha,\ y \ge 0\}.$$

By the duality theorem (Theorem 3.1), we have $b^T y^i = \alpha^T x^i \le \beta$. □

From Theorem 4.5, it directly follows the next method of solving the separation problem for the disjunction of two polyhedra.

Separation procedure *for* conv$(P_1 \cup P_2)$:

Check whether the point $\tilde{x} \in \mathbb{R}^n$ belongs to the set $P_1 \cup P_2$, and if not, find the separating hyperplane by solving the next LP

$$\begin{aligned} &\tilde{x}^T \alpha - \beta \to \max,\\ &(A^i)^T y^i \ge \alpha, \quad i = 1,2,\\ &(b^i)^T y^i \le \beta, \quad i = 1,2,\\ &y^i \in \mathbb{R}^n_+, \quad i = 1,2,\\ &-1 \le \beta \le 1. \end{aligned} \tag{4.14}$$

If α^* and β^* are components of an optimal solution to (4.14) and $(\alpha^*)^T \tilde{x} > \beta$, then the inequality $(\alpha^*)^T x \le \beta$ separates $\tilde{x}$ from conv$(P_1 \cup P_2)$. Otherwise, $\tilde{x} \in$ conv$(P_1 \cup P_2)$.

Solving (4.14) we seek an inequality $\alpha^T x \le \beta$ that holds for conv$(P_1 \cup P_2)$ and that is most violated at the point $\tilde{x}$ under the normalization $|\beta| \le 1$. Without this normalization condition, due to the homogeneity of the remaining constraints, the objective function in (4.14) would be unbounded.

Example 4.5 *We need to solve the IP*

$$\begin{array}{rr} & x_1 + 2x_2 \to \max, \\ 1: & -2x_1 + 3x_2 \le 4, \\ 2: & x_1 + \ \ x_2 \le 5, \\ 3: & x_1 \qquad \ge 0, \\ 4: & x_2 \ge 0, \\ & x_1, x_2 \in \mathbb{Z} \end{array} \tag{4.15}$$

by the cutting plane algorithm that generates only disjunctive cuts.

Solution. Let us start immediately with an optimal basic solution to the relaxation LP for (4.15):

$$I = (1,2), \quad B^{-1} = \begin{bmatrix} -\frac{1}{5} & \frac{3}{5} \\ \frac{1}{5} & \frac{2}{5} \end{bmatrix}, \quad x^{(1)} = \begin{pmatrix} \frac{11}{5} \\ \frac{14}{5} \end{pmatrix}, \quad \pi = \begin{pmatrix} \frac{1}{5} \\ \frac{7}{5} \end{pmatrix}.$$

The solution $x^{(1)}$ is non-integer, and we will try to cut it off. The polytope $P^{(1)}$ of feasible solutions of the relaxation LP is shown in Fig. 4.5.a. Since there are no integer points in the strip $2 < x_1 < 3$, we can remove from $P^{(1)}$ all points from this strip (in Fig. 4.5.a the deleted area is shaded). Now the feasible domain of our problem is contained in the union of two polyhedra, $P_1^{(1)}$ and $P_2^{(1)}$, that are given by the following systems of inequalities:

$$P_1^{(1)}: \begin{array}{rr} 1: & -2x_1 + 3x_2 \le 4, \\ 2: & x_1 + \ \ x_2 \le 5, \\ 3: & x_1 \qquad \le 2, \\ & x_1 \qquad \ge 0, \\ & x_2 \ge 0, \end{array} \qquad P_2^{(1)}: \begin{array}{rr} 1: & -2x_1 + 3x_2 \le 4, \\ 2: & x_1 + \ \ x_2 \le 5, \\ 3: & -x_1 \qquad \le -3, \\ & x_1 \qquad \ge 0, \\ & x_2 \ge 0. \end{array}$$

To separate $x^{(1)}$ from $\operatorname{conv}\left(P_1^{(1)} \cup P_2^{(1)}\right)$, we need to solve the following LP:

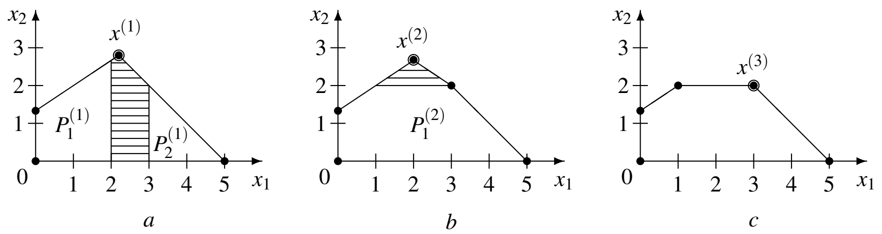

Fig. 4.5 Illustration for Example 4.5

$$\begin{array}{rrrrl}
 & & \tfrac{11}{5}\alpha_1 + \tfrac{14}{5}\alpha_2 - \beta \to \max, & & \\
-2y_1^1 + y_2^1 + y_3^1 & & -\ \alpha_1 & & \geq 0,\\
3y_1^1 + y_2^1 & & -\ \alpha_2 & & \geq 0,\\
4y_1^1 + 5y_2^1 + 2y_3^1 & & -\beta & & \leq 0,\\
 & -2y_1^2 + y_2^2 - y_3^2 - \alpha_1 & & & \geq 0,\\
 & 3y_1^2 + y_2^2 \quad -\ \alpha_2 & & & \geq 0,\\
 & 4y_1^2 + 5y_2^2 - 3y_3^2 \quad -\beta & & & \leq 0,\\
 & & -1 \leq \beta & & \leq 1,\\
 & & y_1^1, y_2^1, y_3^1, y_1^2, y_2^2, y_3^2 & & \geq 0.
\end{array}$$

Using any **LP** solver available to you, you can check that this LP has an optimal solution for which the variables α_1, α_2, and β take the following values:

$$\alpha_1^* = \frac{1}{6}, \quad \alpha_2^* = \frac{1}{4}, \quad \beta^* = 1.$$

Thus, we have found the cut

$$\frac{1}{6}x_1 + \frac{1}{4}x_2 \leq 1,$$

or

$$5: \quad 2x_1 + 3x_2 \leq 12.$$

We add this cut to (4.15) as the 5-th constraint. The polytope $P^{(2)}$ of the feasible solutions of the new relaxation LP is shown in Fig. 4.5.b.

Now let us proceed to the reoptimization. At the point $x^{(1)}$, the added inequality ($s = 5$) is violated by $12 - 2 \cdot \frac{11}{5} - 3 \cdot \frac{14}{5} = -\frac{4}{5}$. We calculate

$$u^T = (2,3) \cdot \begin{bmatrix} -\frac{1}{5} & \frac{3}{5} \\ \frac{1}{5} & \frac{2}{5} \end{bmatrix} = \left(\frac{1}{5}, \frac{12}{5}\right),$$

$$\lambda = \frac{7}{12} \quad \Rightarrow \quad t = 2,\ I = (1,5),$$

and then perform the pivot operation:

$$B^{-1} := B^{-1} \cdot I(2,u)^{-1} = \begin{bmatrix} -\frac{1}{5} & \frac{3}{5} \\ \frac{1}{5} & \frac{2}{5} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ -\frac{1}{12} & \frac{5}{12} \end{bmatrix} = \begin{bmatrix} -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{6} \end{bmatrix},$$

$$\pi = \begin{pmatrix} \frac{1}{5} - \frac{7}{12} \cdot \frac{1}{5} \\ \frac{7}{12} \end{pmatrix} = \begin{pmatrix} \frac{1}{12} \\ \frac{7}{12} \end{pmatrix},$$

$$x^{(2)} = x^{(1)} + \beta B^{-1} \mathbf{e}_2 = \begin{pmatrix} \frac{11}{5} \\ \frac{14}{5} \end{pmatrix} - \frac{4}{5} \cdot \begin{pmatrix} \frac{1}{4} \\ \frac{1}{6} \end{pmatrix} = \begin{pmatrix} 2 \\ \frac{8}{3} \end{pmatrix}.$$

The new solution $x^{(2)}$ is still non-integer, and we will try to cut it off. Since there are no integer points in the strip $2 < x_2 < 3$, we can remove from $P^{(2)}$ all points from this strip (the deleted area is shaded in Fig. 4.5.b). Now the feasible domain of our IP is contained in the union of two polyhedra, $P_1^{(2)}$ and $P_2^{(2)}$, that are the solution sets to the following systems of inequalities:

$$P_1^{(2)}: \begin{array}{rrl} 1: & -2x_1 + 3x_2 & \le 4, \\ 2: & x_1 + x_2 & \le 5, \\ 3: & 2x_1 + 3x_2 & \le 12, \\ 4: & x_2 & \le 2, \\ & x_1 & \ge 0, \\ & x_2 & \ge 0, \end{array} \qquad P_2^{(2)}: \begin{array}{rrl} 1: & -2x_1 + 3x_2 & \le 4, \\ 2: & x_1 + x_2 & \le 5, \\ 3: & 2x_1 + 3x_2 & \le 12, \\ 4: & -x_2 & \le -3, \\ & x_1 & \ge 0, \\ & x_2 & \ge 0. \end{array}$$

From Fig. 4.5.b it is clear that the polyhedron $P_2^{(2)}$ is empty and, therefore, the second system of inequalities is incompatible. But we will not rely on the drawing. To separate $x^{(2)}$ from $\operatorname{conv}\left(P_1^{(2)} \cup P_2^{(2)}\right)$, we solve the following LP:

$$\begin{array}{lllr} & & 2\alpha_1 + \frac{8}{3}\alpha_2 - \beta \to \max, & \\ -2y_1^1 + y_2^1 + 2y_3^1 & & -\ \alpha_1 & \ge 0, \\ 3y_1^1 + y_2^1 + 3y_3^1 + y_4^1 & & -\ \alpha_2 & \ge 0, \\ 4y_1^1 + 5y_2^1 + 12y_3^1 + 2y_4^1 & & -\beta & \le 0, \\ & -2y_1^2 + y_2^2 + 2y_3^2 & -\ \alpha_1 & \ge 0, \\ & 3y_1^2 + y_2^2 + 3y_3^2 - y_4^2 & -\ \alpha_2 & \ge 0, \\ & 4y_1^2 + 5y_2^2 + 12y_3^2 - 3y_4^2 & -\beta & \le 0, \\ & & -1 \le \beta & \le 1, \\ & & y_1^1, y_2^1, y_3^1, y_4^1, y_1^2, y_2^2, y_3^2, y_4^2 & \ge 0. \end{array}$$

This program has an optimal solution with the following components:

$$\alpha_1^* = 0, \quad \alpha_2^* = \frac{1}{2}, \quad \beta^* = 1.$$

Therefore, our second cut is the inequality

$$6: \quad x_2 \le 2,$$

which is added to our already extended IP as the 6-th constraint. The feasible polytope, $P^{(3)}$, of the relaxation LP for this new IP is shown in Fig. 4.5.c.

Now let us proceed to the reoptimization. At the point $x^{(2)}$, the last added inequality ($s = 6$) is violated by $2 - \frac{8}{3} = -\frac{2}{3}$. We calculate

$$u^T = (0,1) \cdot \begin{bmatrix} -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{6} \end{bmatrix} = \left(\frac{1}{6}, \frac{1}{6}\right),$$

$$\lambda = \min\left\{\frac{1}{2}, \frac{7}{2}\right\} = \frac{1}{2} \quad \Rightarrow \quad t = 1,\, I = (6,5),$$

and then perform the pivot operation:

$$B^{-1} := B^{-1} \cdot I(1,u)^{-1} = \begin{bmatrix} -\frac{1}{4} & \frac{1}{4} \\ \frac{1}{6} & \frac{1}{6} \end{bmatrix} \cdot \begin{bmatrix} 6 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -\frac{3}{2} & \frac{1}{2} \\ 1 & 0 \end{bmatrix},$$

$$\pi = \begin{pmatrix} \frac{1}{2} \\ \frac{7}{12} - \frac{1}{2} \cdot \frac{1}{6} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix},$$

$$x^{(3)} = x^{(2)} + \beta B^{-1} \mathbf{e}_1 = \begin{pmatrix} 2 \\ \frac{8}{3} \end{pmatrix} - \frac{2}{3} \cdot \begin{pmatrix} -\frac{3}{2} \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}.$$

Since the point $x^{(3)}$ is integral, then it is an optimal solution to (4.15). □

4.6 Lift And Project

In this section we will consider MIPs in which all integer variables are binary:

$$\begin{aligned} & c^T x \to \max, \\ & Ax \le b, \\ & x_j \in \{0,1\}, \quad j = 1,\dots,p, \end{aligned} \tag{4.16}$$

where A is a real $m \times n$-matrix, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, x is an n-vector of variables, and $0 < p \le n$. To simplify the discussion, we assume that the inequalities $-x_j \le 0$ and $x_j \le 1$ ($j = 1,\dots,p$) are already included in the system $Ax \le b$.

Let X denote the set of feasible solution to (4.16), then $P = P(A,b)$ is a relaxation polyhedron for X ($X \subseteq P$). Next we describe a procedure that, for any $j \in \{1,\dots,p\}$, constructs a polyhedron P_j such that $X \subseteq P_j \subseteq P$.

Lift-and-project:
(*Lift*) Linearize the system of nonlinear inequalities

$$x_j(Ax - b) \le 0, \quad (1 - x_j)(Ax - b) \le 0,$$

by substituting x_j for x_j^2, and a new continuous variable y_i for $x_i x_j$ ($i \ne j$). Let M_j denote the polyhedron of the solutions of the resulting system of linear inequalities.
Project the polyhedron $M_j \subseteq \mathbb{R}^n \times \mathbb{R}^{n-1}$ onto the space of x-variables. Let $P_j = \mathrm{lfpr}(P)$ denotes the resulting polyhedron.

Recall that the *projection* of a set $Q \subseteq U \times V$ onto the set U is the set

$$\mathrm{proj}_U(Q) \stackrel{\mathrm{def}}{=} \{u \in U :\ \exists\,(u,v) \in Q\}.$$

The inclusion $X \subseteq P_j$ follows from the fact that, for $x \in X$, the point (x,y) belongs to M_j if we define $y_i = x_i x_j$ for all $i \neq j$. The inclusion $P_j \subseteq P$ is valid because each of the inequalities $A_i x \le b$ is the sum of two inequalities defining M_j:

$$x_j A_i x \le b_i x_j \quad \text{and} \quad (1-x_j)A_i x \le b_i(1-x_j).$$

The above lift-and-project procedure does not give an explicit (in the form of a system of linear inequalities) descriptions of the polyhedrons P_j. But we can still describe each polyhedron P_j implicitly by providing a separation procedure. The polyhedron M_j is described by the following system of inequalities:

$$\begin{aligned}(A^j - b)x_j + A^{N\setminus\{j\}}y &\le 0,\\ A^{N\setminus\{j\}}x^{N\setminus\{j\}} + bx_j - A^{N\setminus\{j\}}y &\le b,\end{aligned}$$

where $N \stackrel{\mathrm{def}}{=} \{1,\dots,n\}$. The point $\bar{x} \in \mathbb{R}^n$ belongs to P_j if and only if the following system of inequalities is compatible:

$$\begin{aligned}A^{N\setminus\{j\}}y &\le \bar{x}_j(b - A^j),\\ -A^{N\setminus\{j\}}y &\le (1-\bar{x}_j)b - A^{N\setminus\{j\}}\bar{x}^{N\setminus\{j\}}.\end{aligned} \tag{4.17}$$

By Farkas' lemma (Lemma 3.3) the system of inequalities (4.17) has a solution if and only if

$$\bar{x}_j(b-A^j)^T u + ((1-\bar{x}_j)b - A^{N\setminus\{j\}}\bar{x}^{N\setminus\{j\}})^T v \ge 0$$

for all $u, v \in \mathbb{R}^m_+$ such that

$$u^T A^{N\setminus\{j\}} - v^T A^{N\setminus\{j\}} = 0.$$

We can verify this condition of Farkas' lemma by solving the following LP:

$$\bar{x}_j(b-A^j)^T u + ((1-\bar{x}_j)b - A^{N\setminus\{j\}}\bar{x}^{N\setminus\{j\}})^T v \to \min, \tag{4.18a}$$
$$u^T A^{N\setminus\{j\}} - v^T A^{N\setminus\{j\}} = 0, \tag{4.18b}$$
$$\sum_{i=1}^m u_i + \sum_{i=1}^m v_i = 1, \tag{4.18c}$$
$$u \ge 0,\ v \ge 0. \tag{4.18d}$$

Here, (4.18c) is a normalization restriction, which is introduced to ensure that the objective function in (4.18) is bounded.

Let $\bar{z}$ and $(\bar{u},\bar{v})$ be the optimal objective value and an optimal solution of (4.18). If $\bar{z} \ge 0$, then the point $\bar{x}$ belongs to P_j, and if $\bar{z} < 0$, then the inequality

$$x_j(b-A^j)^T \bar{u} + ((1-x_j)b - A^{N\setminus\{j\}}x^{N\setminus\{j\}})^T \bar{v} \ge 0,$$

or after rearranging,

$$(\bar{v}^T b - \bar{u}^T(b - A^j))x_j + \bar{v}^T A^{N\setminus\{j\}} x^{N\setminus\{j\}} \le \bar{v}^T b \tag{4.19}$$

separates $\bar{x}$ from P_j.

Example 4.6 *We need to separate the point $\bar{x} = (1/3, 2)$ from the solution set of the following system:*

$$\begin{array}{rrrl}
1: & 3x_1 + & 2x_2 & \le 5,\\
2: & -x_1 & & \le 0,\\
3: & x_1 & & \le 1,\\
4: & & -x_2 & \le 0,\\
5: & & x_2 & \le 2,\\
 & & x_1 \in & \{0,1\}.
\end{array}$$

Solution. First, for $j = 1$, we write down (4.18) applied to our problem instance:

$$\begin{aligned}
z = \frac{2}{3}u_1 + \frac{1}{3}u_2 + \frac{2}{3}u_5 - \frac{2}{3}v_1 + \frac{2}{3}v_3 + 2v_4 - \frac{2}{3}v_5 \to \min,\\
2u_1 - u_4 + u_5 - 2v_1 + v_2 - v_5 = 0,\\
u_1 + u_2 + u_3 + u_4 + u_5 + v_1 + v_2 + v_3 + v_4 + v_5 = 1,\\
u_1, u_2, u_3, u_4, u_5, v_1, v_2, v_3, v_4, v_5 \ge 0.
\end{aligned}$$

The vectors $\bar{u} = \left(\frac{1}{3}, 0, 0, 0, 0\right)^T$, $\bar{v} = \left(0, 0, 0, 0, \frac{2}{3}\right)^T$ constitute an optimal solution to this LP, and the optimal objective value is $\bar{z} = -\frac{1}{9}$. Since $\bar{z} < 0$, by (4.19), we can write the cut:

$$\left(\frac{4}{3} - \frac{2}{3}\right)x_1 + \frac{2}{3}x_2 \le \frac{4}{3} \quad \text{or} \quad x_1 + x_2 \le 2.$$

□

The following theorem shows that each polyhedron P_j build by the lift-and-project procedure is the convex hull of the union of two polyhedra. This means that the lift-and-project cuts are specialized disjunctive cuts.

Theorem 4.6. $P_j = P_j^* \overset{\text{def}}{=} \operatorname{conv}\left((P \cap \{x \in \mathbb{R}^n : x_j = 0\}) \cup (P \cap \{x \in \mathbb{R}^n : x_j = 1\})\right)$.

Proof. We assume that $P \neq \emptyset$. Otherwise, the result is trivial. First we prove the inclusion $P_j \subseteq P_j^*$.

If $P \cap \{x \in \mathbb{R}^n : x_j = 0\} = \emptyset$, then $P_j^* = P \cap \{x \in \mathbb{R}^n : x_j = 1\}$. We already know that $P_j \subseteq P$. Therefore, to prove the inclusion $P_j \subseteq P_j^*$, it suffices to show that the inequality $x_j \ge 1$ holds for P_j. Since $P \cap \{x \in \mathbb{R}^n : x_j = 0\} = \emptyset$, then $\varepsilon \overset{\text{def}}{=} \min\{x_j : x \in P\} > 0$ and the inequality $x_j \ge \varepsilon$ is valid for P. Since $x_j \ge \varepsilon$ is a linear combination of some inequalities from $Ax \le b$, the inequality $(1 - x_j)x_j \ge (1 - x_j)\varepsilon$ is valid for the nonlinear system constructed in the lift step. Then x_j^2 is replaced with x_j, and we obtain the inequality $(1 - x_j)\varepsilon \ge 0$, from which it follows that $x_j \ge 1$.

The inclusion $P_j \subseteq P_j^*$ is proved similarly when $P \cap \{x \in \mathbb{R}^n : x_j = 1\} = \emptyset$.

Now we consider the case when both sets, $P \cap \{x \in \mathbb{R}^n : x_j = 0\}$ and $P \cap \{x \in \mathbb{R}^n : x_j = 1\}$, are non-empty (this is illustrated in Fig. 4.6). Suppose that the inequality $\alpha^T x \le \beta$ is valid for P_j^*. Since this inequality is also valid for $P \cap \{x \in \mathbb{R}^n : x_j \le 0\}$ and $P \cap \{x \in \mathbb{R}^n : x_j \ge 1\}$, then, by virtue of Proposition 1.3, there exist vectors $\lambda^0, \lambda^1 \in \mathbb{R}^n_+$, and numbers $\gamma_0, \gamma_1 \ge 0$ such that

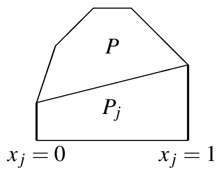

Fig. 4.6

$$\alpha^T = (\lambda^0)^T A + \gamma_0 \mathbf{e}_j \quad \text{and} \quad \beta \ge (\lambda^0)^T b,$$
$$\alpha^T = (\lambda^1)^T A - \gamma_1 \mathbf{e}_j \quad \text{and} \quad \beta \ge (\lambda^1)^T b - \gamma_1.$$

Consequently, the inequalities $\alpha^T x - \gamma_0 x_j \le \beta$ and $\alpha^T x - \gamma_1(1 - x_j) \le \beta$ are valid for P. Therefore, the inequalities

$$(1 - x_j)(\alpha^T x - \gamma_0 x_j - \beta) \le 0,$$
$$x_j(\alpha^T x - \gamma_1(1 - x_j) - \beta) \le 0$$

and their sum
$$\alpha^T x - (\gamma_0 + \gamma_1)(x_j - x_j^2) - \beta \le 0$$

are valid for the nonlinear system build in the lift step. Then x_j^2 is replaced with x_j giving the inequality $\alpha^T x \le b$ that is valid for M_j, and, consequently, for P_j. This completes the proof of the inclusion $P_j \subseteq P_j^*$.

It remains to justify the validity of the inverse inclusion $P_j^* \subseteq P_j$. We assume that $P_j^* \ne \emptyset$; otherwise, the result is trivial. Let the point $\bar{x}$ belong to at least one of the sets $P \cap \{x \in \mathbb{R}^n : x_j = 0\}$ or $P \cap \{x \in \mathbb{R}^n : x_j = 1\}$. Define $\bar{y}_i \overset{\text{def}}{=} \bar{x}_i \bar{x}_j$ for $i \ne j$. As $\bar{x}_j^2 = \bar{x}_j$, then $(\bar{x}, \bar{y}) \in M_j$ and $\bar{x} \in P_j$. Since P_j is a convex set, we conclude that $P_j^* \subseteq P_j$. □

Applying the lift-and-project procedure successively for $j = 1, \dots, p$, we can compute the convex hull of the feasible set X.

Theorem 4.7. *Defining* $\bar{P}_0 = P$ *and* $\bar{P}_k = \mathrm{lfpr}_k(\bar{P}_{k-1})$ *for* $k \ge 1$, *we have the equality* $\bar{P}_p = \mathrm{conv}(X)$.

Proof. By induction, we prove the equality $\bar{P}_j = \mathrm{conv}(X_j)$, where

$$X_j \overset{\text{def}}{=} \{x \in \mathbb{R}^n : Ax \le b,\ x_i \in \{0,1\} \text{ for } i = 1, \dots, j\}.$$

By definition, $\bar{P}_0 = P = X_0$. Let $j \ge 1$ and assume that $\bar{P}_{j-1} = \mathrm{conv}(X_{j-1})$. By Theorem 4.6, we have

$$\begin{aligned}\bar{P}_j &= \mathrm{conv}((\mathrm{conv}(X_{j-1}) \cap \{x : x_j = 0\}) \cup (\mathrm{conv}(X_{j-1}) \cap \{x : x_j = 1\})) \\ &= \mathrm{conv}(\mathrm{conv}(X_{j-1} \cap \{x : x_j = 0\}) \cup \mathrm{conv}(X_{j-1} \cap \{x : x_j = 1\})) \\ &= \mathrm{conv}(X_j).\end{aligned}$$ □

4.7 Separation and Optimization

In the most general form, the *separation problem* is formulated as follows. Given a set $X \subset \mathbb{R}^n$ and a point $\tilde{x} \in \mathbb{R}^n$, we need to prove that $\tilde{x} \in \mathrm{conv}(X)$, or find a hyperplane $H(a,\beta)$ separating $\tilde{x}$ from X, i.e., such that

$$\begin{aligned} &a^T x \le \beta, \quad x \in X,\\ &a^T \tilde{x} > \beta. \end{aligned}$$

We have already considered some special cases of this general problem. In particular, studying the Gomory cuts, we solved the problem of separating a vertex $\tilde{x}$ of a polyhedron $P(A,b)$ from a mixed integer set $X = P(A,b;S)$. Note that the problem of separating an arbitrary point $\tilde{x}$ (not a vertex of $P(A,b)$) from the set $P(A,b;S)$ is **NP**-hard. It is not surprising that the general separation problem is also **NP**-hard, and we can hardly hope to develop an efficient algorithm for solving it. Here we describe a procedure for solving the general separation problem, which is efficient in practice for a number of special sets X. Let us also note that in practice we need very fast separation procedures, since they are repeatedly called by the cutting plane algorithms. For this reason, in the modern **MIP** solvers, very often exact separation procedures are replaced with fast heuristics.

It is known that the separation problem for the set X is polynomially equivalent to the optimization problem

$$\max\{c^T x : x \in X\}. \tag{4.20}$$

Here our main interest is not in investigating the complexity aspects of this equivalence. We are going to present two **LP** based approaches for solving each of the problems, optimization or separation, provided that there is an efficient procedure for solving the other problem.

4.7.1 Markowitz Model for Portfolio Optimization

We already know how separation procedures are used for solving MIPs. In this section on a particular quadratic optimization problem we demonstrate how to apply the cutting plane approach for solving optimization problems with convex constraints that are represented by separation procedures.

We want to invest some amount of money in some of n assets (stocks, bonds, etc.). Let p_i be the *relative change in price* of asset i during some planning horizon (for example, one year), i.e., p_i is the change in the price of this asset during the planning horizon divided by its price at the beginning of the horizon (*return* per one enclosed dollar). We assume that $p_1,\dots,p_n$ are dependent normal random variables, and $p = (p_1,\dots p_n)^T$ is a random price vector with known mean (mathematical expectation) $\bar{p} \in \mathbb{R}^n_+$ and covariance matrix Σ, which is a symmetric positive semidefined $n \times n$-matrix.

A *portfolio* is a vector $x = (x_1, \dots, x_n)^T$, where $x_i \geq 0$ is the share of funds invested in asset i ($i = 1, \dots, n$), $\sum_{i=1}^n x_i = 1$. We will assume that the portfolio is formed in order to remain unchanged for the planning horizon. Therefore, the return of the portfolio x at the end of the planning horizon is a random variable $p^T x$ with the mean value $\bar{p}^T x$ and the variance $x^T \Sigma x$. Markowitz formulated the problem of portfolio optimization as the following quadratic programming problem:

$$\bar{p}^T x \to \max, \tag{4.21a}$$

$$x^T \Sigma x \leq r^2, \tag{4.21b}$$

$$\sum_{i=1}^n x_i = 1, \tag{4.21c}$$

$$x_i \geq 0, \quad i = 1, \dots, n. \tag{4.21d}$$

In this problem, we maximize the average return of the portfolio at a limited risk of r^2 (see Sect. 8.3 for a discussion of risk measures).

To solve (4.21) with the dual simplex method, we need to represent the convex set

$$X_\Sigma \overset{\text{def}}{=} \left\{ x \in \mathbb{R}^n : x^T \Sigma x \leq r^2 \right\}$$

by a separation procedure. In this particular case, the separation algorithm is simple. Since Σ is positive semidefined, it can be factored as $\Sigma = B^T B$, where B is some $m \times n$-matrix. Introducing new variables $y = Bx$, we can rewrite (4.21b) as follows:

$$y^T y \leq r^2, \quad y = Bx.$$

Now, to separate a given point $\tilde{x} \in \mathbb{R}^n$ from the set X_Σ, we first compute $\tilde{y} = B\tilde{x}$. If $\|\tilde{y}\| \leq r$, then $\tilde{x} \in X_\Sigma$. Otherwise, the hyperplane $\tilde{y}^T \left(y - \frac{r}{\|\tilde{y}\|} \tilde{y} \right) = 0$ — which touches the sphere given by $\|y\| = r$ at the intersection point, $\frac{r}{\|\tilde{y}\|} \tilde{y}$, of this sphere with the ray going from the origin through the point $\tilde{y}$ — separates $\tilde{y}$ from the ball $B(r) \overset{\text{def}}{=} \{ y \in \mathbb{R}^n : \|y\| \leq 1 \}$, and $B(r)$ is in the half-space given by the inequality $\tilde{y}^T y \leq r \|\tilde{y}\|$. Therefore, the inequality $(\tilde{y}^T B) x \leq r \|\tilde{y}\|$ is valid for X_Σ but not for $\tilde{x}$.

Example 4.7 *Consider the problem of forming a portfolio of four assets. The mean values and standard deviations of the future random returns of these assets are presented in the following table*

Asset	1	2	3	4
$\bar{p}_i$	1.03	1.06	1.08	1.1
σ_i	0	0.05	0.1	0.2

Here asset 1 *is a risk-free asset with a return of* 3%. *The correlation coefficients between risky assets are the following:* $\rho_{24} = -0.04$, $\rho_{34} = 0.03$, *and* $\rho_{23} = 0$.

We need to find an approximately optimal portfolio which risk is not greater than r^2 *for* $r = 0.04$.

Solution. First, using equations $\Sigma_{ii} = \sigma_i^2$, $\Sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$, we compute the covariance matrix:

$$\Sigma = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.0025 & 0 & -0.0004 \\ 0 & 0 & 0.01 & 0.0006 \\ 0 & -0.0004 & 0.0006 & 0.04 \end{bmatrix}.$$

So we have to solve the following optimization problem:

$$\begin{aligned} &1.03x_1 + 1.06x_2 + 1.08x_3 + 1.1x_4 \to \max, \\ &0.0025x_2^2 + 0.01x_3^2 + 0.04x_3^2 - 0.0008x_2x_4 + 0.0012x_3x_4 \le r^2, \\ &x_1 + x_2 + x_3 + x_4 = 1, \\ &x_1 \ge 0,\ x_2 \ge 0,\ x_3 \ge 0,\ x_4 \ge 0. \end{aligned} \tag{4.22}$$

You can directly verify that $\Sigma \approx B^T B$ for

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & -0.008 \\ 0 & 0 & 0.1 & 0.006 \\ 0 & 0 & 0 & 0.1997 \end{bmatrix}.$$

Therefore,

$$y_1 = 0,\ y_2 = 0.05x_2 - 0.008x_4,\ y_3 = 0.1x_3 + 0.006x_4,\ y_4 = 0.1997x_4.$$

We cannot numerically find an exact optimal solution to (4.22). Let us agree to stop calculations as soon as we have a portfolio x such that $\|Bx\| \le r + \varepsilon$ for $\varepsilon = 10^{-5}$, and then

$$x^T \Sigma x \le (r+\varepsilon)^2 = r^2 + 2 \times 0.04 \times 10^{-5} + 10^{-10} < r^2 + 10^{-6}.$$

We start with solving the LP ($LP(0)$ in what follows) obtained from (4.22) after removing the quadratic inequality. The point $x^{(0)} = (0,0,0,1)^T$ is the only optimal solution to this LP. Then $y^{(0)} = Bx^{(0)} = (0,-0.008,0.006,0.1997)^T$, and since $\|y^{(0)}\| = 0.19995 > r = 0.04$, we compute the first cut:

$$\frac{1}{r}\left(y^{(0)}\right)^T Bx = -0.01x_2 + 0.015x_3 + 0.999502x_4 \le \|y^{(0)}\| = 0.19995.$$

Adding this inequality to $LP(0)$ and reoptimizing the extended LP (denoted as $LP(1)$), we get its optimal solution $x^{(1)}$, which is presented in Table 4.1. Column 2 of this table contains optimal solutions to all the LPs solved by our cutting plane algorithm: $LP(i+1)$ is obtained from $LP(i)$ by adding to the latter the inequality $\frac{1}{r}\left(y^{(i)}\right)^T Bx \le \|y^{(i)}\|$, which is presented in Column 3 of Row i, that separates from X_Σ an optimal solution, $x^{(i)}$, to $LP(i)$. Here $y^{(i)} \overset{\text{def}}{=} Bx^{(i)}$.

Table 4.1 Cutting planes for portfolio problem example

i	$(x^{(i)})^T$	$\frac{1}{r}(y^{(i)})^T Bx \le \|y^{(i)}\|$
0	$(0,0,0,1)$	$-0.010000x_1 + 0.015000x_2 + 0.999502x_3 \le 0.199950$
1	$(0.000000, 0.000000, 0.812139, 0.187861)$	$-0.001879x_2 + 0.205853x_3 + 0.199950x_4 \le 0.090497$
2	$(0.000000, 0.549577, 0.248606, 0.201817)$	$0.032330x_2 + 0.065179x_3 + 0.199950x_4 \le 0.054525$
3	$(0.000000, 0.553723, 0.390366, 0.055911)$	$0.034049x_2 + 0.098430x_3 + 0.056202x_4 \le 0.049161$
4	$(0.000000, 0.704495, 0.202844, 0.092661)$	$0.043104x_2 + 0.052101x_3 + 0.088612x_4 \le 0.044338$
5	$(0.140220, 0.481318, 0.272385, 0.106078)$	$0.029022x_2 + 0.069687x_3 + 0.105297x_4 \le 0.042010$
6	$(0.096975, 0.573079, 0.262967, 0.066979)$	$0.035148x_2 + 0.066747x_3 + 0.065160x_4 \le 0.041017$
7	$(0.117786, 0.560763, 0.227810, 0.093641)$	$0.034111x_2 + 0.058357x_3 + 0.091404x_4 \le 0.040488$
8	$(0.096812, 0.606008, 0.222317, 0.074862)$	$0.037127x_2 + 0.056702x_3 + 0.072099x_4 \le 0.040251$
9	$(0.117884, 0.561278, 0.241584, 0.079254)$	$0.034287x_2 + 0.061585x_3 + 0.077225x_4 \le 0.040121$
10	$(0.112453, 0.580639, 0.223040, 0.083868)$	$0.035451x_2 + 0.057018x_3 + 0.081366x_4 \le 0.040063$
11	$(0.122751, 0.558533, 0.232882, 0.085834)$	$0.034050x_2 + 0.059508x_3 + 0.083699x_4 \le 0.040030$
12	$(0.116129, 0.570827, 0.231828, 0.081216)$	$0.034865x_2 + 0.059175x_3 + 0.078945x_4 \le 0.040016$
13	$(0.118105, 0.569426, 0.227866, 0.084603)$	$\|y^{(13)}\| = 0.040008 < 0.04 + 10^{-5}$

We stopped calculations after 13 iterates because $\|y^{(13)}\| = 0.040008 < r + 10^{-5}$. So, we declare the portfolio $x^{(13)}$ as an approximate optimal solution to our example, the expected return of this portfolio is $\bar{p}^T x^{(13)} = 1.064398$. $\square$

Let us note that an efficient implementation of the dual simplex method as a cutting plane algorithm not only adds cuts to the problem constraints but it also deletes cuts added at some previous stages if those cuts are not tight at an optimal solution to the currently solved LP.

In conclusion, let us also note that the cutting plane algorithm based on the dual simplex method is not the best choice for solving convex (in particular, quadratic) optimization problems. Nevertheless, when an optimization problem with convex constraints involves integer variables, then the representation of these convex constraints by separation procedures withing a branch-and-cut algorithm — which is a combinations of the cutting plane and branch-and-bound methods (see Sect. 6.2) — may be a reasonable solution approach.

4.7.2 *Exact Separation Procedure*

Here we show how to separate a given point $\tilde{x} \in \mathbb{R}^n$ from a given set $X \subseteq \mathbb{R}^n$ using an algorithm for solving (4.20).

First, solving (4.20) with different objective vectors $c = c^1, \dots, c^m$, we find a family of vectors $\{x^1, \dots, x^m\} \in X$. We can also start with an empty family by setting $m = 0$. Then we look for a hyperplane $H(a, \beta)$ separating our point $\tilde{x}$ from the set $\{x^1, \dots, x^m\}$. To do this we solve the following LP:

$$\begin{aligned}
&\tilde{x}^T v - \alpha \to \max,\\
&(x^i)^T v - \alpha \le 0,\ i = 1,\dots,m,\\
&-1 \le v_i \le 1,\ i = 1,\dots,n,\\
&-1 \le \alpha \le 1.
\end{aligned} \tag{4.23}$$

Let $a = v^*$, $\beta = \alpha^*$ denote the components of an optimal solution to (4.23). Since $v = 0$ and $\alpha = 0$ is a feasible solution to (4.23), then $a^T\tilde{x} \ge \beta$. Let us consider two cases.

1. First we consider the case when $a^T\tilde{x} > \beta$. To verify that the entire set X lies in the half-space $a^T x \le \beta$, we find an optimal solution x^* to (4.20) when $c = a$. If $a^T x^* \le \beta$, then $a^T x = \beta$ is a separating hyperplane. Otherwise, we set $x^{m+1} = x^*$ and increment m by 1, then we again solve (4.23) to find a new hyperplane $a^T x = \beta$. We continue to act this way until we find a separating hyperplane, or until we get the equality $a^T\tilde{x} = \beta$.

2. Now we consider the case when $a^T\tilde{x} = \beta$. Let $y^* \in \mathbb{R}^{m+2n+2}$ be a solution to the dual LP for (4.23):

$$\begin{aligned}
&\sum_{i=1}^{2n+2} y_{m+i} \to \min,\\
&\sum_{i=1}^{m} x^i_j y_i + y_{m+i} - y_{m+n+i} = \tilde{x}_j, \quad j = 1,\dots,n,\\
&-\sum_{i=1}^{m} y_i + y_{m+2n+1} - y_{m+2n+2} = -1,\\
&y_i \ge 0, \quad i = 1,\dots,m+2n+2.
\end{aligned}$$

Since the optimal primal and dual objective values are equal, then $y^*_{m+i} = 0$ for $i = 1,\dots,2n+2$ and, therefore, the vector equality $\sum_{i=1}^m y^*_i x^i = \tilde{x}$ holds, which means that

$$\tilde{x} \in \operatorname{conv}(\{x^1,\dots,x^m\}) \subseteq \operatorname{conv}(X).$$

It is clear that in practice the separation procedure described above can be used only for those sets X for which (4.20) is solved easily.

Example 4.8 *We need to separate the point* $\tilde{x} = \left(1, \frac{4}{5}, \frac{1}{5}\right)^T$ *from the knapsack set* $X_1 = \{x \in \{0,1\}^3 :\ 4x_1 + 5x_2 + 5x_3 \le 9\}$.

Solution. First, we set $m = 0$ and solve the next LP:

$$\begin{aligned}
v_1 + \frac{4}{5}v_2 + \frac{1}{5}v_3 - \alpha &\to \max,\\
-1 \le v_1, v_2, v_3 &\le 1,\\
-1 \le \alpha &\le 1.
\end{aligned}$$

Its solution is $v^* = (1,1,1)^T$, $\alpha^* = -1$, and therefore we set $a = v^* = (1,1,1)^T$, $\beta = \alpha^* = -1$. Since $a^T\tilde{x} = 2 > -1 = \beta$, we need to solve the following 0,1-knapsack problem:

$$\begin{aligned} x_1 + x_2 + x_3 &\to \max, \\ 4x_1 + 5x_2 + 5x_3 &\le 9, \\ x_1, x_2, x_3 &\in \{0,1\}. \end{aligned}$$

Its solution is $x^* = (1,1,0)^T$. As $a^T x^* = 2 > -1 = \beta$, we set $x^1 = (1,1,0)^T$ and $m = 1$.

Now we solve the following LP:

$$\begin{aligned} v_1 + \frac{4}{5}v_2 + \frac{1}{5}v_3 - \alpha &\to \max, \\ v_1 + v_2 - \alpha &\le 0, \\ -1 \le v_1, v_2, v_3 &\le 1, \\ -1 \le \alpha &\le 1. \end{aligned}$$

Its solution is $v^* = (1,0,1)^T$, $\alpha^* = 1$, and therefore we set $a = (1,0,1)^T$, $\beta = 1$. Since $a^T \tilde{x} = \frac{6}{5} > 1 = \beta$, we need to solve the following 0,1-knapsack problem:

$$\begin{aligned} x_1 + x_3 &\to \max, \\ 4x_1 + 5x_2 + 5x_3 &\le 9, \\ x_1, x_2, x_3 &\in \{0,1\}. \end{aligned}$$

Its solution is $x^* = (1,0,1)^T$. As $a^T x^* = 2 > 1 = \beta$, we set $x^2 = (1,0,1)^T$ and $m = 2$.

Now we solve the next LP:

$$\begin{aligned} v_1 + \frac{4}{5}v_2 + \frac{1}{5}v_3 - \alpha &\to \max, \\ v_1 + v_2 - \alpha &\le 0, \\ v_1 + v_3 - \alpha &\le 0, \\ -1 \le v_1, v_2, v_3 &\le 1, \\ -1 \le \alpha &\le 1. \end{aligned}$$

Its solution is $v^* = (1,0,0)^T$, $\alpha^* = 1$, and therefore we set $a = (1,0,1)^T$, $\beta = 1$. As $a^T \tilde{x} = 1 = \beta$, then $\tilde{x} \in \operatorname{conv}(\{x^1, x^2\})$. This can also be seen directly:

$$\frac{4}{5}x^1 + \frac{1}{5}x^2 = \frac{4}{5}\begin{pmatrix}1\\1\\0\end{pmatrix} + \frac{1}{5}\begin{pmatrix}1\\0\\1\end{pmatrix} = \begin{pmatrix}1\\ \frac{4}{5}\\ \frac{1}{5}\end{pmatrix}.$$

□

We will solve one more similar example, but with the other outcome.

Example 4.9 *We need to separate the point* $\tilde{x} = \left(\frac{3}{4}, \frac{4}{5}, \frac{1}{2}\right)^T$ *from the knapsack set* $X_2 = \{x \in \{0,1\}^3 : 4x_1 + 5x_2 + 2x_3 \le 9\}$.

Solution. We set $m = 0$ and solve the next LP:

$$\begin{aligned} \frac{3}{4}v_1 + \frac{4}{5}v_2 + \frac{1}{2}v_3 - \alpha &\to \max, \\ -1 \le v_1, v_2, v_3 &\le 1, \\ -1 \le \alpha &\le 1. \end{aligned}$$

Its solution is $v^* = (1,1,1)^T$, $\alpha^* = -1$, and therefore $a = v^* = (1,1,1)^T$, $\beta = \alpha^* = -1$. Since $a^T\tilde{x} = \frac{41}{20} > -1 = \beta$, we need to solve the following 0,1-knapsack problem:

$$\begin{aligned} x_1 + x_2 + x_3 &\to \max, \\ 4x_1 + 5x_2 + 2x_3 &\le 9, \\ x_1, x_2, x_3 &\in \{0,1\}. \end{aligned}$$

Its solution is $x^* = (1,1,0)^T$. As $a^T x^* = 2 > -1 = \beta$, we set $x^1 = (1,1,0)^T$ and $m = 1$.

Now we solve the following LP:

$$\begin{aligned} \frac{3}{4}v_1 + \frac{4}{5}v_2 + \frac{1}{2}v_3 - \alpha &\to \max, \\ v_1 + v_2 - \alpha &\le 0, \\ -1 \le v_1, v_2, v_3 &\le 1, \\ -1 \le \alpha &\le 1. \end{aligned}$$

Its solution is $v^* = (-1,0,1)^T$, $\alpha^* = -1$, and therefore we set $a = (-1,0,1)^T$, $\beta = -1$. Since $a^T\tilde{x} = -\frac{1}{4} > -1 = \beta$, we need to solve the following 0,1-knapsack problem:

$$\begin{aligned} -x_1 + x_3 &\to \max, \\ 4x_1 + 5x_2 + 2x_3 &\le 9, \\ x_1, x_2, x_3 &\in \{0,1\}. \end{aligned}$$

Its solution is $x^* = (0,0,1)^T$. As $a^T x^* = 1 > -1 = \beta$, we set $x^2 = (0,0,1)^T$ and $m = 2$.

Now we solve the following LP:

$$\begin{aligned} \frac{3}{4}v_1 + \frac{4}{5}v_2 + \frac{1}{2}v_3 - \alpha &\to \max, \\ v_1 + v_2 - \alpha &\le 0, \\ v_3 - \alpha &\le 0, \end{aligned}$$

$$\begin{aligned} -1 \le v_1, v_2, v_3 \le 1,\\ -1 \le \alpha \le 1. \end{aligned}$$

Its solution is $v^* = (0,1,1)^T$, $\alpha^* = 1$, and therefore we set $a = (0,1,1)^T$, $\beta = 1$. Since $a^T\tilde{x} = \frac{13}{10} > 1 = \beta$, we need to solve the following 0,1-knapsack problem:

$$\begin{aligned} x_2 + x_3 &\to \max,\\ 4x_1 + 5x_2 + 2x_3 &\le 9,\\ x_1, x_2, x_3 &\in \{0,1\}. \end{aligned}$$

Its solution is $x^* = (0,1,1)^T$. As $a^T x^* = 2 > 1 = \beta$, we set $x^3 = (0,1,1)^T$ and $m = 3$.

Now we solve the next LP:

$$\begin{aligned} \frac{3}{4}v_1 + \frac{4}{5}v_2 + \frac{1}{2}v_3 - \alpha &\to \max,\\ v_1 + v_2 - \alpha &\le 0,\\ v_3 - \alpha &\le 0,\\ v_2 + v_3 - \alpha &\le 0,\\ -1 \le v_1, v_2, v_3 &\le 1,\\ -1 \le \alpha &\le 1. \end{aligned}$$

Its solution is $v^* = (1,0,1)^T$, $\alpha^* = 1$, and therefore we set $a = (1,0,1)^T$, $\beta = 1$. Since $a^T\tilde{x} = \frac{5}{4} > 1 = \beta$, we need to solve the following 0,1-knapsack problem:

$$\begin{aligned} x_1 + x_3 &\to \max,\\ 4x_1 + 5x_2 + 2x_3 &\le 9,\\ x_1, x_2, x_3 &\in \{0,1\}. \end{aligned}$$

Its solution is $x^* = (1,0,1)^T$. As $a^T x^* = 2 > 1 = \beta$, we set $x^4 = (1,0,1)^T$ and $m = 4$.

Now we solve the following LP:

$$\begin{aligned} \frac{3}{4}v_1 + \frac{4}{5}v_2 + \frac{1}{2}v_3 - \alpha &\to \max,\\ v_1 + v_2 - \alpha &\le 0,\\ v_3 - \alpha &\le 0,\\ v_2 + v_3 - \alpha &\le 0,\\ v_1 + v_3 - \alpha &\le 0,\\ -1 \le v_1, v_2, v_3 &\le 1,\\ -1 \le \alpha &\le 1. \end{aligned}$$

Its solution is $v^* = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^T$, $\alpha^* = 1$, and therefore we set $a = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^T$, $\beta = 1$. Since $a^T\tilde{x} = \frac{41}{40} > 1 = \beta$, we need to solve the following 0,1-knapsack problem:

$$\begin{aligned}\frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 &\to \max,\\ 4x_1 + 5x_2 + 2x_3 &\le 9,\\ x_1, x_2, x_3 &\in \{0,1\}.\end{aligned}$$

Its solution is $x^* = (1,1,0)^T$. Since $a^T x^* = 1 = \beta$, then the hyperplane

$$\frac{1}{2}x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_3 = 1$$

separates $\tilde{x}$ from X_2, and the inequality $x_1 + x_2 + x_3 \le 2$ is valid for X_2. □

4.8 Notes

The reviews [88], [144] and [39] are a good addition to the material of this and the next chapters.

***Sect.* 4.2.** Theorem 4.1 was proved by Chvátal [35] for the case when $P(A,b)$ is a polytope. Schrijver showed in [121] that this result is also valid for arbitrary polyhedra $P(A,b)$. Historically, the first cutting plane algorithm for solving MIPs was developed by Gomory [58, 60]. This algorithm uses the cuts described in Theorem 4.2. In the general case, the separation problem for the Chvátal-Gomory inequalities is **NP**-hard in the strong sense [50]. Efficient separation procedures have been proposed for totally tight cuts [86] (see Exercise 4.7), as well as for some special polyhedra, provided that all components of the vector u are 0 or $1/2$ [32].

The inequalities (4.4), known as the *flower* inequalities, were introduced in [48].

***Sect.* 4.3.** Mixed integer rounding was studied in [98, 99]. Many known classes of strong cuts for a series of structured mixed integer sets can be obtained by applying mixed integer rounding [89].

***Sect.* 4.4.** The fractional Gomory cuts were introduced in [59]. In the same paper it was proved that the cutting plane algorithm based on these cuts solves any general MIP in a finite number of steps, provided that the objective function takes integral values on all feasible solutions of this MIP.

***Sect.* 4.5.** The disjunctive principle was first implicitly present in the derivation of the fractional Gomory cuts. The approach we outlined is based on the Balas characterization of the disjunction of polyhedra [11]. The disjunction of polyhedra from different spaces can be represented more efficiently [12] (see Exercise 4.12). In [85] it is shown that many classes of facet defining inequalities for some well known combinatorial optimization problems can be obtained by applying the disjunctive technique.

***Sect.* 4.6.** The lift-and-project procedure presented here was proposed in [13]. A more powerful procedure, in which lifting is performed simultaneously for all binary variables, was studied in [127]. An even more powerful lift-and-project procedure was proposed in [87], but here the separation procedure is reduced to solving a semidefined programming problem.

***Sect.* 4.7.** The equivalence of the optimization and separation problems was established in [63] (see also [64, 122]).

Markowitz received the 1990 Nobel Prize in Economics for his portfolio optimization model [90].

***Sect.* 4.9.** The statements of exercises 4.2, 4.5, 4.6, 4.7, 4.12 are, respectively, taken from [91], [31], [49], [86], [12].

4.9 Exercises

4.1. Let A be a real $m \times n$-matrix, $b \in \mathbb{R}^m$, $S \subseteq \{1,\dots,n\}$. Prove that if the mixed-integer set $P(A,b;S)$ is bounded, then its convex hull, $\mathrm{conv}(P(A,b;S))$, is a polytope.

4.2. Let us consider a very simple generalization of the mixed-integer set X from Lemma 4.1:

$$\bar{X} = \{(x,y) \in \mathbb{Z} \times \mathbb{R}^m : \ x - y_i \le b_i,\ i = 1,\dots,m\}.$$

Prove that $\mathrm{conv}(\bar{X})$ is the solution set of the following system of inequalities:

$$\begin{aligned} x - y_i &\le b_i, \quad i = 1,\dots,m,\\ x - \frac{y_i}{1-f_i} &\le \lfloor b \rfloor, \quad i = 1,\dots,m,\\ y_i &\ge 0, \quad i = 1,\dots,m. \end{aligned}$$

Here $f_i \stackrel{\mathrm{def}}{=} b_i - \lfloor b_i \rfloor$ for $i = 1,\dots,m$.

4.3. Let A be a real $m \times n$-matrix, and $b \in \mathbb{R}^m$. Let $f(r) \stackrel{\mathrm{def}}{=} r - \lfloor r \rfloor$ denote the fractional part of $r \in \mathbb{R}$. Assume that $f(u^T b) < \frac{1}{2}$ holds for all $u \in \mathbb{R}^m_+$, and t is a positive integer such that $\frac{1}{2} \le t f(u^T b) < 1$. Let $\bar{u} = (f(t\,u_1),\dots,f(t\,u_m))$. Prove that, for the set $P(A,b) \cap \mathbb{Z}^n_+$, the cut $\lfloor \bar{u}^T A \rfloor x \le \lfloor \bar{u}^T b \rfloor$ is not weaker than the cut $\lfloor u^T A \rfloor x \le \lfloor u^T b \rfloor$.

4.4. A function $h : \mathbb{R} \to \mathbb{R}$ is called *superadditive*, if

$$h(r_1) + h(r_2) \le h(r_1 + r_2) \quad \text{fot all } r_1, r_2 \in \mathbb{R}.$$

Let $\beta \in (0,1)$ and $f(r) = r - \lfloor r \rfloor$. Prove that the functions $\lfloor r \rfloor$ and

$$g_\beta(r) \stackrel{\mathrm{def}}{=} \lfloor r \rfloor + \frac{\max\{0, f(r) - f(\beta)\}}{1 - f(\beta)}$$

are superadditive.

4.5. Prove the validity of the following generalization of the *rounding principle*: if g is a non-decreasing superadditive function and $g(0) = 0$, then the inequality

$$\sum_{j=1}^{n} g(\alpha_j)x_j \le g(\beta)$$

is valid for all non-negative integer solutions of the inequality

$$\sum_{j=1}^{n} \alpha_j x_j \le \beta.$$

4.6. Prove the validity of the following analogue of Farkas' lemma.

Theorem 4.8. *Let A be a rational $m \times n$-matrix, and b be a rational m-vector. A linear system $Ax = b$ has no integer solutions if and only if there exists a rational m-vector y such that $y^T A$ is an integer-valued n-vector, but $y^T b$ is not integer.*

4.7. Let A be a rational $m \times n$-matrix, and b be a rational m-vector. A Chvátal-Gomory cut $\sum_{j=1}^{n} \lfloor u^T A^j \rfloor x \le \lfloor u^T b \rfloor$ that cuts off a point $\tilde{x} \in P(A,b) \subseteq \mathbb{R}^n_+$ from the set $P(A,b) \cap \mathbb{Z}^n$ is called *totally tight* if $u^T(b - A\tilde{x}) = 0$, i.e., this cut is derived from the inequalities that are satisfied at $\tilde{x}$ as equalities. How can we use the result of Theorem 4.8 to construct a separation procedure for the class of totally tight Chvátal-Gomory cuts?

4.8. For the set

$$X = \{x \in \mathbb{Z}^n_+ : \ x_i + x_j \le 1, \quad i, j = 1, \dots, n, \ i \ne j\},$$

prove that

$$\operatorname{conv}(X) = \left\{ x \in \mathbb{R}^n_+ : \ \sum_{j=1}^{n} x_j \le 1 \right\}$$

and the Chvátal rank of the inequality $\sum_{j=1}^{n} x_j \le 1$ is $O(n \log n)$.

4.9. Prove that the Chvátal rank of the solution set of the following system

$$\begin{aligned} tx_1 + x_2 &\le 1 + t, \\ -tx_1 + x_2 &\le 1, \\ x_1 \qquad &\le 1, \\ x_1, x_2 &\ge 0 \end{aligned}$$

is equal to $t - 1$ for $t = 1, 2, \dots$.

4.10. Apply the cutting plane algorithm that generates only Chvátal-Gomory cuts to solve the following IPs:

$$\text{a)} \quad \begin{aligned} x_1 + x_2 &\to \max, \\ -x_1 + x_2 &\le 1, \\ 3x_1 + 2x_2 &\le 4, \\ x_1, x_2 &\in \mathbb{Z}_+; \end{aligned} \qquad \text{b)} \quad \begin{aligned} x_1 + x_2 + x_3 &\to \max, \\ 2x_1 + 2x_2 + x_3 &\le 6, \\ x_1 + \qquad x_3 &\le 2, \\ x_2 + x_3 &\le 2, \\ x_1, x_2, x_3 &\in \mathbb{Z}_+. \end{aligned}$$

4.11. Apply the cutting plane algorithm that generates only fractional Gomory cuts to solve the following MIPs:

$$\text{a)}\quad \begin{aligned} x_1+2x_2 &\to \max,\\ x_1+\ x_2 &\le 4,\\ -x_1+\ x_2 &\le 1,\\ x_1 &\in \mathbb{Z}_+,\\ x_2 &\ge 0; \end{aligned} \qquad \text{b)}\quad \begin{aligned} x_3 &\to \max,\\ x_1+\ x_2+x_3 &\le 2,\\ -x_1 \qquad +x_3 &\le 0,\\ -x_2+x_3 &\le 0,\\ x_1,x_2 &\in \mathbb{Z}_+,\\ x_3 &\ge 0. \end{aligned}$$

4.12. For $k=1,2$, let $P_k=\{x\in[0,1]^{n_k}: A_kx\ge 1\}$ be a monotone polyhedron, where $A_k=[a^k_{ij}]$ is a non-negative real $m_k\times n_k$-matrix ($a^k_{ij}\ge 0$). Let $N_k \stackrel{\text{def}}{=} \{1,\dots,n_k\}$, $M_k \stackrel{\text{def}}{=} \{1,\dots,m_k\}$, and $\mathscr{F}^k_i \stackrel{\text{def}}{=} \left\{S\subseteq N_k : \sum_{j\in N_k\setminus S} a^k_{ij} < 1\right\}$ for $i\in M_k$.

a) Prove that the convex hull of the union $(P_1\times[0,1]^{n_2})\cup([0,1]^{n_1}\times P_2)$ is described by the following system:

$$\begin{gathered} 0\le x^k_j\le 1,\quad j=1,\dots,n_k,\ k=1,2,\\ \sum_{j\in S_1}\frac{a^1_{i_1,j}}{1-\sum\limits_{l\in N_1\setminus S_1} a^1_{i_1,l}}x^1_j+\sum_{j\in S_2}\frac{a^2_{i_2,j}}{1-\sum\limits_{l\in N_2\setminus S_2} a^2_{i_2,l}}x^2_j\ge 1,\\ S_1\subseteq \mathscr{F}^1_{i_1},\ S_2\subseteq \mathscr{F}^2_{i_2},\ i_1\in M_1,\ i_2\in M_2. \end{gathered} \tag{4.24}$$

b) For $k\in\{1,2\}$, $x\in[0,1]^{n_k}$ and $\alpha\in[0,1]$, let $S^k(\alpha,x)\stackrel{\text{def}}{=}\{j\in N_k : x_j\le\alpha\}$ and, for $i\in M_k$, let

$$\begin{aligned} \delta^k_i(x) &\stackrel{\text{def}}{=} \max\left\{x_j : \sum_{l\in S^k(x_j,x)} a^k_{il}x_l \le x_j\left(1-\sum_{l\in N_k\setminus S^k(x_j,x)} a_{il}\right)\right\},\\ \alpha^k_i(x) &\stackrel{\text{def}}{=} \sum_{j\in S^k(\delta^k_i(x),x)} \frac{a^k_{ij}x_j}{1-\sum\limits_{l\in N_k\setminus S^k(\delta^k_i(x),x)} a^k_{il}}. \end{aligned}$$

Prove that a given point $(\tilde{x}^1,\tilde{x}^2)\in[0,1]^{n_1}\times[0,1]^{n_2}$ satisfies all inequalities in (4.24) if and only if $\alpha^1_{i_1}(\tilde{x}^1)+\alpha^2_{i_2}(\tilde{x}^2)\ge 1$ for all pairs $(i_1,i_2)\in M_1\times M_2$. If for some $(i_1,i_2)\in M_1\times M_2$, $\alpha^1_{i_1}(\tilde{x}^1)+\alpha^2_{i_2}(\tilde{x}^2)<1$, then for $S_1=S^1(\delta^1_{i_1}(\tilde{x}^1),\tilde{x}^1)$ and $S_2=S^2(\delta^2_{i_2}(\tilde{x}^2),\tilde{x}^2)$ the corresponding inequality in (4.12) is violated at $(\tilde{x}^1,\tilde{x}^2)$.

4.13. Elaborate separation procedures for the following sets:

a) (*solution set of a convex quadratic constraint*)

$$X_1=\left\{x\in\mathbb{R}^n : x^TQx+2c^Tx\le d\right\},$$

where Q is a real symmetric positive defined $n\times n$-matrix, $c\in\mathbb{R}^n$, $d\in\mathbb{R}$;

b) (*norm cone*) $X_2 = \{(x,t) \in \mathbb{R}^n \times \mathbb{R}_+ : \|x\| \leq t\}$.

Hint. a) Use the substitution $y = L^T x + L^{-1} c$, where $Q = LL^T$ and L is a non-degenerate $n \times n$-matrix. b) Given a point $(\tilde{x}, \tilde{t})$ such that $\|\tilde{x}\| > \tilde{t}$, compute its projection $(\hat{x}, \hat{t})$ onto X_2, then the hyperplane passing through $(\hat{x}, \hat{t})$ and orthogonal to the vector $(\tilde{x} - \hat{x}, \tilde{t} - \hat{t})$ separates $(\tilde{x}, \tilde{t})$ from X_2.

Chapter 5
Cuts for Structured Mixed-Integer Sets

Many optimization problems include various specific ”local” structures. For example, very often, MIPs contain inequalities involving only binary variables, or, sometimes, a part of the problem constraints formulate a network flow problem. For such local structures, we can get stronger inequalities using specific features of these structures. It should be noted that there exist a great deal of such special structures. In this chapter we consider a number of structural cuts for those mixed-integer sets that are most often encountered in practical problems, and the separation procedures for which have already been included in many modern **MIP** solvers.

5.1 Knapsack Inequalities

Constraints involving binary variables are so common in practice that they deserve special attention. Consider a *knapsack set* of $0,1$-vectors

$$K = \left\{ x \in \{0,1\}^n : \sum_{j=1}^{n} a_j x_j \le b \right\} \tag{5.1}$$

when all coefficients, b and a_j, are positive. A set $C \subseteq N \overset{\text{def}}{=} \{1,\dots,n\}$ is called a *knapsack cover* (or simply *cover*) if its *excess* $\lambda(C) \overset{\text{def}}{=} \sum_{j\in C} a_j - b$ is positive. Each knapsack cover C defines the *cover inequality*:

$$\sum_{j\in C} x_j \le |C| - 1, \tag{5.2}$$

which is valid for K, and which implies that not all variables x_j for $j \in C$ can simultaneously be equal to 1.

A knapsack cover C is called *minimal* if $a_j \ge \lambda(C)$ for all $j \in C$. If C is not a minimal knapsack cover, then the cover inequality written for C is redundant, since it is the sum of the inequalities

$$\sum_{j\in C'} x_j \le |C'|-1 \quad \text{and} \quad x_j \le 1 \text{ for } j \in C\setminus C',$$

where C' is a minimal cover contained in C.

5.1.1 Separation Problem For Cover Inequalities

For a given point $\tilde{x} \in [0,1]^n$, we need to find a violated at this point inequality if such one exists. Let us rewrite (5.2) in the following form:

$$\sum_{j\in C}(1-x_j) \ge 1.$$

Hence, it is clear that in order to solve the separation problem, we need to answer the following question: is there a subset $C \subseteq N$ such that

$$\sum_{j\in C} a_j > b \quad \text{and} \quad \sum_{j\in C}(1-\tilde{x}_j) < 1?$$

This question can be rewritten also as follows:

$$\min_{C\subseteq N}\left\{\sum_{j\in C}(1-\tilde{x}_j):\ \sum_{j\in C} a_j > b\right\} < 1?$$

Let us introduce an n-vector z of binary variables to represent any cover C, where $z_j = 1$ if $j \in C$, and $z_j = 0$ if $j \in N\setminus C$. Now to answer the above question, we have to solve the following optimization problem:

$$\begin{aligned} \sum_{j=1}^{n}(1-\tilde{x}_j)z_j &\to \min,\\ \sum_{j=1}^{n} a_j z_j &> b, \\ z &\in \{0,1\}^n. \end{aligned} \tag{5.3}$$

Summarizing the arguments presented above, we formulate the following result.

Theorem 5.1. *Let ξ^* an z^* be the the optimal objective value and an optimal solution to* (5.3). *If $\xi^* \ge 1$, then the point $\tilde{x}$ satisfies all the cover inequalities. If $\xi^* < 1$, then for $C^* = \{j \in N:\ z_j^* = 1\}$ the inequality $\sum_{j\in C^*} x_j \le |C^*|-1$ is violated at $\tilde{x}$ by* $1-\xi^*$.

We can solve (5.3), which is a 0,1-knapsack problem, of sufficiently large size relatively quickly using the recurrence formula (1.30) (for this, (5.3) must first be represented in the form (1.28) by substituting y_j for $1-z_j$). But separation procedures must be running very quickly, because they are called by the cutting plane algorithms very often. In practice, (5.3) does not need to be solved to optimality, instead, we can solve it approximately, but very quickly. Of course, solving (5.3) approximately, we risk not finding a cover inequality violated at $\tilde{x}$, even if such one exists.

Next we describe a simple efficient algorithm that solves (5.3) approximately.

1. List the ratios $\frac{1-\tilde{x}_j}{a_j}$ in non-decreasing order:

$$\frac{1-\tilde{x}_{\pi(1)}}{a_{\pi(1)}} \le \frac{1-\tilde{x}_{\pi(2)}}{a_{\pi(2)}} \le \dots \le \frac{1-\tilde{x}_{\pi(n)}}{a_{\pi(n)}}.$$

2. Find a minimal index k such that $\sum_{i=1}^{k} a_{\pi(i)} > b$.
3. Return $C = \{\pi(1), \dots, \pi(k)\}$.

The above algorithm is also known as the **LP**-*heuristic* because when all coefficients a_j are integers, it constructs a solution that can be obtained by rounding up the components of an optimal solution to the following LP:

$$\min \left\{ \sum_{j=1}^{n} (1-\tilde{x}_j)\, z_j : \sum_{j=1}^{n} a_j z_j \ge b+1,\ z \in [0,1]^n \right\}.$$

5.2 Lifting Inequalities

The technique of *lifting inequalities* allows us to strengthen an inequality that is valid for some subset of a given discrete set, and obtain an inequality that is valid for the whole set. Its essence is as follows.

Theorem 5.2. *Let $X \subseteq \{0,1\}^n$, $X^\delta \overset{\text{def}}{=} \{x \in X : x_n = \delta\}$ for $\delta \in \{0,1\}$ and the inequality*

$$\sum_{j=1}^{n-1} a_j x_j \le b \tag{5.4}$$

is valid for X^δ.

($\delta = 0$, *lifting up*) *If $X^1 = \emptyset$, then $x_n = 0$ for all $x \in X$. If $X^1 \ne \emptyset$, then the inequality*

$$\sum_{j=1}^{n-1} a_j x_j + \alpha_n x_n \le b \tag{5.5}$$

is valid for X when $\alpha_n \le b - \zeta^1$, where

$$\zeta^1 = \max \left\{ \sum_{j=1}^{n-1} a_j x_j : x \in X^1 \right\}.$$

Moreover, if $\alpha_n = b - \zeta^1$ and (5.4) *is a facet defining inequality for* $\operatorname{conv}(X^0)$*, then* (5.5) *defines a facet for* $\operatorname{conv}(X)$.

($\delta = 1$, *lifting down*) *If* $X^0 = \emptyset$, *then* $x_n = 1$ *for all* $x \in X$. *If* $X^0 \neq \emptyset$, *then the inequality*

$$\sum_{j=1}^{n-1} a_j x_j + \gamma_n x_n \leq b + \gamma_n \tag{5.6}$$

is valid for X *when* $\gamma_n \geq \zeta^0 - b$, *where*

$$\zeta^0 = \max\left\{ \sum_{j=1}^{n-1} a_j x_j : x \in X^0 \right\}.$$

Moreover, if $\gamma_n = \zeta^0 - b$ *and* (5.4) *is a facet defining inequality for* $\operatorname{conv}(X^1)$, *then* (5.6) *defines a facet for* $\operatorname{conv}(X)$.

Proof. We consider only the lifting up case when $\delta = 0$. The lifting down case when $\delta = 1$ is considered in a similar way.

Let $\bar{x} \in X$. If $\bar{x}_n = 0$, then

$$\sum_{j=1}^{n-1} a_j \bar{x}_j + \alpha_n \bar{x}_n = \sum_{j=1}^{n-1} a_j \bar{x}_j \leq b.$$

If $\bar{x}_n = 1$, then, by definition of ζ^1, we have

$$\sum_{j=1}^{n-1} a_j \bar{x}_j + \alpha_n \bar{x}_n = \sum_{j=1}^{n-1} a_j \bar{x}_j + \alpha_n \leq \zeta^1 + \alpha_n \leq b.$$

Suppose now that $X^1 \neq \emptyset$, the dimension of the set X^0 is d, $\alpha_n = b - \zeta^1$ and (5.4) is a facet defining inequality for $\operatorname{conv}(X^0)$. Then X^0 contains d affine-independent points $x^1, \dots, x^d$ for which (5.4) holds as an equality. Let $\zeta^1 = \sum_{j=1}^{n-1} a_j x_j^{d+1}$ for $x^{d+1} \in X^1$. Since $x_n^{d+1} = 1$, then x^{d+1} does not lie in the affine subspace generated by the points $x^1, \dots, x^d$. Hence, we have $d+1$ affine-independent point $x^1, \dots, x^d, x^{d+1}$ satisfying (5.5) as an equality. Therefore, by Proposition 1.4, (5.5) defines a facet for $\operatorname{conv}(X)$. $\square$

The lifting technique described in Theorem 5.2 can be applied successively: the inequality obtained as a result of lifting one variable can be lifted further by another variable. Next, we apply this technique to strengthen the cover inequalities.

5.2.1 Lifted Cover Inequalities

Let $C \subseteq N = \{1, \dots, n\}$ be a knapsack cover for the set K defined by (5.1). We want to strengthen the cover inequality $\sum_{j \in C} x_j \leq |C| - 1$, which is valid for K. We do this using the lifting technique described in Theorem 5.2.

Consider a partition (C_1, C_2) of the set C ($C_1 \cup C_2 = C$ and $C_1 \cap C_2 = \emptyset$) with $C_1 \neq \emptyset$. The inequality

$$\sum_{j \in C_1} x_j \le |C_1| - 1 \tag{5.7}$$

is valid for the set

$$K_0 = \{x \in K : \ x_j = 1 \ \text{ for } j \in C_2\}.$$

Lifting (5.7) up by the variables x_j for $j \in N \setminus C$, and down by the variables x_j for $j \in C_2$, we can obtain a *lifted cover inequality* (*LCI*)

$$\sum_{j \in C_1} x_j + \sum_{j \in N \setminus C_1} \alpha_j x_j \le |C_1| - 1 + \sum_{j \in C_2} \alpha_j \tag{5.8}$$

that is valid for K. For $j \in N \setminus C_1$, the coefficients α_j depend on the order in which the variables x_j are lifted. Let $j_1, \dots, j_k$ be some ordering of the elements of the set $N \setminus C_1$, $\gamma = |C_1| - 1$, $\beta = b - \sum_{j \in C_2} a_j$, and $\chi^{C_2} \in \{0,1\}^N$ be the characteristic vector of C_2 ($\chi_i^{C_2} = 1$ if $i \in C_2$, and $\chi_i^{C_2} = 0$ if $i \in N \setminus C_2$). The coefficients α_{j_i} can be calculated in series as follows.

Suppose that we have already obtained the inequality

$$\sum_{j \in C_1} x_j + \sum_{i=1}^{r-1} \alpha_{j_i} x_{j_i} \le \gamma$$

that is valid for the set K_{r-1}, where

$$K_s \stackrel{\text{def}}{=} \{x \in K : \ x_j = 1, \ j \in C_2 \setminus \{j_1, \dots, j_s\}\}.$$

We calculate the coefficient α_{j_r} for the next variable x_{j_r} to guarantee that the resulting inequality must be valid for the set K_r. By virtue of Theorem 5.2, it is necessary to solve the following 0,1-knapsack problem

$$\begin{aligned} &\sum_{j \in C_1} x_j + \sum_{i=1}^{r-1} \alpha_{j_i} x_{j_i} \to \max, \\ &\sum_{j \in C_1} a_j x_j + \sum_{i=1}^{r-1} a_{j_i} x_{j_i} \le \beta + (-1)^{(1-\chi_{j_r}^{C_2})} a_{j_r}, \\ &x_j \in \{0,1\}, \quad j \in C_1 \cup \{j_1, \dots, j_{r-1}\}. \end{aligned} \tag{5.9}$$

Letting ξ_r denote the optimal objective value in (5.9), we set

$$\begin{aligned} &\alpha_{j_r} = \gamma - \xi_r, \text{ if } j_r \in N \setminus C; \\ &\alpha_{j_r} = \xi_r - \gamma, \ \gamma = \xi_r, \ \beta := \beta + a_{j_r}, \text{ if } j_r \in C_2. \end{aligned}$$

If (5.9) is solved using the recurrence formula (1.31), the above lifting procedure runs in polynomial time. Let us also note again that different orderings of the set $N \setminus C_1$ result in different lifted cover inequalities.

Example 5.1 *Consider the knapsack set*

$$K^1 = \left\{x \in \{0,1\}^6 :\ 5x_1 + 6x_2 + 4x_3 + 6x_4 + 3x_5 + 8x_6 \le 16\right\}$$

and the point $\tilde{x} = \left(0, \frac{2}{3}, \frac{3}{4}, 1, 1, 0\right)$. *We need to find a lifted cover inequality that separates* $\tilde{x}$ *from* K^1.

Solution. First, let us write down (5.3) applied to our knapsack set K^1:

$$\begin{aligned}
z_1 + \frac{1}{3}z_2 + \frac{1}{4}z_3 + 0z_4 + 0z_5 + z_6 &\to \min,\\
5z_1 + 6z_2 + 4z_3 + 6z_4 + 3z_5 + 8z_6 &> 16,\\
z_1, z_2, z_3, z_4, z_5, z_6 &\in \{0,1\}.
\end{aligned}$$

To find a knapsack cover, we apply the LP-heuristic:

1. Sorting the numbers $\frac{1}{5}, \frac{1}{18}, \frac{1}{16}, 0, 0, \frac{1}{8}$, which are the ratios $\frac{1-\tilde{x}_j}{a_j}$, in non-decreasing order, we get the permutation $\pi = (4,5,2,3,6,1)$.
2. Now we decide that $k = 4$ and $C = \{4,5,2,3\}$.

Since $\sum_{j\in C}(1-\tilde{x}_j) = 0 + 0 + \frac{1}{3} + \frac{1}{4} = \frac{7}{12} < 1$, we have found the cover inequality violated at $\tilde{x}$: $x_2 + x_3 + x_4 + x_5 \le 3$.

Next let us lift this cover inequality for the partition $C_1 = C$, $C_2 = \emptyset$, and the lifting sequence $j_1 = 1$, $j_2 = 6$. We set $\beta = b = 16$, $\gamma = |C_1| - 1 = 3$. Note that when $C_2 = \emptyset$, the values of β and γ remain constant during the execution of the lifting algorithm.

To calculate α_1, we solve the following 0,1-knapsack problem:

$$\begin{aligned}
x_2 + x_3 + x_4 + x_5 &\to \max,\\
6x_2 + 4x_3 + 6x_4 + 3x_5 &\le 16 - 5 = 11,\\
x_2, x_3, x_4, x_5 &\in \{0,1\}.
\end{aligned}$$

An optimal solution to this problem is given by $x_2 = x_3 = 1$, $x_4 = x_5 = 0$, and the optimal objective value is $\xi_1 = 2$. Hence, $\alpha_1 = \gamma - \xi_1 = 3 - 2 = 1$.

Similarly, to find α_6, we solve the next 0,1-knapsack problem:

$$\begin{aligned}
x_1 + x_2 + x_3 + x_4 + x_5 &\to \max,\\
5x_1 + 6x_2 + 4x_3 + 6x_4 + 3x_5 &\le 16 - 8 = 9,\\
x_1, x_2, x_3, x_4, x_5 &\in \{0,1\}.
\end{aligned}$$

Its optimal solution is determined by $x_1 = x_5 = 1$, $x_2 = x_3 = x_4 = 0$, and the optimal objective value is $\xi_2 = 2$. Hence, $\alpha_6 = \gamma - \xi_2 = 3 - 2 = 1$.

Now we can write the lifted cover inequality

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \le 3$$

that separates $\tilde{x}$ from K^1. □

Example 5.2 *We need to find a lifted cover inequality that separates the point*

$$\tilde{x} = (0, 0.4, 0.5, 0.5, 0.7, 1.0)$$

from the knapsack set

$$K^2 = \left\{ x \in \{0,1\}^6 : \ 13x_1 + 7x_2 + 6x_3 + 5x_4 + 3x_5 + 10x_6 \le 22 \right\}.$$

Solution. Let us first try to find a maximally violated cover inequality by solving the following 0,1-knapsack problem:

$$\begin{aligned} z_1 + 0.6z_2 + 0.5z_3 + 0.5z_4 + 0.3z_5 + 0z_6 &\to \min, \\ 13z_1 + 7z_2 + 6z_3 + 5z_4 + 3z_5 + 10z_6 &> 22, \\ z_1, z_2, z_3, z_4, z_5, z_6 &\in \{0,1\}. \end{aligned}$$

Its optimal solution is $z^* = (1,0,0,0,0,1)$, and the optimal objective value is $\xi^* = 1$. Therefore, by Theorem 5.1, the point $\tilde{x}$ satisfies all cover inequalities valid for K^2. In particular, $\tilde{x}$ satisfies the cover inequality $x_1 + x_2 \le 1$ written for the cover $C = \{1,6\}$ determined by z^*. Moreover, one can easily verify that, regardless of the choice of the partition (C_1, C_2) and regardless of the ordering of the set $N \setminus C_1$, all coefficients α_j calculated by the lifting procedure are zeros. As a result, we have the inequality $x_1 + x_6 \le 1$, which is not violated at $\tilde{x}$.

Nevertheless, there is still a lifted cover inequality that separates $\tilde{x}$ from K^2. We will find such an inequality later in the continuation of this example. □

Example 5.2 shows that if we are going to solve the separation problem for the class of lifted cover inequalities, the choice of the most violated cover inequality as the starting one is not always justified. This example motivates the use of the following heuristic procedure for finding an initial knapsack cover C.

For $\delta \in \{0,1\}$, we define $N^\delta \overset{\text{def}}{=} \{j \in N : \ \tilde{x}_j = \delta\}$, and then compute $\bar{b} = b - \sum_{j \in N^1} a_j$. Next, sorting the values $\tilde{x}_j$ for $j \in N^2 = N \setminus (N^0 \cup N^1)$ in non-increasing order,

$$\tilde{x}_{j_1} \ge \tilde{x}_{j_2} \ge \dots \ge \tilde{x}_{j_k},$$

we find an index r such that

$$\sum_{i=1}^{r-1} a_{j_i} \le \bar{b} \quad \text{and} \quad \sum_{i=1}^{r} a_{j_i} > \bar{b}.$$

Then we set $C_1 = \{j_1,\dots,j_r\}$, $C_2 = N^1$ and $C = C_1 \cup C_2$. To lift the inequality $\sum_{j\in C_1} x_j \le |C_1|-1$, let us order the elements of the set $N\setminus C_1$ in such a way that the elements $j \in N^2$ are written first and they are listed in non-increasing order of values $\tilde{x}_j$, then the elements from N^1 must follow, and the elements from N^0 are written at the very end.

Example 5.3 (continuation of Example 5.2) *We need to apply the above procedure for separating the point $\tilde{x}$ from the set K^2.*

Solution. First, we write the partition of $N = \{1,2,3,4,5,6\}$: $N^0 = \{1\}$, $N^1 = \{6\}$ and $N^2 = N\setminus(N^0\cup N^1) = \{2,3,4,5\}$. Next we sort the components $\tilde{x}_j$ for $j\in N^2$:

$$\tilde{x}_5 = 0.7 > \tilde{x}_3 = 0.5 = \tilde{x}_4 = 0.5 > \tilde{x}_2 = 0.3.$$

Since $\bar{b} = b - a_6 = 22-10 = 12$, $a_5+a_3 = 3+6 < 12$ and $a_5+a_3+a_4 = 3+6+5 > 12$, then $r=3$ and $C_1 = \{3,4,5\}$, $C_2 = N^1 = \{6\}$, $C = C_1\cup C_2 = \{3,4,5,6\}$. Listing the elements of $N\setminus C_1$ in the order of $(2,6,1)$, we will lift the inequality

$$x_3 + x_4 + x_5 \le 2.$$

Note that this inequality is not valid for K^2. It is valid only for the set $\{x\in K^2 : x_6 = 1\}$. Initially, we set $\beta = b - a_6 = 22-10 = 12$, $\gamma = 2$.

To compute α_2, we solve the following 0,1-knapsack problem:

$$\begin{gathered} x_3+x_4+x_5 \to \max,\\ 6x_3+5x_4+3x_5 \le 12-7 = 5,\\ x_3,x_4,x_5 \in \{0,1\}. \end{gathered}$$

An optimal solution to this problem is given by $x_3 = x_5 = 0$, $x_4 = 1$, and the optimal objective value is $\xi_1 = 1$. Therefore, $\alpha_2 = \gamma - \xi_1 = 2-1 = 1$.

To compute α_6, we solve the next 0,1-knapsack problem:

$$\begin{gathered} \xi_2 = x_2+x_3+x_4+x_5 \to \max,\\ 7x_2+6x_3+5x_4+3x_5 \le 12+10 = 22,\\ x_2,x_3,x_4,x_5 \in \{0,1\}. \end{gathered}$$

Its optimal solution is given by $x_2 = x_3 = x_4 = x_5 = 1$, and the optimal objective value is $\xi_2 = 4$. Therefore, $\alpha_2 = \xi_2 - \gamma = 4-2 = 2$. We also set $\beta := \beta + a_6 = 12+10 = 22$, $\gamma = \xi_2 = 4$.

Now we compute the last coefficient α_1 solving the following 0,1-knapsack problem:

$$\begin{gathered} \xi_3 = x_2+x_3+x_4+x_5+2x_6 \to \max,\\ 7x_2+6x_3+5x_4+3x_5+10x_6 \le 22-13 = 9,\\ x_2,x_3,x_4,x_5,x_6 \in \{0,1\}. \end{gathered}$$

An optimal solution to this problem is determined by $x_2 = x_3 = x_6 = 0$, $x_4 = x_5 = 1$, and the optimal objective value is $\xi_3 = 2$. Therefore, $\alpha_1 = \gamma - \xi_3 = 4 - 2 = 2$.

Now all three coefficients are calculated, and we can write down the resulting lifted cover inequality

$$2x_1 + x_2 + x_3 + x_4 + x_5 + 2x_6 \le 4,$$

which separates $\tilde{x}$ from K^2 since it is violated at $\tilde{x}$ by 0.1. □

5.2.2 Lifting Feasible Set Inequalities

The lifting technique presented in Theorem 5.2 can be used to strengthen any inequalities involving only binary variables. Let us again consider the knapsack set K defined by (5.1). If a set $S \subset N$ is not a knapsack cover (it is called a *feasible* set), then $\sum_{j\in S} a_j \le b$, and, for any vector $w \in \mathbb{Z}^S_{++}$, the inequality

$$\sum_{j\in S} w_j x_j \le \sum_{j\in S} w_j$$

is valid for K. If we lift this trivial inequality, sometimes we can get a much stronger inequality. Let us demonstrate this on an example.

Example 5.4 *Consider the knapsack set*

$$K^3 = \left\{ x \in \{0,1\}^6 :\ 3x_1 + 4x_2 + 6x_3 + 7x_4 + 9x_5 + 18x_6 \le 21 \right\}.$$

Choosing $S = \{1,2,3,4\}$ *and* $w_1 = w_2 = w_3 = w_4 = 1$, *we write the inequality*

$$x_1 + x_2 + x_3 + x_4 \le 4,$$

which, clearly, is valid for K^3. *We need to strengthen this inequality lifting the variables in the following order:* $j_1 = 5$, $j_2 = 6$.

Solution. We have $\gamma = 4$ and $\beta = 21$. To compute α_5, we solve the following 0,1-knapsack problem:

$$\begin{gathered} x_1 + x_2 + x_3 + x_4 \to \max, \\ 3x_1 + 4x_2 + 6x_3 + 7x_4 \le 21 - 9 = 12, \\ x_1, x_2, x_3, x_4 \in \{0,1\}. \end{gathered}$$

Its optimal solution is given by $x_1 = x_2 = 1$, $x_3 = x_4 = 0$, and the optimal objective value is $\xi_1 = 2$. Therefore, $\alpha_5 = \gamma - \xi_1 = 4 - 2 = 2$.

Next, we calculate α_6 solving the problem

$$\begin{aligned} x_1+x_2+x_3+x_4+2x_5 &\to \max,\\ 3x_1+4x_2+6x_3+7x_4+9x_5 &\le 21-18=3,\\ x_1,x_2,x_3,x_4,x_5 &\in \{0,1\}. \end{aligned}$$

An optimal solution to this problem is determined by $x_1=1, x_2=x_3=x_4=x_5=0$, and the optimal objective value is $\xi_2=1$. Hence, $\alpha_6=\gamma-\xi_2=4-1=3$ and the resulting inequality

$$x_1+x_2+x_3+x_4+2x_5+3x_6 \le 4$$

is valid for K^3. □

5.3 Mixed Knapsack Sets

Here, on a simple example, we demonstrate a typical application of the mixed integer rounding (see Sect. 4.3) to generate cuts for simple mixed integer sets. Then we discuss an alternative way of lifting inequalities, when the values of missing coefficients do not depend on the order of their calculation.

Consider the set

$$X^{BK+1}=\left\{(x,y)\in\{0,1\}^n\times\mathbb{R}:\ \sum_{j=1}^n a_jx_j-y\le b\right\},$$

where $a_j\ge 0$ for $j=1,\dots,n$. A subset $C\subseteq\{1,\dots,n\}$ is a (*mixed knapsack*) *cover* for X^{BK+1}, if

1) $\lambda=\lambda(C)\overset{\text{def}}{=}\sum_{j\in C}a_j-b>0$ and
2) $a_k>\lambda$ for $k\in\arg\max_{j\in C}a_j$.

Theorem 5.3. *The following inequality*

$$\sum_{j\in C}\min\{a_j,\lambda\}x_j-y\le\sum_{j\in C}\min\{a_j,\lambda\}-\lambda \tag{5.10}$$

is valid for X^{BK+1}.

Proof. As all a_j are non-negative, then X^{BK+1} is a subset of of the set

$$\tilde{X}^{BK+1}=\left\{(x,y)\in\{0,1\}^n\times\mathbb{R}:\ \sum_{j\in C}a_jx_j-y\le b\right\},$$

and, if (5.10) is valid for $\tilde{X}^{BK+1}$, then it is valid also for X^{BK+1}.

After the substitution $\bar{x}_j$ for $1-x_j$, the inequality

$$\sum_{j\in C}a_jx_j-y\le b$$

takes the form

$$y + \sum_{j \in C} a_j \bar{x}_j \geq \lambda.$$

Dividing this inequality by a_k, we obtain the inequality

$$\sum_{j \in C} -\frac{a_j}{a_k} \bar{x}_j - \frac{1}{a_k} y \leq -\frac{\lambda}{a_k}, \tag{5.11}$$

to which we apply Theorem 4.3. Taking into account that $a_k > \lambda$ and $a_k \geq a_j$ for all $j \in C$, we calculate

$$\begin{aligned} f &= -\frac{\lambda}{a_k} - (-1) = \frac{a_k - \lambda}{a_k}, \\ f_j &= -\frac{a_j}{a_k} - (-1) = \frac{a_k - a_j}{a_k}, \\ \frac{f_j - f}{1 - f} &= \frac{\lambda - a_j}{\lambda} = 1 - \frac{a_j}{\lambda}. \end{aligned}$$

Applying (4.7) to (5.11), we obtain the inequality

$$-\sum_{j \in C} \min\{1, a_j/\lambda\} \bar{x}_j - \frac{1}{\lambda} y \leq -1,$$

which, after multiplication by λ and the inverse substitution $1 - x_j$ for $\bar{x}_j$, transforms into (5.10). □

Example 5.5 *We need to separate the point* $(\tilde{x}, \tilde{y}) = \left(0, 1, 0, 1, 1, \frac{3}{4}, \frac{1}{2}, 0, 0\right)$ *from the mixed integer set*

$$\begin{aligned} X = \{&(x, y) \in \{0,1\}^6 \times \mathbb{R}^3_+ : \\ &5x_1 + x_2 + 3x_3 + 2x_4 + x_5 + 8x_6 - 2y_1 - y_2 + 3y_3 \leq 9\}. \end{aligned}$$

Solution. We should not be confused by the fact that there are more than one real variable here. Since $y_3 \geq 0$, the set X is contained in the set

$$X' = \{(x, y) \in \{0,1\}^6 \times \mathbb{R}^2_+ : \ 5x_1 + x_2 + 3x_3 + 2x_4 + x_5 + 8x_6 - 2y_1 - y_2 \leq 9\},$$

and any inequality valid for X' will be also valid for X. Further, after the substitution s for $2y_1 + y_2$, the set X' transforms into the set

$$\bar{X} = \{(x, s) \in \{0,1\}^6 \times \mathbb{R}_+ : \ 5x_1 + x_2 + 3x_3 + 2x_4 + x_5 + 8x_6 - s \leq 9\}.$$

Let us start with the cover $C = \{2, 4, 5, 6\}$ that is produced by the **LP** heuristic applying to $\tilde{x}$ and the knapsack set obtained from $\bar{X}$ by dropping the variable s. Since $\lambda = 1 + 2 + 1 + 8 - 9 = 3$, by Theorem 5.3, the inequality

$$x_2 + 2x_4 + x_5 + 3x_6 - s \le 1 + 2 + 1 + 3 - 3 = 4$$

is valid for $\bar{X}$. Therefore, the inequality

$$x_2 + 2x_4 + x_5 + 3x_6 - 2y_1 - y_2 \le 4$$

is valid for X, but at $(\tilde{x}, \tilde{y})$ it is violated by $\frac{1}{4}$. □

5.3.1 Sequence Independent Lifting

We can strengthen (5.10) by calculating the coefficients of the variables x_j for $j \notin C$ using a lifting technique, which differs from that discussed in Sect. 5.2. Define the *lifting function*

$$\phi_C(u) = \min \left\{ y + \sum_{j \in C} \min\{a_j, \lambda\}(1 - x_j) - \lambda \; : \right.$$
$$\left. \sum_{j \in C} a_j x_j - y \le b - u, \;\; y \in \mathbb{R}, \;\; x_j \in \{0,1\} \text{ for } j \in C \right\}.$$

Let $C_\lambda \overset{\text{def}}{=} \{j \in C : \, a_j > \lambda\}$ and let $r = |C_\lambda| > 0$. List the elements of C_λ in order of their weights, $a_{\pi_1} \ge a_{\pi_2} \ge \dots a_{\pi_r} > \lambda$, and then set $A_j = \sum_{i=1}^{j} a_{\pi_i}$ for $i = 1, \dots, r$. It is easy to see that

$$\phi_C(u) = \begin{cases} \lambda(j-1) & \text{if } A_{j-1} \le u \le A_j - \lambda, \\ \lambda(j-1) + u - (A_j - \lambda) & \text{if } A_{j-1} - \lambda \le u \le A_j, \\ \lambda(r-1) + u - (A_r - \lambda) & \text{if } u \ge A_r - \lambda, \end{cases}$$

and, therefore, ϕ_C is a *superadditive* function on $\mathbb{R}_+$:

$$\phi_C(u) + \phi_C(v) \le \phi_C(u+v) \quad \text{for all } u, v \in \mathbb{R}.$$

The following theorem establishes the role of superadditivity in the *sequence independent lifting*, i.e, when the order of lifting variables does not affect the lifting coefficients.

Theorem 5.4. *The following inequality*

$$\sum_{j \in C} \min\{a_j, \lambda\} x_j + \sum_{j \in \{1,\dots,n\} \setminus C} \phi_C(a_j) x_j - y \le \sum_{j \in C} \min\{a_j, \lambda\} - \lambda \tag{5.12}$$

is valid for X^{BK+1}.

Proof. It is necessary to prove that any point $(x^*, y^*) \in X^{BK+1}$ satisfies (5.12). Let $Q = \{j \notin C : x_j^* = 1\}$. As ϕ_C is superadditive, we have

$$\begin{aligned}
&\sum_{j\in C} \min\{a_j, \lambda\}(1-x_j^*) - \sum_{j\in\{1,\dots,n\}\setminus C} \phi_C(a_j)x_j^* + y^* - \lambda \\
&= y^* + \sum_{j\in C} \min\{a_j,\lambda\}(1-x_j^*) - \lambda - \sum_{j\in Q} \phi_C(a_j) \\
&\geq \min\left\{ y + \sum_{j\in C} \min\{a_j,\lambda\}(1-x_j) - \lambda \ : \right. \\
&\qquad \left. \sum_{j\in C} a_j x_j - y \leq b - \sum_{j\in Q} a_j,\ y \in \mathbb{R},\ x_j \in \{0,1\} \text{ for } j \in C \right\} - \sum_{j\in Q} \phi_C(a_j) \\
&= \phi_C\left(\sum_{j\in Q} a_j\right) - \sum_{j\in Q} \phi_C(a_j) \geq 0.
\end{aligned}$$

□

5.4 Simple Flow Structures

Let us consider a mixed-integer set

$$X = \{(x,y) \in \mathbb{R}_+^n \times \{0,1\}^n : \sum_{j\in N_1} x_j - \sum_{j\in N_2} x_j \leq b,\ x_j \leq u_j y_j \text{ for } j = 1,\dots,n\},$$

where $b \in \mathbb{R}$, $u \in \mathbb{R}_+^n$, $N_1 \cup N_2 = \{1,\dots,n\}$ and $N_1 \cap N_2 = \emptyset$. Note, that IP (1.23), which is a formulation of the FCNF problem, for each of $|V|$ balance equations, includes two such sets:

$$\begin{aligned}
X_v^+ = \{(x,y) \in \mathbb{R}_+^{E(v)} \times \{0,1\}^{E(v)} : &\sum_{e\in E(V,v)} x_e - \sum_{j\in E(v,V)} x_e \leq d_v, \\
&x_e \leq u_e y_e,\ e \in E(v)\}, \\
X_v^- = \{(x,y) \in \mathbb{R}_+^{E(v)} \times \{0,1\}^{E(v)} : &\sum_{e\in E(v,V)} x_e - \sum_{j\in E(V,v)} x_e \leq -d_v, \\
&x_e \leq u_e y_e,\ e \in E(v)\}.
\end{aligned}$$

Here $E(v) \overset{\text{def}}{=} E(v,V) \cup E(V,v)$.

A pair (C_1, C_2), where $C_1 \subseteq N_1$ and $C_2 \subseteq N_2$, is called a *generalized cover* for the set X if

$$\sum_{j\in C_1} u_j - \sum_{j\in C_2} u_j = b + \lambda(C_1, C_2),$$

where $\lambda(C_1, C_2) > 0$ is the *excess* of the generalized cover.

Theorem 5.5. *Let (C_1, C_2) be a generalized cover, $\lambda = \lambda(C_1, C_2)$ and $L_2 \subseteq N_2 \setminus C_2$. Then the* flow cover inequality

$$\begin{aligned}\sum_{j\in C_1} x_j - \sum_{j\in N_2\setminus(C_2\cup L_2)} x_j + \\ \sum_{j\in C_1} \max\{0, u_j-\lambda\}\cdot(1-y_j) - \lambda\sum_{j\in L_2} y_j \le b + \sum_{j\in C_2} u_j\end{aligned} \tag{5.13}$$

is valid for X.

Proof. We have to prove that any point $(x,y)\in X$ satisfies (5.13). Let $C_1^+ = \{j \in C_1 : u_j > \lambda\}$ and $T = \{j \in N_1 \cup N_2 : y_j = 1\}$. Consider two cases.

1. $|C_1^+ \setminus T| + |L_2 \cap T| = 0.$

$$\begin{aligned}
&\sum_{j\in C_1} x_j + \sum_{j\in C_1}\max\{0,u_j-\lambda\}\cdot(1-y_j)\\
&= \sum_{j\in C_1\cap T} x_j + \sum_{j\in C_1^+\setminus T}(u_j-\lambda)\\
&= \sum_{j\in C_1\cap T} x_j \qquad (\text{since } C_1^+\setminus T=\emptyset)\\
&\le \sum_{j\in N_1} x_j \qquad (\text{since } x_j\ge 0)\\
&\le b+\sum_{j\in N_2} x_j \qquad (\text{by definition of } X)\\
&= b+\sum_{j\in C_2} x_j + \sum_{j\in L_2\cap T} x_j + \sum_{j\in N_2\setminus(C_2\cup L_2)} x_j\\
&\le b+\sum_{j\in C_2} u_j + 0 + \sum_{j\in N_2\setminus(C_2\cup L_2)} x_j \qquad (\text{since } L_2\cap T=\emptyset)\\
&\le b+\sum_{j\in C_2} u_j + \lambda\sum_{j\in L_2} y_j + \sum_{j\in N_2\setminus(C_2\cup L_2)} x_j\,.
\end{aligned}$$

2. $|C_1^+ \setminus T| + |L_2 \cap T| \ge 1.$

$$\begin{aligned}
&\sum_{j\in C_1} x_j + \sum_{j\in C_1}\max\{0,u_j-\lambda\}\cdot(1-y_j)\\
&= \sum_{j\in C_1\cap T} x_j + \sum_{j\in C_1^+\setminus T}(u_j-\lambda)\\
&\le \sum_{j\in C_1} u_j - |C_1^+\setminus T|\cdot\lambda \qquad (\text{since } x_j\le u_j)\\
&\le \sum_{j\in C_1} u_j - \lambda + \lambda\cdot|L_2\cap T| \quad (\text{since } -|C_1^+\setminus T| \le -1+|L_2\cap T|)\\
&= b+\sum_{j\in C_2} u_j + \lambda\sum_{j\in L_2} y_j \le b+\sum_{j\in C_2} u_j + \lambda\sum_{j\in L_2} y_j + \sum_{j\in N_2\setminus(C_2\cup L_2)} x_j\,. \qquad \square
\end{aligned}$$

Inequality (5.13) cuts off from $\operatorname{conv}(X)$ a number of vertices of the relaxation polyhedron

$$\{(x,y) \in \mathbb{R}^n_+ \times [0,1]^n : \sum_{j\in N_1} x_j - \sum_{j\in N_2} x_j \le b, \quad x_j \le u_j y_j \text{ for } j = 1,\dots,n\}.$$

These vertices, $(\bar{x},\bar{y})$, are built as follows:

- $\bar{x}_k = u_k - \lambda$, $\bar{y}_k = (u_k - \lambda)/u_k$ for some $k \in C_1$ with $\lambda < u_k$, or $\bar{x}_k = \lambda$, $\bar{y}_k = \lambda/u_k$ for some $k \in L_2$ with $\lambda < u_k$;
- $\bar{x}_j = u_j$, $\bar{y}_j = 1$ for $j \in (C_1 \cup C_2) \setminus \{k\}$;
- $\bar{x}_j = 0$, $\bar{y}_j = 0 \vee 1$ for $j \in (N_1 \cup N_2) \setminus (C_1 \cup C_2 \cup \{k\})$.

5.4.1 Separation for Flow Cover Inequalities

If we assume that $L_2 = N_2 \setminus C_2$, $x_j = u_j y_j$ and $u_j > \lambda$ for all $j \in C_1$, then (5.13) takes the form

$$\sum_{j\in C_1} u_j y_j + \sum_{j\in C_1} (u_j - \lambda)(1 - y_j) \le b + \sum_{j\in C_2} u_j + \lambda \sum_{j\in N_2\setminus C_2} y_j\,,$$

or, after simplifications,

$$\sum_{j\in C_1} u_j - \lambda \sum_{j\in C_1} (1 - y_j) \le b + \sum_{j\in C_2} u_j + \lambda \sum_{j\in N_2\setminus C_2} y_j\,.$$

Substituting λ for $b - \sum_{j\in C_1} u_j + \sum_{j\in C_2} u_j$ and then dividing the result by λ, we have

$$\sum_{j\in C_1} (1 - y_j) + \sum_{j\in N_2\setminus C_2} y_j \ge 1$$

or

$$\sum_{j\in C_1} (1 - y_j) - \sum_{j\in C_2} y_j \ge 1 - \sum_{j\in N_2} y_j\,.$$

The last inequality is nothing else as a cover inequality for the knapsack set

$$\left\{ y \in \{0,1\}^n : \sum_{j\in N_1} u_j y_j - \sum_{j\in N_2} u_j y_j \le b \right\},$$

that was written for the cover $C = C_1 \cup C_2$ subject to the condition

$$\sum_{j\in C_1} u_j - \sum_{j\in C_2} u_j = b + \lambda \quad \text{and} \quad \lambda > 0.$$

Based on the above reasoning, we can write down the following heuristic that separates a given point $(\tilde{x},\tilde{y})$ from the set X by a flow cover inequality.

1. Solve the following 0,1-knapsack problem:

$$\begin{aligned}\sum_{j\in N_1}(1-\tilde{y}_j)z_j - \sum_{j\in N_2}\tilde{y}_j z_j &\to \min,\\ \sum_{j\in N_1}u_j z_j - \sum_{j\in N_2}u_j z_j &> b,\\ z_j \in \{0,1\}, \quad j &\in N_1\cup N_2\,.\end{aligned}$$

Let z^* be an optimal solution to this problem, and let $C = \{j \in N_1\cup N_2 \,:\, z^*_j = 1\}$.

2. If, for $C_1 = N_1\cap C$, $C_2 = N_2\cap C$ and $L_2 = \{j\in N_2\setminus C_2 \,:\, \lambda\tilde{y}_j < \tilde{x}_j\}$, Ineq. (5.13) is violated at $(\tilde{x},\tilde{y})$, then it is a required cut.

Example 5.6 *We need to separate the point* (x^*,y^*) *with*

$$\begin{aligned}x^* &= (3,0,2,1,0,0),\\ y^* &= (y^*_2,y^*_3,y^*_4,y^*_5,y^*_6) = (0,2/3,1,0,0)\end{aligned}$$

from the set

$$\begin{aligned}X = \{&(x,y)\in\mathbb{R}^6_+\times\{0,1\}^{\{2,3,4,5,6\}} :\\ &x_1+x_2+2x_3-3x_4-2x_5-x_6\le 4,\\ &x_1\le 3,\ x_2\le 3y_2,\ x_3\le 3y_3,\ x_4\le y_4,\ x_5\le 2y_5,\ x_6\le y_6\}.\end{aligned}$$

Solution. Introducing a new binary variable y_1 and changing the variables

$$\bar{x}_1 = x_1,\ \bar{x}_2 = x_2,\ \bar{x}_3 = 2x_3,\ \bar{x}_4 = 3x_4,\ \bar{x}_5 = 2x_5,\ \bar{x}_6 = x_6,$$

we map the set X onto the intersection of the set

$$\begin{aligned}\bar{X} =\{&(\bar{x},y)\in\mathbb{R}^6_+\times\{0,1\}^6 :\\ &\bar{x}_1+\bar{x}_2+\bar{x}_3-\bar{x}_4-\bar{x}_5-\bar{x}_6\le 4,\\ &\bar{x}_1\le 3y_1,\ \bar{x}_2\le 3y_2,\ \bar{x}_3\le 6y_3,\ \bar{x}_4\le 3y_4,\ \bar{x}_5\le 4y_5,\ \bar{x}_6\le y_6\}\end{aligned}$$

and the hyperplane given by the equation $y_1 = 1$. In addition, the point (x^*,y^*) is mapped into the point (x^0,y^0) with

$$x^0 = (3,0,4,3,0,0), \quad y^0 = (1,0,2/3,1,0,0).$$

Next, we try to separate (x^0,y^0) from $\bar{X}$. Therefore, we solve the following 0,1-knapsack problem:

$$\begin{aligned}\frac{1}{3}z_3 - z_4 &\to \min,\\ 3z_1+3z_2+6z_3-3z_4-4z_5-z_6 &> 4,\\ z_1,z_2,z_3,z_4,z_5,z_6 &\in \{0,1\}.\end{aligned}\tag{5.14}$$

We should not be confused by the fact that this problem is not quite similar to the standard $0,1$-knapsack problem (1.28). But after the change of variables

$$\bar{z}_1 = 1 - z_1,\ \bar{z}_2 = 1 - z_2,\ \bar{z}_3 = 1 - z_3,$$
$$\bar{z}_4 = z_4,\ \bar{z}_5 = z_5,\ \bar{z}_6 = z_6,$$

(5.14) is rewritten as

$$1 - \bar{z}_2 + \frac{1}{3}(1 - \bar{z}_3) - \bar{z}_4 \to \min,$$
$$3(1 - \bar{z}_1) + 3(1 - \bar{z}_2) + 6(1 - \bar{z}_3) - 3\bar{z}_4 - 4\bar{z}_5 - \bar{z}_6 \geq 5,$$
$$\bar{z}_1, \bar{z}_2, \bar{z}_3, \bar{z}_4, \bar{z}_5, \bar{z}_6 \in \{0, 1\},$$

or, after rearranging,

$$\bar{z}_2 + \frac{1}{3}\bar{z}_3 + \bar{z}_4 \to \max,$$
$$3\bar{z}_1 + 3\bar{z}_2 + 6\bar{z}_3 + 3\bar{z}_4 + 4\bar{z}_5 + \bar{z}_6 \leq 7,$$
$$\bar{z}_1, \bar{z}_2, \bar{z}_3, \bar{z}_4, \bar{z}_5, \bar{z}_6 \in \{0, 1\}.$$

The point $\bar{z}^* = (0,1,0,1,0,0)$ is an optimal solution to this program. Returning to the original variables, we obtain an optimal solution $z^* = (1,0,1,1,0,0)$ to (5.14).

The point z^* defines the generalized cover $C_1 = \{1,3\}$, $C_2 = \{4\}$ with the excess $\lambda = 2$. Since $L_2 = \{j \in \{5,6\} :\ \lambda y_j^0 < x_j^0\} = \emptyset$, we get the inequality

$$\bar{x}_1 + \bar{x}_3 - \bar{x}_5 - \bar{x}_6 + 1 \cdot (1 - y_1) + 4 \cdot (1 - y_3) \leq 4 + 3.$$

Returning to the original variables x_j, and setting $y_1 = 1$, we obtain the inequality

$$x_1 + 2x_3 - 2x_5 - x_6 - 4y_3 \leq 3,$$

which is valid for X, but is violated at (x^*, y^*) by $\frac{4}{3}$. □

5.5 Generalized Upper Bounds

A *generalized upper bound* (*GUB*) is an inequality $\sum_{j \in C} x_j \leq 1$ that involves only binary variables x_j. In practice, such restrictions are very common. For example, let us recall the set-packing problem (2.1) and the representation of discrete variables (1.2).

Let A be a 0,1-matrix of size $m \times n$, and $\mathbf{e}$ be a vector of m ones. The convex hull of the binary set $\{x \in \{0,1\}^n : Ax \leq \mathbf{e}\}$ is called a *packing polytope*. The intersection graph of the 0,1-matrix A, which is denoted by $G(A)$, has $N \stackrel{\text{def}}{=} \{1, \dots, n\}$ as the set of vertices, and a pair of vertices, (i, j), is an edge in $G(A)$ if the columns A^i and A^j have one in the same row. Using $G(A)$, one can derive a number of classes of inequalities that are valid for the packing polytope.

5.5.1 Clique Inequalities

Let $C \subseteq N$ be the set of vertices of some *clique* in $G(A)$ (any two vertices from C are connected by an edge in $G(A)$). Then the *clique inequality*

$$\sum_{j \in C} x_j \le 1$$

is valid for the packing polytope. Note that in this way we can obtain new inequalities not present in the original system. For example, for the system

$$x_1 + x_2 \le 1,\ x_2 + x_3 \le 1,\ x_3 + x_1 \le 1,$$

the set $\{1,2,3\}$ is a clique in $G(A)$, and the click inequality $x_1 + x_2 + x_3 \le 1$ does not belong to the system.

The separation problem for the class of clique inequalities is formulated as follows: given a point $\tilde{x} \in [0,1]^n$, find in $G(A)$ a clique C^* of maximum weight $w = \sum_{j \in C^*} \tilde{x}_j$. If $w > 1$, the clique inequality $\sum_{j \in C^*} x_j \le 1$ is violated at $\tilde{x}$; otherwise, $\tilde{x}$ satisfies all the clique inequalities.

It is known that this separation problem for the clique inequalities is **NP**-hard. Surprisingly, there is a wider class of inequalities that includes all clique inequalities and for which there is a polynomial separation algorithm. Unfortunately, this polynomial algorithm is too time consuming to be used in practice. Therefore, in practice, the separation problem for the clique inequalities is solved with the help of several heuristic procedures. One of them was specially developed for the case when $\tilde{x}$ is a solution (or only a part of solution) to the relaxation LP of a MIP containing a system of binary inequalities, $Ax \le \mathbf{e}$, among its constraints.

Suppose that one of the inequalities from the system $Ax \le \mathbf{e}$ is fulfilled as an equality. Let this be an inequality i_0 and let $C = \{j :\ a_{i_0,j} = 1,\ \tilde{x}_j > 0\}$. First, we sort the components $\tilde{x}_j$ for $j \in N \setminus C$ in non-increasing order. Let $j_1, \dots, j_k$ be a required order. Then for $i = 1, \dots, k$, if $C \cup \{j_i\}$ is a clique in $G(A)$, add j_i to C. If we can thus add at least one index j_i with $x_{j_i} > 0$, then as a result we get a clique inequality $\sum_{j \in C} x_j \le 1$ that is violated at $\tilde{x}$.

5.5.2 Odd Cycle Inequalities

A list $(j_1, \dots, j_k, j_{k+1} = j_1)$ of vertices is a cycle of length k in $G(A)$ if any of its two neighboring vertices are *adjacent* (connected by an edge). The vertex set, $C = \{j_1, \dots, j_k\}$, of an odd cycle (k is odd) in $G(A)$ induces the *odd cycle inequality*

$$\sum_{j \in C} x_j \le \frac{|C| - 1}{2}, \tag{5.15}$$

which is valid for the packing polytope. In fact, (5.15) is a Chvátal-Gomory cut since we can derive it by adding together the inequalities

$$x_{j_1} + x_{j_k} \le 1 \quad \text{and} \quad x_{j_i} + x_{j_{i+1}} \le 1,\ i = 1,\dots,k-1,$$

dividing the result by 2, and rounding down the right-hand side.

For the class of odd hole inequalities, there exists a polynomial separation procedure. We define a bipartite digraph H with the vertex set $N \cup N'$, where $N' = \{n+1,\dots,2n\}$, and, for each edge (i,j) from $G(A)$, H has two arcs, $(i,n+j)$ and $(n+i,j)$ of weight $1-2\tilde{x}_i$, and two arcs, $(j,n+i)$ and $(n+j,i)$ of weight $1-2\tilde{x}_j$. It is not hard to see that a shortest path in H from a vertex i to the vertex $n+i$ (if such a path exists) corresponds to the shortest odd cycle in $G(A)$ passing through i. We denote by C^* the set of vertices of this cycle. If $\sum_{j\in C^*}(1-2\tilde{x}_j)$ is less than 1, then $\sum_{j\in C^*}\tilde{x}_j > \frac{|C^*|-1}{2}$, and the odd cycle inequality $\sum_{j\in C^*} x_j \le \frac{|C^*|-1}{2}$ is violated at $\tilde{x}$.

In conclusion, we note that, since H has no cycles of negative weight, then a shortest path between any pair of its vertices can be found in polynomial time.

5.5.3 Conflict Graphs

A conflict graph represents logical dependencies between binary variables of a MIP. Let us consider a solution set $X = P(A,b;S)$ of a MIP involving binary variables. The *conflict graph* G_X for the set X contains two vertices, j^0 and j^1, for each binary variable x_j. For $\alpha,\beta \in \{0,1\}$, the edge (i^α, j^β) belongs to the conflict graph if the set $\{x \in X:\ x_i = \alpha,\ x_j = \beta\}$ is empty.

For a given MIP, the conflict graph subsumes the intersection graph, i.e., any edge of the intersection graph is also an edge of the conflict graph. The conflict graphs can be used for generating cuts in the same way as the intersection graphs.

If a vertex set C induces a clique in G_X, then the inequality

$$\sum_{j^1\in C} x_j + \sum_{j^0\in C} (1-x_j) \le 1$$

is valid for X. For example, for the set X of the solutions to the following system

$$\begin{aligned} x_1 + 2x_2 - \ \ x_3 &\le 1,\\ 3x_1 + \ \ x_2 - 2x_3 &\le 2,\\ x_1, x_2, x_3 &\in \{0,1\}, \end{aligned}$$

the conflict graph, G_X, has three edges $(1^1,2^1)$, $(2^1,3^0)$ and $(1^1,3^0)$. Therefore, the set of vertices, $C = \{1^1,2^1,3^0\}$, is a clique in G_X, and the inequality

$$x_1 + x_2 + (1-x_3) \le 1 \quad \text{or} \quad x_1 + x_2 - x_3 \le 0$$

is valid for X.

Similarly, if C is the vertex set of an odd cycle in G_X, then the inequality

$$\sum_{j^1 \in C} x_j + \sum_{j^0 \in C} (1 - x_j) \le \frac{|C| - 1}{2}$$

is valid for X.

5.6 Notes

The idea of obtaining strong inequalities exploiting the structure of an **NP**-hard problem being solved has its roots in [44, 43].

***Sect.* 5.1.** The cover inequalities, as is often the case, independently discovered several authors [10, 72, 104, 138].

***Sect.* 5.2.** The concept of lifting inequalities was already present in [61], later it was extended in [103, 139]. The technique of sequential lifting was proposed in [104] and has since become widespread and is now implemented in many commercial **MIP** libraries (see [67]). The idea of lifting inequalities for feasible sets was proposed in [136].

***Sect.* 5.3.** The role of superadditivity in the sequence-independent lifting was investigated in [69, 140]. Theorem 5.4 was proved in [89].

***Sect.* 5.4.** The flow cover inequalities were introduced in [108]. The lifting procedure for these inequalities was elaborated in [68].

***Sect.* 5.5.** The packing polytope was actively studied in the 1970s. For a discussion of the results of these studies and bibliographic references, see, for example, [98]. The clique inequalities appeared in [52, 103]. The separation problem for the clique inequalities is **NP**-hard (see Theorem 9.2.9 in [64]). Therefore, it is surprising that there is a wider class of inequalities that includes all clique inequalities, and the separation problem for which is polynomially solvable (see [64, 87]).

The inequalities for odd cycles were introduced in [103]. The separation algorithm for these inequalities is based on the algorithm from Lemma 9.1.11 in [64].

The use of the conflict graphs is discussed in [8].

***Sect.* 5.7.** The statements of Exercises 5.4 and 5.5 were taken, respectively, from [108] and [9].

5.7 Exercises

5.1. Write down an inequality that cuts off just one given point $a \in \{0,1\}^n$ from the 0,1-cube $\{0,1\}^n$.

5.2. Find a lifted cover inequality that separates a given point $\tilde{x}$ from a given set X:

a) $\tilde{x} = \left(0,0,\frac{3}{4},\frac{3}{4},1\right)^T$, $X = \{x \in \{0,1\}^5 : 8x_1 + 7x_2 + 6x_3 + 6x_4 + 5x_5 \le 14\}$;
b) $\tilde{x} = \left(0,\frac{6}{8},\frac{3}{4},\frac{3}{4},0\right)^T$, $X = \{x \in \{0,1\}^5 : 12x_1 + 8x_2 + 6x_3 + 6x_4 + 7x_5 \le 15\}$;
c) $\tilde{x} = \left(0,0,\frac{1}{2},\frac{1}{6},1\right)^T$, $X = \{x \in \{0,1\}^5 : 10x_1 - 9x_2 + 8x_3 + 6x_4 - 3x_5 \le 2\}$.

5.3. For given set X and point $(\tilde{x},\tilde{y})$, find a flow cover inequality that separates $(\tilde{x},\tilde{y})$ from X:

a) $$X = \{(x,y) \in \mathbb{R}^3_+ \times \{0,1\}^3 : x_1 + x_2 + x_3 = 7,\ x_1 \le 3y_1,\ x_2 \le 5y_2,\ x_3 \le 6y_3\},$$
$$(\tilde{x},\tilde{y}) = \left(2,5,0;\frac{2}{3},1,0\right)^T;$$

b) $$X = \{(x,y) \in \mathbb{R}^6_+ \times \{0,1\}^6 : 2x_1 + x_2 + x_3 - x_4 - 2x_5 - x_6 = 4,$$
$$x_1 \le \frac{3}{2}y_1,\ x_2 \le 3y_2, x_3 \le 6y_3,\ x_4 \le 3y_4,\ x_5 \le \frac{5}{2}y_5,\ x_6 \le y_6\},$$
$$(\tilde{x},\tilde{y}) = \left(\frac{3}{2},3,0,0,1,0;1,1,0,0,\frac{2}{5},0\right)^T.$$

Here, in the representations of $(\tilde{x},\tilde{y})$, the $\tilde{x}$ and $\tilde{y}$ parts are separated by semicolons.

5.4. Consider the set
$$X = \left\{(x,y) \in \{0,1\}^n \times \mathbb{R}^n_+ : \sum_{j=1}^n y_j \le b,\ y_j \le a_j x_j \text{ for } j = 1,\dots,n\right\}.$$

Let $C \subseteq \{1,\dots,n\}$ and $\lambda \overset{\text{def}}{=} \sum_{j\in C} a_j - b > 0$. Using Theorem 5.3, prove that the inequality
$$\sum_{j\in C} (y_j + \max\{0, a_j - \lambda\}(1 - x_j)) \le b$$
is valid for X.

5.5. Consider the solution set X of the following system:
$$\begin{aligned} s + \sum_{j\in N_1} x_j - \sum_{j\in N_2} x_j &\ge b,\\ 0 \le x_j \le y_j, \quad j &= 1,\dots,n,\\ s \ge 0,\ y &\in \mathbb{Z}^n_+, \end{aligned} \tag{5.16}$$

where (N_1,N_2) is a partition of the set $N = \{1,\dots,n\}$. Let $f = b - \lfloor b \rfloor$. Prove the following statements:

a) conv(X) is described by the inequalities that determine the relaxation polyhedron for (5.16), and the inequalities
$$s + f\sum_{j\in L_1} y_j + \sum_{j\in R_1} x_j \ge f\lceil b\rceil + \sum_{j\in L_2} (x_j - (1-f)y_j), \tag{5.17}$$

for all partitions (L_1, R_1) of N_1, and all $L_2 \subseteq N_2$;

b) a point $(s, x, y) \in \mathbb{R}_+ \times \mathbb{R}^n_+ \times \mathbb{Z}^n_+$ satisfies (5.17) if and only if the following non-linear inequality holds:

$$s + \sum_{j \in N_1} \min\{f y_j, x_j\} \geq f\lceil b \rceil + \sum_{j \in N_2} \max\{0, x_j - (1-f)y_j\}.$$

5.6. Let us consider (1.7), which is the system of inequalities that describes the truth sets of the CNF given by (1.6). Suppose that $q \in S^1_k \cap S^0_l$. Prove that the following inequality

$$\sum_{j \in (S^1_k \cup S^1_l) \setminus \{q\}} x_j + \sum_{j \in (S^0_k \cup S^0_l) \setminus \{q\}} (1 - x_j) \geq 1$$

is valid for the truth sets of this CNF.

5.7. Prove that the constraint matrix in (1.15), which is a formulation of the transportation problem, is totally unimodular.

5.8. The *parity polytope* is the convex hull of the set of 0, 1-solutions to the comparison

$$\sum_{j=1}^n x_j \equiv 0 \pmod 2.$$

Prove that this polytope coincides with the set of solutions to the following system of linear inequalities:

$$\sum_{j \in S} x_j - \sum_{j \in \{1,\dots,n\} \setminus S} x_j \leq |S| - 1, \quad S \subseteq \{1, \dots, n\},\ |S| \text{ is odd},$$
$$0 \leq x_j \leq 1, \quad j = 1, \dots, n.$$

How to solve the separation problem for this system of inequalities?

5.9. *Minimum distance of a linear code*. A binary $m \times n$ matrix H (*parity check matrix*) is given. Among the binary vectors x satisfying the comparison $Hx \equiv 0 \pmod 2$, we need to find a nonzero vector of minimum weight (with a minimum number of ones). Using Exercise 5.8, formulate this problem as an IP.

5.10. Consider the packing polytope P and let G denote the intersection graph of the constraint matrix defining P (see Sect. 5.5 for the definitions and notations used below). Show that

a) a click inequality is facet defining for P if and only if it is induced by the maximal (by inclusion) click in G;

b) an odd cycle inequality is facet defining for P if and only if it is induced by a cordless odd cycle (also known as *odd hole*), i.e., no two non-neighboring vertices of this cycle are adjacent in G.

Chapter 6
Branch-And-Cut

Currently the main method of solving MIPs is the branch-and-cut method since it is used in all (in all!) known competitive modern **MIP** solvers. Briefly, the branch-and-cut method is a combination of the branch-and-bound and cutting-plane algorithms. In this chapter we present a general schema of the branch-and-cut method, and also discuss its most important components. In the last section of this chapter we demonstrate an application of this method for solving MIPs with exponentially many inequalities. Specifically, we consider a branch-and-cut algorithm for the traveling salesman problem. This algorithm is considered as one of the most impressive successful applications of the branch-and-cut method.

6.1 Branch-And-Bound

We will consider MIP (1.1) with two sided constraints. The basic structure of the *branch-and-bound method* is the *search tree*. The *root* (or *root node*) of the search tree corresponds to the original MIP. The search tree grows through a process called *branching* that creates two or more descendants for one of the leaves of the current search tree. Each of the MIPs in the child nodes is obtained from the parent MIP by adding one or more new constraints that are usually upper or lower bounds for integer variables. It should also be noted that in the process of branching, we should not lose feasible solutions: the union of the feasible domains of the child MIPs must be the feasible domain for their parent MIP.

But if the search tree only grew (via branching), then even for relatively small MIPs the tree size could be huge. On the contrary, one of the main ideas of the branch-and-bound method is to prevent an uncontrolled growth of the search tree This is achieved by cutting off the ”hopeless” branches of the search tree. We evaluate the prospects of the nodes of the search tree by comparing their upper bounds with the current lower bound. In the **LP** based branch-and-bound method, the *upper bound* at any node k is the optimal objective value, $\gamma(k)$, of the relaxation LP at this node. The *lower bound* (or *record*), R, is the largest value of the objective function attained on the already found feasible solutions of the original MIP. The best of these solutions is called a *record solution*. If $\gamma(k) \le R$, then node k and all its descendants are cut off from the search tree.

```
branch-and-bound($c, b^1, b^2, A, d^1, d^2, S; x^R, R$);
{
  Compute $x^0 \in \arg\max\{c^T x : b^1 \le Ax \le b^2,\ d^1 \le x \le d^2\}$;
  if ($x^0_S \in \mathbb{Z}^S$) { // change the record and record solution
    $R = c^T x^0$; $x^R = x^0$; return;
  }
  initialize the list of active nodes with one node $(x^0, d^1, d^2)$;
  while (the list of active nodes is not empty) {
    select a node, $N = (x^0, d^1, d^2)$, from the list of active nodes;
    if ($c^T x^0 \le R$)
      continue;
    select a fractional component $x^0_i$ for $i \in S$;
    compute $x^1 \in \arg\max\{c^T x : b^1 \le Ax \le b^2,\ d^1 \le x \le d^2(i, \lfloor x^0_i \rfloor)\}$;
    if ($c^T x^1 > R$) {
      if ($x^1_S \in \mathbb{Z}^S$) { // change the record and record solution
        $R = c^T x^1$; $x^R = x^1$;
      }
      else
        add the node $(x^1, d^1, d^2(i, \lfloor x^0_i \rfloor))$ to the list of active nodes;
    }
    compute $x^2 \in \arg\max\{c^T x : b^1 \le Ax \le b^2,\ d^1(i, \lceil x^0_i \rceil)) \le x \le d^2\}$;
    if ($c^T x^2 > R$) {
      if ($x^2_S \in \mathbb{Z}^S$) { // change the record and record solution
        $R = c^T x^2$; $x^R = x^2$;
      }
      else
        add the node $(x^2, d^1(i, \lceil x^0_i \rceil), d^2)$ to the list of active nodes;
    }
  }
}
```

Listing 6.1. Branch-and-bound method for solving MIPs

A very general version of the **LP**-based branch-and-bound method for solving MIPs is shown in Listing 6.1. The input of the *branch-and-bound* procedure consists of the parameters describing a MIP, a constraint matrix A, vectors c, b^1, b^2, d^1, d^2 and a set S of integer variables, as well as a feasible solution x^R (initial record solution) and $R = c^T x^R$ (initial record). There are MIPs for which it is difficult to find an initial feasible solution, in such cases R is set to $-\infty$. If $R = -\infty$ when the *branch-and-bound* procedure terminates, then the MIP being solved does not have feasible solutions; otherwise, x^R is an optimal solution to this MIP. In the description of the method we use the following notation

$$d(i,\alpha) \overset{\text{def}}{=} \begin{cases} d_j, & \text{if } j \neq i, \\ \alpha, & \text{if } j = i. \end{cases}$$

It follows from the description of the *branch-and-bound* procedure that each node of the search tree stores not only the upper and lower bounds for the values of variables but also an optimal solution to the relaxation LP. In practice, instead of an optimal solution to the relaxation LP, it is better to store at each node a description of an optimal basis, so that the dual simplex method later will be able to quickly reoptimize the relaxation LPs for the child nodes.

The *branch-and-bound* procedure from Listing 6.1 allows for ambiguities in the selection of a node from the list of active nodes and in the choice of a variable for branching on it if there are several integer variables taking fractional values. There exist a few competitive strategies to make the procedure unambiguous. A simple and at the same time quite efficient method is to select a node with the maximum upper bound. Unfortunately, there is no rule — both simple and efficient for most MIPs — to select a variable for branching. In the literature, the most commonly mentioned rule recommends choosing a variable with the most fractional value (which fractional part is closest to $\frac{1}{2}$). But computational experiments have proved that this rule is no better than the one that recommends choosing a variable for branching randomly. Let us postpone a detailed discussion of the branching rules until Sect. 6.3.

We should not be confused by the fact that the search tree is not mentioned in the *branch-and-bound* procedure. In fact, the procedure "builds" (although implicitly) a search tree, and the leaves of this tree constitute the list of active nodes.

Let us demonstrate the work of the branch-and-bound method on a simple example.

Example 6.1 *We need to solve the following IP:*

$$\begin{array}{rr} & x_1 + 2x_2 \to \max, \\ 1: & -2x_1 + 3x_2 \le 4, \\ 2: & 2x_1 + 2x_2 \le 11, \\ 3: & 1 \le x_1 \le 4, \\ 4: & 1 \le x_2 \le 5, \\ & x_1, x_2 \in \mathbb{Z}. \end{array}$$

Solution. The search tree is shown in Fig. 6.1. The nodes of this tree are numbered from 0 (the root node corresponding to the original MIP) to 5. Each node is represented as a rectangle that, for the relaxation LP at this node, stores the feasible intervals of both variables, as well as an optimal solution and the optimal objective value (upper bound). The relaxation LPs are solved by the dual simplex method starting with an optimal basis for the parent node relaxation LP.

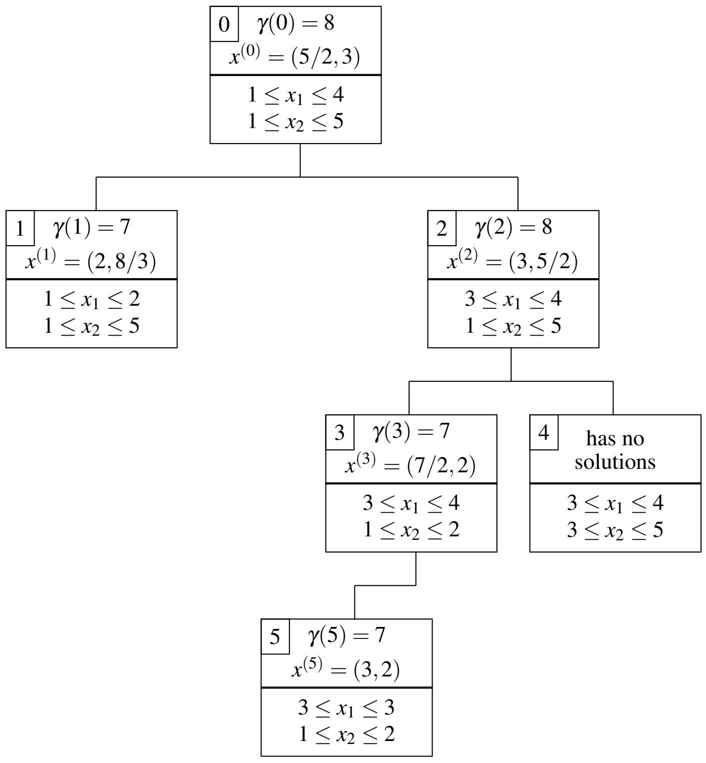

Fig. 6.1 Search tree for IP of Example 6.1

Initially, we set $R = -\infty$. In this example, the objective function takes only integer values for all feasible solutions, therefore, we take as the upper bound, $\gamma(k)$, not the value of $c^T x^{(k)}$ but its integer part $\lfloor c^T x^{(k)} \rfloor$. Here $x^{(k)}$ denotes an optimal solution to the relaxation LP at node k. The iterations performed by the algorithm follow below. They are numbered by the pairs $i.j$, where i is the node index, and j is the iteration index of the dual simplex method.

0.0. $I = (3,4)$, $B^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $x = (4,5)^T$, $\pi = (1,2)^T$.

0.1. $s = 1$, $u = (-2,3)^T$, $\lambda = \frac{2}{3}$, $t = 2$, $I = (3,1)$, $B^{-1} = \begin{bmatrix} 1 & 0 \\ \frac{2}{3} & \frac{1}{3} \end{bmatrix}$, $x = (4,4)^T$, $\pi = (\frac{7}{3}, \frac{2}{3})^T$.

0.2. $s=2$, $u=(\frac{10}{3},\frac{2}{3})^T$, $\lambda=\frac{7}{10}$, $t=1$, $I=(2,1)$, $B^{-1}=\begin{bmatrix}\frac{3}{10} & -\frac{1}{5}\\ \frac{1}{5} & \frac{1}{5}\end{bmatrix}$,
$x=x^{(0)}=(\frac{5}{2},3)^T$, $\pi=(\frac{7}{10},\frac{1}{5})^T$, $\gamma(0)=8$.

Since the solution $x^{(0)}$ to the relaxation LP is not integral, we form the root (node 0) of the search tree, and then we perform branching on the variable x_1 taking a fractional value.

1.1. $s=3$, $u=(\frac{3}{10},-\frac{1}{5})^T$, $\lambda=\frac{7}{3}$, $t=1$, $I=(3,1)$, $B^{-1}=\begin{bmatrix}1 & 0\\ \frac{2}{3} & \frac{1}{3}\end{bmatrix}$,
$x^{(1)}=(2,\frac{8}{3})^T$, $\pi=(\frac{7}{3},\frac{2}{3})^T$, $\gamma(1)=7$.

Since the solution $x^{(1)}$ is not integer, and $\gamma(1)=7>-\infty=R$, we add node 1 to the search tree.

2.1. $s=-3$, $u=(-\frac{3}{10},\frac{1}{5})^T$, $\lambda=1$, $t=2$, $I=(2,-3)$, $B^{-1}=\begin{bmatrix}0 & -1\\ \frac{1}{2} & 1\end{bmatrix}$,
$x^{(2)}=(3,\frac{5}{2})^T$, $\pi=(1,1)^T$, $\gamma(2)=8$

Since the solution $x^{(2)}$ at node 2 is also not integer and $\gamma(2)=8>-\infty=R$, we add this node to the search tree.

Among the active nodes, which are the tree leaves, node 2 has the maximum upper bound. So, we choose this node to branch on the variable x_2.

3.1. $s=4$, $u=(\frac{1}{2},1)^T$, $\lambda=1$, $t=2$, $I=(2,4)$, $B^{-1}=\begin{bmatrix}\frac{1}{2} & -1\\ 0 & 1\end{bmatrix}$,
$x^{(3)}=(\frac{7}{2},2)^T$, $\pi=(\frac{1}{2},1)^T$, $\gamma(3)=7$.

Since the solution $x^{(3)}$ at node 3 is not integer, and $\gamma(3)=7>-\infty=R$, we add this node to the search tree.

4.1. $s=-4$, $u=(-1/2,-1)^T$. Since all components of vector u are non-positive, then the relaxation LP at this node has no feasible solutions. In this case, we do not need to add node 4 to the search tree. Nevertheless, in Fig. 6.1 this node is present there to make the tree more informative.

From two active nodes, 1 and 3, with the maximal upper bound 7, we select node 3, which is farther from the root, and perform branching on the variable x_1.

5.1. $s=3$, $u=(1/2,-1)^T$, $\lambda=1$, $t=1$, $I=(3,4)$, $B^{-1}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$,
$x^{(5)}=(3,2)^T$, $\pi=(1,2)^T$, $\gamma(5)=7$.

Since the solution $x^{(5)}$ is integer and $\gamma(5)=7>-\infty=R$, we change the record and the record solution: $R=7$, $x^R=(3,2)^T$. In Fig. 6.1, node 5 is also presented for informative purposes.

As the upper bounds for the nodes 1 and 3 are equal to the current record, then node 1 and the right branch of node 3, which we have not created yet, must be cut off.

Since there are no more unprocessed nodes in the search tree (the list of active nodes is empty), the current record solution $x^R = (3,2)^T$ is optimal. □

6.2 Branch-And-Cut

The *branch-and-cut method* is a branch-and-bound method in which cuts are generated when solving the relaxation LPs at all (or only some) nodes of the search tree. At first glance it may seem that these changes are insignificant. In practice, this changed the whole philosophy of integer programming. A simplified block diagram of the branch-and-cut method is shown in Fig. 6.2.

Two values that determine the behavior of the branch-and-cut method (as well as the branch-and-bound method) are the lower bound (record) and the upper bounds (optimal objective values for relaxation LPs) at the nodes of the search tree. In the branch-and-bound method, when processing a node of the search tree, the main goal is to solve its relaxation LP as quickly as possible. The branch-and-cut method performs much more work at each node generating cuts in order to minimize its upper bound. In this case, the *duality gap* (the difference between the upper and lower bounds) at the node also decreases.

Unlike the ”pure” cutting plane algorithms, we now do not expect that adding cuts will be sufficient to find an optimal solution. It is also worth noting that earlier, as a rule, only one inequality was generated to cut off the solution of the relaxation LP. Today such a strategy is considered bad, now cuts are added by groups of many inequalities.

In practice, it is very important to determine when to stop generating new cuts and proceed to branching. If many cuts are added at each node, it may take significantly longer to reoptimize the relaxation LPs. A reasonable strategy is to watch how the duality gap is decreasing. If there is no substantial progress for several cut generation rounds, then it is time to stop. In this case, after each cut generation round, it is reasonable to remove from the active (solved at the moment) LP those cuts that are not satisfied ”almost” as equalities. Some of such cuts are permanently removed from the system, and the remaining ones are moved to a special repository called the *cut pool*. Before adding the processed node to the list of active nodes (to the search tree), all cuts present in the constraint system of the active LP but not in the pool are recorded there. Later, when this node is selected for processing, its relaxation LP is restored by extracting all the necessary inequalities (cuts) from the pool.

It may seem that the pool is needed only to restrict the sizes of the node LPs. But if cuts are added not only at the root node of the search tree, then we need the pool for one more important reason. The problem is that not all generated cuts are *global inequalities*, i.e., they are valid for all nodes of the search tree. This happens, in particular, because the cut generating procedures use the bounds imposed on the variables (inequalities of the form $x_j \le (\ge) d$), which are different for different nodes of the search tree. A *local inequality* is valid for a particular node and all

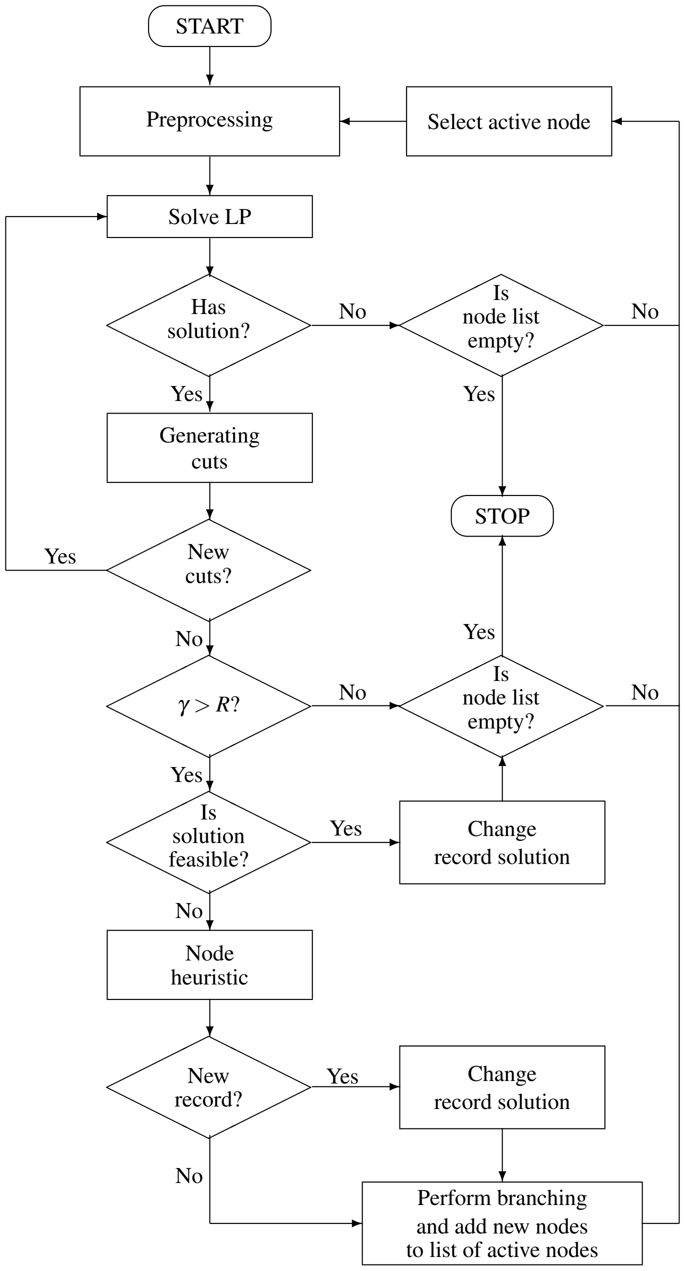

Fig. 6.2 Block diagram of the branch-and-cut method

its descendants; for other nodes it may not be valid. Therefore, such an inequality can not be present in the active constraint matrix permanently, and the pool is the best place to store it.

Another way to reduce the duality gap at a node is to use node heuristics to increase the lower bound (record). The idea of the *node heuristics* is simple. Each time some tree node is processed, we can try to somehow "round off" a solution to its relaxation LP. Usually the rounding consists in performing some type of "diving", when the values of a group of integer variables with "almost integer" values are fixed, the resulting LP is reoptimized, then another group of variables is fixed, and so on until an integer solution is obtained, or it is proved that fixing variables resulted in an inconsistent constraint system. When we are lucky to get a new feasible solution better than the record one, then the lower bound is increased allowing us to eliminate some active nodes of the search tree, and thereby speed up the solution process.

Example 6.2 *We need to solve the following IP:*

$$\begin{aligned} x_1 + 3x_2 + \ \ x_3 + 2x_4 &\to \max,\\ 4x_1 + 7x_2 + 3x_3 + 5x_4 &\le 10,\\ 5x_1 + 4x_2 + 6x_3 + 2x_4 &\le 9,\\ x_1, x_2, x_3, x_4 &\in \{0,1\}. \end{aligned} \tag{6.1}$$

Solution. Let us agree to generate only the cover inequalities, which are always global and, therefore, they are valid for all nodes of the search tree shown in Fig. 6.3.

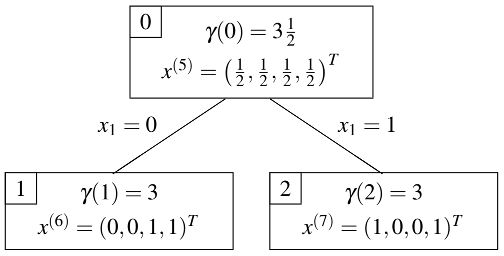
0
$\gamma(0) = 3\frac{1}{2}$
$x^{(5)} = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)^T$
$x_1 = 0$
$x_1 = 1$
1
$\gamma(1) = 3$
$x^{(6)} = (0,0,1,1)^T$
2
$\gamma(2) = 3$
$x^{(7)} = (1,0,0,1)^T$

Fig. 6.3 Search tree for IP (6.1)

0. First, we solve the relaxation LP for (6.1). Its optimal solution is the point $x^{(1)} = \left(0, 1, 0, \frac{3}{5}\right)^T$, which violates the inequality

$$x_2 + x_4 \le 1$$

written for the knapsack cover $C_1^1 = \{2,4\}$ of the first knapsack constraint. Adding this inequality to the constrain system, after reoptimizing, we get the solution $x^{(2)} = \left(\frac{1}{3}, 1, \frac{5}{9}, 0\right)^T$, which violates the inequality

$$x_1 + x_2 \le 1$$

induced by the cover $C_2^1 = \{1,2\}$ of the first knapsack inequality. Adding this inequality and reoptimizing, we get the third solution $x^{(3)} = \left(0,1,\frac{5}{6},0\right)^T$, which violates the inequality

$$x_2 + x_3 \le 1$$

written for the cover $C_1^2 = \{2,3\}$ of the second knapsack constraint. Again, adding this inequality and reoptimizing, we find the solution $x^{(4)} = \left(\frac{5}{9},\frac{4}{9},\frac{5}{9},\frac{5}{9}\right)^T$, which violates the inequality

$$x_1 + x_3 \le 1$$

induced by the cover $C_2^2 = \{1,3\}$ of the second knapsack constraint. Adding this inequality and reoptimizing, we obtain the solution $x^{(5)} = \left(\frac{1}{2},\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)^T$, which satisfies all cover inequalities for both knapsack sets of our IP. So, we turn to branching on the variable x_1.

1. Now we solve the relaxation LP for node 1 ($x_1 = 0$). Its optimal solution, $x^{(6)} = (0,0,1,1)^T$, is integer and, therefore, it is declared as the first record solution, $x^R = (0,0,1,1)^T$, and the record is set to $R = c^T x^R = 3$.

2. An optimal solution $x^{(7)} = (1,0,0,1)^T$ to the relaxation LP at node 2 ($x_1 = 1$) is also integer. The objective value on this solution is equal to 3. Therefore, both points, $x^{(6)}$ and $x^{(7)}$, are optimal solutions to (6.1). □

6.3 Branching

The branching strategy used in a particular implementation of the branch-and-cut method significantly affects the performance of this implementation, especially in cases where MIP formulations are not strong enough, and the cuts are inefficient. A branching strategy includes a rule for selecting the next active node and a branching rule. In practice, there are many different rules for selecting a node for branching. For example, let us mention only a few such rules:

- **maxCOST**: first choose a node with the maximum *upper bound*, which is the optimal objective value of the relaxation LP;
- **DFS**: first choose a node of maximum depth (the depth of a node is the distance from it to the root of the search tree);
- **DFS/maxCOST**: **DFS** before obtaining the first valid solution, then switching to the **maxCOST** strategy;
- for the first k (say $k = 255$) nodes, use the **maxCOST** strategy, then if no feasible solution was found, switch to the **DFS/maxCOST** strategy.

When a node, say node q, has been selected, we need to determine a variable on which to branch. Let $\bar{x}^q$ and $z(q)$ be an optimal solution and the optimal objective value of the relaxation LP at node q, and let $\tilde{z}_i^-(q)$ and $\tilde{z}_i^+(q)$ be the optimal objective values of the relaxation LPs for the left (with the restriction $x_i \le \lfloor \bar{x}_i^q \rfloor$) and right (with the restriction $x_i \ge \lceil \bar{x}_i^q \rceil$) descendants of node q. As a rule, in practice, we are limited to computing some estimates $\lambda_i^-(q)$ and $\lambda_i^+(q)$ for the decrements $z(q) - z_i^-(q)$ and $z(q) - z_i^+(q)$ of the objective value at node q. To perform branching, we select a variable for which the weighted estimate

$$\lambda_i(q) = (1-\mu)\cdot\min\{\lambda_i^-(q), \lambda_i^+(q)\} + \mu\cdot\max\{\lambda_i^-(q), \lambda_i^+(q)\}$$

is maximum. Here μ is a number between zero and one.

Next, we discuss some of the most commonly used rules for calculating the estimates $\lambda_i(q)$. Note that in practice, various combinations of those rules are also applied. In addition, branching on variables is not the only branching method used in the branch-and-cut algorithms. We will discuss some other branching methods in Sects. 6.3.2, 7.2.2 and 7.3.3.

Branching on Most Fractional Variables

This is one of the simplest and most frequently used rules according to which a variable with a fractional part closest to 0.5 is chosen. This corresponds to the calculation of the estimates by the following rule:

$$\lambda_i(q) = 0.5 - |\bar{x}_i - \lfloor \bar{x}_i \rfloor - 0.5|.$$

In other words, this rule chooses for branching that variable, which is ”harder” to round off. Unfortunately, numerical experiments have proved that in practice this rule is not better than the rule that chooses a variable for branching randomly.

Pseudocost Branching

This complex rule remembers all successful branchings over all variables. The values

$$\zeta_i^-(q) \overset{\text{def}}{=} (z(q) - z_i^-(q))/(\bar{x}_i^q - \lfloor \bar{x}_i^q \rfloor),$$
$$\zeta_i^+(q) \overset{\text{def}}{=} (z(q) - z_i^+(q))/(\lceil \bar{x}_i^q \rceil - \bar{x}_i^q)$$

define the average (per unit of change of the variable) decrement of the objective function for the left and right descendants of node q, respectively. By this definition, the value of $\zeta_i^-(q)$ or $\zeta_i^+(q)$ is infinity if the corresponding relaxation LP does not have a solution. In such a case, we can set either of these values to be equal to the integrality gap at node q if the latter is finite, or to some predefined penalty. Further, let $\xi_i^-(q)$ (resp., $\xi_i^+(q)$) denote the sum of $\zeta_i^-(q')$ (resp., $\zeta_i^+(q')$) for all nodes q' processed before the processing of node q starts, and for which the branching was performed on the variable x_i. Let η_i^- (resp., η_i^+) be the number of such nodes. For any variable x_i, the *left* and *right pseudocosts* are determined by the formulas

$$\Psi_i^-(q) \stackrel{\text{def}}{=} \xi_i^-(q)/\eta_i^-(q)\,, \quad \Psi_i^+(q) \stackrel{\text{def}}{=} \xi_i^+(q)/\eta_i^+(q)\,.$$

Defining

$$\begin{aligned} \lambda_i^-(q) &= \Psi_i^-(q)\cdot(\bar{x}_i^q - \lfloor \bar{x}_i^q \rfloor), \\ \lambda_i^+(q) &= \Psi_i^+(q)\cdot(\lceil \bar{x}_i^q \rceil - \bar{x}_i^q), \end{aligned}$$

we determine the *pseudocost branching rule*.

In early stages of the branch-and-cut algorithm almost all pseudocosts are zeroes. Therefore, applying pseudocost branching in the earliest most important stages of the algorithm, in fact, we will choose a variable for branching randomly. In practice, hybrid methods are used to select a variable for branching: at early stages some other branching rule is applied, and the algorithm switches to the pseudocost branching after it accumulates enough information.

Strong Branching

In its pure form, *strong branching* assumes the calculation of exact estimates

$$\lambda_i^-(q) = z(q) - z_i^-(q), \quad \lambda_i^+(q) = z(q) - z_i^+(q)$$

for all integer variables taking fractional values. Since the calculation of all the estimates takes too much time, the strong branching procedure can be modified as follows. First, a relatively small subset C of integer variables with fractional values (for example, 10% of all candidates with largest pseudocosts) is chosen. The next simplification is that, when estimating the decrements $z(q) - z_i^-(q)$ and $z(q) - z_i^+(q)$, only some fixed number of iterations of the dual simplex method is accomplished. This simplification is motivated by the fact that, for the dual simplex method, the average per iteration decrease of the objective value usually decreases with the number of iterations performed. However, this observation is not valid for many problems with built-in combinatorial structures, since, as a rule, the relaxation LPs of such problems are strongly degenerate.

6.3.1 Priorities

Assigning *priorities* to the variables allow us to establish a partial order relation on the set of integer variables: the higher the priority of a variable the more important it is. When a variable is selecting for branching, the priorities play a dominant role: a variable to branch on is usually selected from the variables with the highest priority.

As an example of a situation where setting priorities can significantly speed up the solution process, we can mention the multiperiod planning problems (see Sects. 2.4, 2.6, 2.11 and 2.14). Here, the decisions made in the early periods have a greater impact on future decisions than future decisions have on the past ones. Therefore, the decision variables for early periods should receive higher priorities.

The priorities allow us to resolve the following difficulty. Very often, we know that some variables automatically take integer values. As a simple example, we can mention a situation where in an equation with integer coefficients all variables except one are integer. It is obvious that this single non-integer variable can take only integer values. Another example is the family of x_{ij} variables in (2.4), which is the formulation of the facility location problem. The submatrix of the constraint matrix, composed of columns corresponding to the x_{ij} variables, is totally unimodular (see Exercise 1.12). Therefore, any basic feasible solution to the relaxation LP for (2.4) has integer components x_{ij} if all y_i take integer values. If we declare the variables x_{ij} to be integers, then any of them can be chosen for branching, which is undesirable. On the other hand, by declaring variables x_{ij} as integer-valued, we can benefit by generating stronger cuts. A simple solution to this problem is to declare the x_{ij} variables as integer-valued and assign them the lowest priority.

6.3.2 Special Ordered Sets

Let us recall the representation (1.2) of the discrete variable (see Sect. 1.1.1). Suppose that we have a MIP with such a constraint. If, in an optimal solution to the relaxation LP, not all values $\tilde{\lambda}_i$ of the variables λ_i are integers, then using any standard branching rule, we choose a fractional component $\tilde{\lambda}_{i^*}$ to divide the feasible domain of λ variables,

$$K = \left\{ \lambda \in \{0,1\}^k : \sum_{i=1}^{k} \lambda_i = 1 \right\},$$

into two subsets: $K_0 = \{\lambda \in K : \lambda_{i^*} = 0\}$ of cardinality $k-1$, and $K_1 = \{\lambda \in K : \lambda_{i^*} = 1\}$ of cardinality 1. Since the set K_0 is usually much larger than the set K_1, the search tree turns out to be *unbalanced*. The situation can be corrected if we use a *balanced branching*, when the set K is divided into two subsets:

$$\begin{aligned}
\bar{K}_0 &= \{\lambda \in K : \lambda_i = 0,\ i = 1,\dots,r\},\\
\bar{K}_1 &= \{\lambda \in K : \lambda_i = 0,\ i = r+1,\dots,k\},
\end{aligned}$$

where

$$r = \arg\min_{1 \le j < k} \left| \sum_{i=1}^{j} \tilde{\lambda}_i - \sum_{i=j+1}^{k} \tilde{\lambda}_i \right|.$$

Such a way of branching is known as *SOS1-branching*.

A special way of balanced branching is also used if a MIP contains SOS2-constraints, which are used to represent piecewise linear approximations of non-linear functions (see Sect. 1.1.3). Let $\tilde{\lambda}_i$ denote the values of the variables λ_i in a SOS2-constraint given by the equation

$$\sum_{i=1}^{k} \lambda_i = 1$$

and the requirement that no more than two components λ_i take non-zero values, and if there are two of such components, they must be consecutive. Let

$$i_1 = \min\{i : \tilde{\lambda}_i > 0\}, \quad i_2 = \max\{i : \tilde{\lambda}_i > 0\}.$$

If $i_2 - i_1 > 1$, then after branching the feasible set

$$K = \left\{ \lambda \in [0,1]^k : \sum_{i=1}^{k} \lambda_i = 1 \right\}$$

is divided into two subsets

$$K_1 = \{\lambda \in K : \lambda_i = 0,\ i = 1, \dots, r\},$$
$$K_2 = \{\lambda \in K : \lambda_i = 0,\ i = r, \dots, k\},$$

where $r = \lfloor (i_1 + i_2)/2 \rfloor$. This way of branching is called a SOS2-branching.

6.4 Global Gomory Cuts

Let us recall that global cuts are valid for all nodes of the search tree, and local cuts are valid only for a particular node and all its descendants. Therefore, as a rule, global cuts are more useful in practice. Recall that we declare a cut as being local if in its derivation we used other local inequalities. Most often, such inequalities are the lower and upper bounds for the values of integer variables. Changing a bound (lower or upper) of a binary variable means fixing its value. This simple observation makes it possible to generate a global fractional Gomory cut each time when no local inequalities, other than the bounds for binary variables, were used in the derivation of the base inequality (for which mixed integer rounding is applied). Let us demonstrate this with a simple example.

Example 6.3 *Let us imagine that we are solving the following IP*

$$\begin{aligned} 4x_1 + 2x_2 + 5x_3 &\to \max, \\ 3x_1 + 2x_2 + 2x_3 &\le 4, \\ x_1, x_2, x_3 &\in \{0,1\} \end{aligned} \tag{6.2}$$

by the branch-and-cut method. We know an optimal solution, $x^* = \left(\frac{2}{3}, 0, 1\right)$, to the root relaxation LP, and now we need to process the child node obtained from the root node after fixing x_1 to 1.

Solution. First, let us write down the relaxation LP for this node:

$$\begin{array}{rl} & 4x_1 + 2x_2 + 5x_3 \to \max, \\ 1: & 3x_1 + 2x_2 + 2x_3 \le 4, \\ 2: & 1 \le x_1 \le 1, \\ 3: & 0 \le x_2 \le 1, \\ 4: & 0 \le x_3 \le 1. \end{array}$$

Its optimal basic feasible solution, basic set and inverse basic matrix are the following:

$$\bar{x} = \left(1, 0, \frac{1}{2}\right)^T, \ I = (1, -2, -3) \text{ and } B^{-1} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ \frac{1}{2} & \frac{3}{2} & 1 \end{bmatrix}.$$

As x_3 is the only variable taking a fractional value, we will build the fractional Gomory cut starting with the equation:

$$x_3 + \frac{1}{2}s_1 + \frac{3}{2}s_2 + s_3 = \frac{1}{2}. \tag{6.3}$$

Next, we compute the coefficients

$$f_0 = \frac{1}{2}, \quad \frac{f_0}{1 - f_0} = 1, \quad f_1 = \frac{1}{2}, \quad f_2 = \frac{1}{2}, \quad f_3 = 0$$

and write down the cut

$$\frac{1}{2}s_1 + \frac{1}{2}s_2 \ge \frac{1}{2},$$

or

$$s_1 + s_2 \ge 1.$$

Substituting $4 - 3x_1 - 2x_2 - 2x_3$ for s_1, and $-1 + x_1$ for s_2, after simplifications and rearranging, we get the cut in the initial variables:

$$x_1 + x_2 + x_3 \le 1.$$

This inequality, as it should, cuts off the point $\bar{x}$, but is not global, since it also cuts off the point $(0, 1, 1)^T$, which is a feasible solution to (6.2). This happened because, when derivating this cut, we used the local bound $x_1 \ge 1$.

Now, let us build a global cut. Since we have the equation $x_1 = 1$, we can include into the basic set any of two inequalities: $x_1 \ge 1$, which is local, or $x_1 \le 1$, which is global. This time we include into the basic set the global inequality $x_1 \le 1$. So, we have the basic set $\bar{I} = (1, 2, -3)$ and the inverse basic matrix

$$\bar{B}^{-1} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ \frac{1}{2} & -\frac{3}{2} & 1 \end{bmatrix}.$$

Now we will build the cut starting with the equation:

$$x_3 + \frac{1}{2}s_1 - \frac{3}{2}s_2 + s_3 = \frac{1}{2}. \tag{6.4}$$

Before continuing, note that we could write down (6.4) directly by the matrix B^{-1} (without writing $\bar{B}^{-1}$): (6.4) is obtained from (6.3) by changing the sign of the coefficient of the variable s_2, which corresponds to the inequality $x_1 \le 1$.

Let us continue building the cut. We calculate the coefficients

$$f_0 = \frac{1}{2}, \quad \frac{f_0}{1-f_0} = 1, \quad f_1 = \frac{1}{2}, \quad f_2 = -\frac{3}{2} - (-2) = \frac{1}{2}, \quad f_3 = 0$$

and write down the cut

$$\frac{1}{2}s_1 + \frac{1}{2}s_2 \ge \frac{1}{2},$$

or

$$s_1 + s_2 \ge 1.$$

Substituting $4 - 3x_1 - 2x_2 - 2x_3$ for s_1, and $1 - x_1$ for s_2, after simplifications and rearranging, we get the cut in the initial variables:

$$2x_1 + x_2 + x_3 \le 2.$$

In the derivation of this cut, we did not use local inequalities, and therefore this cut is global. □

6.5 Preprocessing

We have not yet discussed one position in the block diagram of the branch-and-cut method shown in Fig. 6.2. It is about preprocessing, or automatic reformulation. All modern commercial **MIP** solvers begin solving any MIP with an attempt to simplify it (narrow the feasible intervals for variables or even fix their values, classify variables and constraints, strengthen inequalities, scale the constraint matrix, etc.). The following simple statements give an idea what actions are being performed in the *preprocessing* step.

Proposition 6.1. *Given the inequalities $\sum_{j=1}^n a_j x_j \le b$ and $l \le x \le u$, where $a, l, u \in \mathbb{R}^n$. Let*

$$v_i = \sum_{j:\ j\ne i,\, a_j>0} a_j l_j + \sum_{j:\ j\ne i,\, a_j<0} a_j u_j.$$

1) (***Bounds on variables***) *If $a_i > 0$, then*

$$x_i \le \min\left\{u_i, \frac{b - \nu_i}{a_i}\right\},$$

and if $a_i < 0$, then

$$x_i \ge \max\left\{l_i, \frac{b - \nu_i}{a_i}\right\}.$$

For an integer variable x_i, we have

$$\lceil l_i \rceil \le x_i \le \lfloor u_i \rfloor.$$

2) (***Redundancy***) *The inequality $\sum_{j=1}^{n} a_j x_j \le b$ is redundant if*

$$\sum_{j:\ a_j>0} a_j u_j + \sum_{j:\ a_j<0} a_j l_j \le b.$$

3) (***Infeasibility***) *The system $\sum_{j=1}^{n} a_j x_j \le b$ and $l \le x \le u$ is infeasible if*

$$\sum_{j:\ a_j>0} a_j l_j + \sum_{j:\ a_j<0} a_j u_j > b.$$

Proposition 6.2. (***Fixing variables***) *Consider the LP*

$$\max\{c^T x :\ Ax \le b,\ l \le x \le u\}.$$

If $A^j \ge 0$ and $c_j \le 0$, then $x_j = l_j$. Conversely, if $A^j \le 0$ and $c_j \ge 0$, then $x_j = u_j$.

Example 6.4 *We need to apply Propositions 6.1 and 6.2 to strengthen the following formulation:*

$$\begin{array}{rl}
 & 3x_1 + 2x_2 - x_3 \to \max,\\
1: & 4x_1 - 3x_2 + 2x_3 \le 13,\\
2: & 7x_1 + 3x_2 - 4x_3 \ge 8,\\
3: & x_1 + 2x_2 - x_3 \ge 4,\\
 & x_1 \in \mathbb{Z}_+,\\
 & x_2 \in \{0,1\},\\
 & x_3 \ge 1.
\end{array}$$

Solution. In view of Proposition 6.2, we fix the value of x_3 to 1. Further, from Ineqs. 1 and 2, we have

$$\begin{aligned}
x_1 &\le \lfloor (13 + 3x_2 - 2x_3)/4 \rfloor \le \lfloor (13 + 3\cdot 1 - 2\cdot 1)/4 \rfloor = 3,\\
x_1 &\ge \lceil (8 - 3x_2 + 4x_3)/7 \rceil \ge \lceil (8 - 3\cdot 1 + 4\cdot 1)/7 \rceil = 2.
\end{aligned}$$

From Ineq. 3, we obtain the lower bound:

$$x_2 \ge \lceil (4 - x_1 + x_3)/2 \rceil \ge \lceil (4 - 3 - 1)/2 \rceil = 1.$$

Hence, $x_2 = 1$, and from Ineq. 3 we establish the bound:

$$x_1 \geq 4 - 2x_2 + x_3 = 4 - 2 \cdot 1 + 1 = 3.$$

So, we have fixed the values of all variables and found the solution to this example, $x^* = (3, 1, 1)$, already at the preprocessing stage. Of course, this is rather an exception than a rule. □

Another simple way to strengthen the formulation of a given MIP is to change the constraint coefficients. Let us start with a specific example. The inequality

$$\sum_{j=1}^{n} a_j x_j \geq \beta,$$

in which all the variables x_j are binary and the coefficients a_j and β are positive, is equivalent to the inequality

$$\sum_{j=1}^{n} \min\{a_j, \beta\} x_j \geq \beta.$$

Geometrically, decreasing some coefficients a_j, we rotate the hyperplane $H(a, \beta)$ in such a way that it touches more feasible solutions.

Proposition 6.3. *Let X be the solution set of the system*

$$\alpha y + \sum_{j=1}^{n} a_j x_j \leq \beta, \tag{6.5}$$

$$y \in \{0, 1\} \text{ and } l_j \leq x_j \leq u_j,\ j = 1, \dots, n, \tag{6.6}$$

and let

$$U = \sum_{j: a_j > 0} a_j u_j + \sum_{j: a_j < 0} a_j l_j.$$

If $\alpha < 0$, then for $\bar{\alpha} = \max\{\alpha, \beta - U\}$ the inequality

$$\bar{\alpha} y + \sum_{j=1}^{n} a_j x_j \leq \beta \tag{6.7}$$

is valid for X and is not weaker than (6.5).

If $\alpha > 0$, then for $\hat{\alpha} = \min\{\alpha, U - \beta + \alpha\}$ the inequality

$$\hat{\alpha} y + \sum_{j=1}^{n} a_j x_j \leq \beta - \alpha + \hat{\alpha} \tag{6.8}$$

is valid for X and is not weaker than (6.5).

Proof. First we consider the case when $\alpha < 0$. Inequalities (6.5) and (6.7) are equivalent if $y = 0$ or $\bar{\alpha} = \alpha$. Therefore, suppose that $y = 1$ and $\bar{\alpha} > \alpha$. Then

$$-\alpha > -\bar{\alpha} = -\beta + U \geq -\beta + \sum_{j=1}^{n} a_j x_j$$

and, consequently,

$$\bar{\alpha} \cdot 1 + \sum_{j=1}^{n} a_j x_j \leq \beta.$$

Now consider the case when $\alpha > 0$. Let us rewrite (6.5) in the form

$$-\alpha(1-y) + \sum_{j=1}^{n} a_j x_j \leq \beta - \alpha. \qquad (6.9)$$

From the already proved part of this proposition for the case $\alpha < 0$, it follows that, for

$$-\hat{\alpha} = \max\{-\alpha, \beta - \alpha - U\},$$

the inequality

$$-\hat{\alpha}(1-y) + \sum_{j=1}^{n} a_j x_j \leq \beta - \alpha$$

is valid for X and is not weaker than (6.9). □

Example 6.5 *We need to reduce the coefficients of the binary variables in the inequality*

$$92x_1 - 5x_2 + 72x_3 - 10x_4 + 2x_5 \leq 100 \qquad (6.10)$$

under the following conditions:

$$x_1, x_2, x_3 \in \{0,1\},\ 0 \leq x_4 \leq 5,\ 0 \leq x_5 \leq 2.$$

Solution. Let us apply Proposition 6.3 sequentially for the variables x_1, x_2 and x_3. Since $\alpha = a_1 = 92 > 0$, $U = -5 \cdot 0 + 72 - 10 \cdot 0 + 2 \cdot 2 = 76$, then

$$\hat{\alpha} = \min\{92, 76 - 100 + 92\} = 68,$$

and we obtain the inequality

$$68x_1 - 5x_2 + 72x_3 - 10x_4 + 2x_5 \leq 100 - 92 + 68 = 76.$$

As $\alpha = a_2 = -5 < 0$, $U = 68 + 72 + 4 = 144$, then

$$\bar{\alpha} = \max\{-5, 76 - 144\} = -5,$$

and therefore we cannot change the coefficient of x_2.

Since $\alpha = a_3 = 72 > 0$, $U = 68 + 4 = 72$, then

$$\hat{\alpha} = \min\{72, 72 - 76 + 72\} = 68,$$

and we obtain the final inequality

$$68x_1 - 5x_2 + 68x_3 - 10x_4 + 2x_5 \le 76 - 72 + 68 = 72. \qquad \square$$

After solving the relaxation LP for some node of the search tree, we have additional information that can be used to strengthen the formulation at this node.

Proposition 6.4. *Let γ be the optimal objective value of the relaxation LP for some node being processed, and R be the record value at that time. Let x_j^* and $\bar{c}_j \neq 0$ be the value and the reduced cost of an integer variable x_j, $l_j \le x_j \le u_j$. Further, let $\delta = \lfloor (\gamma - R)/\bar{c}_j \rfloor$. Then, for any optimal solution of the relaxation LP, the following holds:*

if $x_j^* = l_j$, then $x_j \le l_j + \delta$, and if $x_j^* = u_j$, then $x_j \ge u_j - \delta$.

6.5.1 Disaggregation of Inequalities

There exist methods that allows us to replace (*aggregate*) a system of linear inequalities with a single linear inequality so that the integer solutions of the system and this single inequality are the same. But in practice it is better to *disaggregate* inequalities; usually, this strengthens existing formulations. In Example 1.1 we strengthened an **IP** formulation of a binary set by replacing an inequality with a family of inequalities that imply the original inequality. The next proposition presents a simple but rather general disaggregation technique.

Proposition 6.5. *Let us consider the solution set X to the inequality*

$$\sum_{j\in B} f_j x_j + \sum_{i\in I} a_i y_i \le b, \tag{6.11}$$

where all variables x_j $(j \in B)$ are binary, all variables y_i $(i \in I)$ are integer (or binary), all coefficients f_j $(j \in B)$ are positive and $\sum_{j\in B} f_j \le 1$, b and all coefficients a_i $(i \in I)$ are integers. Then the inequalities

$$x_j + \sum_{i\in I} a_i y_i \le b, \quad j \in B, \tag{6.12}$$

are also valid for X, and (6.11) *is a consequence of* (6.12).

Proof. The inequalities

$$f_j x_j + \sum_{i\in I} a_i y_i \le b, \quad j \in B,$$

are valid for X since all f_j and x_j are non negative. As x_j are binaries, and $b - \sum_{i\in I} a_i y_i$ takes only integer values, all inequalities from (6.12) are also valid for X.

Multiplying the j-th inequality in (6.12) by $f_j / \sum_{k \in B} f_k$, and then summing together all $|B|$ resulting inequalities, we get the inequality

$$\frac{1}{\sum_{j \in B} f_j} \sum_{j \in B} f_j x_j + \sum_{i \in I} a_i y_i \le b,$$

which is not weaker than (6.11).

Thus, we have proven that (6.12) is not weaker (usually much stronger) formulation for X than (6.11). $\square$

6.5.2 Probing

Solving a MIP with a substantial share of binary variables, we can also apply the technique of *probing* the values of binary variable. At each iteration of the probing procedure, the value of one binary variable, x_i, is fixed to $\alpha \in \{0,1\}$, the basic preprocessing techniques are applied, and then we explore the consequences.

1. If infeasibility is detected, then, for any feasible solution, the variable x_i cannot take the value of α, and, therefore, its value must be set to $1-\alpha$. For demonstration, consider a simple example:

$$2x_1 - 2x_2 + x_3 \le 0, \quad x_1 + x_2 \le 1, \quad x_1, x_2 \in \{0,1\}, \quad x_3 \ge 1.$$

Setting $x_1 = 1$, from the second inequality we obtain that $x_2 = 0$, and from the first one we derive the upper bound $x_3 \le -2$, which contradicts to the lower bound $x_3 \ge 1$. Therefore, we can set $x_1 = 0$.

2. If one of the bounds of a constraint $l \le a^T x \le u$ is changed, then we can strengthen this constraint. Let $\bar{l}, \bar{u}$ be new bounds established as a result of preprocessing. Then the following inequalities hold

$$\begin{aligned} l + (\bar{l} - l)(1 - x_i) \le a^T x \le u - (u - \bar{u})(1 - x_i) \quad &\text{if } \alpha = 0, \\ l + (\bar{l} - l)x_i \le a^T x \le u - (u - \bar{u})x_i \quad &\text{if } \alpha = 1. \end{aligned}$$

In the example

$$x_1 - x_2 \le 0, \quad x_1 - x_3 \le 0, \quad x_2 + x_3 \ge 1,$$

after setting $x_1 = 1$, we obtain the inequalities $x_2 \ge 1, x_3 \ge 1$, and a new lower bound for the third inequality: $x_2 + x_3 \ge 2$. Therefore, we can write down the inequality

$$x_2 + x_3 \ge 1 + (2-1)x_1 \quad \text{or} \quad -x_1 + x_2 + x_3 \ge 1.$$

We can also consider the bounds on variables as inequalities. Consider the system

$$5x_1 + 3x_2 + 2x_3 \le 10, \quad x_1 \in \{0,1\}, \quad x_2 \le 3, \quad x_3 \le 5, \quad x_2, x_3 \in \mathbb{Z}_+.$$

Fixing x_1 to 1, from the first inequality we get the upper bounds $x_2 \le 1$ and $x_3 \le 2$. This allows us to derive the following inequalities:

$$x_2 \le 3 - (3-1)x_1 \quad \text{and} \quad x_3 \le 5 - (5-2)x_1\,,$$

or

$$2x_1 + x_2 \le 3 \quad \text{and} \quad 3x_1 + x_3 \le 5.$$

Despite the apparent simplicity, the probing procedure is a very powerful tool for enhancing the formulations of MIPs with binary variables. Besides, the probing techniques subsume some of the preprocessing techniques that we discussed earlier.

Disaggregation of Inequalities

Proposition 6.5 demonstrates a way of strengthening an inequality by replacing it with a family of inequalities that imply the original constraint. In some cases, we can automate the disaggregation process by probing binary variables.

Consider the inequality

$$x_1 + x_2 + \cdots + x_n \le k \cdot y,$$

where all variables are binary, and $1 \le k \le n$. Fixing y to 0, we get the new upper bounds $x_i \le 0$, $i = 1, \dots, n$. Consequently, the following inequalities are valid:

$$x_i \le y, \quad i = 1, \dots, n.$$

Changing Coefficients of Binary Variables

Probing binary variables, we can achieve much more than Proposition 6.3 allows.

Example 6.6 *We need to strengthen the first inequality in the following system*

$$\begin{aligned} 4x_1 + 5x_2 + 7x_3 + 2x_4 &\le 9, \\ x_1 + x_2 &\le 1, \\ x_1, x_2, x_3, x_4 &\in \{0,1\}. \end{aligned}$$

Solution. First we note that we cannot strengthen the first inequality in the way specified in Proposition 6.3. Therefore, let us proceed to probing the variables.

Setting $x_1 = 1$, from the second inequality we conclude that $x_2 = 0$, and from the first inequality it follows that $x_3 = 0$. Then the maximum value of the left-hand side of the first inequality is $u = 4 + 2 = 6 < 9$. Therefore, if $x_1 = 1$, this inequality becomes redundant, and it can be strengthened as follows:

$$4x_1 + 5x_2 + 7x_3 + 2x_4 \le 9 - (9-6)x_1$$

or

$$7x_1 + 5x_2 + 7x_3 + 2x_4 \le 9. \tag{*}$$

Setting $x_2 = 1$ and arguing in a similar way as above, we can further strengthen (*) to the inequality

$$7x_1 + 7x_2 + 7x_3 + 2x_4 \le 9. \qquad \square$$

6.6 Traveling Salesman Problem

Often, a formulation of a MIP contains too many inequalities, and all of them cannot be stored in the computer memory. In such cases, some of the inequalities are excluded from the formulation, the truncated problem is solved by the branch-and-cut method, and the excluded inequalities are considered as cuts. But such cuts, which constitute a part of a MIP formulation, must be represented by an exact separation procedure. Otherwise, we could get an infeasible solution to our MIP. Let us demonstrate what has been said with a famous example of the *minimum Hamiltonian cycle problem*.

Given a (undirected) graph $G = (V,E)$, each edge $e \in E$ of which is assigned a *cost* c_e, we need to find a *Hamiltonian cycle* (a simple cycle that covers all vertices) with the minimum total cost of edges. We note that the minimum Hamiltonian cycle problem on complete graphs is also called the *traveling salesman problem* (*TSP*) because of the following interpretation. There are n cities and the distance c_{ij} is known between any pair of cities i and j. A traveling salesman wants to find the shortest ring route, which visits each of the n cities exactly once. As a subproblem, the TSP appears in practical applications in the following context. A multifunctional device, processing a unit of some product, performs over it n operations in any order. The readjustment time of the device after performing operation i for operation j is t_{ij}. It is necessary to find the order of performing operations for which the total time spent on readjustments is minimum.

Introducing binary variables x_e, $e \in E$, with $x_e = 1$ if edge e is included in the Hamiltonian cycle, and $x_e = 0$ otherwise, we can formulate the minimum Hamiltonian cycle problem as follows:

$$\sum_{e\in E} c_e x_e \to \min, \tag{6.13a}$$

$$\sum_{e\in E(v,V)} x_e = 2, \quad v \in V, \tag{6.13b}$$

$$\sum_{e\in E(S,V\setminus S)} x_e \ge 2, \quad \emptyset \ne S \subset V, \tag{6.13c}$$

$$x_e \in \{0,1\}, \quad e \in E. \tag{6.13d}$$

Here we use the notation $E(S,T)$ for the subset of edges from E with one end vertex in S, and the other in T.

Equations (6.13b) require that each vertex be incident to exactly two selected edges. The *subtour elimination inequalities* (6.13c) are needed to exclude "short cycles" (see below the solution of Example 6.7). System (6.13c) contains too many inequalities. Even for relatively small graphs (say with 50 vertices) we can not store in the memory of a modern computer any conceivable description of all subtour elimination inequalities. But we can treat the subtour elimination inequalities as cuts and add them to the active node LPs as needed. To do this, we only need to solve the following separation problem:

> *given a point $\tilde{x} \in [0,1]^E$ that satisfies (6.13b), it is needed to verify whether all inequalities in (6.13c) are valid, and if there exist violated inequalities, find one (or a few) of them.*

This separation problem can be formulated as the *minimum cut problem* in which we need to find a proper subset $\tilde{S}$ of the vertex set V ($\emptyset \neq \tilde{S} \subset V$) such that the value $q = \sum_{e \in E(\tilde{S}, V \setminus \tilde{S})} \tilde{x}_e$ is minimum. If $q < 2$, then the inequality $\sum_{e \in E(\tilde{S}, V \setminus \tilde{S})} x_e \geq 2$ is violated at $\tilde{x}$; otherwise $\tilde{x}$ satisfies all inequalities from (6.13c).

To find a minimum cut, there are effective deterministic and probabilistic algorithms. We can use one of them to solve the separation problem for the subtour elimination inequalities. In addition, we can find several minimum cuts (violated inequalities) at once by constructing the Gomory-Hu tree. It is said that a cut $(S, V \setminus S)$ separates two vertices s and t if exactly one of these vertices belongs to S; such a cut is also called an *s,t-cut*. The Gomory-Hu tree, $T_{GH} = (V, \tilde{E})$, is defined on the vertex set V of the graph G, but the edges $e \in \tilde{E}$ need not be edges in G. Each edge $e \in \tilde{E}$ is assigned a number f_e. For given two vertices $s,t \in V$, we can find a minimum s,t-cut as follows. First, on a single path connecting s and t in T_{GH}, we need to find an edge e with the minimal value f_e. Removing this edge e from the tree T_{GH}, we get two subtrees. Let S and $V \setminus S$ be the vertex sets of these subtrees, then the partition $(S, V \setminus S)$ is a minimum s,t-cut. In spite of the fact that n-vertex graphs have $n(n-1)/2$ different s,t-cuts (different pairs s,t), we can build the Gomory-Hu tree by a procedure that solves only $n-1$ minimum s,t-cut problems.

Example 6.7 *Consider an example of the minimum Hamiltonian cycle problem defined on the graph depicted in Fig. 6.4, where the numbers near the edges are their costs. We need to solve this example by the branch-and-cut method, when a) $\alpha = 4$, $\beta = 10$; b) $\alpha = \beta = 0$.*

Solution. In both cases, we begin by solving the LP

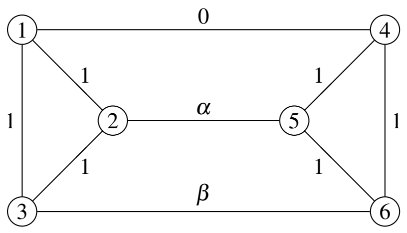

Fig. 6.4 An example of the Hamiltonian cycle problem

$$
\begin{array}{l}
x_{1,2}+x_{2,3}+x_{3,1}+x_{4,5}+x_{5,6}+x_{6,4}+\qquad \alpha x_{2,5}+\beta x_{3,6}\to\min,\\
x_{1,2}+\qquad x_{3,1}+\qquad\qquad\qquad x_{1,4}\qquad\qquad\qquad =2,\\
x_{1,2}+x_{2,3}+\qquad\qquad\qquad\qquad\qquad x_{2,5}\qquad\quad =2,\\
\qquad x_{2,3}+x_{3,1}+\qquad\qquad\qquad\qquad\qquad\qquad x_{3,6}=2,\\
\qquad\qquad\qquad x_{4,5}+\qquad x_{6,4}+x_{1,4}\qquad\qquad\qquad =2,\\
\qquad\qquad\qquad x_{4,5}+x_{5,6}+\qquad\qquad x_{2,5}\qquad\quad =2,\\
\qquad\qquad\qquad\qquad x_{5,6}+x_{6,4}+\qquad\qquad\qquad x_{3,6}=2,\\
0\le x_{1,2},x_{2,3},x_{3,1},x_{4,5},x_{5,6},x_{6,4},x_{1,4},x_{2,5},x_{3,6}\le 1.
\end{array}
\tag{6.14}
$$

Here the variable $x_{i,j}$ corresponds to the variable x_e for the edge $e=(i,j)$.

Case a). Use your favorite LP solver to verify that, for $\alpha=4$ and $\beta=10$, an optimal solution to (6.14) is the point $x^{(1)}$ with the coordinates:

$$
\begin{aligned}
x^{(1)}_{1,2}=x^{(1)}_{2,3}=x^{(1)}_{3,1}=x^{(1)}_{4,5}=x^{(1)}_{5,6}=x^{(1)}_{6,4}=1,\\
x^{(1)}_{1,4}=x^{(1)}_{2,5}=x^{(1)}_{3,6}=0.
\end{aligned}
$$

Note that two short cycles correspond to the point $x^{(1)}$:

$$1\to 2\to 3\to 1 \quad\text{and}\quad 4\to 5\to 6\to 4.$$

The point $x^{(1)}$ violates the subtour elimination inequality for $S=\{1,2,3\}$:

$$x_{1,4}+x_{2,5}+x_{3,6}\ge 2.$$

Adding this inequality to the constraints of LP (6.14), after the reoptimization, we obtain the solution $x^{(2)}$ with the coordinates:

$$
\begin{aligned}
x^{(2)}_{3,1}=x^{(2)}_{2,3}=x^{(2)}_{2,5}=x^{(2)}_{5,6}=x^{(2)}_{6,4}=x^{(2)}_{1,4}=1,\\
x^{(2)}_{1,2}=x^{(2)}_{3,6}=x^{(2)}_{4,5}=0.
\end{aligned}
\tag{6.15}
$$

The integer point $x^{(2)}$ determines a minimum Hamiltonian cycle

$$1\to 3\to 2\to 5\to 6\to 4\to 1,$$

which cost equals 8.

Case b). Now we solve (6.14) with $\alpha = \beta = 0$. Its solution is the point $x^{(3)}$ with the following coordinates:

$$x^{(3)}_{1,2} = x^{(3)}_{2,3} = x^{(3)}_{3,1} = x^{(3)}_{4,5} = x^{(3)}_{5,6} = x^{(3)}_{6,4} = \frac{1}{2},$$
$$x^{(3)}_{1,4} = x^{(3)}_{2,5} = x^{(3)}_{3,6} = 1.$$

For such a small example, it is not difficult to verify (even without solving the separation problem) that the point $x^{(3)}$ satisfies all the subtour elimination inequalities. This suggests that Formulation (6.13), containing so many inequalities, is not ideal. Many other classes of inequalities are known for the minimum Hamiltonian cycle problem. But, unlike the subtour elimination inequalities, all other classes of inequalities are usually not part of the problem formulation.

It is easy to see that of the six edges

$$(1,2),\ (2,3),\ (3,1),\ (1,4),\ (2,5),\ (3,6)$$

no more than four can be on a Hamiltonian cycle. Therefore, the next inequality must hold

$$x_{1,2} + x_{2,3} + x_{3,1} + x_{1,4} + x_{2,5} + x_{3,6} \le 4, \tag{6.16}$$

which is violated at $x^{(3)}$. Adding this inequality to the constraints of (6.16), after reoptimization, we again get the solution $x^{(2)}$ given by (6.15), but now the cost of $x^{(2)}$ is 4. □

Inequality (6.16) belongs to a large class of inequalities called comb-inequalities. Suppose that a family of vertex subsets $H \subseteq V$ and $T_i \subseteq V$ for $i = 1,\dots,k$, satisfies the following conditions :

$$\begin{aligned} &|H \cap T_i| \ge 1,\ |T_i \setminus H| \ge 1, \quad i = 1,\dots,k,\\ &T_i \cap T_j = \emptyset, \quad i = 1,\dots,k-1,\ j = i+1,\dots,k, \end{aligned} \tag{6.17}$$

where $k \ge 3$ is an odd integer. The configuration $C = (H, T_1,\dots,T_k)$ is called a *comb* in graph G with the *handle H* and the *teeth T_i* (for illustration see Fig. 6.5). Denote by $x(S)$ the sum $\sum_{e \in E(S,S)} x_e$. The inequality

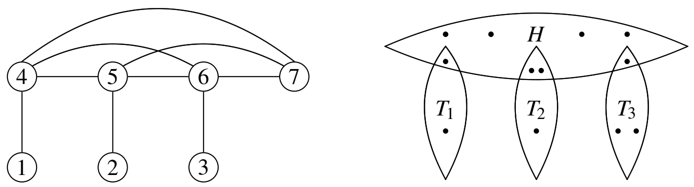

Fig. 6.5 Graphical representation of comb-inequalities

$$x(H)+\sum_{i=1}^{k} x(T_i) \le |H| + \sum_{i=1}^{k}(|T_i|-2) + \left\lfloor \frac{k}{2} \right\rfloor \tag{6.18}$$

is called a *comb-inequality*.

Let us show that (6.18) is a Chvátal-Gomory cut. Using (6.13b) and the following equivalent representation for the subtour elimination inequalities

$$x(S) \le |S|-1, \quad \emptyset \ne S \subset V,$$

we can write down the following chain of inequalities:

$$\begin{aligned}2x(H)+2\sum_{i=1}^{k} x(T_i) &\le x(H,V)+\sum_{i=1}^{k}(x(T_i)+x(T_i\cap H)+x(T_i\setminus H))\\ &\le 2|H|+\sum_{i=1}^{k}((|T_i|-1)+(|T_i\cap H|-1)+(|T_i\setminus H|-1))\\ &= 2|H|+2\sum_{i=1}^{k}(|T_i|-2)+k.\end{aligned}$$

Dividing both sides of the resulting inequality by 2 and rounding the right-hand side, we obtain (6.18).

Even for small n, the total number of comb-inequalities is huge, there are much more of them than there are the subtour elimination inequalities. To use the comb-inequalities in computational algorithms, we need an efficient separation procedure for these inequalities. In the general case, the separation problem for comb-inequalities is not solved. But there are several heuristic separation procedures. These procedures may not find an inequality violated at a given point, even if such inequalities exist.

An efficient exact separation procedure is known only for a subclass of comb-inequalities, when

$$|H\cap T_i|=1,\ |T_i\setminus H|=1, \quad i=1,\dots,k.$$

Such comb-inequalities are also called *flower inequalities* because these inequalities are sufficient to describe the 2-*matching polyhedron*, which is the convex hull of points $x\in\{0,1\}^E$ satisfying (6.13b).

From a practical point of view, the main difference between the cuts that are in the problem formulation, and the usual cuts, is that the exact separation procedure is necessary for the former, and heuristic separation procedures can be used for the latter. There are many examples where, even when there are theoretically efficient separation procedures, in practice, preference is given to faster heuristics.

6.7 Notes

***Sect.* 6.1.** The **LP**-based branch-and-cut method for integer programming was proposed by Land and Doig in [83].

***Sect.* 6.2.** Articles [62, 106] were among the first to describe the use of cuts in the branch-and-bound method.

***Sect.* 6.3.** Now standard branching on an integer variable with a fractional value appeared in [41], pseudocost branching was proposed in [23], strong branching was introduced in CPLEX 7.5 (see also [6]), and GUB/SOS-branching was proposed in [19].

***Sect.* 6.4.** The idea to generate global cuts avoiding local bounds for binary variables was specified in [14].

***Sect.* 6.5.** Many preprocessing methods are considered as folklore, since it is very difficult to trace their origins. Various aspects of preprocessing are discussed in [4, 30, 70, 74, 119, 129, 132].

***Sect.* 6.6.** Danzig, Falkerson and Johnson [44] were the first to use cuts for solving a traveling salesman problem with 49 cities. Later, their method was significantly expanded and improved by many researchers. An overview of these results is provided in [96]. An implementation of the branch-and-cut method for solving very big traveling salesman problems is discussed in [7].

The minimum cut problem can be efficiently solved by both deterministic [97] and probabilistic [79] algorithms. An efficient implementation of the procedure for constructing the Gomory-Hu tree was proposed in [71].

The comb inequalities were introduced in [36] for a particular case of comb structures, where each tooth contains exactly one vertex from the handle, and the general comb inequalities appeared in [65]. The separation procedure for flower inequalities was developed in [105]. A number of heuristic separation procedures for comb-like inequalities where described in [107].

***Sect.* 6.8.** The statement of Exercise 6.11 was taken from [92] (see also [110]).

6.8 Exercises

6.1. Consider the following IP

$$\max\{-x_{n+1} \,:\, 2x_1+2x_2+\cdots+2x_n+x_{n+1}=n,\ x\in\{0,1\}^{n+1}\}.$$

Prove that for odd n, the branch-and-bound method from Listing 6.1 processes an exponential (in n) number of nodes.

6.2. How many branchings can the branch-and-bound method perform in the worst case when solving an IP with one integer variable?

6.3. Solve again Example 6.1 by the branch-and-bound method, but now first apply preprocessing.

6.4. Solve the following IPs by the branch-and-bound method:

a)
$$\begin{aligned} 4x_1+5x_2+x_3 &\to \max,\\ 3x_1+2x_2 \quad\quad &\le 10,\\ x_1+4x_2 \quad\quad &\le 11,\\ 3x_1+3x_2+x_3 &\le 13,\\ 0\le x_1 &\le 4,\\ 0\le x_2 &\le 3,\\ 0\le x_3 &\le 5,\\ x_1,x_2,x_3 &\in \mathbb{Z}; \end{aligned}$$

b)
$$\begin{aligned} 10x_1+14x_2+21x_3 &\to \min,\\ 2x_1+\ 2x_2+\ 7x_3 &\ge 14,\\ 8x_1+11x_2+\ 9x_3 &\ge 12,\\ 9x_1+\ 6x_2+\ 3x_3 &\ge 10,\\ 0\le x_1 &\le 2,\\ 0\le x_2 &\le 2,\\ 0\le x_3 &\le 3,\\ x_1,x_2,x_3 &\in \mathbb{Z}. \end{aligned}$$

6.5. Using the result of Exercise 3.4, solve the following 0,1-knapsack problem by the branch-and-bound method:

$$\begin{aligned} 16x_1+6x_2+14x_3+19x_4 &\to \max,\\ 6x_1+3x_2+\ 7x_3+\ 9x_4 &\le 13,\\ x_1,x_2,x_3,x_4 &\in \{0,1\}. \end{aligned}$$

6.6. Solve the next IP by the branch-and-cut method that at each node generates only fractional Gomory cuts:

$$\begin{aligned} 3x_1-\ \ x_2 &\to \max,\\ 3x_1-2x_2 &\le 3,\\ -5x_1-4x_2 &\le -10,\\ 2x_1+\ \ x_2 &\le 5,\\ 0\le x_1 &\le 2,\\ 0\le x_2 &\le 3,\\ x_1,x_2 &\in \mathbb{Z}. \end{aligned}$$

6.7. Apply probing to strengthen the formulation

$$\begin{aligned} 5x_1-8x_2-12x_3+3x_4-3x_5 &\to \max,\\ x_1+3x_2-\ 4x_3+2x_4+5x_5 &\le 0,\\ 3x_1+7x_2-\ 2x_3+2x_4+3x_5 &\ge 4,\\ -2x_1+\qquad\ 2x_3+\ x_4-\ x_5 &\ge 2,\\ x_1,x_2,x_3,x_4,x_5,x_6 &\in \{0,1\}. \end{aligned}$$

6.8. Prove that in the system

$$\begin{aligned} -10^6y+999995x_1+999995x_2+x_3+x_4-x_5-x_6-x_7-x_8 &\le -3,\\ y &\ge 1,\\ (y,x) &\in \mathbb{Z}\times\{0,1\}^8 \end{aligned}$$

its first inequality can be replaced with the following much simpler inequality

$$x_1 + x_2 - y \le 0.$$

Generalize your proof to propose a useful preprocessing technique.

6.9. *Chvátal-Gomory strengthening of inequalities*. Given an integer set $X = \{x \in \mathbb{Z}^n_+ : \sum_{j=1}^n a_j x_j \ge b\}$ with $a_j > 0$ for $j = 1, \dots, n$. Let $q > 0$ be a scalar such that

$$\lceil qa_j \rceil \times b / \lceil qb \rceil \le a_j, \quad j = 1, \dots, n,$$

and al least one of the above inequalities is strict. Prove that the inequality

$$\sum_{j=1}^n \lceil qa_j \rceil x_j \ge \lceil qb \rceil$$

is valid for X, and it is stronger than $\sum_{j=1}^n a_j x_j \ge b$.

6.10. Prove Proposition 6.4.

6.11. Prove that in Formulation (6.13) of the minimum Hamiltonian cycle problem, System (6.13c) of the subtour elimination inequalities can be replaced with the following more compact system

$$u_v - u_w + (n-1)x_{(v,w)} \le n-2, \quad (v,w) \in E,\ v,w \ne s, \tag{6.19}$$

where s is an arbitrary vertex from V, and u_v ($v \in V \setminus \{s\}$) are continuous variables. Give a meaningful interpretation for the variables u_v. Which of two systems, (6.13c) or (6.19), is stronger?

6.12. *Acyclic subgraph*. Given a digraph $G = (V,E)$, each arc $e \in E$ of which is assigned a weight w_e; we need to find an *acyclic subgraph* (having no directed cycles) $G' = (V,E')$ ($E' \subseteq E$) of the maximum weight $\sum_{e \in E'} w_e$. Let $\mathscr{C}_G$ denote the family of all directed cycles in G (more precisely, every element in $\mathscr{C}_G$ is the set of edges of some directed cycle in G). Introducing binary variables x_e for all $e \in E$ ($x_e = 1$ only if $e \in E'$), we can write down the following formulation:

$$\sum_{e \in E} w_e x_e \to \max, \tag{6.20a}$$

$$\sum_{e \in C} x_e \le |C| - 1, \quad C \in \mathscr{C}_G, \tag{6.20b}$$

$$x_e \in \{0,1\}, \quad e \in E. \tag{6.20c}$$

How can we solve the separation problem for Ineqs. (6.20b)?

Chapter 7
Branch-And-Price

If formulated briefly, then the *branch-and-price method* is a combination of the branch-and-bound method with the column generation approach, which is used for solving large LPs. In this chapter, we first consider the column generation algorithm applied to the one-dimensional cutting stock problem. Then we discuss the general scheme of the branch-and-price method and consider two specific application of this method for solving 1) the generalized assignment problem, and 2) the problem of designing a reliable telecommunication network.

7.1 Column Generation Algorithms

The column generation method is used for solving LPs with a large (usually exponentially large) number of columns (variables). Technically, this method is similar to the cutting plane method. A standard problem for demonstrating this method is the one-dimensional cutting stock problem.

7.1.1 One-Dimensional Cutting Stock Problem

Materials such as paper, textiles, cellophane and metal foil are produced in rolls of great length, from which short stocks are then cut out. For example, from a roll with a length of 1000 cm, we can cut out 20 stocks of 30 cm in length, and 11 stocks with a length of 36 cm, with 4 cm going to waste. When it is required to cut out many different types of stocks in different quantities, it is not always easy to find the most economical way (with the minimum amount of waste) to do this.

The problem of finding the most economical method of cutting is known as the *cutting stock problem*. In the simplest form, it is formulated as follows. From rolls of length L, we need to cut out pieces of length $l_1,\dots,l_m$, respectively, in the quantities $q_1,\dots,q_m$. Our goal is to use the minimum number of rolls.

A vector $a = (a_1, \dots, a_m)^T$ with non-negative integer components is called a *pattern* if $\sum_{i=1}^m a_i l_i \le L$, i.e., from a roll of length L one can simultaneously cut out a_1 stocks of length l_1, a_2 stocks of length l_2, and so on a_m stocks of length l_m. Let $a^j = (a_1^j, \dots, a_m^j)^T$, $j = 1, \dots, n$, be the set of all possible patterns, and let x_j be a variable which value is the number of rolls to be cut along the pattern a^j. Then the (one-dimensional) cutting stock problem is formulated as the next IP:

$$\begin{aligned} &\sum_{j=1}^n x_j \to \min, \\ &\sum_{j=1}^n a_i^j x_j \ge q_i, \quad i = 1, \dots, m, \\ &x_j \in \mathbb{Z}_+, \quad j = 1, \dots, n. \end{aligned} \tag{7.1}$$

Remark. This base model can be easily modified for the case when it is necessary to cut rolls of different lengths. When cutting expensive materials, such as silk, a more appropriate criterion is to minimize the cost of leftovers $\sum_{j=1}^n c_j x_j$, where c_j is the waste cost for the pattern a^j.

7.1.2 Column Generation Approach

It should be noted that the number of variables, n, in (7.1) may be astronomically large. Therefore, even the relaxation LP for (7.1) cannot be solved in the usual way. But such LPs with exponentially many variables (columns) can be solved using the *column-generation* technique which, in its essence, consists in the following. At the beginning, a relatively small family of patterns is selected — without loss of generality we may assume that these are the first k patterns, $a^1, \dots, a^k$, from our list of all patterns — so that the following *truncated LP*

$$\begin{aligned} &\sum_{j=1}^k x_j \to \min, \\ &\sum_{j=1}^k a_i^j x_j \ge q_i, \quad i = 1, \dots, m, \\ &x_j \ge 0, \quad j = 1, \dots, k, \end{aligned} \tag{7.2}$$

has a feasible solution. Let x^* and y^* be optimal primal and dual basic solutions to this LP. The solution x^* can be extended to the solution of the *full LP* (relaxation LP for (7.1)) if we set to zero the values of all variables x_j for $j = k+1, \dots, n$. Clearly, this extended solution is optimal to (7.1) if y^* is its optimal dual solution. By the complementary slackness condition (see item c) of Theorem 3.2), the latter is valid if all reduced costs are non-negative:

$$\bar{c}_j = 1 - \sum_{i=1}^{m} a_i^j y_i^* \ge 0, \quad j = 1, \dots, n. \tag{7.3}$$

To verify (7.3), it is enough to solve the following pricing problem:

$$\begin{aligned} &\sum_{i=1}^{m} y_i^* z_i \to \max, \\ &\sum_{i=1}^{m} l_i z_i \le L, \\ &z_i \in \mathbb{Z}_+, \ i = 1, \dots, m. \end{aligned} \tag{7.4}$$

which is an integer knapsack problem, and it can be solved by dynamic programming using the recurrence formula (1.29).

Let z^* be an optimal solution to (7.4). If $\sum_{i=0}^{m} y_i^* z_i^* \le 1$, then all inequalities in (7.3) are satisfied at z^*. Otherwise, since z^* is a pattern, we add to (7.2) a new column $a^{k+1} = z^*$ with a variable x_{k+1}, increment k by 1, and solve this extended truncated LP. We continue adding (generating) new columns until (7.3) is satisfied.

A commonly accepted approach for solving the cutting-stock IP (7.1) is to solve its relaxation LP and then round up the components of its optimal solution. For problems with a small number of stocks that need to be cut out in large quantities, this approach usually gives a near optimum solution. Alternatively, after solving the relaxation LP, we can continue solving the problem by the branch-and-cut algorithm without generating columns.

7.1.3 Finding a Good Initial Solution

When applying a column generation algorithm, it is highly desirable to start with a such truncated LP which objective value is close to the optimal objective value of the full LP. Here we present a heuristic that builds a reasonably good set of patterns. This heuristic is based on the assumption that long stocks are cut out first, and short stocks are cut out from leftovers.

Initialization. List stocks in decreasing order of their lengths:

$$l_{\pi(1)} > l_{\pi(2)} > \cdots > l_{\pi(m)}.$$

Set $b = q$, $I = (\pi(1), \dots, \pi(m))$, $k = 0$.

General step. While $I \ne \emptyset$, do the following.

```
Set $k := k+1$, $W = L$, $x_k = \infty$. For all $i \notin I$, set $a_i^k = 0$.
For $i = 1, \dots, |I|$,
    set $a^k_{I[i]} = \left\lfloor \frac{W}{l_{I[i]}} \right\rfloor$, $W := W - a^k_{I[i]} \cdot l_{I[i]}$;
```

if $x_k a^k_{I[i]} > b_{I[i]}$, set $x_k = \left\lceil \frac{b_{I[i]}}{a^k_{I[i]}} \right\rceil$.
Set $b := b - x_k a^k$ and remove from I all stocks s such that $b_s \le 0$, preserving the order of the elements that are left.

7.1.4 Cutting Stock Example

To illustrate how the column generation algorithm works, let us apply it to solve an instance of the cutting-stock problem with the following numeric parameters: $L = 100$, $l_1 = 45$, $l_2 = 36$, $l_3 = 31$, $l_4 = 14$, $q_1 = 97$, $q_2 = 610$, $q_3 = 395$, $q_4 = 211$.

First, let us apply the heuristic from Sect. 7.1.3 to find an initial set of patterns.

Initialization. Set $b = (97, 610, 395, 211)$, $I = (1, 2, 3, 4)$, $k = 0$.

Step 1. Set $W = 100$, $x_1 = \infty$. Compute in sequence

$$a^1_1 = \left\lfloor \frac{100}{45} \right\rfloor = 2, \quad W = 100 - 2 \cdot 45 = 10, \quad x_1 = \left\lceil \frac{97}{2} \right\rceil = 49;$$
$$a^1_2 = \left\lfloor \frac{10}{36} \right\rfloor = 0; \quad a^1_3 = \left\lfloor \frac{10}{31} \right\rfloor = 0; \quad a^1_4 = \left\lfloor \frac{10}{14} \right\rfloor = 0.$$

Set $b = (97, 610, 395, 211) - 49(2, 0, 0, 0) = (-1, 610, 395, 211)$, $\quad I = (2, 3, 4)$.

Step 2. Set $W = 100$, $x_2 = \infty$, $a^2_1 = 0$. Compute in sequence

$$a^2_2 = \left\lfloor \frac{100}{36} \right\rfloor = 2; \quad W = 100 - 2 \cdot 36 = 28, \quad x_2 = \left\lceil \frac{610}{2} \right\rceil = 305;$$
$$a^2_3 = \left\lfloor \frac{28}{31} \right\rfloor = 0;$$
$$a^2_4 = \left\lfloor \frac{28}{14} \right\rfloor = 2, \quad W = 28 - 2 \cdot 14 = 0, \quad x_2 = \left\lceil \frac{211}{2} \right\rceil = 106.$$

Set $b = (-1, 610, 395, 211) - 106(0, 2, 0, 2) = (-1, 398, 395, -1)$, $I = (2, 3)$.

Step 3. Set $W = 100$, $x_3 = \infty$, $a^3_1 = 0$, $a^3_4 = 0$. Compute in sequence

$$a^3_2 = \left\lfloor \frac{100}{36} \right\rfloor = 2; \quad W = 100 - 2 \cdot 36 = 28, \quad x_3 = \left\lceil \frac{398}{2} \right\rceil = 199;$$
$$a^3_3 = \left\lfloor \frac{28}{31} \right\rfloor = 0.$$

Set $b = (-1, 398, 395, -1) - 199(0, 2, 0, 0) = (-1, 0, 395, -1)$, $I = (3)$.

Step 4. Set $W = 100$, $x_4 = \infty$, $a^4_1 = 0$, $a^4_2 = 0$, $a^4_4 = 0$. Compute

$$a_3^4 = \left\lfloor \frac{100}{31} \right\rfloor = 3; \quad W = 100 - 3 \cdot 31 = 7, \quad x_4 = \left\lceil \frac{395}{3} \right\rceil = 132.$$

Set $b = (-1,0,395,-1) - 132(0,0,3,0) = (-1,0,-1,-1)$, $I = \emptyset$.

Having an initial set of patterns, we can continue solving our cutting stock example by the column generation algoritm. First, we write down the truncated LP based on the patterns a^j, $j = 1,\dots,4$:

$$\begin{aligned}
x_1 + \ x_2 + \ x_3 + \ x_4 &\to \min,\\
2x_1 \qquad\qquad\qquad &\ge 97,\\
2x_2 + 2x_3 \qquad &\ge 610,\\
3x_4 &\ge 395,\\
2x_2 \qquad\qquad &\ge 211,\\
x_1, x_2, x_3, x_4 &\ge 0.
\end{aligned} \tag{7.5}$$

Its optimal primal and dual solutions are, respectively, the following vectors:

$$x^1 = \left(48\frac{1}{2}, 105\frac{1}{2}, 199\frac{1}{2}, 131\frac{2}{3}\right)^T, \quad y^1 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{3}, 0\right)^T.$$

Next we write down the pricing problem:

$$\begin{aligned}
\frac{1}{2}z_1 + \ \frac{1}{2}z_2 + \ \frac{1}{3}z_3 + \ 0z_4 &\to \max,\\
45z_1 + 36z_2 + 31z_3 + 14z_4 &\le 100,\\
z_1, z_2, z_3, z_4 &\in \mathbb{Z}_+.
\end{aligned}$$

The vector $z^1 = (0,1,2,0)^T$ is its optimal solution. Since $(y^1)^T z^1 = 1/2 + 2/3 = 7/6 > 1$, then the column $a^5 = z^1$ is added to (7.5), and, as a result, we get the following LP:

$$\begin{aligned}
x_1 + \ x_2 + \ x_3 + \ x_4 + \ x_5 &\to \min,\\
2x_1 \qquad\qquad\qquad\qquad &\ge 97,\\
2x_2 + 2x_3 + \qquad\quad x_5 &\ge 610,\\
3x_4 + 2x_5 &\ge 395,\\
2x_2 \qquad\qquad\qquad &\ge 211,\\
x_1, x_2, x_3, x_4, x_5 &\ge 0.
\end{aligned}$$

After reoptimizing, we obtain the following optimal primal and dual solutions to the above LP:

$$x^2 = \left(48\frac{1}{2}, 105\frac{1}{2}, 100\frac{3}{4}, 0, 197\frac{1}{2}\right)^T, \quad y^2 = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{4}, 0\right)^T.$$

Now we solve the next pricing problem:

$$\begin{aligned} \frac{1}{2}z_1 + \frac{1}{2}z_2 + \frac{1}{4}z_3 + 0z_4 &\to \max,\\ 45z_1 + 36z_2 + 31z_3 + 14z_4 &\le 100,\\ z_1, z_2, z_3, z_4 &\in \mathbb{Z}_+ . \end{aligned}$$

The point $z^2 = (1,1,0,0)$ is its optimal solution. Since $(y^2)^T z^2 = 1$, then x^2 determines an optimal solution to the full relaxation LP.

Rounding up the solution x^2, we get the following approximate solution of our example: cut out 49 rolls according to the pattern $(2,0,0,0)$, 106 rolls according to the pattern $(0,2,0,2)$, 101 according to the pattern $(0,2,0,0)$, and 198 rolls according to the pattern $(0,1,2,0)$. In this case, 454 rolls are used in total.

Since any cutting plan must use at least

$$\lceil x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \rceil = \left\lceil 452\frac{1}{4} \right\rceil = 453$$

rolls, then if there is a more economical way of cutting, it will save only one roll. As a rule, a cutting plan obtained by rounding up a solution to the relaxation LP is very "close" to the optimal ones. Therefore, in practice, we can almost always limit ourselves to the search for such approximate cutting plans. □

7.2 Dancig-Wolfe Reformulation

Danzig-Wolfe decomposition was originally developed (in 1960) for solving large structured LPs. Much later in the 1980-th this method began to be used to reformulate MIPs in order to strengthen them.

Consider the IP in the following form:

$$\begin{aligned} &\sum_{k=1}^{K} (c^k)^T x^k \to \max,\\ &\sum_{k=1}^{K} A^k x^k \le b, \qquad (7.6)\\ &x^k \in X^k, \quad k = 1,\dots,K, \end{aligned}$$

where $b \in \mathbb{R}^m$, and, for $k = 1,\dots,K$, $c^k \in \mathbb{R}^{n_k}$, A^k is a real $m \times n_k$-matrix, $X^k \subset \mathbb{Z}^{n_k}$ is a finite set.

We can write

$$X^k = \left\{ x^k = \sum_{a \in X^k} \lambda_a^k a : \sum_{a \in X^k} \lambda_a^k = 1,\ \lambda_a^k \in \{0,1\} \text{ for } a \in X^k \right\}.$$

Substituting the expressions for x^k into (7.6), we have

$$\begin{aligned}
&\sum_{k=1}^{K} \sum_{a\in X^k} (a^T c^k)\lambda_a^k \to \max,\\
&\sum_{k=1}^{K} \sum_{a\in X^k} (A^k a)\lambda_a^k \le b,\\
&\qquad\quad \sum_{a\in X^k} \lambda_a^k = 1, \qquad\qquad k=1,\dots,K,\\
&\qquad\qquad \lambda_a^k \in \mathbb{Z}_+, \qquad a\in X^k,\ k=1,\dots,K.
\end{aligned} \tag{7.7}$$

A natural question arises: what gives us the transition from Formulation (7.6), which is compact, to Formulation (7.7) with a huge number of variables (columns)? One of the advantages is that, in cases where each of the sets X^k is a set of integer points of a polyhedron, (7.7) is usually stronger than (7.6), since only points x^k from $\mathrm{conv}(X^k)$ are feasible to the relaxation LP for (7.7). For example, if

$$X^k = \{x\in\mathbb{Z}^2:\ 3x_1+2x_2\le 4,\ 0\le x_1,x_2\le 1\},$$

then the point $\bar{x}=\left(1,\frac{1}{2}\right)$, which satisfies all inequalities defining X^k, does not belong to the convex hull of all integer points, $(0,0)$, $(1,0)$ and $(0,1)$, from X^k.

Let z_{LPM} denote the optimal objective value of the relaxation LP for (7.7). It is not difficult to see that the following equality holds

$$\begin{aligned}
z_{LPM} = z_{CUT} \overset{\text{def}}{=} \max &\sum_{k=1}^{K} (c^k)^T x^k,\\
&\sum_{k=1}^{K} A^k x^k \le b,\\
&x^k \in \mathrm{conv}(X^k), \quad k=1,\dots,K.
\end{aligned}$$

This means that the branch-and-bound method applied to (7.7) is equivalent (from the point of view of the accuracy of upper bounds) to the the branch-and-cut method applied to (7.6) assuming that this method uses exact separation procedures for the sets $\mathrm{conv}(X^k)$.

7.2.1 Master and Pricing Problems

The number of variables in (7.7) can be astronomically large. Therefore, its relaxation LP can only be solved by a column generation algorithm. First, relatively small subsets $S^k\subseteq X^k$ are chosen and the *master problem* is solved:

$$
\begin{aligned}
&\sum_{k=1}^{K}\sum_{a\in S^k}(a^T c^k)\lambda_a^k \to \max,\\
&\sum_{k=1}^{K}\sum_{a\in S^k}(A^k a)\lambda_a^k \le b,\\
&\qquad\sum_{a\in S^k}\lambda_a^k = 1, \quad k=1,\dots,K,\\
&\qquad\lambda_a^k \ge 0, \quad a\in S^k,\ k=1,\dots,K.
\end{aligned}
\tag{7.8}
$$

Let $(y,v)\in\mathbb{R}_+^m\times\mathbb{R}^K$ be an optimal dual solution to (7.8). The nonzero components of an optimal primal solution to (7.8) are nonzero components of an optimal solution to the relaxation LP for (7.7) if the inequalities

$$y^T A^k a + v_k \ge a^T c^k, \quad a\in X^k,\ k=1,\dots,K, \tag{7.9}$$

hold. To verify this condition, for each $k=1,\dots,K$, we need to solve the following *pricing problem*

$$
\begin{aligned}
&((c^k)^T - y^T A^k)x^k \to \max,\\
&x^k \in X^k.
\end{aligned}
\tag{7.10}
$$

If for some k the optimal objective value in (7.10) is greater than v_k, then an optimal solution to (7.10), $\bar{x}^k$, is added to the set S^k. Next we solve the extended master LP. We continue to act in this way until an optimal solution to the relaxation LP for (7.7) is found.

7.2.2 Branching

Applying the column generation technique makes it difficult (or even impossible) to use the conventional branching on integer variables in the branch-and-bound method. For example, if at a particular node of the search tree we set some variable λ_a^k to zero, then for an optimal solution to the relaxation LP at this node the reduced cost of λ_a^k can be positive, and this makes it possible that the column corresponding to λ_a^k can be an optimal solution to the pricing problem. To prevent this, an additional constraint must be added to the pricing problem. As a result, even at the search tree nodes of low height, initially a relatively easy to solve pricing problem can turn into a difficult IP.

Everything is greatly simplified for problems involving only binary variables. The point $\tilde{x}^k = \sum_{a\in S^k}\lambda_a^k a$ belongs to $\{0,1\}^{n_k}$ if and only if all λ_a^k are integer-valued. Therefore, if there are variables λ_a^k taking fractional values, then $\tilde{x}^k$ has fractional components $\tilde{x}_j^k$, one of which can be chosen for branching on it.

When processing a branch $x_j^k = \alpha$ ($\alpha\in\{0,1\}$), all elements a with $a_j = 1-\alpha$ must be removed from the set S^k. In addition, we need to exclude the generation of such elements (columns) when solving the corresponding pricing problem. The latter can be done by setting $x_j^k = \alpha$ in the pricing problem. If $\alpha = 1$, then we also need to exclude from the other sets $S^{\bar{k}}$ $(\bar{k} \neq k)$ all elements a with $a_j = 1$. In the pricing problems it is necessary to set $x_j^{\bar{k}} = 0$ for all $\bar{k} \neq k$. Note that the addition of such simple constraints usually does not destroy the structure of each pricing problem.

This combination of the branch-and-bound and column generation methods is known as the *branch-and-price method*.

7.3 Generalized Assignment Problem

The generalized assignment problem can be considered as the following parallel machine scheduling problem. Each of m independent tasks must be processed without interruptions by one of K parallel machines; it takes p_i^k units of time and costs c_i^k if task i is processed on machine k $(i = 1,\dots,m,\ k = 1,\dots,K)$. The *load* (total running time) of machine k must not exceed its *capacity* l_k. A *schedule* is represented by an $m \times K$-matrix $X = \{x_i^k\}$ with $x_i^k = 1$ if task i is assigned (for processing) to machine k. The *generalized assignment problem* (*GAP*) is to find such a schedule that admits all the above requirements and has the minimum total assignment cost. It is formulated as follows:

$$\sum_{k=1}^{K}\sum_{i=1}^{m} c_i^k x_i^k \to \min, \tag{7.11a}$$

$$\sum_{k=1}^{K} x_i^k = 1, \quad i = 1,\dots,m, \tag{7.11b}$$

$$\sum_{i=1}^{m} p_i^k x_i^k \le l_k, \quad k = 1,\dots,K, \tag{7.11c}$$

$$x_i^k \in \{0,1\}, \quad i = 1,\dots,m,\ k = 1,\dots,K. \tag{7.11d}$$

Objective (7.11a) is to minimize the total assignment cost. Equations (7.11b) require that each task be assigned to exactly one machine. Inequalities (7.11c) impose the *capacity restrictions*.

Setting

$$X^k = \{z \in \{0,1\}^m : \sum_{i=1}^{m} p_i^k z_i \le l_k\},$$

we can reformulate (7.11) in the following way:

$$\sum_{k=1}^{K} \sum_{a \in X^k} \left(\sum_{i=1}^{m} c_i^k a_i \right) \lambda_a^k \to \min, \tag{7.12a}$$

$$\sum_{k=1}^{K} \sum_{a \in X^k} a \lambda_a^k = \mathbf{e}, \tag{7.12b}$$

$$\sum_{a \in X^k} \lambda_a^k \le 1, \quad k = 1, \dots, K, \tag{7.12c}$$

$$\lambda_a^k \in \{0,1\}, \quad a \in X^k,\ k = 1, \dots, K. \tag{7.12d}$$

Here $\mathbf{e}$ denotes the m-vector of all ones. Observe that we write down (7.12c) as inequalities (instead of equations in accordance with (7.7)) since X^k contains the point $\mathbf{0} \in \mathbb{R}^m$.

7.3.1 Master Problem

For $k = 1, \dots, K$, let $S^k \subseteq X^k$. To simplify the construction of an initial master problem, as well as the processing of infeasible subproblems, we introduce slack variables s_i ($i = 1, \dots, m$) to formulate the master problem in the following way:

$$-\sum_{k=1}^{K} \sum_{a \in S^k} \left(\sum_{i=1}^{m} c_i^k a_i \right) \lambda_a^k - M \sum_{i=1}^{m} s_i \to \max, \tag{7.13a}$$

$$\sum_{k=1}^{K} \sum_{a \in S^k} a \lambda_a^k + s = \mathbf{e}, \tag{7.13b}$$

$$\sum_{a \in S^k} \lambda_a^k \le 1, \quad k = 1, \dots, K, \tag{7.13c}$$

$$\lambda_a^k \ge 0, \quad a \in S^k,\ k = 1, \dots, K, \tag{7.13d}$$

$$s_i \ge 0, \quad i = 1, \dots, m, \tag{7.13e}$$

where M is a sufficiently big number, for example,

$$M \ge \sum_{i=1}^{m} \max_{1 \le k \le K} c_i^k.$$

Introducing slack variables also allows us to start with the simplest master problem when all $S^k = \emptyset$.

7.3.2 Pricing Problem

Let $(\tilde{y},\tilde{v}) \in \mathbb{R}^m \times \mathbb{R}_+^K$ be an optimal dual solution to (7.13). Here the dual variable y_i corresponds to the i-th equation in (7.13b), and the dual variable v_k corresponds to the k-th inequality in (7.13c). For $k = 1,\dots,K$, the *pricing problem* is the following 0,1-knapsack problem:

$$\begin{aligned} &\sum_{i=1}^m (-c_i^k - \tilde{y}_i) z_i \to \max,\\ &\sum_{i=1}^m p_i^k z_i \le l_k, \\ &z_i \in \{0,1\}, \quad i = 1,\dots,m. \end{aligned} \tag{7.14}$$

This pricing problem can be solved by the recurrence formula (1.30).

Let $z^* \in \{0,1\}^m$ be an optimal solution to (7.14). If the inequality

$$\sum_{i=1}^m (-c_i^k - \tilde{y}_i) z_i^* > \tilde{v}_k$$

holds, then z^* is added to the set S^k, and thereby the column$(z^*, \mathbf{e}_k) \in \mathbb{R}^m \times \mathbb{R}^K$ must be added to the master problem. The objective coefficient for this column is $-\sum_{i=1}^m c_i^k a_i$.

7.3.3 Branching

Branching on the variables λ_a^j is not efficient due to two reasons. First, such branching results in a non-balanced search tree, since the branch $\lambda_a^j = 0$ excludes only one column while the branch $\lambda_a^j = 1$ excludes all columns that have 1 at least in one row i such that $a_i = 1$. Second, such branching violates the structure of the pricing problem (see Sect. 7.3.2).

As we have already noted, if in an optimal solution of the master LP, $\tilde{\lambda}$, one of the values $\tilde{\lambda}_a^k$ is non-integer, then the vector

$$\tilde{x}^k = \sum_{a \in S^k} \tilde{\lambda}_a^k a$$

has non-integer components as well. Let δ_q be the minimum of the values

$$\delta_k = \left(1 - \sum_{a \in S^k} \tilde{\lambda}_a^k\right)^2 + \sum_{a \in S^k} \left(\tilde{\lambda}_a^k\right)^2, \quad k = 1,\dots,K.$$

Among the variables x_i^q, we choose for branching a variable x_r^q whose current value $\tilde{x}_r^q$ is closest to 0.5.

When implementing this branch-and-price algorithm, at each node of the search tree, it is necessary to store a list of variables x_i^k together with their fixed values. These data allow us to correctly modify the 0,1-knapsack pricing problems. More precisely, when solving at some node the pricing problem for machine k, if a variable x_i^k is fixed to α, the variable z_i must be set to α. In addition, for all $\bar{k} \neq k$ and for all variables $x_i^{\bar{k}}$ fixed to 1, we also need to set $z_i = 0$.

7.3.4 Example

After we have specified all the elements of the branch-and-price method, let us apply it for solving a small example.

Example 7.1 *We need to solve an instance of GAP in which three ($m = 3$) tasks must be performed on two machines ($K = 2$) with the following parameters:*

$$[c_i^k] = \begin{bmatrix} 1 & 2 \\ 2 & 2 \\ 2 & 1 \end{bmatrix}, \quad [p_i^k] = \begin{bmatrix} 3 & 2 \\ 2 & 2 \\ 2 & 3 \end{bmatrix}, \quad l = \begin{pmatrix} 4 \\ 4 \end{pmatrix}.$$

Solution. We start with $S^1 = \emptyset$, $S^2 = \emptyset$ and $M = 6$. The search tree processed by the branch-and price method is depicted in Fig. 7.1. Below are presented the steps performed by the method to solve this example: step $i.j$ describes iteration j of the column generation algorithm when processing node i of the search tree.

0.1. Solve the initial master LP:

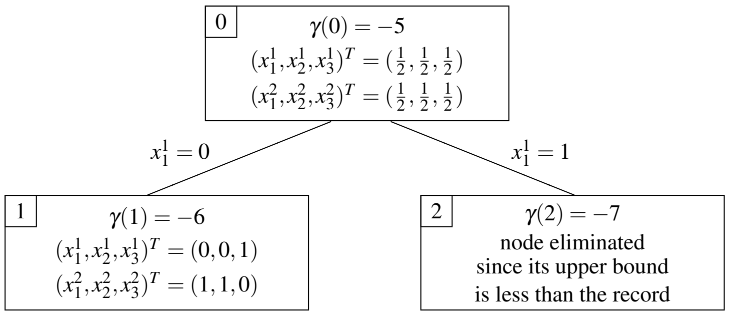
0
$\gamma(0) = -5$
$(x_1^1, x_2^1, x_3^1)^T = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
$(x_1^2, x_2^2, x_3^2)^T = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$
$x_1^1 = 0$
$x_1^1 = 1$
1
$\gamma(1) = -6$
$(x_1^1, x_2^1, x_3^1)^T = (0,0,1)$
$(x_1^2, x_2^2, x_3^2)^T = (1,1,0)$
2
$\gamma(2) = -7$
node eliminated
since its upper bound
is less than the record

Fig. 7.1 Search tree for Example 7.1

$$\begin{aligned} -6s_1 - 6s_2 - 6s_3 &\to \max,\\ s_1 \qquad\qquad &= 1,\\ s_2 \qquad &= 1,\\ s_3 &= 1,\\ 0 &\le 1,\\ 0 &\le 1,\\ s_1, s_2, s_3 &\ge 0. \end{aligned}$$

Its optimal dual solution is given by $y = (-6,-6,-6)$ and $v_1 = v_2 = 0$.

Now we solve two pricing problems. For machine 1, we solve the following 0,1-knapsack problem:

$$\begin{aligned} \xi = 5z_1 + 4z_2 + 4z_3 &\to \max,\\ 3z_1 + 2z_2 + 2z_3 &\le 4,\\ z_1, z_2, z_3 &\in \{0,1\}. \end{aligned}$$

The point $z^* = (0,1,1)^T$ and $\xi^* = 8$ are its optimal solution and objective value. Since $\xi^* = 8 > 0 = v_1$, we add z^* to the set S^1: $S^1 = \{(0,1,1)^T\}$.

For machine 2, we solve the next 0,1-knapsack problem:

$$\begin{aligned} \xi = 4z_1 + 4z_2 + 5z_3 &\to \max,\\ 2z_1 + 2z_2 + 3z_3 &\le 4,\\ z_1, z_2, z_3 &\in \{0,1\}. \end{aligned}$$

Its optimal solution and objective value are $z^* = (1,1,0)^T$ and $\xi^* = 8$. Since $\xi^* = 8 > 0 = v_2$, we add z^* to the set S^2: $S^2 = \{(1,1,0)^T\}$.

0.2. Solve the next extended master LP:

$$\begin{aligned} -4\lambda_1^1 - 4\lambda_1^2 - 6s_1 - 6s_2 - 6s_3 &\to \max,\\ \lambda_1^2 + s_1 \qquad\qquad &= 1,\\ \lambda_1^1 + \lambda_1^2 + \qquad s_2 \qquad &= 1,\\ \lambda_1^1 + \qquad\qquad\qquad s_3 &= 1,\\ \lambda_1^1 \qquad\qquad\qquad &\le 1,\\ \lambda_1^2 \qquad\qquad &\le 1,\\ \lambda_1^1, \lambda_1^2, s_1, s_2, s_3 &\ge 0. \end{aligned}$$

Its optimal dual solution is given by $y = (-6,2,-6)$ and $v_1 = v_2 = 0$.

Now let us solve the pricing problems. For machine 1, we solve the following 0,1-knapsack problem:

$$\begin{aligned} \xi = 5z_1 - 4z_2 + 4z_3 &\to \max,\\ 3z_1 + 2z_2 + 2z_3 &\le 4,\\ z_1, z_2, z_3 &\in \{0,1\}. \end{aligned}$$

Its optimal solution and objective value are $z^* = (1,0,0)^T$ and $\xi^* = 5$. Since $\xi^* = 5 > 0 = v_1$, we add z^* to S^1: $S^1 = \{(0,1,1)^T, (1,0,0)^T\}$.

For machine 2, we solve the next 0,1-knapsack problem:

$$\begin{aligned} \xi = 4z_1 - 4z_2 + 5z_3 &\to \max, \\ 2z_1 + 2z_2 + 3z_3 &\le 4, \\ z_1, z_2, z_3 &\in \{0,1\}. \end{aligned}$$

Its optimal solution and objective value are $z^* = (0,0,1)^T$ and $\xi^* = 5$. Since $\xi^* = 5 > 0 = v_2$, we add z^* to S^2: $S^2 = \{(1,1,0)^T, (0,0,1)^T\}$.

0.3. We need to solve one more master LP:

$$\begin{array}{l} -4\lambda_1^1 - \lambda_2^1 - 4\lambda_1^2 - \lambda_2^2 - 6s_1 - 6s_2 - 6s_3 \to \max, \\ \qquad\;\; \lambda_2^1 + \lambda_1^2 + \qquad s_1 \qquad\qquad\quad = 1, \\ \lambda_1^1 + \qquad\; \lambda_1^2 + \qquad\qquad s_2 \qquad\; = 1, \\ \lambda_1^1 + \qquad\qquad\; \lambda_2^2 + \qquad\qquad\quad s_3 = 1, \\ \lambda_1^1 + \lambda_2^1 \qquad\qquad\qquad\qquad\qquad\quad \le 1, \\ \qquad\qquad\; \lambda_1^2 + \lambda_2^2 \qquad\qquad\qquad\quad \le 1, \\ \qquad\qquad\quad \lambda_1^1, \lambda_2^1, \lambda_1^2, \lambda_2^2, s_1, s_2, s_3 \ge 0. \end{array}$$

Its optimal dual solution is given by $y = (-1,-3,-1)$ and $v_1 = v_2 = 0$.

Now we solve the pricing problems. For machine 1, we solve the following 0,1-knapsack problem:

$$\begin{aligned} \xi = 0z_1 + z_2 - z_3 &\to \max, \\ 3z_1 + 2z_2 + 2z_3 &\le 4, \\ z_1, z_2, z_3 &\in \{0,1\}. \end{aligned}$$

Its optimal solution and objective value are $z^* = (0,1,0)^T$ and $\xi^* = 1$. Since $\xi^* = 1 > 0 = v_1$, we add z^* to S^1: $S^1 = \{(0,1,1)^T, (1,0,0)^T, (0,1,0)^T\}$.

For machine 2, we solve the next 0,1-knapsack problem:

$$\begin{aligned} \xi = -z_1 + z_2 + 0z_3 &\to \max, \\ 2z_1 + 2z_2 + 3z_3 &\le 4, \\ z_1, z_2, z_3 &\in \{0,1\}. \end{aligned}$$

Its optimal solution and objective value are $z^* = (0,1,0)^T$ and $\xi^* = 1$. Since $\xi^* = 1 > 0 = v_2$, we add z^* to S^2: $S^2 = \{(1,1,0)^T, (0,0,1)^T, (0,1,0)^T\}$.

0.4. Solve the next master LP:

$$\begin{array}{rl}
-4\lambda_1^1-\lambda_2^1-2\lambda_3^1-4\lambda_1^2-\lambda_2^2-2\lambda_3^2-6s_1-6s_2-6s_3 & \to \max,\\
\lambda_2^1+\lambda_1^2+s_1 & =1,\\
\lambda_1^1+\lambda_3^1+\lambda_1^2+\lambda_3^2+s_2 & =1,\\
\lambda_1^1+\lambda_2^2+s_3 & =1,\\
\lambda_1^1+\lambda_2^1+\lambda_3^1 & \le 1,\\
\lambda_1^2+\lambda_2^2+\lambda_3^2 & \le 1,\\
\lambda_1^1,\lambda_2^1,\lambda_3^1,\lambda_1^2,\lambda_2^2,\lambda_3^2,s_1,s_2,s_3 & \ge 0.
\end{array}$$

The non-zero components of its optimal primal and dual solutions are the following:

$$\begin{gathered}
\lambda_1^1=\lambda_2^1=\lambda_1^2=\lambda_2^2=\frac{1}{2},\\
y=(-2,-3,-2),\ v_1=v_2=1.
\end{gathered} \tag{7.15}$$

Next we solve the pricing problems. For machine 1, we solve the following 0,1-knapsack problem:

$$\begin{aligned}
&\xi=z_1+z_2+0z_3\to\max,\\
&3z_1+2z_2+2z_3\le 4,\\
&z_1,z_2,z_3\in\{0,1\}.
\end{aligned}$$

Its optimal solution and optimal objective value are $z^*=(1,0,0)^T$ and $\xi^*=1$. Since $\xi^*=1=v_1$, we cannot extend the set S^1.

For machine 2, we solve the next 0,1-knapsack problem:

$$\begin{aligned}
&\xi=0z_1+z_2+z_3\to\max,\\
&2z_1+2z_2+3z_3\le 4,\\
&z_1,z_2,z_3\in\{0,1\}.
\end{aligned}$$

Its optimal solution and optimal objective value are $z^*=(0,1,0)^T$ and $\xi^*=1$. Since $\xi^*=1=v_2$, we cannot extend the set S^2.

We were not able to extend both sets S^1 and S^2. This means that the current solution given by (7.15) is optimal for the root node LP. Knowing the values of the variables λ_j^k, we calculate the values of the variables x_i^k:

$$\begin{pmatrix}x_1^1\\x_2^1\\x_3^1\end{pmatrix}=\frac{1}{2}\cdot\begin{pmatrix}0\\1\\1\end{pmatrix}+\frac{1}{2}\cdot\begin{pmatrix}1\\0\\0\end{pmatrix}=\begin{pmatrix}\frac{1}{2}\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix},$$

$$\begin{pmatrix}x_1^2\\x_2^2\\x_3^2\end{pmatrix}=\frac{1}{2}\cdot\begin{pmatrix}1\\1\\0\end{pmatrix}+\frac{1}{2}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix}=\begin{pmatrix}\frac{1}{2}\\\frac{1}{2}\\\frac{1}{2}\end{pmatrix}.$$

Now we select the variable x_1^1, with a non-integer value of $\frac{1}{2}$, to branch on it at the root node.

1.1. Since $x_1^1 = 0$ at node 1, then all vectors a with $a_1 = 1$ must be excluded from the set S^1 that is inherited from the parent node 0. As a result, we have

$$S^1 = \{(0,1,1)^T, (0,1,0)^T\}.$$

In addition, for this node and all its descendants, when setting the pricing problems for machine 1, it will always be necessary to set $z_1 = 0$. The set S^2 remains the same as that of the parent node[1]: $S^2 = \{(1,1,0)^T, (0,0,1)^T, (0,1,0)^T\}$.

Now we solve the following master LP:

$$\begin{aligned}
-4\lambda_1^1 - 2\lambda_2^1 - 4\lambda_1^2 - \lambda_2^2 - 2\lambda_3^2 - 6s_1 - 6s_2 - 6s_3 &\to \max,\\
\lambda_1^2 + s_1 &= 1,\\
\lambda_1^1 + \lambda_2^1 + \lambda_1^2 + \lambda_3^2 + s_2 &= 1,\\
\lambda_1^1 + \lambda_2^2 + s_3 &= 1,\\
\lambda_1^1 + \lambda_2^1 &\le 1,\\
\lambda_1^2 + \lambda_2^2 + \lambda_3^2 &\le 1,\\
\lambda_1^1, \lambda_2^1, \lambda_1^2, \lambda_2^2, \lambda_3^2, s_1, s_2, s_3 &\ge 0.
\end{aligned}$$

Its optimal dual solution is given by $y = (-6, -\frac{1}{2}, -\frac{7}{2})$ and $v_1 = 0$, $v_2 = \frac{5}{2}$.

Next we solve the pricing problems. For machine 1, we solve the following 0,1-knapsack problem:

$$\begin{gathered}
\xi = 5z_1 - \frac{3}{2}z_2 + \frac{3}{2}z_3 \to \max,\\
3z_1 + 2z_2 + 2z_3 \le 4,\\
z_1 = 0,\ z_2, z_3 \in \{0,1\}.
\end{gathered}$$

Its optimal solution and objective value are $z^* = (0,0,1)^T$ and $\xi^* = \frac{3}{2}$. Since $\xi^* = \frac{3}{2} > 0 = v_1$, we add z^* to S^1: $S^1 = \{(0,1,1)^T, (0,1,0)^T, (0,0,1)^T\}$.

For machine 2, we solve the next 0,1-knapsack problem:

$$\begin{gathered}
\xi = 4z_1 - \frac{3}{2}z_2 + \frac{5}{2}z_3 \to \max,\\
2z_1 + 2z_2 + 3z_3 \le 4,\\
z_1, z_2, z_3 \in \{0,1\}.
\end{gathered}$$

Its optimal solution and objective value are $z^* = (1,0,0)^T$ and $\xi^* = 4$. Since $\xi^* = 4 > \frac{5}{2} = v_2$, we add z^* to S^2: $S^2 = \{(1,1,0)^T, (0,0,1)^T, (0,1,0)^T, (1,0,0)^T\}$.

1.2. Now we need to solve the next master LP:

[1] Since we only have two machines, task 1 must be processed by machine 2. Therefore, we could remove from the set S^2 all vectors a with $a_1 = 0$. These are the second and third vectors. But we do not do this, because we do not want to use any specifics of this concrete example.

$$
\begin{array}{rrrrrrrrrrl}
-4\lambda_1^1 - 2\lambda_2^1 - 2\lambda_3^1 & -4\lambda_1^2 - \lambda_2^2 - 2\lambda_3^2 - 2\lambda_4^2 & -6s_1 - 6s_2 - 6s_3 & \to \max, \\
& \lambda_1^2 + \qquad\qquad \lambda_4^2 + & s_1 & = 1, \\
\lambda_1^1 + \lambda_2^1 + \qquad & \lambda_1^2 + \quad \lambda_3^2 + \qquad & s_2 & = 1, \\
\lambda_1^1 + \qquad \lambda_3^1 + & \lambda_2^2 + \qquad\qquad & s_3 & = 1, \\
\lambda_1^1 + \lambda_2^1 + \lambda_3^1 & & & \le 1, \\
& \lambda_1^2 + \lambda_2^2 + \lambda_3^2 + \lambda_4^2 & & \le 1, \\
& \lambda_1^1, \lambda_2^1, \lambda_3^1, \lambda_1^2, \lambda_2^2, \lambda_3^2, \lambda_4^2, s_1, s_2, s_3 & & \ge 0.
\end{array}
$$

The non-zero components of its optimal primal and dual solutions are the following:

$$
\begin{aligned}
\lambda_3^1 = \lambda_1^2 = 1, \\
y = (-3,-2,-2),\ v_1 = 0,\ v_2 = 1.
\end{aligned}
\tag{7.16}
$$

Next we solve the pricing problems. For machine 1, we solve the following 0,1-knapsack problem:

$$
\begin{aligned}
\xi = 2z_1 + 0z_2 + 0z_3 \to \max, \\
3z_1 + 2z_2 + 2z_3 \le 4, \\
z_1 = 0,\ z_2, z_3 \in \{0,1\}.
\end{aligned}
$$

Its optimal solution and objective value are $z^* = (0,0,0)^T$ and $\xi^* = 0$. Since $\xi^* = 0 = v_1$, we cannot extend the set S^1.

For machine 2, we solve the next 0,1-knapsack problem:

$$
\begin{aligned}
\xi = z_1 + 0z_2 + z_3 \to \max, \\
2z_1 + 2z_2 + 3z_3 \le 4, \\
z_1, z_2, z_3 \in \{0,1\}.
\end{aligned}
$$

Its optimal solution and objective value are $z^* = (0,0,1)^T$ and $\xi^* = 1$. Since $\xi^* = 1 = v_2$, we cannot extend the set S^2.

We were not able to extend both sets S^1 and S^2. This means that the current solution given by (7.16) is optimal for node 1. Since all λ_j^k are integers, we can compute a feasible solution to the original problem:

$$
\begin{pmatrix} x_1^1 \\ x_2^1 \\ x_3^1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \qquad \begin{pmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}.
$$

According to this solution, task 3 is processed on machine 1, and tasks 1 and 2 are processed on machine 2. This is our first solution, and therefore it is remembered as a record solution in the form of the vector $\pi^R = (2,2,1)^T$ ($\pi_i^R = k$ if task i is assigned to machine k). The optimal objective value, which is equal to -6, of the solved relaxation LP is our new record: $R = -6$. Note that the cost of the record assignment, π^R, is $-R = 6$.

2.1. Since $x_1^1 = 1$ at node 2, which means that task 1 is assigned to machine 1, we must exclude all vectors a with $a_1 = 1$ from the set S^2 that is inherited from the parent node 0. As a result, we have

$$S^2 = \{(0,0,1)^T, (0,1,0)^T\}.$$

In addition, for this node and all its descendants, when setting the pricing problems for machine 1, it will always be necessary to set $z_1 = 1$, and for all other machines (in this example only for machine 2), we need to set $z_1 = 0$. The set S^1 remains the same as that at the parent node: $S^1 = \{(0,1,1)^T, (1,0,0)^T, (0,1,0)^T\}$.

Now we solve the next master LP:

$$\begin{array}{l}
-4\lambda_1^1 - \lambda_2^1 - 2\lambda_3^1 - \lambda_1^2 - 2\lambda_2^2 - 6s_1 - 6s_2 - 6s_3 \to \max,\\
\qquad \lambda_2^1 + s_1 = 1,\\
\lambda_1^1 + \lambda_3^1 + \lambda_2^2 + s_2 = 1,\\
\lambda_1^1 + \lambda_1^2 + s_3 = 1,\\
\lambda_1^1 + \lambda_2^1 + \lambda_3^1 \le 1,\\
\lambda_1^2 + \lambda_2^2 \le 1,\\
\lambda_1^1, \lambda_2^1, \lambda_3^1, \lambda_1^2, \lambda_2^2, s_1, s_2, s_3 \ge 0.
\end{array}$$

The non-zero components of its optimal primal and dual solutions are the following:

$$\begin{aligned}
\lambda_1^1 = \lambda_2^1 = \lambda_1^2 = \lambda_2^2 = s_1 = \frac{1}{2},\\
y = (-6,-5,-4),\ v_1 = 5,\ v_2 = 3.
\end{aligned} \tag{7.17}$$

Next we solve the pricing problems. For machine 1, we solve the following 0,1-knapsack problem:

$$\begin{aligned}
\xi = 5z_1 + 3z_2 + 2z_3 \to \max,\\
3z_1 + 2z_2 + 2z_3 \le 4,\\
z_1 = 1,\ z_2, z_3 \in \{0,1\}.
\end{aligned}$$

Its optimal solution and objective value are $z^* = (1,0,0)^T$ and $\xi^* = 5$. Since $\xi^* = 5 = v_1$, we cannot extend the set S^1.

For machine 2, we solve the next 0,1-knapsack problem:

$$\begin{aligned}
\xi = 4z_1 + 3z_2 + 3z_3 \to \max,\\
2z_1 + 2z_2 + 3z_3 \le 4,\\
z_1 = 0,\ z_2, z_3 \in \{0,1\}.
\end{aligned}$$

Its optimal solution and objective value are $z^* = (0,0,1)^T$ and $\xi^* = 3$. Since $\xi^* = 3 = v_2$, we cannot extend the set S^2.

We were not able to extend both sets S^1 and S^2. This means that the current solution given by (7.17) is optimal for node 2. But since the upper bound at this node is -7 and it is less than the current record equal to -6, then node 2 must be eliminated from the search tree.

Since there are no more unprocessed nodes in the search tree, the current record assignment $\pi^R = (2,2,1)^T$ is optimal. $\square$

After solving this example, someone might have doubts about the efficiency of the branch-and-price algorithms (at least the one th6at we just used to solve our example): so many calculations to solve an almost trivial example. In fact, this is not so. But even this simple example should convince you that the implementation of almost any branch-and-price algorithm is not a trivial exercise.

7.4 Symmetry Issues

As we noted in Sect. 7.2, one of the reasons why Formulation (7.7) with a huge number of variables may be preferable to the compact formulation (7.6) is that the first one provides more accurate upper bounds. We also noted that, we will get the same bounds if we solve (7.6) by the branch-and cut method using exact separation procedures for the sets $\mathrm{conv}(X^k)$. Another reason why we can prefer (7.7) is the presence of symmetric structures, when, for some k, the objects, X^k, A^k and c^k, are the same. In such cases, there are many symmetric (essentially identical) solutions: by interchanging the values of the corresponding components of the vectors x^{k_1} and x^{k_2} for two identical structures k_1 and k_2, we get a feasible solution with the same objective value. As a rule, the branch-and-cut algorithms are very inefficient in solving problems with symmetric structures. The reason here is that, by adding a cut valid for the set X^{k_1} or by changing a bound for some variable $x_j^{k_1}$, we do not exclude the later appearance in the search tree of a node with the symmetric LP solution obtained from the previously cut off solution by just exchanging the vectors x^{k_1} and x^{k_2}. In some cases, we can cut off some of the symmetric solutions by adding new constraints. But it is not always possible to completely overcome this *symmetry issues*. More effectively, these symmetry issues can be resolved by developing a specialized branching scheme. Often, the transition to Formulation (7.7) and the use of a special branching scheme allows us to completely eliminate the symmetry. Next we demonstrate this with the example of the generalized assignment problem.

Let us modify the statement of the problem given in Sect. 7.3. Suppose now that we have K groups of machines, and group k contains n_k identical machines. Now the parameters p_i^k and c_i^k characterize all machines of group k, they are, respectively, the processing time and cost of performing task i on any machine from group k. For a new statement, the master problem (7.13) changes only slightly: we only need to replace (7.13c) with the following inequalities:

$$\sum_{a \in S^k} \lambda_a^k \le n_k, \quad k = 1,\dots,K. \tag{7.18}$$

At the root node of the search tree, the pricing problem (7.14) for each group of machines remains unchanged. At other nodes, additional constraints will be added to their pricing problem.

To eliminate symmetry, we now do not distinguish machines from one group. For this reason branching on the variables x_i^k is impossible. On the other hand, branching on variables λ_a^k is inefficient. We can still develop an efficient branching scheme based on the following observation.

Proposition 7.1. *Let $\tilde{\lambda}$ be a solution to the master LP at some node of the search tree. If not all values $\tilde{\lambda}_a^k$ $(a \in S^k)$ are integers, then there exist two tasks, r and s, such that*

$$\sum_{a\in S^k:\, a_r=a_s} \tilde{\lambda}_a^k < 1. \tag{7.19}$$

So, if some all k not all $\tilde{\lambda}_a^k$ are integers, we seek a pair of tasks, r and s, such that (7.19) is satisfied. Branching is performed by dividing the set of feasible solutions into two subsets, the first of which includes all assignments in which the tasks r and s are performed on the same machine, and the second includes the assignments in which the tasks r and s are performed on different machines. In the first case, when solving the pricing problem for group k, the tasks r and s are combined into one task of processing time $p_r^k + p_s^k$ and cost $c_r^k + c_s^k$. At the same time, it is necessary to exclude from the node master LP all variables λ_a^k for those k and a such that $a_r \neq a_s$. In the second case, we need to add the inequality $z_r + z_s \le 1$ to the pricing problem for group k, and exclude from the master LP all variables λ_a^k for those k and a such that $a_r = a_s$. Of course, the addition of new inequalities destroys the structure of the pricing problem, and now it is not a 0,1-knapsack problem and, therefore, it must be solved as an ordinary IP. But, despite this, computational experiments have proved the efficiency of such branching.

7.5 Designing Telecommunication Networks

In this section we show that applications of the branch-and-price method are not limited to the framework of the decomposition approach of Danzig and Wolfe. In a number of cases, with a natural choice of variables, their number can be exponentially large (recall the cutting stock problem from Sect. 7.1.1). In addition, there may be restrictions that are difficult to compactly formulate by means of linear inequalities. Sometimes, it is better to take into account such complex restrictions in pricing problems, which can be solved in some other way (say, by dynamic or constraint programming).

The input data in the problem of designing a reliable telecommunication network are specified by two graphs: the *channel graph* $G = (V,E)$ and the *demand graph* $H = (\overline{V},D)$. The set V consists of logical network nodes (offices, routers, etc.), $\overline{V}$ is a subset of V (offices are in $\overline{V}$, but routers are not). The edges $e \in E$ of the channel graph represent the set of physical communication lines (channels) that can potentially be installed. Different types of communication lines (representing various technologies, for example, optical fiber, telephone lines, Ethernet, etc.) are represented by parallel edges. The capacity of channel $e \in E$ is u_e, and the cost of establishing this channel is c_e. For each edge $q = (v,w) \in D$ of the demand graph, we know the *demand*, d_q, for communication between nodes v and w: the total capacity of communication lines between nodes v and w must be at least d_q. The values of d_q are determined statistically on the basis of the forecast for the development of the information services market.

It is common to call a network *reliable* if it is designed in such a way that it continues to function even after some of its components fail. Let $G_s = (V_s, E_s)$ be the subgraph obtained from the graph G by deleting all its components damaged in the emergency state s, $s = 1,\dots,S$ (as a rule, each emergency state corresponds to the failure of one node or channel; if a node is damaged, then all channels incident to it also are not able to function). It is required that, for each demand edge $q = (v,w) \in D$ and for each emergency state s, the survived subnetwork still be able to route at least $\rho_{sq} d_q$ information units between nodes v and w, where $0 < \rho_{sq} \le 1$.

In the case of congestion of communication channels, some connections, which are the paths carrying information between pairs of nodes, can be very long. Due to various reasons (for example, to reduce delays of establishing connections, or to reduce the load of computers), it is necessary to limit the length (number of edges) of the communication paths. Let l_q be the maximum path length for a demand $q \in D$.

Now we can formulate the problem of *designing a reliable telecommunication network* as follows: it is needed to find in G a subgraph $G' = (V, E')$ with the minimum total cost of its edges, $c(G') \overset{\text{def}}{=} \sum_{e \in E'} c_e$, provided that the capacities of edges in G' are sufficient for routing (taking into account the restriction on the path lengths) the required amounts of information in the normal and all emergency states.

Let us agree to consider the normal (failure-free) state of the network as state 0. We set $\rho_{0,q} = 1$ for all $q \in D$. For each state $s = 0,\dots,S$ and each demand $q = (v,w) \in D$, let $\mathscr{P}(s,q)$ denote the set of feasible v,w-paths. For the normal state ($s = 0$), a feasible v,w-path is any path from v to w of length at most l_q in the graph $G_0 = (V_0 = V, E_0 = E)$. For any emergency state $s = 1,\dots,S$, a feasible v,w-path is any path from v to w in G_s.

For each state $s = 0,\dots,S$, each demand $q = (v,w) \in D$, and each path $P \in \mathscr{P}(s,q)$, we introduce a *flow* variable f_P^{sq} that represents the amount of information circulating between nodes v and w along path P in state s. For each edge $e \in E$ we introduce a binary variable x_e with $x_e = 1$ only if channel e is included into the communication network ($e \in E'$). In these variables, the telecommunication network design problem is formulated as follows:

$$\sum_{e \in E} c_e x_e \to \min, \tag{7.20a}$$

$$\sum_{q \in D} \sum_{P \in \mathscr{P}(s,q):\, e \in E(P)} f_P^{sq} \le u_e x_e, \quad e \in E_s,\ s = 0,\dots,S, \tag{7.20b}$$

$$\sum_{P \in \mathscr{P}(s,q)} f_P^{sq} \ge \rho_{sq} d_q, \quad q \in D,\ s = 0,\dots,S, \tag{7.20c}$$

$$f_P^{sq} \geq 0, \quad P \in \mathscr{P}(s,q), q \in D, \ s = 0,\dots,S, \tag{7.20d}$$

$$x_e \in \{0,1\}, \quad e \in E. \tag{7.20e}$$

Here $E(P)$ denotes the set of edges on a path P.

Objective (7.20a) is to minimize the network cost. Inequalities (7.20b) impose the capacity restrictions for the communication channels: if channel $e \in E$ is included in the network G' ($x_e = 1$), then the total (along all paths) information flow, transmitted through this channel in any of the states, must not exceed the channel capacity; if channel e is not included in the network ($x_e = 0$), the information can not be transmitted via this channel, i.e., $f_P^{sq} = 0$ for all paths P passing through edge e ($e \in E(P)$). Inequalities (7.20c) guarantee that for any demand in any state the network will be able to transmit the required amount of information.

7.5.1 Master Problem

Since the number of variables in (7.20) is huge, we can solve it only by the branch-and-price method. The *master problem* is written very simply:

$$\sum_{e\in E} c_e x_e + M \cdot z_{sq} \to \min, \tag{7.21a}$$

$$\sum_{q\in D} \sum_{P\in\hat{\mathscr{P}}(s,q):\, e\in E(P)} f_P^{sq} \leq u_e x_e \quad e \in E_s, \ s = 0,\dots,S, \tag{7.21b}$$

$$\sum_{P\in\hat{\mathscr{P}}(s,q)} f_P^{sq} + z_{sq} \geq \rho_{sq} d_q, \quad q \in D, \ s = 0,\dots,S, \tag{7.21c}$$

$$f_P^{sq} \geq 0, \quad P \in \hat{\mathscr{P}}(s,q), \ q \in D, \ s = 0,\dots,S, \tag{7.21d}$$

$$x_e \in \{0,1\}, \quad e \in E. \tag{7.21e}$$

Here $\hat{\mathscr{P}}(s,q)$ is a subset of the set $\mathscr{P}(s,q)$. We have also introduced slack variables, z_{sq}, so that the master MIP always had a solution. In particular, this allows us to start the branch-and-price algorithm with the empty sets $\hat{\mathscr{P}}(s,q)$. As it is common in **LP**, M denotes a sufficiently large number.

Before proceeding to the discussion of the pricing problem, let us note that, in our branch-and price algorithm we can apply the standard branching on integer variables as all integer variables, x_e, are always present in the active master problem,

7.5.2 Pricing Problem

Formulating (7.20), we hid the restriction on the length of the communication paths — which is difficult to formulate by a system of inequalities — in the definition of the sets $\mathscr{P}(s,q)$, thereby moving the accounting of this requirement into the pricing problem.

Let us denote the dual variables for the master relaxation LP as follows:

- $\alpha_{se} \le 0$ is associated with the inequality in (7.21b) written for $s \in \{0,\dots,S\}$ and $e \in E$;
- $\beta_{sq} \ge 0$ corresponds to the inequality in (7.21c) written for $s \in \{0,\dots,S\}$ and $q \in D$.

Let the pair of $\tilde{\alpha} \in \mathbb{R}^{\{0,\dots,S\}} \times \mathbb{R}^E$ and $\tilde{\beta} \in \mathbb{R}^{\{0,\dots,S\}} \times \mathbb{R}^D$ constitute an optimal dual solution to the relaxation LP for (7.21). For given $s \in \{0,\dots,S\}$, $q \in D$ and $P \in \mathscr{P}(s,q)$, the reduced cost of the variable f_P^{sq} is

$$-\sum_{e\in E(P)} \tilde{\alpha}_{se} - \tilde{\beta}_{sq}.$$

Therefore, the *pricing problem* is formulated as follows:

$$\min\left\{-\sum_{e\in E(P)} \tilde{\alpha}_{se} :\, P \in \mathscr{P}(s,q)\right\} \tag{7.22}$$

This is the shortest path problem between nodes v and w in G_s, when the weight of every edge $e \in E_s$ is $-\tilde{\alpha}_{se} \ge 0$. If the weight of a shortest path, $\hat{P}$, is less than $\tilde{\beta}_{sq}$, then $\hat{P}$ is added to the set $\hat{\mathscr{P}}(s,q)$.

For emergency states ($s = 1,\dots,S$), (7.22) is the shortest path problem in the undirected graph. For the normal state ($s = 0$), (7.22) is the problem of finding a shortest path of limited length. For those who do not know how to solve this problem, let us say that some shortest path algorithms are easily adapted to search for shortest paths of limited length. In particular, if the path length should not exceed k, the famous Bellman-Ford algorithm must execute only k iterates (if, of course, the algorithm does not stop earlier).

The shortest path algorithms are studied almost in every manual on graph or network flow theory, and therefore are not discussed here.

7.6 Notes

***Sect.* 7.1.** The works [54, 55], devoted to solving the one-dimensional cutting stock problem, can be considered the first application of the column generation technique for solving IPs.

***Sect.* 7.2.** The work of Danzig and Wolfe [45] is fundamental for using decomposition in **LP**. Johnson [76] was one of the first to realize both the potential and the implementation complexity of the branch-and-price algorithms. The ideas and methodology of the branch-and-price method are discussed in [18], where an overview of some its applications is also given. A particular success of the branch-and-price method is associated with the routing and scheduling problems [46].

***Sect.* 7.3.** The generalized assignment problem has become standard for demonstrating the branch-and-price method since the work [118].

***Sect.* 7.5.** Minoux [93] was the first who studied the problem of reliability (survivability) of the generalized multicommodity flow networks. The review [3] is devoted to the designing reliable telecommunication networks.

***Sect.* 7.7.** The idea of Exercise 7.6 to formulate a vehicle routing problem as a set partitioning problem originally appeared in [16].

7.7 Exercises

7.1. Solve an instance of the cutting stock problem in which from rolls of length 128 it is necessary to cut out 12 stocks of lengths 66, 34 of length 25, 9 of length 85, 20 of length 16, and 7 of length 45.

7.2. Write down a compact formulation for the one-dimensional cutting stock problem from Sect. 7.1.1.

Hint. Suppose that the desired number of stocks can be cut out from no more than k rolls. For example, as k, we can take the required number of rolls for a cutting plan, built by the heuristic algorithm from Sect. 7.1.3. Use the following variables:

- x_{ij}: number of stocks of type i to be cut out from roll j, $i = 1,\dots,m$, $j = 1,\dots,k$;
- $y_j = 1$ if roll j is cut out, and $y_j = 0$ otherwise, $j = 1,\dots,k$.

7.3. Use the branch-and-price method to solve the generalized assignment problem with the following parameters: $m = 3$, $K = 2$,

$$[c_i^k] = \begin{bmatrix} 1 & 2 \\ 2 & 2 \\ 2 & 1 \end{bmatrix}, \quad [p_i^k] = \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}, \quad l = \begin{pmatrix} 2 \\ 2 \end{pmatrix}.$$

7.4. *Clustering problem*. Given a graph $G = (V,E)$, each edge $e \in E$ of which is assigned a cost c_e, and each vertex $v \in V$ is associated with a weight w_v. We need to partition the set of vertices V into K clusters $V_1,\dots,V_K$ (some of them may be empty) so that the sum of the vertex weights in each of the clusters does not exceed a given limit W, and the sum of the costs of the edges between the clusters is minimum.

Formulate this clustering problem as an IP so that the branch-and-price method can be used to solve that IP, write down the master and pricing problems, elaborate a branching rule.

7.5. If you have mastered Exercise 7.4, you can test yourself by numerically solving an example of the clustering problem on the complete graph of three vertices with three possible clusters, when the weights of all vertices and the cost of all edges are equal to one, and the sum of node weights in any cluster is at most two.

7.6. Formulate the single depot vehicle routing problem (VRP) from Sect. 2.16 as a set partitioning problem defined on a hypergraph which vertices correspond to the customers, and hyperedges represent all feasible routes. Elaborate a branch-and-price algorithm that solve the **IP** formulation for this set partition problem: write down the master and pricing problems, specify a branching rule.

7.7. Let us remember the problem of designing a reliable telecommunication network from Sect. 7.5. Suppose now that a designing network will not reroute information flows in case of failure of some its elements. In the design of such networks, to increase reliability, a *diversification strategy* is used, which requires that, for each demand edge $q=(v,w)\in D$, no more than $\delta_q d_q$ $(0<\delta_q\le 1)$ information units, circulating between nodes v and w, pass through any channel and any node. The implementation of this diversification strategy will guarantee that any damage of one channel or one node will not affect $(1-\delta_q)\,100\,\%$ of the total amount of information between nodes v and w.

Write down a compact IP as a model of the problem of designing diversified telecommunication networks.

7.8. Elaborate a branch-and-price algorithm for the facility location problem formulated in Sect. 2.2.

7.9. Reformulate the problem of detailed placement from Sect. 2.8 as an IP so that the branch-and-price method can be used to solve that IP, write down the master and pricing problems, show how to solve the pricing problem.

7.10. In Sect. 7.2 for IP (7.7) we have established the equivalence of two bounds z_{LPM} and z_{CUT}. Prove that the *Lagrangian relaxation* provides the same bound:

$$z_{LPM}=z_{CUT}=z_{LD}\overset{\text{def}}{=}\min_{u\in\mathbb{R}^m_+}L(u),$$

where

$$L(u)=\sum_{k=1}^{K}\max\{(c^k-u^TA^k)^Tx^k:\ x^k\in X^k\}+u^Tb.$$

7.11. *Steiner tree problem*. Given a graph $G=(V,E)$ and a set $T\subseteq V$ of *terminals*. A *Steiner tree* is a minimum (by inclusion) subgraph in G having a path between any pair of terminals. Let each edge $e\in E$ be assigned a cost $c_e\ge 0$. The Steiner tree problem is to find in G a Steiner tree of minimum cost, where the cost of a tree is the sum of the costs of its edges.

Let $\mathscr{P}$ denote the set of all paths in G connecting any pair of terminals, and let $V(P)$ and $E(P)$ denote the sets of, respectively, vertices end edges on a path P. Introducing two families of binary variables

- $x_e=1$ if $e\in E$ is an edge of the Steiner tree, and $x_e=0$ otherwise,
- $y_P=1$ if $P\in\mathscr{P}$ is a path in the Steiner tree, and $y_P=0$ otherwise,

we formulate the Steiner tree problem as the following IP:

$$\begin{aligned}
&\sum_{e\in E} c_e x_e \to \min, \\
&\sum_{P\in\mathscr{P}:\, t\in V(P)} y_P \ge 1, \quad t\in T, \\
&y_P \le x_e, \quad e\in E,\ P\in\mathscr{P},\ e\in E(P), \\
&y_P \in \{0,1\}, \quad P\in\mathscr{P}, \\
&x_e \in \{0,1\}, \quad e\in E.
\end{aligned} \tag{7.23}$$

Elaborate a branch-and-price algorithm for solving (7.23), specify the pricing problem.

7.12. *Steiner tree packing problem*. Given a graph $G=(V,E)$ and a family of terminal sets $T_i \subseteq V$, $i=1,\dots,k$. Let each edge $e\in E$ be assigned a cost $c_e \ge 0$. The Steiner tree packing problem is to find a family of k edge-disjoint Steiner trees so that each terminal set T_i is connected by exactly one tree, and the total cost of all edges included in the trees is minimum. It is worth noting that this packing problem is a model for the *routing problem* in VLSI circuit design, where each Steiner tree represents an electrical circuit connecting a set of terminals.

Proceeding from Formulation (7.23) for the Steiner tree problem, write down an **IP** formulation for the Steiner tree packing problem. Elaborate a branch-and-price algorithm for solving your formulation.

Chapter 8
Optimization With Uncertain Parameters

The formulations of many optimization problems include uncertain parameters. There are several approaches to solving such problems. In *stochastic programming* it is assumed that all uncertain parameters are random variables with known probability distributions. *Robust optimization* is used when it is required that the solution be acceptable for all possible values of uncertain parameters. The latter is very important in those situations where small changes in the input of the problem can significantly affect its solution.

Usually the solution of an optimization problem with uncertain parameters is reduced to the solution of its deterministic equivalent, which, as a rule, is an optimization problem that is much larger than the original problem. In this chapter we will study only such models of stochastic programming and robust optimization, the deterministic versions of which are MIPs.

8.1 Two-Stage Stochastic Programming Problems

Stochastic programming models include two types of variables: expected and adaptive. *Expected variables* represent those decisions that are taken *here-and-now*: they do not depend on the future implementation of the random parameters. Decisions described by *adaptive variables* are accepted after the values of the random parameters become known.

For example, consider the *two-stage stochastic programming problem* that is formulated as follows:

$$\begin{array}{rrl} c^T x + & E(h(\omega)^T y(\omega)) & \to \max, \\ A(\omega)x + & G(\omega)\, y(\omega) & \le b(\omega), \\ x & & \in X, \\ & y(\omega) & \in \mathbb{R}^{n_y}_+. \end{array} \tag{8.1}$$

In (8.1) a decision for use in the current (first) period is represented by the vector $x \in X$ of the expected variables, where $X \subseteq R^{n_x}$ is some set (for example, $X = \mathbb{R}^{n_x}_+$, $X = \mathbb{Z}^{n_x}_+$ or $X = P(\bar{A}, \bar{b}; S)$). A decision $x \in X$ must be made in the current period before an elementary event ω from the probability space $(\Omega, \mathscr{A}, \mathbb{P})$ occurs in the next period. A decision $y(\omega)$ is made in this next period after observing ω. Therefore, the vector y of adaptive variables is a vector-function of ω. The system $A(\omega)x + G(\omega)y(\omega) \le b(\omega)$ of stochastic constraints connects the expected and adaptive variables. The objective function in (8.1) is the sum of two terms: deterministic $c^T x$, estimating the quality of the solution x, and the expected value $E(h(\omega)^T y(\omega))$ of the random variable $h(\omega)^T y(\omega)$, which estimates the quality of the solution $y(\omega)$.

Problem (8.1) can be reformulated as follows:

$$\max\{f(x) : x \in X\}, \tag{8.2}$$

where $f(x) = E(f(x,\omega))$, and the random variable $f(x,\omega)$ (see Exercise 8.3) is determined by the rule:

$$\begin{aligned} f(x,\omega) &\overset{\text{def}}{=} c^T x + \max h(\omega)^T y(\omega),\\ G(\omega)y(\omega) &\le b(\omega) - A(\omega)x,\\ y(\omega) &\in \mathbb{R}^{n_y}_+. \end{aligned} \tag{8.3}$$

If the sample space Ω is infinite, then computing $f(x)$ can be a very difficult problem. One approach is to approximate an infinite probability space with a finite space. Discussion of how this is done is beyond the scope of this book. In what follows we assume that $\Omega = \{\omega_1, \dots, \omega_K\}$ is a finite set and the event (scenario) ω_k occurs with probability p_k ($k = 1, \dots, K$). For $k = 1, \dots, K$, we introduce the following notations: $h_k = h(\omega_k)$, $w_k = p_k h_k$, $A_k = A(\omega_k)$, $G_k = G(\omega_k)$, $b_k = b(\omega_k)$, $y_k = y(\omega_k)$, $n_k = n_y$. The *deterministic equivalent* of the stochastic problem (8.1) is written as follows:

$$\begin{aligned} &c^T x + \sum_{k=1}^{K} w_k^T y_k \to \max,\\ &A_k x + \quad G_k y_k \le b_k, \qquad k = 1, \dots, K,\\ &x \qquad\qquad\quad \in X,\\ &\qquad\qquad\quad y_k \in \mathbb{R}^{n_k}_+, \quad k = 1, \dots, K. \end{aligned} \tag{8.4}$$

Having solved (8.4), we get a decision x for use in the current period. This decision, x, must be adequate to everything that can happen in the next period. If we knew what scenario ω_k would happen in the next period, we would solve the problem

$$\begin{aligned} &c^T x + h_k^T y_k \to \max,\\ &A_k x + G_k y_k \le b_k,\\ &x \qquad\quad \in X,\\ &\qquad\quad y_k \in \mathbb{R}^{n_k}_+ \end{aligned}$$

that takes into account the constraints only for this scenario. But since we do not know which scenario will be realized in the future, in (8.4) we require that the constraints $A_k x + G_k y_k \le b_k$ be valid for all scenarios $k = 1,\dots,K$.

8.2 Benders' Reformulation

If the number of scenarios K is very large, then (8.4) can be a very difficult optimization problem. Obviously, with a huge number of scenarios, the effect of each of them on the solution x to be made in the current period is different. The Benders decomposition approach allows us to reformulate the problem in such a way that information about the scenarios will be provided through cuts.

Let us first assume that the vector of expected variables x is fixed. We can find the values of the remaining variables y_k by solving K LPs:

$$z_k(x) \overset{\text{def}}{=} \max\{w_k^T y_k : G_k y_k \le b_k - A_k x,\ y_k \ge 0\}, \quad k = 1,\dots,K. \tag{8.5}$$

Now let us write down the dual for each of these LPs:

$$z_k(x) = \min\{u_k^T (b_k - A_k x) : G_k^T u_k \ge w_k,\ u_k \ge 0\}, \quad k = 1,\dots,K.$$

We denote by Q_k the polyhedron $\{u_k \in \mathbb{R}_+^{m_k} : G_k^T u_k \ge w_k\}$ of the k-th dual LP. Here m_k denotes the number of inequalities in the system $A_k x + G_k y_k \le b_k$. In view of Theorem 1.1, if the polyhedron Q_k is not empty, it can be represented in the form $Q_k = \text{conv}(V_k) + \text{cone}(C_k)$, where V_k is the set of vertices of Q_k, and C_k is the set of directing vectors for the extreme rays of the cone $\{u_k \in \mathbb{R}_+^{m_k} : G_k^T u_k \ge 0\}$.

Theorem 8.1. *If at least one polyhedron Q_k is empty, then* (8.4) *does not have a solution. If all polyhedra Q_k $(k = 1,\dots,K)$ are nonempty, then* (8.4) *is equivalent to the following problem:*

$$\eta \to \max, \tag{8.6a}$$

$$c^T x + \sum_{k=1}^{K} u_k^T (b_k - A_k x) \ge \eta, \quad (u_1,\dots,u_K) \in V_1 \times \cdots \times V_K, \tag{8.6b}$$

$$u_k^T (b_k - A_k x) \ge 0, \quad u_k \in C_k,\ k = 1,\dots,K, \tag{8.6c}$$

$$x \in X,\ \eta \in \mathbb{R}. \tag{8.6d}$$

Proof. If $Q_k = \emptyset$ for some k, then by Theorem 3.1 (of duality), for all x, LP (8.5), written for this given k, does not have a solution (its objective function is unbounded or there are no feasible solutions). Therefore, in this case, (8.4) does not have a solution as well.

Suppose now that all sets Q_k are nonempty. Then (8.4) can be formulated as follows:

$$\begin{aligned}
c^T x + \sum_{k=1}^{K} \min_{u_k \in Q_k} u_k^T (b_k - A_k x) &\to \max,\\
u_k^T (b_k - A_k x) \ge 0, \quad u_k &\in C_k,\ k = 1,\dots,K,\\
x &\in X.
\end{aligned} \tag{8.7}$$

Introducing a new variable

$$\eta = \sum_{k=1}^{K} \min_{u_k \in Q_k} u_k^T (b_k - A_k x),$$

we can rewrite (8.7) in the form (8.6). □

Problem (8.6) is *Benders' reformulation* of Problem (8.4). Despite the fact that the number of inequalities in (8.6b) and (8.6c) can be huge, we can still solve (8.6) by the branch-and-cut method that generates these inequalities, also known as *Benders' cuts*, in a separation procedure. Obtaining an input vector $\bar{x} \in \mathbb{R}_+^{n_x}$ and a number $\bar{\eta}$, this procedure can work as follows.

Separation procedure. For $k = 1,\dots,K$, solve the LP:

$$\min\{(b_k - A_k\bar{x})^T u_k \,:\, G_k^T u_k \ge w_k,\ u_k \in \mathbb{R}_+^m\}. \tag{8.8}$$

If for some k this LP has no feasible solutions, and $\bar{u}_k$ is a certificate of infeasibility (see Sect. 3.4), then return the cut

$$\bar{u}_k^T (b_k - A_k x) \ge 0.$$

If (8.8) has an optimal solution $\bar{u}_k$ for each k, and if $\sum_{k=1}^K \bar{u}_k^T (b_k - A_k\bar{x}) < \eta - c^T\bar{x}$, then return the cut

$$c^T x + \sum_{k=1}^{K} \bar{u}_k^T (b_k - A_k x) \ge \eta.$$

Otherwise, $(\bar{x},\bar{\eta})$ satisfies all inequalities in both families, (8.6b) and (8.6c).

What gives us the transition from Formulation (8.4), which is relatively compact, to Formulation (8.6) with a huge number of constraints? The obvious plus of the transition to Benders' reformulation is that we essentially reduced the number of continuous variables, more precisely, we excluded the vectors y_k for $k = 1,\dots,K$, and added only one new variable η. Another advantage is not so obvious. As a rule, the fewer continuous variables we have, the stronger cuts (for example, the fractional Gomory cuts) we can generate.

Example 8.1 *We need to solve the next MIP preliminary having carried out Benders' reformulation:*

$$\begin{aligned} x_1 + 3x_2 + 2y_1 - 2y_2 + 2y_3 &\to \max, \\ 2x_1 + 4x_2 + 2y_1 + \ \ y_2 + \ \ y_3 &\le 10, \\ -x_1 + \ \ x_2 + \ \ y_1 - \ \ y_2 + 2y_3 &\le -2 \\ x_1, x_2 \in \mathbb{Z}_+,\ x_1 &\le 3, \\ y_1, y_2, y_3 &\ge 0. \end{aligned} \tag{8.9}$$

Solution. Here $K = 1$ and

$$c = \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \quad A_1 = \begin{bmatrix} 2 & 4 \\ -1 & 1 \end{bmatrix}, \quad w_1 = \begin{pmatrix} 2 \\ -2 \\ 2 \end{pmatrix}, \quad G_1 = \begin{bmatrix} 2 & 1 & 1 \\ 1 & -1 & 2 \end{bmatrix}, \quad b_1 = \begin{pmatrix} 10 \\ -2 \end{pmatrix}.$$

Let us define $X = \{x \in \mathbb{Z}^2_+ :\ x_1 \le 3\}$. The system of inequalities

$$\begin{aligned} 2u_1 + u_2 &\ge 2, \\ u_1 - u_2 &\ge -2, \qquad (G_1^T u \ge w_1) \\ u_1 + 2u_2 &\ge 2 \end{aligned}$$

defines the polyhedron Q_1 depicted in Fig. 8.1. This polyhedron has three vertices

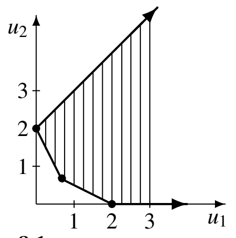

Fig. 8.1

$$u^1 = (0,2)^T, \quad u^2 = \left(\frac{2}{3}, \frac{2}{3}\right)^T, \quad u^3 = (2,0)^T,$$

and two extreme rays with direction vectors

$$\hat{u}^1 = (1,0)^T, \quad \hat{u}^2 = (1,1)^T.$$

So, we can reformulate (8.9) as follows:

$$\begin{aligned} \eta &\to \max, \\ x_1 + 3x_2 + 2\,(-2 + x_1 - x_2) &\ge \eta, \\ x_1 + 3x_2 + \frac{2}{3}\,(10 - 2x_1 - 4x_2) + \frac{2}{3}\,(-2 + x_1 - x_2) &\ge \eta, \\ x_1 + 3x_2 + 2\,(10 - 2x_1 - 4x_2) &\ge \eta, \\ 10 - 2x_1 - 4x_2 &\ge 0, \\ (10 - 2x_1 - 4x_2) + (-2 + x_1 - x_2) &\ge 0, \\ x_1, x_2 \in \mathbb{Z}_+,\ x_1 &\le 3, \end{aligned}$$

or after simplifications and rearranging

$$\begin{aligned} \eta &\to \max, \\ \eta - 3x_1 - x_2 &\le -4, \\ 3\eta - x_1 + x_2 &\le 16, \\ \eta + 3x_1 + 5x_2 &\le 20, \\ x_1 + 2x_2 &\le 5, \\ x_1 + 5x_2 &\le 8, \\ x_1 &\le 3, \\ x_1, x_2 &\in \mathbb{Z}_+. \end{aligned} \tag{8.10}$$

MIP (8.10) can be solved quite easily. From the inequality $x_1 + 5x_2 \le 8$, by integrality of x_2, we have

$$x_2 \le \lfloor (8 - x_1)/5 \rfloor \le \lfloor 8/5 \rfloor = 1.$$

Taking into account the inequalities $0 \le x_1 \le 3$ and $0 \le x_2 \le 1$, from the first three inequalities we calculate an upper bound for η:

$$\begin{aligned} \eta &\le -4 + 3x_1 + x_2 \le 6, \\ \eta &\le (16 + x_1 - x_2)/3 \le 19/3, \\ \eta &\le 20 - 3x_1 - 5x_2 \le 20. \end{aligned}$$

As $\eta \le 6$, a feasible solution $x_1^* = 3$, $x_2^* = 1$, $\eta^* = 6$ is optimal to (8.10). Substituting $x_1 = 3$ and $x_2 = 1$ into (8.9), we obtain the next LP

$$\begin{aligned} 2y_1 - 2y_2 + 2y_3 &\to \max, \\ 2y_1 + \ \ y_2 + \ \ y_3 &\le 0, \\ y_1 - \ \ y_2 + 2y_3 &\le 0, \\ y_1, y_2, y_3 &\ge 0, \end{aligned}$$

which has a unique solution $y_1^* = y_2^* = y_3^* = 0$. □

8.3 Risks

Maximization of the expected profit implies that the decision making process is repeated a sufficiently large number of times under the same conditions. Only then the asymptotic statements, such as the law of large numbers, guarantee the convergence in probability terms of random variables to their expected values. In other situations, we cannot ignore the risk of obtaining a profit, which is significantly lower than the expected value. The identification of suitable risk measures is the subject of active research. We are not going to investigate the problem of risk modeling in its entirety, but we will only discuss one concept of risk that is convenient for use in optimization models.

Here we will try to expand the two-stage model of stochastic programming (8.1), adding to it a system of inequalities that limits the risk of the decision x. The concept of risk is more conveniently to introduce in terms of the loss function $g(x,\omega)$ that depends on the solution x and is a random variable defined on some probability space $(\Omega,\mathscr{A},\mathbb{P})$ $(\omega\in\Omega)$.

Historically, the first and perhaps most famous notion of risk was introduced by H. Markowitz, Nobel Prize winner in Economics in 1990. He defined the *risk* as the variation of the random loss value:

$$\mathrm{var}(g(x,\omega)) = E((g(x,\omega)-E(g(x,\omega)))^2).$$

Conceptually, this measure of risk has several drawbacks. The most important of them is that this measure is symmetric: it equally penalizes for receiving both smaller and larger losses than the expected value. From the point of view of the use in **MIP**, a drawback is that using this risk measure means introducing a quadratic (nonlinear) constraint in optimization models.

Another not less well-known risk measure, called the Value-at-Risk, was developed by the financial engineers of J. P. Morgan. Let

$$G(x,\eta) \stackrel{\text{def}}{=} \mathbb{P}\{\omega\in\Omega:\ g(x,\omega)\le\eta\}$$

be the distribution function of the random variable $g(x,\omega)$. For a given probability $0<\alpha<1$, the risk of making a decision x is

$$\mathrm{VaR}_\alpha(x) \stackrel{\text{def}}{=} \min\{\eta:\ G(x,\eta)\ge\alpha\},$$

which is the maximum loss that occurs with probability at least α.

For a finite probability space, when $\Omega=\{\omega_1,\dots,\omega_K\}$ and when event ω_k occurs with probability p_k $(k=1,\dots,K)$, we compute $\mathrm{VaR}_\alpha(x)$ as follows:

1) list the values $g_k(x)\stackrel{\text{def}}{=} g(x,\omega_k)$ in non-decreasing order:

$$g_{\pi(1)}(x)\le g_{\pi(2)}(x)\le\dots\le g_{\pi(K)}(x);$$

2) find the minimum index j such that $\sum_{i=1}^{j} p_{\pi(i)}\ge\alpha$, and set $\mathrm{VaR}_\alpha(x)=g_{\pi(j)}(x)$.

As an example, consider a finite probability space

$$\Omega=\{0,1,2,3,4,5,6,7,8,9\},$$

when the first three events, $0,1,2$, occur with probability $\frac{1}{6}$, the next two events, $3,4$, with probability $\frac{1}{8}$, and the remaining events, $5,6,7,8,9$, with probability $\frac{1}{20}$. Let $\alpha=0.9$ and $g(x,\omega)=x-\omega$ for $x\in\mathbb{Z}_+$. As $\omega_k=k-1$, then for $x=4$ we have $g_k(4)=g(4,k-1)=5-k$, $k=1,\dots,10$. Next we need to sort the values $g_k(4)$:

i	1	2	3	4	5	6	7	8	9	10
$\pi(i)$	10	9	8	7	6	5	4	3	2	1
$g_{\pi(i)}(4)$	-5	-4	-3	-2	-1	0	1	2	3	4
$p_{\pi(i)}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$

Since
$$\sum_{i=1}^{8} p_{\pi(i)} = 3\frac{1}{6} + 2\frac{1}{8} + 3\frac{1}{20} = 0.9,$$
then $j = 8$ and $\text{VaR}_{0.9}(4) = g_{\pi(8)}(4) = g_3(4) = 2$.

The risk measure VaR is widely used in the financial industry, and its calculation is one of the standard attributes of most financial analysis programs. Despite its popularity, the VaR measure is also not without drawbacks. One of these drawbacks is that this measure does not in any way estimate the amount of losses exceeding $\text{VaR}_\alpha(x)$. Another disadvantage of the VaR measure is that the function $\text{VaR}_\alpha(x)$ is not superadditive (see Exercise 4.4). In financial terminology superadditivity expresses the fact that diversification of investments reduces the risk. Another disadvantage of the VaR measure is that it is difficult to use the function $\text{VaR}_\alpha(x)$ in optimization because it is not convex. These and other shortcomings of the VaR measure motivated the appearance of a number of its modifications.

The measure $\text{CVaR}_\alpha(x)$ (Conditional-Value-at-Risk) is defined as the expected (average) loss, provided that the losses are not less than $\text{VaR}_\alpha(x)$. Formally, the value of $\text{CVaR}_\alpha(x)$ is defined as the expectation of the random variable $g(x,\omega)$ with the so-called α-tail distribution

$$G_\alpha(x,\eta) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } \eta < \text{VaR}_\alpha(x), \\ \dfrac{G(x,\eta)-\alpha}{1-\alpha} & \text{if } \eta \ge \text{VaR}_\alpha(x). \end{cases}$$

By definition, $\text{CVaR}_\alpha(x) = \int_{-\infty}^{\infty} \eta \, dG_\alpha(x,\eta)$. This definition is difficult to use in calculations. The following statement removes this difficulty.

Theorem 8.2. *The following equalities are valid*

$$\text{CVaR}_\alpha(x) = \min_{\eta\in\mathbb{R}} g_\alpha(x,\eta) = g_\alpha(x, \text{VaR}_\alpha(x)),$$

where
$$g_\alpha(x,\eta) \overset{\text{def}}{=} \eta + \frac{1}{1-\alpha}\int_\Omega \max\{g(x,\omega)-\eta, 0\}\mathbb{P}(d\omega).$$

As before, let us restrict ourselves to the scenario approach. Therefore, we assume again that $\Omega = \{\omega_1,\dots,\omega_K\}$ is a finite set and event ω_k occurs with probability p_k, $k = 1,\dots,K$. In this case

$$g_\alpha(x,\eta) = \eta + \frac{1}{1-\alpha}\sum_{k=1}^{K} p_k \max\{g_k(x)-\eta, 0\}, \tag{8.11}$$

where $g_k(x) \overset{\text{def}}{=} g(x, \omega_k)$.

Continuing the example in which we calculated $\text{VaR}_{0.9}(4) = 2$, we compute

$$\begin{aligned}\text{CVaR}_{0.9}(4) &= g_{0.9}(4, \text{VaR}_{0.9}(4)) = g_{0.9}(4,2)\\ &= 2 + \frac{1}{1-0.9} \cdot \frac{1}{20} \cdot (2+1) = 2 + \frac{3}{2} = 3.5.\end{aligned}$$

8.3.1 Extended Two-Stage Model

Again, consider the two-stage stochastic program (8.1). But now we want to maximize the expected profit by limiting the risk: $\text{CVaR}_\alpha(x) \le r$, where r is the maximum allowable risk level. Introducing new variables z_k to represent $\max\{g_k(x) - \eta, 0\}$ in Formula (8.11), we can extend the deterministic equivalent (8.4) as follows:

$$\begin{aligned} c^T x + \sum_{k=1}^K w_k^T y_k &\to \max,\\ A_k x + G_k y_k &\le b_k, \qquad k = 1,\dots,K,\\ \eta + \frac{1}{1-\alpha} \sum_{k=1}^K p_k z_k &\le r, \\ g_k(x) - \eta - z_k &\le 0, \qquad k = 1,\dots,K,\\ \eta \in \mathbb{R},\ x &\in X,\\ z_k \ge 0,\ y_k &\in \mathbb{R}_+^{n_k}, \qquad k = 1,\dots,K. \end{aligned} \tag{8.12}$$

Program (8.12) is a MIP if the functions $g_k(x)$ are linear and X is a mixed-integer set $P(\bar{A}, \bar{b}; S)$. Of particular interest is also the case when $g(x, \omega) = -f(x, \omega)$, where $f(x, \omega)$ is defined in (8.3) and is a nonlinear function. Then (8.12) can be rewritten as follows:

$$\begin{aligned} c^T x + \sum_{k=1}^K w_k^T y_k &\to \max,\\ A_k x + G_k y_k &\le b_k, \qquad k = 1,\dots,K,\\ \eta + \frac{1}{1-\alpha} \sum_{k=1}^K p_k z_k &\le r, \\ c^T x + h_k^T y_k + \eta + z_k &\ge 0, \qquad k = 1,\dots,K,\\ \eta \in \mathbb{R},\ x &\in X,\\ z_k \ge 0,\ y_k &\in \mathbb{R}_+^{n_k}, \qquad k = 1,\dots,K. \end{aligned} \tag{8.13}$$

Let us verify that both programs, (8.12) and (8.13), are equivalent when $g(x, \omega) = -f(x, \omega)$. By definition

$$g_k(x) = -c^T x - \max\{h_k^T y_k :\ G_k y_k \le b_k - A_k x\},$$

and since

$$c^T x + \sum_{k=1}^{K} w_k^T y_k = \sum_{k=1}^{K} p_k (c^T x + h_k^T y_k),$$

then in an optimal solution $(x^*; y_1^*, \dots, y_K^*; \eta^*; z_1^*, \dots, z_K^*)$ to (8.13) each point y_k^* is an optimal solution to the LP

$$\max\{h_k^T y_k : \, G_k y_k \le b_k - A_k x^*\},$$

and, therefore, $g_k(x^*) = -c^T x^* - h_k^T y_k^*$.

8.3.2 Credit Risk

Credit risk is the risk caused by the fact that the obligors do not fully fulfill their obligations, or by a decrease in the market price of assets due to the fall of credit ratings. For example, a portfolio of bonds from emerging markets (Brazil, India, Russia, etc.) can most likely generate revenue, but at the same time there is a small probability of large losses. For such investments, the distribution functions of returns (future incomes) are asymmetric and, consequently, symmetric risk measures are not entirely appropriate here. But the VaR measure (as well as its derivative CVaR) was invented to assess the risks in such situations.

Consider a problem of optimizing a portfolio of n potential investments (such as shares). We need to determine the share x_j of each investment j in the portfolio. Then the portfolio is presented by the vector $x = (x_1, \dots, x_n)^T$. The set X of feasible solutions (portfolios) is described by the system

$$\sum_{j=1}^{n} x_j = 1,$$
$$l_j \le x_j \le u_j, \quad j = 1, \dots, n,$$

where l_j and u_j are minimum and maximum possible shares of investment j.

Assume that K scenarios are possible after the completion of some planning horizon. Let p_k be the probability of occurrence of scenario k, and let μ^k be the return vector for this scenario, where μ_j^k is the return (per one enclosed dollar) of investment j. Then $\mu = \sum_{k=1}^{K} p_k \mu^k$ is the vector of expected returns. The loss of a portfolio x, if scenario k occurs, is determined by the formula:

$$g_k(x) = (q - \mu^k)^T x,$$

where q is the return vector, provided that the credit rating of each investment does not change. We determine the risk of the portfolio x to be $\mathrm{CVaR}_\alpha(x)$ and limit this risk to a given value r. Note that with this setting, the "security level" of our portfolio x is determined by choosing two parameters, α and r.

Under the above assumptions, the problem of maximizing the expected return of the portfolio at a limited risk is written as follows:

$$\mu^T x \to \max, \tag{8.14a}$$

$$\eta + \frac{1}{1-\alpha}\sum_{k=1}^{K} p_k z_k \le r, \tag{8.14b}$$

$$(q-\mu^k)^T x - \eta - z_k \le 0, \quad k=1,\dots,K, \tag{8.14c}$$

$$\sum_{j=1}^{n} x_j = 1, \tag{8.14d}$$

$$l_j \le x_j \le u_j, \quad j=1,\dots,n, \tag{8.14e}$$

$$z_k \ge 0, \quad k=1,\dots,K, \tag{8.14f}$$

$$\eta \in \mathbb{R}. \tag{8.14g}$$

Note that (8.14) is an LP. However, it can turn into a MIP after taking into account a number of standard for portfolio optimization additional logical conditions. One of these conditions is the requirement to diversify the investments. Suppose that the set $N = \{1,\dots,n\}$ of all investments is divided into subsets (groups) $N_1,\dots,N_m$, say, according to the sectoral or territorial principle. It is required that no more than n_i different investments be present in the portfolio from group N_i, and also that the portfolio had investments from at least s groups.

We introduce two families of binary variables:

- $y_j = 1$ if investment j is present in the portfolio, and $y_j = 0$ otherwise ($j = 1,\dots,n$);
- $\delta_i = 1$ if at least one investment from group i is present in the portfolio, and $\delta_i = 0$ otherwise ($i = 1,\dots,m$).

To take into account the above requirements, we need to replace (8.14e) with the following system:

$$\begin{aligned}
l_j y_j \le x_j \le u_j y_j, &\quad j=1,\dots,n,\\
\sum_{j\in N_i} y_i \le n_i, &\quad i=1,\dots,m,\\
y_j \le \delta_i, &\quad j\in N_i,\ i=1,\dots,m,\\
\sum_{i=1}^{m} \delta_i \ge s, &\\
y_j \in \{0,1\}, &\quad j=1,\dots,n,\\
\delta_i \in \{0,1\}, &\quad i=1,\dots,m.
\end{aligned}$$

8.4 Multistage Stochastic Programming Problems

Although multistage models of stochastic programming have been studied for several decades, they have not been used in practice to solve problems of required sizes. Only recently, with the advent of sufficiently powerful computers, stochastic programming models began to be used in practice, and stochastic programming itself began to develop at a rapid pace.

Multistage problems of stochastic programming are applied when the planning horizon includes more than one period (stage). Let T denote the number of periods, and $\omega^t \in \Omega^t$ are events that can occur in period t, $t = 1,\dots,T$. At the beginning of the planning horizon (at stage 0), an expected solution x is taken when the event ω^1 has not yet occurred. A decision $y(\omega^1,\dots,\omega^t)$ is made at stage t, when the events $\omega^1,\dots,\omega^{t-1}$ have already occurred, and the event ω^t has not yet happened. The decision $y(\omega^1,\dots,\omega^t)$ depends on the decision $y(\omega^1,\dots,\omega^{t-1})$ made at the previous stage. The multistage model is written as follows:

$$
\begin{aligned}
c^T x + \sum_{t=1}^{T} h(\omega^1,\dots,\omega^t)^T y(\omega^1,\dots,\omega^t) &\to \max,\\
A(\omega^1)x + G(\omega^1)y(\omega^1) &\le b(\omega^1),\\
A(\omega^1,\omega^2)y(\omega^1) + G(\omega^1,\omega^2)y(\omega^1,\omega^2) &\le b(\omega^1,\omega^2),\\
A(\omega^1,\omega^2,\omega^3)y(\omega^1,\omega^2) + G(\omega^1,\omega^2,\omega^3)y(\omega^1,\omega^2,\omega^3) &\le b(\omega^1,\omega^2,\omega^3),\\
&\ddots \vdots \ddots\\
A(\omega^1,\dots,\omega^T)y(\omega^1,\dots,\omega^{T-1}) + {}&\\
G(\omega^1,\dots,\omega^T)y(\omega^1,\dots,\omega^T) &\le b(\omega^1,\dots,\omega^T),\\
x &\in X,\\
y(\omega^1,\dots,\omega^t) \in Y_t, &\quad t = 1,\dots,T.
\end{aligned}
\tag{8.15}
$$

Suppose that, for $t = 1,\dots,T$, the sample space Ω^t is finite, and a sequence of the events

$$(\omega_1,\dots,\omega_t) \in \Omega^1 \times \dots \times \Omega^t$$

occurs with probability $p(\omega_1,\dots,\omega_t)$. Obviously, the following equality must hold

$$\sum_{(\omega_1,\dots,\omega_t)\in\Omega^1\times\dots\times\Omega^t} p(\omega_1,\dots,\omega_t) = 1.$$

Since many of these probabilities may be zeros, to formulate the deterministic equivalent of the stochastic problem (8.15), it is convenient to introduce the concept of the *scenario tree*.

The nodes of the scenario tree are numbered from 0 to n. Node 0 is the root of the tree. The nodes that are at a distance of t from the root belong to stage t. We denote by $t(i)$ the stage to which node i belongs. We assume that the edges are oriented in the direction from the root to the leaves, and the directed edges are called arcs. Note that any node j, except for the root, is entered by only one arc (i,j), and then node i is called the *parent* of node j and is denoted by $parent(j)$. Each arc (i,j) of the tree is associated with an event $\omega(i,j)$ from $\Omega^{t(i)}$. The problem input data are distributed among the tree nodes as follows. The set X and the vector $c_0 = c$ are assigned to node 0. Each of the remaining nodes, $j \in \{1,\dots n\}$, is associated with the following parameters:

$$\begin{aligned}
p_j &\stackrel{\text{def}}{=} p(\omega(0,i_1),\omega(i_1,i_2),\dots,\omega(i_{t(j)-1},j)),\\
c_j &\stackrel{\text{def}}{=} h(\omega(0,i_1),\omega(i_1,i_2),\dots,\omega(i_{t(j)-1},j)) \times p_j,\\
b_j &\stackrel{\text{def}}{=} b(\omega(0,i_1),\omega(i_1,i_2),\dots,\omega(i_{t(j)-1},j)),\\
A_j &\stackrel{\text{def}}{=} A(\omega(0,i_1),\omega(i_1,i_2),\dots,\omega(i_{t(j)-1},j)),\\
G_j &\stackrel{\text{def}}{=} G(\omega(0,i_1),\omega(i_1,i_2),\dots,\omega(i_{t(j)-1},j)),
\end{aligned}$$

where the sequence $(0,i_1,\dots,i_{t(j)-1},j)$ specifies the only path in the tree leading from the root (node 0) to node j. Note that, by definition of p_j, the following equations hold:

$$\begin{aligned}
\sum_{j:\,t(j)=\tau} p_j &= 1, \quad \tau = 1,\dots,T,\\
\sum_{j:\,parent(j)=i} p_j &= p_i \quad \text{for all } i \ge 0 \text{ such that } t(i) < T.
\end{aligned}$$

Denoting by x_j the vector of variables describing a decision taken at node j ($x_0 = x$), we write down the deterministic equivalent of (8.15) as follows:

$$\begin{aligned}
&\sum_{j=0}^{n} c_j^T x_j \to \max,\\
&A_j x_{parent(j)} + G_j x_j \le b_j, \quad j = 1,\dots,n,\\
&x_0 \in X,\\
&x_j \in Y_{t(j)}, \quad j = 1,\dots,n.
\end{aligned} \tag{8.16}$$

Solving (8.16), in particular, we find a decision $x = x_0$ to be taken at the beginning of the planning horizon. Decisions x_j at other nodes of the scenario tree are related to later periods and, therefore, are not implemented in practice. When the first period is over, today's future will become reality and we will already know which of the first stage scenarios was realized (we will know the value of event ω^1). To find a decision to be taken in period 1 for the realized scenario, we will need to build a new scenario tree and solve a new instance of (8.16) when the new planning horizon starts at the beginning of period 2 of the original planning horizon. Therefore, it is said that (8.16) is applied dynamically.

It should be noted that (8.16) is a MIP only if X and all Y_t are polyhedral mixed integer sets, i.e., $X = P(\bar{A}^0, \bar{b}^0, S^0)$ and $Y^t = P(\bar{A}^t, \bar{b}^t, S^t)$ for $t = 1, \dots, T$. In the next two sections we consider two concrete examples of the multistage stochastic programming problem.

8.5 Synthetic Options

When forming an investment portfolio, one of the most important goals is to prevent the portfolio yield to fall below some critical level. This can be done by including in the portfolio derivative financial assets, such as options. In situations where derivative assets are not available, we can achieve the desired result by forming a portfolio based on the ”synthetic option” strategy.

The input for the portfolio optimization parameters are the following:

- n: number of assets;
- T: number of periods in the planning horizon, period t begins at time $t-1$ and ends at time t;
- z_0: amount of cash at the beginning of the planning horizon;
- x_{i0}: amount of money invested in asset i at the beginning of the planning horizon;
- R: interest on capital (1 + rate of interest) in terms of one period;
- $r_{it} = r_i(\omega^1, \dots, \omega^t)$: random return (per one enclosed dollar) of asset i in period t;
- ρ_{it}: transaction cost when buying and selling asset i in period t; it is assumed that all transactions are made at the very beginning of each period;
- q_i: maximum share of asset i in the portfolio.

Expected variables:

- x^b_{i1}: amount of money spent in period 1 on buying asset i;
- x^s_{i1}: amount of money received in period 1 from selling asset i.

Adaptive variables:

- $x_{it} = x_i(\omega^1, \dots, \omega^t)$: amount of money invested in asset i in period t, $t = 1, \dots, T$;
- $z_t = z_i(\omega^1, \dots, \omega^t)$: amount of cash at the end of period t, $t = 1, \dots, T$;
- $x^b_{it} = x^b_i(\omega^1, \dots, \omega^{t-1})$: amount of money spent in period t on buying asset i, $i = 1, \dots, n$, $t = 2, \dots, T$;
- $x^s_{it} = x^s_i(\omega^1, \dots, \omega^{t-1})$: amount of money received in period t from selling asset i, $i = 1, \dots, n$, $t = 2, \dots, T$;
- $\xi = \xi(\omega^1, \dots, \omega^T)$: random component of the portfolio value at the end of the planning horizon (at the end of period T);
- w: risk-free (attained if the worst scenario occurs) component of the portfolio value at the end of the planning horizon.

In the selected variables, the portfolio optimization problem is formulated as follows:

$$\lambda w + (1-\lambda)E(\xi) \to \max, \tag{8.17a}$$

$$z_{t-1} + \sum_{i=1}^{n}(1-\rho_{it})x_{it}^{s} - \sum_{i=1}^{n}(1+\rho_{it})x_{it}^{b} = \frac{1}{R}z_t, \quad t=1,\dots,T, \tag{8.17b}$$

$$x_{i,t-1} + x_{it}^{b} - x_{it}^{s} = \frac{1}{r_{it}}x_{it}, \quad i=1,\dots,n,\ t=1,\dots,T, \tag{8.17c}$$

$$x_{it} - q_i\left(z_t + \sum_{j=1}^{n} x_{jt}\right) \le 0, \quad i=1,\dots,n,\ t=1,\dots,T, \tag{8.17d}$$

$$z_T + \sum_{i=1}^{n}(1-\rho_{iT})x_{iT} = w + \xi, \tag{8.17e}$$

$$x_{it}^{b}, x_{it}^{s}, x_{it} \ge 0, \quad i=1,\dots,n,\ t=1,\dots,T, \tag{8.17f}$$

$$z_t \ge 0, \quad t=1,\dots,T. \tag{8.17g}$$

Objective (8.17a) is to maximize the weighted ($0 \le \lambda \le 1$) combination of two components of the portfolio value: risk-free, w, and expected (over all possible outcomes), $E(\xi)$. Equations (8.17b) are the cash balance relations: in each period t, the amount of cash at the end of the previous period, z_{t-1}, plus the amount of money obtained from selling assets, $\sum_{i=1}^{n}(1-\rho_{it})x_{it}^{s}$, minus the amount of money spent on purchasing assets, $\sum_{i=1}^{n}(1+\rho_{it})x_{it}^{b}$, equals the amount of cash at the end of this period, z_t, divided by the interest on capital, R. Similarly, Eqs. (8.17c) establish the balance for each asset i in each period t: the amount of money invested in the asset at the end of the previous period, $x_{i,t-1}$, plus the amount of money invested in the asset, x_{it}^{b}, minus the amount of money received from selling the asset, x_{it}^{b}, equals the amount of money invested in the asset at the end of this period, x_{it}^{b}, divided by the return of the asset in this period, r_{it}. Inequalities (8.17d) limit the shares of all assets in the portfolio. Finally, Eq. (8.17e) isolates the risk-free component of the portfolio value at the end of the planning horizon.

Example 8.2 *An investor wants to invest an amount of z_0 in one risky asset. The planning horizon consists of $T=2$ periods. As for the general model, let R denote the interest on capital in one period. In period* 1 *the return of this asset is r_1^+ or r_1^- with equal probability, and in period* 2 *the return is r_2^+ with probability $\frac{2}{3}$, and r_2^- with probability $\frac{1}{3}$. The cost of the transaction when buying and selling one asset unit is constant and equal to ρ. It is necessary to write a deterministic equivalent for* (8.17) *applied to this investment problem.*

Solution. By assumption, at the beginning of the planning horizon, the investor has an amount of z_0, and no investment in this asset, $x_0 = 0$. The scenario tree for this example is shown in Fig. 8.2. It has 7 nodes, the number near each non-root node is the probability that the corresponding to this node scenario occurs. In stage 1, the scenarios 1 and 2 happen with probability $\frac{1}{2}$. In stage 2 (by the end of the planning horizon), the scenarios 3 and 5 occur with probability $\frac{1}{3}$, and the scenarios 4 and 6

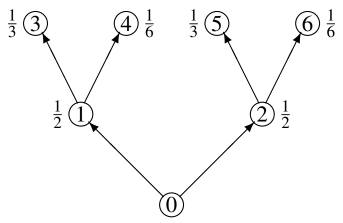

Fig. 8.2 Scenario tree for Example 8.2

occur with probability $\frac{1}{6}$. The decisions made at node i ($i = 0, 1, 2$) are represented by the variables:

- x_i^b: amount of money spent on buying the asset at node i;
- x_i^s: amount of money received from selling the asset at node i.

Saying here that some decision is made at a node, we mean that this decision is made when the scenario associated with this node occurs.

At nodes $i = 1, \dots, 6$, we also need the following auxiliary variables:

- x_i: amount of money invested at node i;
- z_i: amount of cash at node i.

Note that only two variables, x_0^b and x_0^s, are expected, and the other ones are adaptive. In these variables, the deterministic equivalent of (8.17) applied to our instance is written as follows:

$$\lambda \cdot w + (1-\lambda)\left(\frac{1}{3}\xi_3 + \frac{1}{6}\xi_4 + \frac{1}{3}\xi_5 + \frac{1}{6}\xi_6\right) \to \max,$$

$$z_0 + (1-\rho)x_0^s - (1+\rho)x_0^b = \frac{1}{R} z_1, \qquad \text{node 1}$$

$$x_0 + x_0^b - x_0^s = \frac{1}{r_1^+} x_1,$$

$$z_0 + (1-\rho)x_0^s - (1+\rho)x_0^b = \frac{1}{R} z_2, \qquad \text{node 2}$$

$$x_0 + x_0^b - x_0^s = \frac{1}{r_1^-} x_2,$$

$$z_1 + (1-\rho)x_1^s - (1+\rho)x_1^b = \frac{1}{R} z_3, \qquad \text{node 3}$$

$$x_1 + x_1^b - x_1^s = \frac{1}{r_2^+} x_3,$$

$$z_1 + (1-\rho)x_1^s - (1+\rho)x_1^b = \frac{1}{R} z_4, \qquad \text{node 4}$$

$$x_1 + x_1^b - x_1^s = \frac{1}{r_2^-} x_4,$$

$$
\begin{aligned}
z_2 + (1-\rho)x_2^s - (1+\rho)x_2^b &= \frac{1}{R} z_5, && \text{node 5}\\
x_2 + x_2^b - x_2^s &= \frac{1}{r_2^+} x_5, &&\\
z_2 + (1-\rho)x_2^s - (1+\rho)x_2^b &= \frac{1}{R} z_6, && \text{node 6}\\
x_2 + x_2^b - x_2^s &= \frac{1}{r_2^-} x_6, &&\\
z_3 + (1-\rho)x_3 &= w + \xi_3, && \text{isolating}\\
z_4 + (1-\rho)x_4 &= w + \xi_4, && \text{risk-free part}\\
z_5 + (1-\rho)x_5 &= w + \xi_5, && \text{of portfolio}\\
z_6 + (1-\rho)x_6 &= w + \xi_6, && \text{value}\\
x_1, z_1, x_2, z_2, x_3, z_3, x_4, z_4, x_5, z_5, x_6, z_6 &\ge 0, &&\\
x_0^b, x_0^s, x_1^b, x_1^s, x_2^b, x_2^s &\ge 0, &&\\
\xi_3, \xi_4, \xi_5, \xi_6 &\ge 0. && \square
\end{aligned}
$$

8.6 Yield Management

Yield management is an approach to revenue maximization for service firms that exhibit the following characteristics:

1. *Relatively fixed capacity*. Service firms with substantial investment in facilities (e.g., hotels and airlines) are capacity-constrained (once all the seats on a flight are sold, further demand can be met only by booking passengers on a later flight).
2. *Ability to segment its market* into different customer classes. Developing various price-sensitive classes of service gives firms more flexibility in different seasons of the year.
3. *Perishable inventory*. Revenue from an unsold seat in a plane or from unsold room in a hotel is lost forever.
4. *Reservation* systems are adopted by service firms to sell capacity in advance of use. However, managers are faced with uncertainty of whether to accept an early reservation at a discount price or to wait in hope to sell later seats or rooms at a higher price.
5. *Fluctuating demand*. To sell more seats or rooms and increase revenue, in periods of slow demand managers can lower prices, while in periods of high demand prices are higher.

Now let us consider a concrete problem. An airline starts selling tickets for a flight to a particular destination D days before the departure. The time horizon of D days is divided into T periods of unequal length (for example, a time horizon of

$D = 60$ days can be divided into $T = 4$ periods of length 30, 20, 7 and 3 days). The airline can use one of K available planes, the seats in all planes are divided into the same number, I, of classes. Plane k ($k = 1,\dots,K$) costs f_k to hire, and has q_{ki} of seats of class i ($i = 1,\dots,I$). For example, plane k may have $q_{k,1} = 30$ first class seats, $q_{k,2} = 40$ business class seats, and $q_{k,3} = 60$ economy class seats. In plane k, up to r^l_{ki} and r^h_{ki} seats of class i can be transformed into seats of lower, $i-1$, and higher, $i+1$, classes, $i = 1,\dots,I$. It is assumed that $r^l_{k,1} = 0$ and $r^h_{kI} = 0$.

For administrative simplicity, in each period t ($t = 1,\dots,T$) only O price options can be used, and let c_{tio} denote the price of a seat of class i ($i = 1,\dots,I$) in period t if option o is used.

Demand is uncertain but it is affected by ticket prices. Let us assume that S scenarios are possible in each period. The probability of scenario s ($1 \le s \le S$) in period t is p_{ts}, $\sum_{s=1}^S p_{ts} = 1$. The results of demand forecasting are at our disposal: if scenario s occurs in period t, and price option o is used in this period, then the demand for seats of class i will be d_{tsoi}.

We have to choose a plane to hire, and to decide, for each of T periods, which price option to use, how many seats to sell in each class (depending on demand). Our goal is to maximize the expected yield.

Let us also note that period t starts at time $t-1$ and ends at time t. Therefore, it is assumed that the decision which option to use in period t is made at time $t-1$. The other decision how many seats of each class to sell depends on the demand in this period; therefore, this decision is assumed to be made at time t (the end of period t).

To write a deterministic model for this stochastic problem, we need to describe a *scenario tree*. In this application the scenario tree has $n+1 = \sum_{t=0}^T |V_t|$ nodes, where V_t denotes the set of nodes in level t, $t = 0,1,\dots,T$. Let us also assume that the root of the scenario tree is indexed by 0, then $V_0 = \{0\}$ and $V = \cup_{t=0}^T V_t$.

Each node $j \in V_t$ ($t = 1,\dots,T$) corresponds to one of the *histories*, $h(j) = (s_1,s_2,\dots,s_t)$, that may happen after t periods, where s_τ is an index of a scenario for period τ. By definition, the history of the root node is empty, $h(0) = ()$. The *parent* of node j, denoted by *parent*(j), is that node in V_{t-1} which history is $(s_1,s_2,\dots,s_{t-1})$, i.e, $h(j) = (h(parent(j)),s_t)$. Note, that the root node 0 is the parent of all nodes in V_1 (of level 1). In what follows, if we say that something is doing at node j it means that this is doing when the history $h(j)$ is realized.

For $j \in V \setminus \{0\}$, the likelihood of history $h(j)$ is $\bar{p}_j \overset{\text{def}}{=} \prod_{\tau=1}^t p_{\tau,s_\tau}$, $\bar{p}_0 \overset{\text{def}}{=} 1$. If price option o is used at node j, the demand for seats of class i is $\bar{d}_{joi} \overset{\text{def}}{=} d_{t,s_t,o,i}$, and their price is c_{toi}. Let us define $\bar{c}_{joi} \overset{\text{def}}{=} \bar{p}_j c_{toi}$.

Now we introduce the variables. The first family of binary variables is to decide which plane to use. For each plane k we define

- $v_k = 1$ if plane k is used, and $v_k = 0$ otherwise.

Having hired a plane, we need to decide how to transform the seats in that plane. So, we define two families of integer variables:

- w^l_i: number of seats of class i to be transformed into seats of class $i-1$, $i = 2,\dots,I$;

- w_i^h: number of seats of class i to be transformed into seats of class $i+1$, $i = 1,\dots,I-1$.

With each node $j \in V \setminus V_T$ of the scenario tree we associate the following decision variables:

- $y_{jo} = 1$ if price option o is used at node j, and $y_{jo} = 0$ otherwise.

We also introduce two families of auxiliary variables. For each node $j \in V \setminus \{0\}$ we use the following variables:

- x_{joi}: number of seats of class i ($i = 1,\dots,I$) for sale at node j using price option o ($o = 1,\dots,O$);
- z_{ji}: total number of seats of class i ($i = 1,\dots,I$) for sale at node j.

In these variables we write down the following deterministic model:

$$-\sum_{k=1}^{K} f_k v_k + \sum_{j \in V \setminus V_T} \sum_{o=1}^{O} \sum_{i=1}^{I} \bar{c}_{joi} x_{joi} \to \max, \tag{8.18a}$$

$$\sum_{k=1}^{K} v_k = 1, \tag{8.18b}$$

$$u_i = \sum_{k=1}^{K} q_{ki} v_k, \quad i = 1,\dots,I, \tag{8.18c}$$

$$w_i^l \le \sum_{k=1}^{K} t_{ki}^l v_k, \quad i = 1,\dots,I, \tag{8.18d}$$

$$w_i^h \le \sum_{k=1}^{K} t_{ki}^h v_k, \quad i = 1,\dots,I, \tag{8.18e}$$

$$\sum_{o=1}^{O} y_{jo} = 1, \quad j \in V \setminus V_T, \tag{8.18f}$$

$$x_{joi} \le \bar{d}_{jio}\, y_{parent(j),o}, \quad j \in V \setminus \{0\},\ i = 1,\dots,I,\ o = 1,\dots,O, \tag{8.18g}$$

$$z_{ji} = z_{parent(j),i} + \sum_{o=1}^{O} x_{joi}, \quad i = 1,\dots,I,\ j \in V \setminus \{0\}, \tag{8.18h}$$

$$z_{j,1} + w_1^h \le u_1 + w_2^l, \quad j \in V_T, \tag{8.18i}$$

$$z_{ji} + w_i^l + w_i^h \le u_i + w_{i-1}^h + w_{i+1}^l, \quad i = 2,\dots,I-1,\ j \in V_T, \tag{8.18j}$$

$$z_{jI} + w_I^l \le u_I + w_{I-1}^h, \quad j \in V_T, \tag{8.18k}$$

$$x_{joi} \in \mathbb{Z}_+, \quad j \in V \setminus \{0\},\ o = 1,\dots,O,\ i = 1,\dots,I, \tag{8.18l}$$

$$y_{jo} \in \{0,1\}, \quad j \in V \setminus V_T,\ o = 1,\dots,O, \tag{8.18m}$$

$$z_{ji} \in \mathbb{Z}_+, \quad j \in V,\ i = 1,\dots,I, \tag{8.18n}$$

$$z_{0i} = 0, \quad i = 1,\dots,I, \tag{8.18o}$$

$$v_k \in \mathbb{Z}_+, \quad k = 1,\dots,K, \tag{8.18p}$$

$$w_i^l, w_i^h \in \mathbb{Z}_+, \quad i = 1,\dots,I. \tag{8.18q}$$

Objective (8.18a) is to maximize the profit from selling seats minus the expenses for hiring a plane. Equation (8.18b) prescribes to hire just one plane, and Eqs. (8.18c) determine the capacities of all seat classes in the hired plane. Inequalities (8.18d) and (8.18e) restrict the number of seats in any class that can be transformed into seats of the lower and higher classes. Equations (8.18f) prescribe to choose only one price option at each not leaf node. The variable upper bounds (8.18g) guarantee that, in any period and for any price option, the number of sold seats of each class does not exceed the demand for these seats. Equations (8.18h) calculate the total number of seats in each class that are sold in any of T periods. Inequalities (8.18i)–(8.18k) imply that, for each class, the number of sold seats plus the number of seats transformed into seats of the adjacent classes does not exceed the total number of seats of this class plus the number of seats of the adjacent classes transformed into seats of this class.

When (8.18) is solved, we know which option, o^1, to use and how many seats of each class for sale in period 1. When period 1 is over, we will know the actual number of seats, s_i^1, of each class i sold in this period. To determine a price option and the number of seats of each class for sale in period 2, we will solve a new planning problem for the time horizon that extends from period 2 to the end of the planning horizon. Writing (8.18) for this new problem, we need to modify (8.18c) in order to take into account the seats sold in period 1:

$$u_i = \sum_{k=1}^{K} q_{ki} v_k - s_i^1, \quad i = 1, \dots, I.$$

Similarly, we will determine a price option and the number of seats of each class for sale in any subsequent period t when period $t-1$ is over.

8.7 Robust MIPs

Considering stochastic programming applications, it was assumed that we are given a set of scenarios with prescribed probabilities. In fact, the problem of generating scenarios is far from being trivial. Moreover, in many (or even in most) practical applications it is impossible to produce a reasonable set of scenarios because of lack of statistically reliable data. We can say that a robust MIP is a MIP in which a part of its parameters (coefficients) are random variables with given ranges of values, but with unknown probability distributions. The robust approach to solving an optimization problem first imposes some restrictions on varying the values of problem uncertain parameters and then seeks a solution that optimizes the objective in the worst case, i.e., when the uncertain parameters take their worst possible values.

A rather general *robust MIP* is formulated as follows:

$$\begin{aligned} c^T x &\to \min, \\ Ax &\le b \quad \text{for all } A \in \mathscr{A}, \\ l \le x &\le u, \\ x_j &\in \mathbb{Z}, \quad j \in S, \end{aligned} \tag{8.19}$$

where $b \in \mathbb{R}^m$, $c, l, u \in \mathbb{R}^n$, $\mathscr{A}$ is a set of real $m \times n$-matrices, x is an n-vector of variables, and $S \subseteq \{1, \dots, n\}$ is a set of integer variables.

You probably already noticed that in the above general model, the vectors b, c, l and u are deterministic. This is not a limitation of the generality of (8.19). For example, if b contains uncertain components, then we can ”move” these uncertainties into the constraint matrix by introducing an additional variable x_{n+1} with a fixed value:

$$Ax - bx_{n+1} \le 0, \;\; x_{n+1} = 1.$$

Similarly, if c contains uncertain components, then we can also ”move” these uncertainties into the constraint matrix by introducing an additional continuous variable z to represent the objective function:

$$\begin{aligned} z &\to \min, \\ c^T x - z &\le 0. \end{aligned}$$

It is natural to assume that there is a certain level of conservatism when changing the uncertain parameters in (8.19). We can quantitatively represent this level of conservatism defining

$$\mathscr{A} = \{A : \; \|A - \bar{A}\| \le \varepsilon\},$$

where the $m \times n$-matrix $\bar{A}$ is some standard value for A, and the number $\varepsilon \ge 0$ quantitatively determines the level of conservatism. This natural representation of conservatism proved to be very effective for the robust LPs (when $S = \emptyset$), since it allows us to reduce a robust LP to a conic linear program[1], which can be solved efficiently (in polynomial time). From a computational point of view, such a representation of conservatism is not entirely appropriate for the robust MIPs ($S \ne \emptyset$) because we are not able yet to solve efficiently conic linear programs with integer variables.

8.7.1 Row-Wise Uncertainties

It is difficult to imagine a situation where all coefficients of the constraint matrix of a non-trivial robust MIP are interdependent. It is natural to assume that the dependent uncertain parameters are in one matrix row. The *robust MIP with row-wise uncertainties* is the following optimization problem:

[1] A *conic linear program* is a problem of minimizing a linear function over the intersection of an affine subspace and a convex cone. In particular, an LP in standard form is a conic linear program which convex cone is polyhedral.

$$\begin{aligned}
&c^T x \to \min,\\
&\sup_{a\in\mathscr{A}_i} a^T x \le b_i, \quad i=1,\dots,m,\\
&l \le x \le u,\\
&x_j \in \mathbb{Z}, \quad j \in S,
\end{aligned} \tag{8.20}$$

where, for $i=1,\dots,m$, $b_i \in \mathbb{R}$ and $\mathscr{A}_i$ is a non-empty set from $\mathbb{R}^n$, $l,u \in \mathbb{R}^n$, x is an n-vector of variables, and $S \subseteq \{1,\dots,n\}$ is a subset of integer variables. Note that this problem can be rewritten in Form (8.19) if we define

$$\mathscr{A} = \{A:\ A_i \in \mathscr{A}_i,\ i=1,\dots,m\}.$$

In general, we can solve (8.20) by the branch-and-cut algorithm if the sets

$$X_i \overset{\text{def}}{=} \{x \in \mathbb{R}^n:\ a^T x \le b_i,\ a \in \mathscr{A}_i\}$$

are represented by efficient separation procedures. More precisely, to separate a given point $\tilde{x} \in \mathbb{R}^n$ from X_i, we need to solve the following optimization problem

$$\max\{\tilde{x}^T a:\ a \in \mathscr{A}_i\}. \tag{8.21}$$

To simplify further arguments, let us assume that this problem has a solution denoted by $a(\tilde{x})$. If $a(\tilde{x})^T\tilde{x} > b_i$, then the inequality $a(\tilde{x})^T x \le b_i$ is valid for X_i but not for $\tilde{x}$; otherwise, $\tilde{x}$ belongs to X_i.

For example, let us consider the case of *ellipsoidal uncertainties* when, for $i = 1,\dots,m$, $\mathscr{A}_i = \{a \in \mathbb{R}^n:\ a = a^i + P_i u,\ \|u\| \le 1\}$ with $a^i \in \mathbb{R}^n$ and P_i symmetric and positive defined $n \times n$-matrix. Then (8.21) is rewritten as follows:

$$\max\{\tilde{x}^T a^i + (P_i\tilde{x})^T u):\ \|u\| \le 1\}.$$

The point $u^* = \frac{1}{\|P_i\tilde{x}\|}P_i\tilde{x}$ is the only optimal solution to this problem (prove this!). Therefore, for $a(\tilde{x}) = a^i + \frac{1}{\|P_i\tilde{x}\|}P_i^2\tilde{x}$, if $a(\tilde{x})^T\tilde{x} = a^i\tilde{x} + \|P_i\tilde{x}\| > b_i$, then the inequality $a(\tilde{x})^T x \le b_i$ is valid for X_i but not for $\tilde{x}$; otherwise, $\tilde{x}$ belongs to X_i.

8.7.2 Polyhedral Uncertainties

A widely used solution approach applied to many robust optimization problems reduces a robust program to its deterministic equivalent, which is also called the *robust counterpart*. The computational difficulty of a robust program depends on how efficiently its robust counterpart can be solved. Now we begin studying those robust MIPs which robust counterparts are MIPs.

Problem (8.20) is a *robust MIP with polyhedral uncertainties* if all sets $\mathscr{A}_i \subseteq \mathbb{R}^n$ are non-empty polyhedra.

Theorem 8.3. *If, for $i = i,\dots,m$, $\mathscr{A}_i = \{a \in \mathbb{R}^n : H_i a \le g_i\}$, where H_i is a real $m_i \times n$-matrix, and $g_i \in \mathbb{R}^{m_i}$, then* (8.20) *is equivalent to the following MIP:*

$$\begin{aligned}
c^T x &\to \min,\\
g_i^T z^i &\le b_i, \quad i=1,\dots,m,\\
H_i^T z^i &= x, \quad i=1,\dots,m,\\
z^i &\ge 0, \quad i=1,\dots,m,\\
l \le x &\le u,\\
x_j &\in \mathbb{Z}, \quad j \in S,
\end{aligned} \tag{8.22}$$

which variables are $x = (x_1,\dots,x_n)^T$, and $z^i = (z^i_1,\dots,z^i_{m_i})^T$ for $i = 1,\dots,m$.

Proof. For $i = 0,\dots,m$, let us consider the computation of

$$\gamma_i \overset{\text{def}}{=} \sup_{a\in\mathscr{A}_i} a^T x$$

as an LP with a as the vector of variables, and then let us write down the dual to this LP:

$$\gamma_i = \max\{x^T a : H_i a \le g_i\} = \min\{g_i^T z^i : H_i^T z^i = x,\ z^i \ge 0\}.$$

Substituting these expressions into (8.20), we obtain (8.22). $\square$

8.7.3 Combinatorial Uncertainties

In this section we consider a model in which the uncertain parameters of the constraint matrix take values from given intervals, and the level of conservatism is expressed by the combinatorial requirement that the number of uncertain parameters with values different from the standard ones is limited in each row of the constraint matrix.

Let us consider (8.20) when, for $i = 1,\dots,m$,

$$\mathscr{A}_i = \Big\{(a_{i1},\dots,a_{in})^T : a_{ij} \in [\bar a_{ij} - \alpha_{ij}, \bar a_{ij} + \alpha_{ij}] \text{ for } j=1,\dots,n,\\ \sum_{j=1}^n \frac{|a_{ij} - \bar a_{ij}|}{\alpha_{ij}} \le q_i\Big\}, \tag{8.23}$$

where q_i ($i = 1,\dots,m$) are parameters that controls the degree of conservatism. Here we also have a robust problem with polyhedral uncertainties because each set $\mathscr{A}_i$ ($i = 1,\dots,m$) is the polyhedron that is the projection (on the space of variables $a_{i1},\dots,a_{in}$) of the polyhedron given by the following system of inequalities:

$$\begin{aligned}
\bar{a}_{ij} - \alpha_{ij}\delta_{ij} \le a_{ij} \le \bar{a}_{ij} + \alpha_{ij}\delta_{ij}, &\quad j = 1,\dots,n,\\
\sum_{j=1}^n \delta_{ij} \le q_i, &\\
0 \le \delta_{ij} \le 1, &\quad j = 1,\dots,n.
\end{aligned}$$

We cannot apply Theorem 8.3 directly here, since the polyhedra $\mathscr{A}_i$ ($i = 1,\dots,m$) are not represented in the required way. Nevertheless, we can formulate a similar theorem which proof differs only in details from the proof of Theorem 8.3.

Theorem 8.4. *Robust MIP* (8.20) *with the uncertainties given by* (8.23) *is equivalent to the following MIP:*

$$\sum_{j=1}^n c_j x_j \to \min, \tag{8.24a}$$

$$\sum_{j=1}^n \bar{a}_{ij} x_j + q_i v_i + \sum_{j=1}^n w_{ij} \le b_i, \quad i = 1,\dots,m, \tag{8.24b}$$

$$v_i + w_{ij} \ge \alpha_{ij} y_j, \quad j = 1,\dots,n,\ i = 1,\dots,m, \tag{8.24c}$$

$$-y_j \le x_j \le y_j, \quad j = 1,\dots,n, \tag{8.24d}$$

$$l_j \le x_j \le u_j, \quad j = 1,\dots,n, \tag{8.24e}$$

$$x_j \in \mathbb{Z}, \quad j \in S, \tag{8.24f}$$

$$v_i \ge 0, \quad i = 1,\dots,m, \tag{8.24g}$$

$$w_{ij} \ge 0, \quad i = 1,\dots,m,\ j = 1,\dots,n. \tag{8.24h}$$

Proof. For $i = 1,\dots,m$ and a fixed x, we define

$$\begin{aligned}
\gamma_i(x) \overset{\text{def}}{=} \max \sum_{j=1}^n x_j a_{ij}, &\\
\bar{a}_{ij} - \alpha_{ij}\delta_{ij} \le a_{ij} \le \bar{a}_{ij} + \alpha_{ij}\delta_{ij}, &\quad j = 1,\dots,n,\\
\sum_{j=1}^n \delta_{ij} \le q_i, &\\
0 \le \delta_{ij} \le 1, &\quad j = 1,\dots,n.
\end{aligned}$$

It is easy to see that this LP has an optimal solution $(a_{i1},\dots,a_{in};\delta_{i1},\dots,\delta_{in})$ such that

$$a_{ij} - \bar{a}_{ij} = \begin{cases} \alpha_{ij}\delta_{ij} & \text{if } x_j \ge 0,\\ -\alpha_{ij}\delta_{ij} & \text{if } x_j < 0. \end{cases}$$

Therefore, $a_{ij}x_j = \bar{a}_{ij}x_j + \alpha_{ij}\delta_{ij}|x_j|$, and by Theorem 3.1 (of duality) we have

$$\begin{aligned}
&\gamma_i(x) - \sum_{j=1}^{n} \bar{a}_{ij}x_j = \\
&= \max\left\{ \sum_{j=1}^{n} \alpha_{ij}|x_j|\delta_{ij} : \sum_{j=1}^{n} \delta_{ij} \le q_i,\ 0 \le \delta_{ij} \le 1 \text{ for } j = 1,\dots,n \right\} \qquad (8.25)\\
&= \min\left\{ q_i v_i + \sum_{j=1}^{n} w_{ij} : v_i + w_{ij} \ge \alpha_{ij}|x_j|,\ \ v_i \ge 0,\ w_{ij} \ge 0 \text{ for } j = 1,\dots,n \right\}.
\end{aligned}$$

Now, to get (8.24), it remains to replace $\sup_{a\in\mathscr{A}_i} a^T x$ in (8.20) with the above expression for $\gamma_i(x)$. Note that the variables y_j are introduced in (8.24) to represent the modules $|x_j|$. Therefore, for all non-negative variables x_j, it is better to substitute x_j for y_j. $\square$

If the value of the parameter q_i is non-negative integer, then the first LP in (8.25) always has an integer solution. and, therefore, the definition of $\mathscr{A}_i$ means that no more than q_i of uncertain elements a_{ij} ($\alpha_{ij} > 0$) in row i may take values other than $\bar{a}_{ij}$.

Example 8.3 *We need to solve the following robust IP*

$$\begin{array}{l}
3x_1 + 2x_2 + 2x_3 \to \max,\\
x_1 + \ \ x_2 + \ \ x_3 \le 3,\\
x_1 + \ \ x_2 \qquad\ \ \le 2,\\
\quad x_1, x_2, x_3 \in \mathbb{Z}_+,
\end{array} \qquad (8.26)$$

when, in each inequality, at most one non-zero coefficient can vary by not more than one.

Solution. In this example, $q_1 = q_2 = 1$, $\alpha_{1,1} = \alpha_{1,2} = \alpha_{1,3} = 1$ $\alpha_{2,1} = \alpha_{2,2} = 1$ and $\alpha_{2,3} = 0$. Now, let us write down (8.24) applied to our instance:

$$\begin{aligned}
-3x_1 - 2x_2 - 2x_3 &\to \min,\\
x_1 + x_2 + x_3 + v_1 + w_{11} + w_{12} + w_{13} &\le 3,\\
x_1 + x_2 + v_2 + w_{21} + w_{22} &\le 2,\\
v_1 + w_{11} \ge x_1,\ v_1 + w_{12} \ge x_2,\ v_1 + w_{13} &\ge x_3,\\
v_2 + w_{21} \ge x_1,\ v_2 + w_{22} &\ge x_2,\\
x_1, x_2, x_3 &\in \mathbb{Z}_+,\\
v_1, v_2 &\ge 0,\\
w_{11}, w_{12}, w_{13}, w_{21}, w_{22} &\ge 0.
\end{aligned} \qquad (8.27)$$

Note that here we have excluded the variables y_j since all the variables x_j are non-negative and, consequently, for any optimal solution, $y_j = x_j$ for $j = 1,2,3$.

It is easy to verify that an optimal solution to (8.27) has the following components:

$$x_1 = 1,\ x_2 = 0,\ x_3 = 1, \quad v_1 = v_2 = 1,$$
$$w_{11} = w_{12} = w_{13} = w_{21} = w_{22} = 0.$$

Consequently, the point $x^* = (1,0,1)^T$ is a solution to our robust program. It is worth noting that x^* would not be an optimal solution to (8.26) if this program were not robust. □

8.7.4 Robust Single-Product Lot-Sizing Problem

Let us consider again a single-product version of the lot-sizing problem studied in Sect. 1.7.1. A firm is producing some product. The planning horizon consists of T periods, where period t starts at time $t-1$ and ends at time t, $t = 1,\dots,T$. Inventory of the product in the warehouse before the start of the planning horizon is s_0. For each period $t = 1,\dots,T$, we know

- u_t^p: production capacity (in product units);
- u_t^s: storage capacity (in product units);
- f_t: fixed production cost;
- c_t: unit production cost;
- h_t: unit storage cost;
- p_t: unit shortage cost (penalty for not supplying a unit of product to satisfy the demands in the first t periods).

In period $t = 1,\dots,T$, the demand for product, d_t is an uncertain parameter that takes values from an interval $[\bar{d}_t - \alpha_t, \bar{d}_t + \alpha_t]$. To rule out large deviations in the cumulative demands in all periods, we impose the following restrictions on the values of uncertain parameters:

$$\sum_{\tau=1}^{t} \frac{|d_\tau - \bar{d}_\tau|}{\alpha_\tau} \le q_t, \quad t = 1,\dots,T.$$

It is required that the *budgets of uncertainty*, q_t, are increasing in t, reflecting the fact that uncertainty increases with the number of periods passed. It is also assumed that $q_t - q_{t-1} \le 1$ for all $t = 2,\dots,T$, which means that the budget of uncertainty at any period does not exceed the number of involved uncertain parameters.

It is necessary to determine how many product units to produce in each period in order to minimize the maximum (over all possible demands) total production, storage and shortage cost during the planning horizon.

For $t = 1,\dots,T$, we introduce the following variables:

- x_t: amount of product produced in period t;
- s_t: amount of product stored in the warehouse at the end of period t;
- $y_t = 1$ if the product is produced in period t, and $y_t = 0$ otherwise.

It is important to notice that the decision on the production level, x_t, in period t is taken at the very beginning of period t (at time $t-1$), and the demand in this period will be known only at the end of the period (at time t).

The dynamic of stock during the planning horizon is described by the following balance equations:

$$s_{t-1} + x_t = d_t + s_t, \quad t = 1, \dots, T. \tag{8.28}$$

Let us note that here s_0 is not a variable but a constant. Each balance equation in (8.28) relates two neighboring periods: the amount of product, s_{t-1}, in the warehouse at the end of period $t-1$ plus the amount, x_t, produced in period t equals the demand, d_t, in period t plus the amount, s_t, stored in the warehouse at the end of period t.

Using (8.28) we can get explicit expressions for the stock variables:

$$s_t = s_0 + \sum_{\tau=1}^{t} (x_\tau - d_\tau), \quad t = 1, \dots, T. \tag{8.29}$$

It is important to notice that, from the point of view of robust optimization, the balance relations written in Form (8.29) are preferable to those written in Form (8.28). This is because in (8.29) the expression for s_t contains all uncertain parameters, $d_1, \dots, d_t$, affecting the value of s_t.

Having determined

$$\begin{aligned}\mathscr{A}_t \stackrel{\text{def}}{=} \Big\{(d_1, \dots, d_t) : &\sum_{\tau=1}^{t} \frac{|d_\tau - \bar{d}_\tau|}{\alpha_\tau} \le q_t,\\ &\bar{d}_\tau - \alpha_\tau \le d_\tau \le \bar{d}_\tau + \alpha_\tau \text{ for } \tau = 1, \dots, t\Big\}\end{aligned}$$

for $t = 1, \dots, T$, we can write the following robust problem:

$$\sum_{t=1}^{T} (f_t y_t + c_t x_t + z_t) \to \min, \tag{8.30a}$$

$$s_0 + \sum_{\tau=1}^{t} (x_\tau - d_\tau) \le \frac{1}{h_t} z_t, \quad (d_1, \dots, d_t) \in \mathscr{A}_t,\ t = 1, \dots, T, \tag{8.30b}$$

$$-s_0 - \sum_{\tau=1}^{t} (x_\tau - d_\tau) \le \frac{1}{p_t} z_t, \quad (d_1, \dots, d_t) \in \mathscr{A}_t,\ t = 1, \dots, T, \tag{8.30c}$$

$$s_0 + \sum_{\tau=1}^{t} (x_\tau - d_\tau) \le u_t^s, \quad (d_1, \dots, d_t) \in \mathscr{A}_t,\ t = 1, \dots, T, \tag{8.30d}$$

$$0 \le x_t \le u_t^p y_t, \quad t = 1, \dots, T, \tag{8.30e}$$

$$y_t \in \{0, 1\}, \quad t = 1, \dots, T. \tag{8.30f}$$

Objective (8.30a) is to minimize the total expenses over all T periods. Here, in view of (8.29), (8.30b) and (8.30c), each variable z_t represents the value of $\max\{h_t s_t, -p_t s_t\}$, which is either the cost of storing s_t product units in period t

if $s_t \ge 0$, or, if $s_t < 0$, the penalty for not supplying $-s_t$ product units in the first t periods. Inequalities (8.30d) express the storage capacity restrictions. Inequalities (8.30e) impose the production capacity restrictions and the implications: $y_t = 0 \Rightarrow x_t = 0$.

Theorem 8.5. *The robust single-product lot-sizing problem* (8.30) *is equivalent to the following MIP:*

$$\sum_{t=1}^{T} (f_t y_t + c_t x_t + z_t) \to \min, \tag{8.31a}$$

$$s_0 + \sum_{\tau=1}^{t} (x_\tau - \bar{d}_\tau) + q_t v_t + \sum_{\tau=1}^{t} w_{t\tau} \le \frac{1}{h_t} z_t, \quad t = 1, \dots, T, \tag{8.31b}$$

$$-s_0 - \sum_{\tau=1}^{t} (x_\tau - \bar{d}_\tau) + q_t v_t + \sum_{\tau=1}^{t} w_{t\tau} \le \frac{1}{p_t} z_t, \quad t = 1, \dots, T, \tag{8.31c}$$

$$s_0 + \sum_{\tau=1}^{t} (x_\tau - d_\tau) + q_t v_t + \sum_{\tau=1}^{t} w_{t\tau} \le u_t^s, \quad t = 1, \dots, T, \tag{8.31d}$$

$$v_t + w_{t\tau} \ge \alpha_t, \quad \tau = 1, \dots, t, \quad t = 1, \dots, T, \tag{8.31e}$$

$$0 \le x_t \le u_t^p y_t, \quad t = 1, \dots, T, \tag{8.31f}$$

$$z_t,\ v_t \ge 0, \quad t = 1, \dots, T, \tag{8.31g}$$

$$w_{t\tau} \ge 0, \quad \tau = 1, \dots, t, \quad t = 1, \dots, T, \tag{8.31h}$$

$$y_t \in \{0, 1\}, \quad t = 1, \dots, T. \tag{8.31i}$$

Proof. We cannot apply Theorem 8.4 directly because all uncertain parameters d_t are not in the constraint matrix but their linear combinations are the constant terms in the inequalities (8.30b), (8.30c) and (8.30d). Nevertheless, we can replace in (8.30b) and (8.30d) each occurrence of any uncertain parameter d_τ with $-d_\tau \gamma_\tau^-$, where γ_τ^- is a new variable set to -1. Similarly, we replace in (8.30c) each occurrence of any uncertain parameter d_τ with $d_\tau \gamma_\tau^+$, where γ_τ^+ is a new variable set to 1. Then we apply Theorem 8.4 to this modified version of (8.30) to obtain an equivalent MIP that is transformed into (8.31) after substituting -1 for any variable γ_τ^-, and 1 for any variable γ_τ^+. $\square$

8.8 Notes

The most comprehensively stochastic programming is presented in [28, 113]. A simple and accessible introduction to the subject is given in [77].

***Sect.* 8.1.** The deterministic equivalent (8.4) of the two-stage model (8.1) is also known from linear stochastic programming (see [28, 77, 113]).

***Sect.* 8.2.** Benders' decomposition applied to MIPs was described in [22].

***Sect.* 8.3.** Theorem 8.2 was proved in [115]. The problem of risk measuring in stochastic programming models with integer variables is discussed in [124]. The application from Sect. 8.3.2 is explored in more detail in [5].

***Sect.* 8.4.** A more detailed introduction to multi-stage models of stochastic integer programming is given in [116]. Synthetic options are considered in [146]. The model of yield management in airline industry was adapted from [137].

***Sect.* 8.7.** Theorem 8.4 was proved in [26]. The single-product lot-sizing robust problem was studied in [27]. The book [24] and the survey [25] are devoted to robust optimization and its applications.

***Sect.* 8.9.** See [28] if you have problems with Exercise 8.2.

8.9 Exercises

8.1. You want to invest \$50000. Today the XYZ shares are sold at \$20 per share. A \$700 European option gives the right (but does not obligate) in six months to buy 100 of XYZ shares at \$15 per share. In addition, six-month risk-free bonds with a face value of \$100 are now sold at \$90. You decided not to buy more than 20 options.

Six months later, three equally likely scenarios for the XYZ share price are possible: 1) the price will not change; 2) the price will rise to \$40; 3) the price will drop to \$12.

Formulate and solve three MIPs in which you want to form a portfolio in order to maximize:

a) expected income;
b) expected income provided that the income must not be less than \$2000 for any of three scenarios;
c) risk-free income that is determined as the income in the worst of three possible scenarios.

Compare optimal solutions to your three models.

8.2. A newspaper seller decides how many newspapers to buy at a price of α to sell them at $\alpha+\beta$, provided that the demand, u, is a random variable with a distribution function G. Seller's goal is to maximize his profit. Solve this stochastic programming problem.

8.3. Prove that, for a fixed x, the function $f(x,\omega)$ defined by (8.3) is in fact a random variable.

8.4. Specify how to calculate VaR_α and CVaR_α for a discrete probability space when all events are equally likely.

8.5. Write explicitly (as in Example 8.1) and then solve Benders' reformulation for the next MIP:

$$\begin{aligned} 2x_1 + 3x_3 + 6y_1 + 4y_2 &\to \max, \\ x_1 + 2x_3 + 4y_3 + 3y_3 &\le 8, \\ -x_1 + 3x_3 - 2y_1 + 4y_2 &\le 10, \\ x_1, x_2 \in \mathbb{Z}_+,\ x_1 &\le 4, \\ y_1, y_2 &\ge 0. \end{aligned}$$

8.6. Write Benders' reformulation for (1.20), which is a **MIP** formulation of the single-product lot-sizing problem.

8.7. The *robust optimization problem with scenario-type uncertainties* is formulated as follows:

$$\min_{0 \le k \le K} \max\{c_k^T x :\ x \in X\}, \tag{8.32}$$

where $X \subseteq \mathbb{R}^n$, and $c_k \in \mathbb{R}^n$ is an objective vector for scenario k, $k = 1, \dots, K$.

Explain why Problem (8.32) can be **NP**-hard even for those sets X for which the problem

$$\max\{c^T x :\ x \in X\}$$

is polynomially solvable.

8.8. An *LP with probability constraints* is written as follows:

$$\begin{aligned} c^T x &\to \max, \\ Ax &\le b, \\ \mathbb{P}\{Hx \ge \xi(\omega)\} &\ge \alpha, \end{aligned} \tag{8.33}$$

where x is a vector of n variables, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $\alpha \in (0,1)$, an $m \times n$-matrix A and an $s \times n$-matrix H are deterministic parameters, ω is an elementary event from a probability space $(\Omega, \mathscr{A}, \mathbb{P})$, and $\xi : \Omega \to \mathbb{R}^s$ is a random vector. Any feasible solution $\bar{x}$ to (8.33) is a solution to the system $Ax \le b$, and the probability that $\bar{x}$ satisfies the system of random inequalities $Hx \ge \xi(\omega)$ is at least α.

Prove that for a finite probability space, when $\Omega = \{\omega_1, \dots, \omega_K\}$ is a finite set and event (scenario) ω_k occurs with probability p_k ($k = 1, \dots, K$), (8.33) can be formulated as a MIP.

8.9. Consider a variation of the problem of designing a telecommunication network from Sect. 2.7 when $d \in \mathbb{R}^E_+$ is a random demand vector that takes a value of $d^k \in \mathbb{R}^E_+$ with probability p_k, $k = 1, \dots, K$. We also assume that Ineqs. (2.9c) and (2.9d) must be satisfied with probability $\alpha \in (0,1)$.

Formulate this variant of the telecommunication network design problem as a MIP.

8.10. Reformulate the short-term financial management problem from Sect. 2.11 in such a way that it can be used for medium and long-term planning. Write down a stochastic programming model and its deterministic equivalent.

References

1. Abara, J.: Applying integer linear programming to the fleet assignment problem. Interfaces **19**, 20–28 (1989)
2. van den Akker, M., Van Hoesen, P.M., Savelsbergh, M.W.P.: A polyhedral approach to single-machine scheduling problems. Math. Program. **85**, 541–572 (1999)
3. Alevras, D., Grötschel, M., Wessäly, R.: Capacity and survivability models for telecommunication networks. Tech. Rep. Technical Report SC 97-22, Konrad-Zuse-Zentrum für Informationstechnik, Berlin (1997)
4. Andersen, E., Andersen, K.: Presolving in linear programming. Math. Program. **71**, 221–245 (1995)
5. Anderson, F., Mausser, H., Rosen, D., Uryasev, S.: Credit risk optimization with conditional value-at-risk criterion. Math. Program. **89**, 273–291 (2001)
6. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Finding cuts in the TSP. Tech. Rep. DIMACS Technical Report 95-05, Rutgers University, New Brunswick, NJ (1995)
7. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: Implementing the Dancig-Fulkerson-Johnson algorithm for large traveling salesman problem. Math. Program. **97**, 91–153 (2003)
8. Atamtürk, A., Nemhauser, G.L., Savelsberg, M.W.P.: Conflict graphs in solving integer programming problems. European Journal of Oper. Res. **121**, 40–45 (2000)
9. Atamtürk, A., Rajan, D.: On splittable and unsplittable flow capacitated network design arc-set polyhedra. Math. Program. **92**, 315–333 (2002)
10. Balas, E.: Facets of the knapsack polytope. Math. Program. **8**, 146–164 (1975)
11. Balas, E.: Disjunctive programming. Annals of Discrete Mathematics **5**, 3–51 (1979)
12. Balas, E., Bockmayr, A., Pisaruk, N., Wolsey, L.: On unions and dominants of polytopes. Math. Program. **99**, 223–239 (2004)
13. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. Math. Program. **58**, 295–324 (1993)
14. Balas, E., Ceria, S., Cornuéjols, G., Natraj, N.: Gomory cuts revisited. Oper. Res. Lett. **19**, 1–9 (1996)
15. Balinski, M.L.: On finding integer solution to linear programs. Tech. rep., Mathematica, Princeton, N.J. (1964)
16. Balinski, M.L., Ouandt, R.: On an integer program for a delivery problem. Oper. Res. **12**, 300–304 (1964)
17. Barany, I., Van Roy, T., Wolsey, L.A.: Uncapacitated lot-sizing: the convex hull of solutions. Math. Program. Study **22**, 32–43 (1984)
18. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.: Branch-and-price: column generation for solving huge integer programs. Oper. Res. **46**, 316–329 (1998)
19. Beale, E.M.L., Tomlin, J.A.: Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. In: Proceedings of the Fifth Annual Conference on Operational Research, pp. 447–454. J. Lawrence (ed.), Tavistock Publications (1970)
20. Beasley, J.E.: An exact two-dimensional non-guillotine cutting tree search procedure. Oper. Res. **33**, 49–64 (1985)
21. Belvaux, G., Boissin, N., Sutter, A., Wolsey, L.A.: Optimal placement of add/drop multiplexers: static and dynamic models. European Journal of Oper. Res. **108**, 26–35 (1998)
22. Benders, J.F.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik **4**, 238–252 (1962)
23. Benichou, M., Gauthier, J., Girodet, P., Hentges, G., Ribiere, G., Vincent, O.: Experiments in mixed-integer programming. Math. Program. **1**, 76–94 (1971)
24. Bental, A., Ghaoui, L.E., Nemirovski, A.: Robust Optimization. Princeton University Press (2009)
25. Bertsimas, D., Brown, D.B., Caramanis, C.: Theory and applications of robust optimization. SIAM Rev. **53**(3), 464–501 (2011)
26. Bertsimas, D., Sim, M.: The price of robustness. Oper. Res. **52**, 35–53 (2004)

27. Bertsimas, D., Thiele, A.: A robust optimization approach to supply chain management. In: D. Bienstock, G. Nemhauser (eds.) Integer Programming and Combinatorial Optimization. IPCO 2004. Lecture Notes in Computer Science, vol. 3064, pp. 86–100. Springer, Berlin, Heidelberg (2004)
28. Birge, J.R., Louveaux, F.V.: Introduction to stochastic programming. Springer Verlag, New York (2011)
29. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge University Press (2004)
30. Brearley, A.L., Mitra, G., Williams, H.P.: Analysis of mathematical programming problems prior to applying the simplex algorithm. Math. Program. **8**, 54–83 (1975)
31. Burdett, C.A., Johnson, E.L.: A subadditive approach to solve linear integer programs. Ann. Discrete Math. **1**, 117–144 (1977)
32. Caprara, A., Fischetti, M.: 0,1/2-chvatal-gomory cuts. Math. Program. **74**, 221–236 (1996)
33. Charnes, A.: Measuring the efficiency of decision-making units. European Journal of Operational Research **2**, 429–444 (1978)
34. Chernikov, S.N.: Linear Inequalities (in Russian). Nauka, Moscow (1968)
35. Chvátal, V.: Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Math. **4**, 305–337 (1973)
36. Chvátal, V.: Edmonds polytopes and a weakly hamiltonian graphs. Math. Program. **5**, 29–40 (1973)
37. Chvátal, V.: Linear Programming. Freeman, New York (1983)
38. Cordeau, J.F., Laporte, G., Savelsbergh, M.W., Vigo, D.: Chapter 6 vehicle routing. In: C. Barnhart, G. Laporte (eds.) Transportation, *Handbooks in Operations Research and Management Science*, vol. 14, pp. 367–428. Elsevier (2007)
39. Cornuéjols, G.: Valid inequalities for mixed integer linear programs. Math. Program. **112**, 3–44 (2008)
40. Cornuéjols, G., Fisher, M.L., Nemhauser, G.L.: Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. Management Science **23**, 789–810 (1977)
41. Dakin, R.: Valid inequalities for mixed integer linear programs. Computer Journal **8**, 250–255 (1965)
42. Dancig, G.: Linear programming and extentions. Princeton University Press, Princeton (1963)
43. Dancig, G., Fulkerson, D., Johnson, S.: On a linear programming combinatorial approach to the traveling salesman problem. Oper. Res. **7**, 58–66 (1959)
44. Dancig, G.B., Fulkerson, D., Johnson, S.: Solution of a large-scale traveling salesman problem. Oper. Res. **2**, 393–410 (1954)
45. Dantzig, G., Wolfe, P.: Decomposition principle for linear programs. Oper. Res. **8**, 101–111 (1960)
46. Desrosiers, J., Dumas, Y., Solomon, M., Soumis, F.: Time constrained routing and scheduling. Handbooks in operations research and management science **8**, 35–139 (1995)
47. Dyer, M., Wolsey, L.: Formulating the single-machine sequencing problem with release dates as a mixed integer program. Discrete Appl. Math. **26**, 255–270 (1990)
48. Edmonds, J.: Paths, trees and flowers. Canadian Journal of Mathematics **17**, 449–467 (1965)
49. Edmonds, J., Giles, R.: A min-max relations for submodular functions on graphs. Ann. Discrete Math. **1**, 185–204 (1977)
50. Eisenbrand, F.: On the membership problem for the elementary closure of a polyhedron. Combinatorica **19**, 297–300 (1999)
51. Forrest, J.J., Goldfarb, D.: Steepest-edge simplex algorithms for linear programming. Math. Program. **57**, 341–374 (1992)
52. Fulkerson, D.R.: Blocking and anti-blocking pairs of polyhedra. Math. Program. **1**, 160–194 (1971)
53. Gavish, B., Graves, S.: The traveling salesman problem and related problems. Tech. rep., Graduate School of Management, University of Rochester, New York (1979). Working Paper
54. Gilmore, P.C., Gomory, R.E.: A linear programming approach to cutting stock problem. Oper. Res. **9**, 849–859 (1961)

55. Gilmore, P.C., Gomory, R.E.: A linear programming approach to cutting stock problem: Part ii. Oper. Res. **11**, 863–888 (1963)
56. Gilmore, P.C., Gomory, R.E.: The theory and computation of knapsack functions. Oper. Res. **14**, 1045–1077 (1966)
57. Goldfarb, D., Reid, J.K.: A practical steepest-edge simplex algorithm. Math. Program. **12**, 361–371 (1977)
58. Gomory, R.E.: Outline of an algorithm for integer solutions to linear programs. Bull. Amer. Soc. **64**, 275–278 (1958)
59. Gomory, R.E.: An algorithm for the mixed integer problem. Tech. Rep. Technical Report RM-2597, The RAND Cooperation (1960)
60. Gomory, R.E.: Solving linear programming problems in integers. In: Proceedings of Symposia in Applied Mathematics, vol. 10 (1960)
61. Gomory, R.E.: Some polyhedra related to corner problems. Linear Algebra and its Applications **2**, 451–588 (1969)
62. Grötschel, M., Jünger, M., Reinelt, G.: A cutting plane algorithm for the linear ordering problem. Oper. Res. **32**, 1195–1220 (1984)
63. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatorica **1**, 169–197 (1981)
64. Grötschel, M., Lovász, L., Schrijver, A.: Geometric algorithms and combinatorial optimization. Springer, Berlin (1988)
65. Grötschel, M., Padberg, M.: On the symmetric travellling salesman problem ii: lifting theorems and facets. Math. Program. **16**, 281–302 (1979)
66. Grunbaum, B.: Convex polytopes. Wiley, New York (1967)
67. Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P.: Lifted cover inequalities for 0-1 integer programs: computation. INFORMS J. Comput. **10**, 427–437 (1998)
68. Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P.: Lifted flow cover inequalities for mixed 0-1 programs. Math. Program. A **85**, 436–467 (1999)
69. Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P.: Sequence independent lifting in mixed integer programming. J. Combinat. Optim. **4**, 109–129 (2000)
70. Guignard, M., Spielberg, K.: Logical reduction methods in zero-one programming. Oper. Res. **29**, 49–74 (1981)
71. Gusfield, D.: Very simple method for all pairs network flow analysis. SIAM J. Comput. **19**, 143–155 (1990)
72. Hammer, P.L., Johnson, E.L., Peled, U.N.: Facets of regular 0-1 polytopes. Math. Program. **8**, 179–206 (1975)
73. Heller, I., Tompkins, C.B.: An extension of a theorem of dancig's. In: Linear inequalities and related systems, ed H.W. Kuhn and A.W. Tucker, pp. 247–252. Princeton University Press, Princeton, N.J. (1956)
74. Hoffman, K., Padberg, M.: Improving representations of zero-one linear programs for branch-and-cut. ORSA Journal of Computing **3**, 121–134 (1991)
75. Ibarra, O.H., Kim, C.E.: Fast approximations algorithms for the knapsack and sum of subset problems. Journal of the ACM **22**, 463–468 (1975)
76. Johnson, E.L.: Modeling and strong linear programs for mixed integer programming. In: Wallace S.W. (eds) Algorithms and Model Formulations in Mathematical Programming. NATO ASI Series (Series F: Computer and Systems Sciences), vol. 51, pp. 1–41. Springer, Berlin, Heidelberg (1989)
77. Kall, P., Wallace, S.: Stochastic programming. Wiley (1994)
78. Kallenberg, L.C.M.: Linear programming and finite markovian control problems. Tech. Rep. 148, Mathematisch Centrum, Math. Centre Tract, Amsterdam (1983)
79. Karger, D.R.: Minimum cuts in near-linear time. Journal of the ACM **47**, 46–76 (2000)
80. Kaufmann, A., Henry-Labordére, A.: Méthodes et modeles de la recherche operationnelle. Dunon, Paris-Bruxelles-Montreal (1974)
81. Khachian, L.G.: Complexity of linear programming problems (in Russian). Moscow (1987)
82. Kondili, E., Pantelides, C.C., Sargent, R.W.H.: A general algorithm for short-term scheduling of batch operations – I. MILP formulation. Computers chem. Engng. **17**, 211–227 (1993)

83. Land, A.H., Doig, A.G.: An automatic method for solving discrete programming problems. Econometrica **28**, 497–520 (1960)
84. Laporte, G., Nobert, Y.: A branch and bound algorithm for the capacitated vehicle routing problem. OR Spektrum **5**, 77–85 (1983)
85. Letchford, A.L.: On disjunctive cuts for combinatorial optimization. Journal of Combinatorial Optimization **5**, 299–315 (2001)
86. Letchford, A.L.: Totally tight chvátal-gomory cuts. Oper. Res. Lett. **30**, 71–73 (2002)
87. Lovász, L., Schrijver, A.: Cones of matrices and set-functions and 0-1 optimization. SIAM J. Optim. **1**, 166–190 (1991)
88. Marchand, H., Martin, A., Weismantel, R., Wolsey, L.: Cutting planes in integer and mixed integer programming. Discrete Appl. Math. **123**, 397–446 (2002)
89. Marchand, H., Wolsey, L.A.: Aggregation and mixed integer rounding to solve mips. Oper. Res. **49**, 363–371 (2001)
90. Markowitz, H.: Portfolio Selection: Efficient Diversification of Investments. Wiley, New York (1959)
91. Miller, A.J., Wolsey, L.A.: Tight formulations for some simple mixed integer programs and convex objective integer programs. Math. Program. **98**, 73–88 (2003)
92. Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulations and the traveling salesman problem. J. Assoc. Comput. Mach. **7**, 326–329 (1960)
93. Minoux, M.: Optimum synthesis of a network with non-simultaneous multicommodity flow requirements. In: P. Hansen (ed.) Studies on Graphs and Discrete Programming, pp. 269–277. North-Holland Publishing Company (1981)
94. Minoux, M.: Programmation Mathémattique. Bordas et C.N.E.T.-E.N.S.T., Paris (1989)
95. Murty, R.G., Yu, F.T.: Linear complementarity, linear and nonlinear programming (internet edition). http://ioe.engin.umich.edu/people/fac/books/murty/linear_complementarity_webbook (1993)
96. Naddef, D.: Polyhedral theory and branch-and-cut algorithms for the symmetric tsp. In: G. Gutin, A. Punnen (eds.) The traveling salesman problem and its variations, pp. 29–116. Kluwer Academic Publishers (2002)
97. Nagamochi, H., Ibaraki, T.: Computing edge connectivity in multigraphs and capacitated graphs. SIAM J. Disc. Math. **5**, 54–66 (1992)
98. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. Wiley (1988)
99. Nemhauser, G.L., Wolsey, L.A.: A recursive procedure to generate all cuts for 0–1 mixed integer programs. Math. Program. **46**, 379–390 (1990)
100. Nisan, N.: Bidding and allocation in combinatorial auctions. In: Proceedings ACM Conference on Electronic Commerce (EC-00), pp. 1–12. Minneapolis, MN (2000)
101. Orgler, Y.: An unequal period model for cash management decisions. Management Science **16**, B77–B92 (1969)
102. Padberg, M.: Linear Optimization and Extensions. Springer-Verlag, Berlin, Heidelberg (1995)
103. Padberg, M.W.: On the facial structure of set packing polyhedra. Math. Program. **5**, 199–215 (1973)
104. Padberg, M.W.: A note on zero-one programming. Oper. Res. **23**, 833–837 (1975)
105. Padberg, M.W., Rao, M.R.: Odd minimum cut-sets and b-matchings. Math. Oper. Res. **7**, 67–80 (1982)
106. Padberg, M.W., Rinaldi, G.: Optimization of a 532 city symmetric traveling salesman problem by branch and cut. Oper. Res. Lett. **6**, 1–7 (1987)
107. Padberg, M.W., Rinaldi, G.: Facet identification for the symmetric traveling salesman polytope. Math. Program. **47**, 219–257 (1990)
108. Padberg, M.W., Van Roy, T.J., Wolsey, L.A.: Valid linear inequalities for fixed charge problems. Oper. Res. **33**, 842–861 (1985)
109. Papadimitriou, C.H.: Computational complexity. Addison-Wesley Publishing Company (1994)
110. Papadimitriou, C.H., Stieglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, NJ (1982)

111. Pinedo, M.L.: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Springer (2012). Discussion of the basic properties of scheduling models, provides an up-to-date coverage of important theoretical models in the scheduling literature as well as significant scheduling problems that occur in the real world.
112. Precido-Walters, F., Rardin, R., Langer, M., Thai, V.: A coupled column generation, mixed integer approach to optimal planning of intensity modulated radiation therapy for cancer. Math. Program. **101**, 319–338 (2004)
113. Prékopa, A.: Stochastic programming. Kluwer Academic Publishers, Dordrecht (1995)
114. Queyranne, M.A., Schulz, A.S.: Polyhedral approaches to machine scheduling. Tech. Rep. Technical report 408/1994, Institut für Mathematik, Technische Universität Berlin, Berlin (1994)
115. Rockafellar, R.T., Uryasev, S.: Optimization of conditional value-at-risk. J. Risk **2**, 21–41 (2000)
116. Römisch, W., Schultz, R.: Multistage stochastic integer programs: an introduction. In: M. Grötschel, S.O. Krumke, J. Rambau (eds.) Online Optimization of Large Scale Systems, pp. 581–600. Springer, Berlin, Heidelberg (2001)
117. Saigal, R.: Linear programming: a modern integrated analysis. Kluwer Academic Publishers, Boston/Dordrecht/London (1995)
118. Savelsbergh, M.W.P.: A branch and price algorithm for the generalized assignment problem. Tech. Rep. COC-93-02, Computational Optimization Center, Georgia Institute of Technology, Atlanta (1993)
119. Savelsbergh, M.W.P.: Preprocessing and probing techniques for mixed integer programming problems. ORSA Journal on Computing **6**, 445–454 (1994)
120. Scholl, A.: Balancing and sequencing of assembly lines. Physica-Verlag, Berlin, Heidelberg (1999)
121. Schrijver, A.: On cutting planes. Ann. Discrete Math. **9**, 291–296 (1980)
122. Schrijver, A.: Theory of linear and integer programming. Wiley (1986)
123. Schrijver, A.: Combinatorial optimization. Springer Verlag (2004)
124. Schultz, R.: Stochastic programming with integer variables. Math. Program. **97**, 285–309 (2003)
125. Shahookar, K., Mazumder, P.: VLSI cell placement techniques. ACM Computing Surveys **23**, 143–220 (1991)
126. Sheble, G.B., Fahd, G.N.: Unit commitment literature synopsis. IEEE Transactions on Power Systems **9**, 128–135 (1994)
127. Sherali, H., Adams, W.: A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. SIAM J. Discr. Math. **3**, 311–430 (1990)
128. Sönke, H., Briskorn, D.: A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Oper. Res. **207**, 1–14 (2010)
129. Suhl, U.H., Szymanski, R.: Supernode processing of mixed-integer models. Computational Optimization and Applications **3**, 317–331 (1994)
130. Sutanthavibul, S., Shragowitz, E., Rosen, J.B.: An analytical approach to floorplan design and optimization. IEEE Transactions on Computer-Aided Design **10**, 761–769 (1991)
131. Sutter, A., Vanderbeck, F., Wolsey, L.A.: Optimal placement of add/drop multiplexers: heuristic and exact algorithms. Oper. Res. **46**, 719–728 (1998)
132. Tomlin, J.A.: On scaling linear programming problems. Math. Program. **4**, 144–166 (1975)
133. Van Vyve, M., Wolsey, L.A.: Approximate extended formulations. Math. Program. **105**, 501–522 (2006)
134. Vanderbei, R.J.: Linear Programming: Foundations and Extensions. Kluwer Academic Publishers (2001)
135. Wagner, H.M., Whitin, T.M.: Dynamic version of the economic lot size model. Management Science **5**, 89–96 (1958)
136. Weismantel, R.: On the 0/1 knapsack polytope. Math. Program. **77**, 49–68 (1997)
137. Williams, H.P.: Model Building in Mathematical Programming, 5th Edition. Wiley (2013)

138. Wolsey, L.A.: Faces for a linear inequality in 0-1 variables. Math. Program. **8**, 165–178 (1975)
139. Wolsey, L.A.: Facets and strong valid inequalities for integer programs. Oper. Res. **24**, 367–372 (1976)
140. Wolsey, L.A.: Valid inequalities and superadditivity for 0/1 integer programs. Math. Oper. Res. **2**, 66–77 (1977)
141. Wolsey, L.A.: Strong formulations for mixed integer programs: a survey. Math. Program. **45**, 173–191 (1989)
142. Wolsey, L.A.: Integer Programming. Wiley (1998)
143. Wolsey, L.A.: Solving multi-item lot-sizing problems with an mip solver using classification and reformulation. Management Science **48**, 1587–1602 (2002)
144. Wolsey, L.A.: Strong formulations for mixed integer programs: valid inequalities and extended formulations. Math. Program. **97**, 423–447 (2003)
145. Yannakakis, M.: Expressing combinatorial optimization problems by linear programs. J. Comp. Syst. Sci. **43**, 441–466 (1991)
146. Zhao, Y., Ziemba, W.T.: The russell-yasuda model: A stochastic programming model using an endogenously determined worst case risk measure for dynamic asset allocation. Math. Program. **89**, 293–309 (2001)

Index