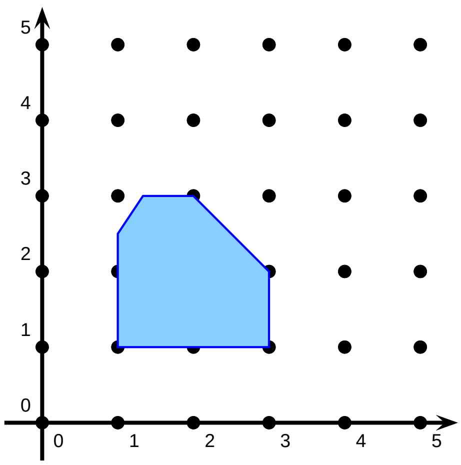
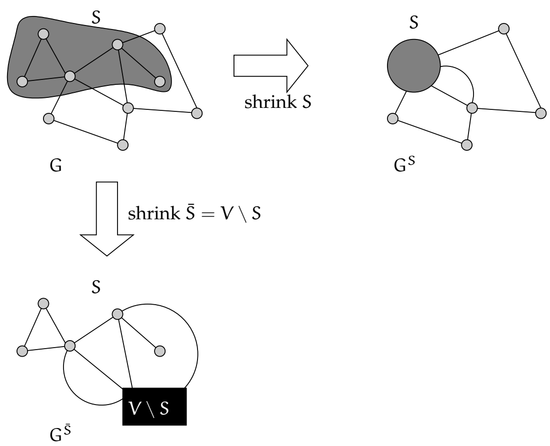
Sven O. Krumke

Integer Programming

Polyhedra and Algorithms

Draft: January 4, 2006





Technische Universität Kaiserslautern

These course notes are based on my lectures »Integer Programming: Polyhedral Theory« and »Integer Programming: Algorithms« at the University of Kaiserslautern.

I would be happy to receive feedback, in particular suggestions for improvement and notificiations of typos and other errors.

Sven O. Krumke
krumke@mathematik.uni-kl.de

Contents

1

Introduction

Linear Programs can be used to model a large number of problems arising in practice. A standard form of a Linear Program is

$$\text{(LP)} \quad \max \quad c^T x \tag{1.1a}$$
$$Ax \leq b \tag{1.1b}$$
$$x \geq 0, \tag{1.1c}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ are given vectors and $A \in \mathbb{R}^{m \times n}$ is a matrix. The focus of these lecture notes is to study extensions of Linear Programs where we are given additional integrality conditions on all or some of the variables.

Problems with integrality constraints on the variables arise in a variety of applications. For instance, if we want to place facilities, it makes sense to require the number of facilites to be an integer (it is not clear what it means to build 2.28 fire stations). Also, frequently, one can model decisions as 0-1-variables: the variable is zero if we make a negative decision and one otherwise.

1.1 Integer Linear Programs

We first state the general form of a Mixed Integer Program as it will be used throughout these notes:

Definition 1.1 (Mixed Integer Linear Program (MIP))
A ***Mixed Integer Linear Program (MIP)*** *is given by vectors $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, a matrix $A \in \mathbb{R}^{m \times n}$ and a number $p \in \{0, \dots, n\}$. The goal of the problem is to find a vector $x \in \mathbb{R}^n$ solving the following optimization problem:*

$$\textit{(MIP)} \quad \max \quad c^T x \tag{1.2a}$$
$$Ax \leq b \tag{1.2b}$$
$$x \geq 0 \tag{1.2c}$$
$$x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}. \tag{1.2d}$$

If $p = 0$, then then there are no integrality constraints at all, so we obtain the Linear Program (1.1). On the other hand, if $p = n$, then all *variables are required to be integral. In this case, we speak of an* Integer Linear Program (IP)*: we note again for later reference:*

$$\textit{(IP)} \quad \max \quad c^T x \tag{1.3a}$$
$$Ax \leq b \tag{1.3b}$$
$$x \geq 0 \tag{1.3c}$$
$$x \in \mathbb{Z}^n \tag{1.3d}$$

If in an (IP) all variables are restricted to values from the set $\mathbb{B} = \{0, 1\}$*, we have a* 0-1-Integer Linear Program *or* Binary Linear Integer Program*:*

$$\begin{aligned}
&(1.4a) & (BIP) \quad \max \quad & c^T x \\
&(1.4b) & & Ax \leq b \\
&(1.4c) & & x \geq 0 \\
&(1.4d) & & x \in \mathbb{B}^n
\end{aligned}$$

Most of the time we will be concerned with Integer Programs (IP) and Binary Integer Programs (BIP).

Example 1.2
Consider the following Integer Linear Program:

$$\begin{aligned}
&(1.5a) & \max \quad & x + y \\
&(1.5b) & & 2y - 3x \leq 2 \\
&(1.5c) & & x + y \leq 5 \\
&(1.5d) & & 1 \leq x \leq 3 \\
&(1.5e) & & 1 \leq y \leq 3 \\
&(1.5f) & & x, y \in \mathbb{Z}
\end{aligned}$$

The feasible region S of the problem is depicted in Figure 1.1. It consists of the integral points emphasized in red, namely

$$S = \{(1, 1), (2, 1), (3, 1), (1, 2), (2, 2), (3, 2), (2, 3)\}.$$

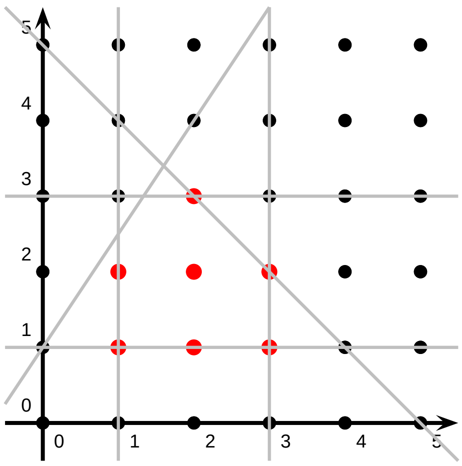


Figure 1.1: Example of the feasible region of an integer program.

◁

Example 1.3 (Knapsack Problem)
A climber is preparing for an expedition to Mount Optimization. His equipment consists of n items, where each item i has a profit $p_i \in \mathbb{Z}_+$ and a weight $w_i \in \mathbb{Z}_+$. The climber knows that he will be able to carry items of

total weight at most $b \in \mathbb{Z}_+$. He would like to pack his knapsack in such a way that he gets the largest possible profit without exceeding the weight limit.

We can formulate the KNAPSACK problem as a (BIP). Define a decision variable x_i, $i = 1, \ldots, n$ with the following meaning:

$$x_i = \begin{cases} 1 & \text{if item } i \text{ gets packed into the knapsack} \\ 0 & \text{otherwise} \end{cases}$$

Then, KNAPSACK becomes the (BIP)

$$\max \sum_{i=1}^{n} p_i x_i \tag{1.6a}$$

$$\sum_{i=1}^{n} w_i x_i \leq b \tag{1.6b}$$

$$x \in \mathbb{B}^n \tag{1.6c}$$

$\triangleleft$

In the preceeding example we have essentially identified a subset of the possible items with a 0-1-vector. Given a ground set E, we can associate with each subset $F \subseteq E$ an *incidence vector* $\chi^F \in \mathbb{R}^F$ by setting

$$\chi_e^F = \begin{cases} 1 & \text{if } e \in F \\ 0 & \text{if } e \notin F. \end{cases}$$

Then, we can identify the vector χ^F with the set F and vice versa. We will see numerous examples of this identification in the rest of these lecture notes. This identification appears frequently in the context of combinatorial optimization problems.

Definition 1.4 (Combinatorial Optimization Problem)
A ***combinatorial optimization problem*** *is given by a finite ground set* N*, a weight function* $c \colon N \to \mathbb{R}$ *and a family* $\mathcal{F} \subseteq 2^N$ *of feasible subsets of* N*. The goal is to solve*

$$\textit{(COP)} \quad \min \left\{ \sum_{j \in S} c_j : S \in \mathcal{F} \right\}. \tag{1.7}$$

Thus, we can write KNAPSACK also as (COP) by using:

$$N := \{1, \ldots, n\}$$
$$\mathcal{F} := \left\{ S \subseteq N : \sum_{i \in S} w_i \leq b \right\},$$

and $c(i) := -w_i$. We will see later that there is a close connection between combinatorial optimization problems (as stated in (1.7)) and Integer Linear Programs.

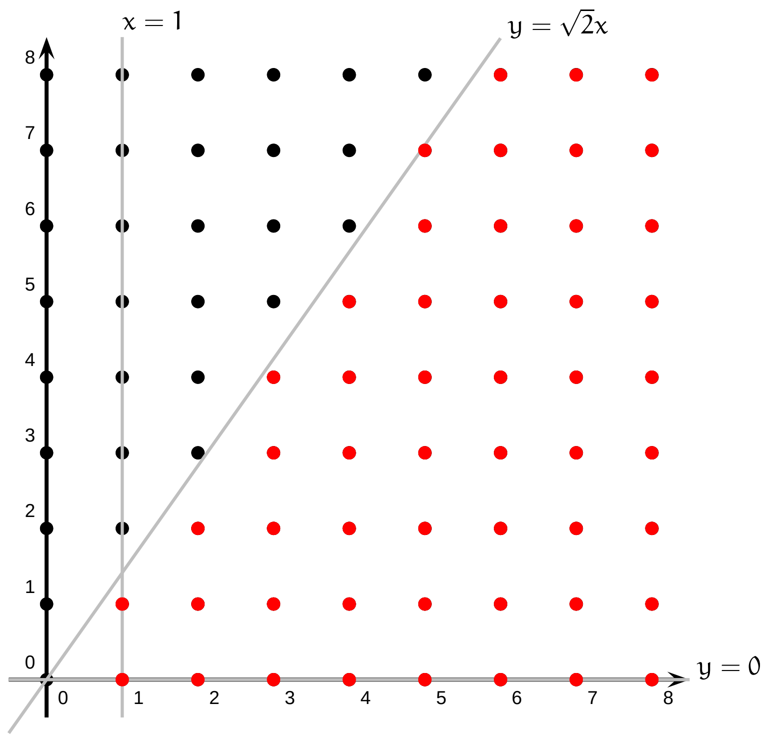


Figure 1.2: An IP without optimal solution.

1.2 Notes of Caution

Integer Linear Programs are qualitatively different from Linear Programs in a number of aspects. Recall that from the Fundamental Theorem of Linear Programming we know that, if the Linear Program (1.1) is feasible and bounded, it has an optimal solution.

Now, consider the following Integer Linear Program:

$$
\begin{aligned}
&(1.8a) && \max && -\sqrt{2}x + y \\
&(1.8b) && && -\sqrt{2}x + y \leq 0 \\
&(1.8c) && && x \geq 1 \\
&(1.8d) && && y \geq 0 \\
&(1.8e) && && x, y \in \mathbb{Z}
\end{aligned}
$$

The feasible region S of the IP (1.8) is depicted in Figure 1.2. The set of feasible solutions is nonempty (for instance $(1, 0)$ is a feasible point) and by the constraint $-\sqrt{2}x+y \leq 0$ the objective is also bounded from above on S. However, the IP does not have an optimal solution!

To see this, observe that from the constraint $-\sqrt{2}x + y \leq 0$ we have $y/x \leq \sqrt{2}$ and, since we know that $\sqrt{2}$ is irrational, $-\sqrt{2}x + y < 0$ for any $x, y \in \mathbb{Z}$. On the other hand, for integral x, the function $-\sqrt{2}x + \lfloor\sqrt{2}x\rfloor$ gets arbitrarily close to 0.

Remark 1.5 An IP with irrational input data can be feasible and bounded but may still not have an optimal solution.

The reason why the IP (1.8) does not have an optimal solution lies in the fact that we have used irrational data to specify the problem. We will see later that under the assumption of rational data any feasible and bounded MIP must have an optimal solution. We stress again that for standard Linear Programs *no* assumption about the input data is needed.

At first sight it might sound like a reasonable idea to simply drop the integrality constraints in an IP and to "round the corresponding" solution. But, in general, this is not a good idea for the following reasons:

- The rounded solution may be infeasible (see Figure 1.3(a)), or
- the rounded solution may be feasible but far from the optimum solution (see Figure 1.3(b)).

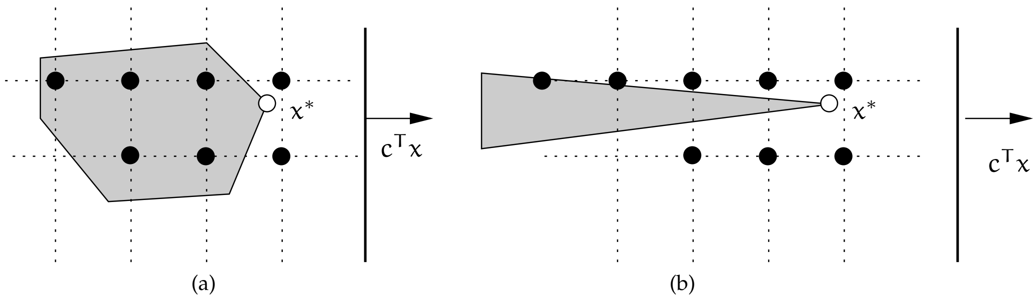


Figure 1.3: Simply rounding a solution of the LP-relaxation to an IP may give infeasible or very bad solutions.

1.3 Examples

In this section we give various examples of integer programming problems.

Example 1.6 (Assignment Problem)
Suppose that we are given n tasks and n people which are available for carrying out the tasks. Each person can carry out exactly one job, and there is a cost c_{ij} if person i serves job j. How should we assign the jobs to the persons in order to minimize the overall cost?

We first introduce binary decision variables x_{ij} with the following meaning:

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ carrys out job } j \\ 0 & \text{otherwise.} \end{cases}$$

Given such a binary vector x, the number of persons assigned to job j is exactly $\sum_{i=1}^{n} x_{ij}$. Thus, the requirement that each job gets served by exactly one person can be enforced by the following constraint:

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \text{for } j = 1, \dots, n.$$

Similarly, we can ensure that each person does exactly one job by having the following constraint:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \text{for } i = 1, \dots, n.$$

We obtain the following BIP:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} & \\
& \sum_{i=1}^{n} x_{ij} = 1 & \text{for } j = 1,\dots,n \\
& \sum_{j=1}^{n} x_{ij} = 1 & \text{for } i = 1,\dots,n \\
& x \in \mathbb{B}^{n^2}
\end{aligned}
$$

◁

Example 1.7 (Uncapacitated Facility Location Problem)
In the *uncapacitated facility location problem (*UFL*)* we are given a set of potential depots $M = \{1, \dots, m\}$ and a set $N = \{1, \dots, n\}$ of clients. Opening a depot at site j involves a fixed cost f_j. Serving client i by a depot at location j costs c_{ij} units of money. The goal of the UFL is to decide at which positions to open depots and how to serve all clients such as to minimize the overall cost.

We can model the cost c_{ij} which arises if client i is served by a facility at j with the help of binary variables x_{ij} similar to the assignment problem:

$$
x_{ij} = \begin{cases} 1 & \text{if client } i \text{ is served by a facility at } j \\ 0 & \text{otherwise.} \end{cases}
$$

The fixed cost f_j which arises if we open a facility at j can be handled by binary variables y_j where

$$
y_j = \begin{cases} 1 & \text{if a facility at } j \text{ is opened} \\ 0 & \text{otherwise.} \end{cases}
$$

Following the ideas of the assignment problem, we obtain the following BIP:

$$
\begin{aligned}
&\text{(1.9a)} \qquad \min \quad && \sum_{j=1}^{m} f_j y_j + \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} & \\
&\text{(1.9b)} && \sum_{j=1}^{m} x_{ij} = 1 && \text{for } i = 1,\dots,n \\
&\text{(1.9c)} && x \in \mathbb{B}^{nm}, y \in \mathbb{B}^{m}
\end{aligned}
$$

As in the assignment problem, constraint (1.9b) enforces that each client is served. But our formulation is not complete yet! The current constraints allow a client i to be served by a facility at j which is not opened, that is, where $y_j = 0$. How can we ensure that clients are only served by open facilities?

One option is to add the nm constraints $x_{ij} \leq y_j$ for all i, j to (1.9). Then, if $y_j = 0$, we must have $x_{ij} = 0$ for all i. This is what we want. Hence, the UFL can be formulated as follows:

$$\min \sum_{j=1}^{m} f_j y_j + \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} \tag{1.10a}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \text{for } i = 1, \ldots, n \tag{1.10b}$$

$$x_{ij} \leq y_j \qquad \text{for } i = 1, \ldots, n \text{ and } j = 1, \ldots, m \tag{1.10c}$$

$$x \in \mathbb{B}^{nm}, y \in \mathbb{B}^{m} \tag{1.10d}$$

A potential drawback of the formulation (1.10) is that it contains a large number of constraints. We can formulate the condition that clients are served only by open facilities in another way. Observe that $\sum_{i=1}^{n} x_{ij}$ is the number of clients assigned to facility j. Since this number is an integer between 0 and n, the condition $\sum_{i=1}^{n} x_{ij} \leq n y_j$ can also be used to prohibit clients being served by a facility which is not open. If $y_j = 0$, then $\sum_{i=1}^{n} x_{ij}$ must also be zero. If $y_j = 1$, we have the constraint $\sum_{i=1}^{n} x_{ij} \leq n$ which is always satisfied since there is a total of n clients. This gives us the alternative formulation of the UFL:

$$\min \sum_{j=1}^{m} f_j y_j + \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} \tag{1.11a}$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \text{for } i = 1, \ldots, n \tag{1.11b}$$

$$\sum_{i=1}^{n} x_{ij} \leq n y_j \qquad \text{for } j = 1, \ldots, m \tag{1.11c}$$

$$x \in \mathbb{B}^{nm}, y \in \mathbb{B}^{m} \tag{1.11d}$$

Are there any differences between the two formulations as far as solvability is concerned? We will explore this question later in greater detail. $\triangleleft$

Example 1.8 (Traveling Salesman Problem)
In the *traveling salesman problem* (TSP) a salesman must visit each of n given cities $V = \{1, \ldots, n\}$ exactly once and then return to his starting point. The distance between city i and j is c_{ij}. The salesman wants to find a tour of minimum length.

The TSP can be modeled as a graph problem by considering a complete directed graph $G = (V, A)$, that is, a graph with $A = V \times V$, and assigning a cost $c(i, j)$ to every arc $a = (i, j)$. A tour is then a cycle in G which touches every node in V exactly once.

We formulate the TSP as a BIP. We have binary variables x_{ij} with the following meaning:

$$x_{ij} = \begin{cases} 1 & \text{if the salesman goes from city } i \text{ directly to city } j \\ 0 & \text{otherwise.} \end{cases}$$

Then, the total length of the tour taken by the salesman is $\sum_{i,j} c_{ij}x_{ij}$. In a feasible tour, the salesman enters each city exactly once and also leaves each city exactly once. Thus, we have the constraints:

$$\sum_{j:j\neq i} x_{ji} = 1 \qquad \text{for } i = 1, \dots, n$$

$$\sum_{j:j\neq i} x_{ij} = 1 \qquad \text{for } i = 1, \dots, n.$$

However, the constraints specified so far do not ensure that a binary solution forms indeed a tour.

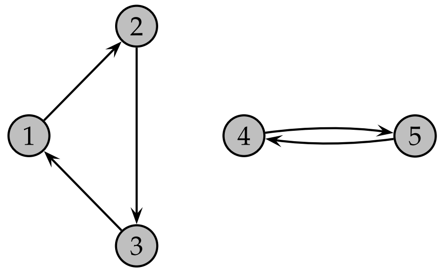


Figure 1.4: Subtours are possible in the TSP if no additional constraints are added.

Consider the situation depicted in Figure 1.4. We are given five cities and for each city there is exactly one incoming and one outgoing arc. However, the solution is not a tour but a collection of directed cycles called *subtours*.

To elminimate subtours we have to add more constraints. One possible way of doing so is to use the so-called *subtour elimination constraints*. The underlying idea is as follows. Let $\emptyset \neq S \subset V$ be a subset of the cities. A feasible tour (on the whole set of cities) must leave S for some vertex outside of S. Hence, the number of arcs that have both endpoints in S can be at most $|S| - 1$. On the other hand, a subtour which is a directed cycle for a set $\emptyset \neq S \subset V$, has exactly $|S|$ arcs with both endpoints in S. We obtain the following BIP for the TSP:

$$\text{(1.12a)} \qquad \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{(1.12b)} \qquad \sum_{j:j\neq i} x_{ij} = 1 \qquad \text{for } i = 1, \dots, n$$

$$\text{(1.12c)} \qquad \sum_{i:i\neq j} x_{ij} = 1 \qquad \text{for } j = 1, \dots, n$$

$$\text{(1.12d)} \qquad \sum_{i\in S} \sum_{j\in S} x_{ij} \leq |S| - 1 \qquad \text{for all } \emptyset \subset S \subset V.$$

$$\text{(1.12e)} \qquad x \in \mathbb{B}^{n(n-1)}$$

As mentioned before, the constraints (1.12d) are called *subtour elimination constraints*. ◁

Example 1.9 (Set-Covering, Set-Packing and Set-Partitioning Problem)
Let U be a finite ground set and $F \subseteq 2^U$ be a collection of subsets of U. There is a cost/benefit c_f associated with every set $f \in F$. In the the set covering

(set packing, set partitioning) problem we wish to find a subcollection of the sets in F such that each element in U is covered at least once (at most once, exactly once). The goal in the set covering and set partitioning problem is to minimize the cost of the chosen sets, whereas in the set packing problem we wish to maximize the benefit.

We can formulate each of these problems as a BIP by the following approach. We choose binary variables y_f, $f \in F$ with the meaning that

$$x_f = \begin{cases} 1 & \text{if set } f \text{ is chosen to be in the selection} \\ 0 & \text{otherwise.} \end{cases}$$

Let $A = (a_{uf})$ be the $|U| \times |F|$-matrix which reflects the element-set containment relations, that is

$$a_{uf} := \begin{cases} 1 & \text{if element } u \text{ is contained in set } f \\ 0 & \text{otherwise.} \end{cases}$$

Then, the constraint that every element is covered at least once (at most once, exactly once) can be expressed by the linear constraints $Ax \geq \mathbb{1}$ ($Ax \leq \mathbb{1}$, $Ax = \mathbb{1}$), where $\mathbb{1} = (1, \ldots, 1) \in \mathbb{R}^U$ is the vector consisting of all ones. $\triangleleft$

Example 1.10 (Minimum Spanning Tree Problem)
A *spanning tree* in an undirected graph $G = (V, E)$ is a subgraph $T = (V, F)$ of G which is connected and does not contain cycles. Given a cost function $c\colon E \to \mathbb{R}_+$ on the edges of G the *minimum spanning tree problem (MST-Problem)* asks to find a spanning tree T of minimum weight $c(F)$.

We choose binary indicator variables x_e for the edges in E with the meaning that $x_e = 1$ if and only if e is included in the spanning tree. The objective function $\sum_{e\in E} c_e x_e$ is now clear. But how can we formulate the requirement that the set of edges chosen forms a spanning tree?

A *cut* in an undirected graph G is a partition $S \cup \bar{S} = V$, $S \cap \bar{S} = \emptyset$ of the vertex set. We denote by $\delta(S)$ the set of edges in the cut, that is, the set of edges which have exactly one endpoint in S. It is easy to see that a subset $F \subseteq E$ of the edges forms a connected spanning subgraph (V, F) if and only if $F \cap \delta(S) \neq \emptyset$ for all subsets S with $\emptyset \subset S \subset V$. Hence, we can formulate the requirement that the subset of edges chosen forms a connected spanning subgraph by having the constraint $\sum_{e\in\delta(S)} x_e \geq 1$ for each such subset. This gives the following BIP:

$$\min \sum_{e\in E} c_e x_e \tag{1.13a}$$

$$\sum_{e\in\delta(S)} x_e \geq 1 \quad \text{for all } \emptyset \subset S \subset V \tag{1.13b}$$

$$x \in \mathbb{B}^E \tag{1.13c}$$

How do we incorporate the requirement that the edge set chosen should be without cycles? The answer is that we do not need to, as far as optimality is concerned! The reason behind that is the following: if χ^F is a feasible solution for (1.13) and F contains a cycle, we can remove one edge e from the cycle and $F \setminus \{e\}$ is still feasible. Since c is nonnegative, the vector $\chi^{F\setminus\{e\}}$ is a feasible solution for (1.13) of cost at most that of χ^F. $\triangleleft$

1.4 Literature

These notes are a revised version of [Kru04], which followed more closely the book by Laurence Wolsey [Wol98]. Classical books about Linear and Integer Programming are the books of George Nemhauser and Laurence Wolsey [NW99] and Alexander Schrijver [Sch86]. You can also find a lot of useful stuff in the books [CC$^+$98, Sch03, GLS88] which are mainly about combinatorial optimization. Section 5.3 discusses issues of complexity. A classical book about the theory of computation is the book by Garey and Johnson [GJ79]. More books from this area are [Pap94, BDG88, BDG90].

1.5 Acknowledgements

I wish to thank all students of the lecture »Optimization II: Integer Programming« at the Technical University of Kaiserslautern (winter semester 2003/04) for their comments and questions. Particular thanks go to all students who actively reported typos and provided constructive criticism: Robert Dorbritz, Ulrich Dürholz, Tatjana Kowalew, Erika Lind, Wiredu Sampson and Phillip Süß. Needless to say, all remaining errors and typos are solely my faults!

2

Basics

In this chapter we introduce some of the basic concepts that will be useful for the study of integer programming problems.

2.1 Notation

Let $A \in \mathbb{R}^{m \times n}$ be a matrix with row index set $M = \{1, \dots, m\}$ and column index set $N = \{1, \dots, n\}$. We write

$$\begin{aligned} A &= (a_{ij})_{\substack{i=1,\dots,m \\ j=1,\dots,n}} \\ A_{\cdot,j} &= \begin{pmatrix} a_{1j} \\ \vdots \\ a_{mj} \end{pmatrix} : j\text{th column of } A \\ A_{i,\cdot} &= (a_{i1}, \dots, a_{n1}) : i\text{th row of } A. \end{aligned}$$

For subsets $I \subseteq M$ and $J \subseteq N$ we denote by

$$A_{I,J} := (a_{ij})_{\substack{i \in I \\ j \in J}}$$

the submatrix of A formed by the corresponding indices. We also set

$$\begin{aligned} A_{\cdot,J} &:= A_{M,J} \\ A_{I,\cdot} &:= A_{I,N}. \end{aligned}$$

For a subset $X \subseteq \mathbb{R}^n$ we denote by

$$\text{lin}(X) := \{ x = \sum_{i=1}^{k} \lambda_i v_i : \lambda_i \in \mathbb{R} \text{ and } v_1, \dots, v_k \in X \}$$

the *linear hull* of X.

2.2 Convex Hulls

Definition 2.1 (Convex Hull)
Given a set $X \subseteq \mathbb{R}^n$, the ***convex hull*** *of X, denoted by $\text{conv}(X)$ is defined to be the set of all convex combinations of vectors from X, that is,*

$$\text{conv}(X) := \{ x = \sum_{i=1}^{k} \lambda_i v_i : \lambda_i \geq 0, \sum_{i=1}^{k} \lambda_i = 1 \textit{ and } v_1, \dots, v_k \in X \}$$

Suppose that $X \subseteq \mathbb{R}^n$ is some set, for instance X is the set of incidence vectors of all spanning trees of a given graph (cf. Example 1.10). Suppose that we wish to find a vector $x \in X$ maximizing $c^T x$.

If $x = \sum_{i=1}^{k} \lambda_i v_i \in \text{conv}(X)$ is a convex combination of the vectors $v_1, \dots, v_k$, then

$$c^T x = \sum_{i=1}^{k} \lambda_i c^T v_i \leq \max\{c^T v_i : i = 1, \dots, k\}.$$

Hence, we have that

$$\max\{\, c^T x : x \in X \,\} = \max\{\, c^T x : x \in \text{conv}(X) \,\}$$

for any set $X \subseteq \mathbb{R}^n$.

Observation 2.2 *Let* $X \subseteq \mathbb{R}^n$ *be any set and* $c \in \mathbb{R}^n$ *be any vector. Then*

$$\max\{\, c^T x : x \in X \,\} = \max\{\, c^T x : x \in \text{conv}(X) \,\}. \tag{2.1}$$

Proof: See above. □

Observation 2.2 may seem of little use, since we have replaced a discrete finite problem (left hand side of (2.1) by a continuous one (right hand side of (2.1)). However, in many cases conv(X) has a nice structure that we can exploit in order to solve the problem. It turns out that "most of the time" conv(X) is a polyhedron $\{\, x : Ax \leq b \,\}$ (see Section 2.3) and that the problem $\max\{\, c^T x : x \in \text{conv}(X) \,\}$ is a Linear Program.

Example 2.3
We return to the IP given in Example 1.2:

$$\begin{aligned} \max \quad & x + y \\ & 2y - 3x \leq 2 \\ & x + y \leq 5 \\ & 1 \leq x \leq 3 \\ & 1 \leq y \leq 3 \\ & x, y \in \mathbb{Z} \end{aligned}$$

We have already noted that the set X of feasible solutions for the IP is

$$X = \{(1,1), (2,1), (3,1), (1,2), (2,2), (3,2), (2,3)\}.$$

Observe that the constraint $y \leq 3$ is actually superfluous, but that is not our main concern right now. What is more important is the fact that we obtain the same feasible set if we *add* the constraint $x - y \geq -1$ as shown in Figure 2.1. Moreover, we have that the convex hull conv(X) of all feasible solutions for the IP is described by the following inequalities:

$$\begin{aligned} 2y - 3x &\leq 2 \\ x + y &\leq 5 \\ -x + y &\leq 1 \\ 1 \leq x &\leq 3 \\ 1 \leq y &\leq 3 \end{aligned}$$

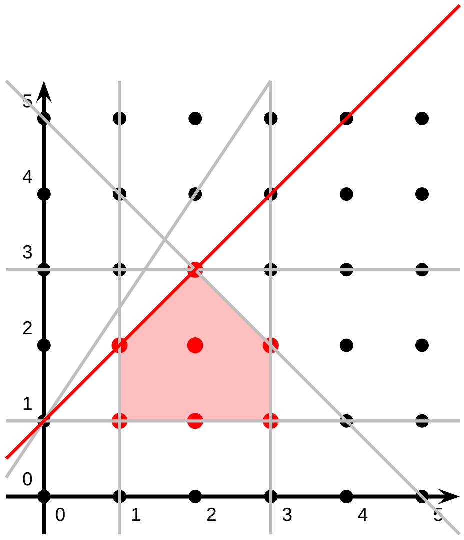


Figure 2.1: The addition of a the new constraint $x - y \geq -1$ (shown as the red line) leads to the same set feasible set.

Observation 2.2 now implies that instead of solving the original IP we can also solve the *Linear Program*

$$\begin{aligned} \max \quad & x + y \\ & 2y - 3x \leq 2 \\ & x + y \leq 5 \\ & -x + y \leq 1 \\ & 1 \leq x \leq 3 \\ & 1 \leq y \leq 3 \end{aligned}$$

that is, a *standard Linear Program* without integrality constraints. $\triangleleft$

In the above example we reduced the solution of an IP to solving a standard Linear Program. We will see later that in principle this reduction is always possible (provided the data of the IP is rational). However, there is a catch! The mentioned reduction might lead to an exponential increase in the problem size. Sometimes we might still overcome this problem (see Section 5.4).

2.3 Polyhedra and Formulations

Definition 2.4 (Polyhedron, polytope)
A ***polyhedron*** *is a subset of* $\mathbb{R}^n$ *described by a finite set of linear inequalities, that is, a polyhedron is of the form*

(2.2) $$P(A, b) := \{ x \in \mathbb{R}^n : Ax \leq b \},$$

where A *is an* $m \times n$*-matrix and* $b \in \mathbb{R}^m$ *is a vector. The polyhedron* P *is a* ***rational polyhedron*** *if* A *and* b *can be chosen to be rational. A bounded polyhedron is called* ***polytope****.*

In Section 1.3 we have seen a number of examples of Integer Linear Programs and we have spoken rather informally of a *formulation* of a problem. We now formalize the term *formulation*:

Definition 2.5 (Formulation)
A polyhedron $P \subseteq \mathbb{R}^n$ *is a* ***formulation*** *for a set* $X \subseteq \mathbb{Z}^p \times \mathbb{R}^{n-p}$, *if* $X = P \cap (\mathbb{Z}^p \times \mathbb{R}^{n-p})$.

It is clear, that in general there is an infinite number of formulations for a set X. This naturally raises the question about "good" and "not so good" formulations.

We start with an easy example which provides the intuition how to judge formulations. Consider again the set $X = \{(1,1),(2,1),(3,1),(1,2),(2,2),(3,2),(2,3)\} \subset \mathbb{R}^2$ from Examples 1.2 and 2.3. Figure 2.2 shows our known two formulations P_1 and P_2 together with a third one P_3.

$$P_1 = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} : \begin{array}{l} 2y - 3x \leq 2 \\ x + y \leq 5 \\ 1 \leq x \leq 3 \\ 1 \leq y \leq 3 \end{array} \right\}$$

$$P_2 = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} : \begin{array}{l} 2y - 3x \leq 2 \\ x + y \leq 5 \\ -x + y \leq 1 \\ 1 \leq x \leq 3 \\ 1 \leq y \leq 3 \end{array} \right\}$$

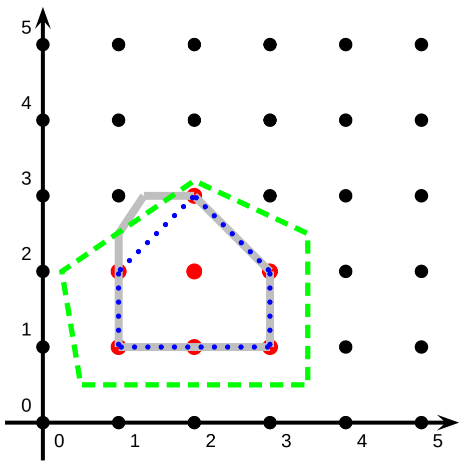


Figure 2.2: Different formulations for the integral set $X = \{(1,1),(2,1),(3,1),(1,2),(2,2),(3,2),(2,3)\}$

Intuitively, we would rate P_2 much higher than P_1 or P_3. In fact, P_2 is an *ideal formulation* since, as we have seen in Example 2.3 we can simply solve a Linear Program over P_2, the optimal solution will be an extreme point which is a point from X.

Definition 2.6 (Better and ideal formulations)
Given a set $X \subseteq \mathbb{R}^n$ *and two formulations* P_1 *and* P_2 *for* X, *we say that* P_1 *is* ***better*** *than* P_2, *if* $P_1 \subset P_2$.

A formulation P *for* X *is called* ***ideal***, *if* $P = \text{conv}(X)$.

We will see later that the above definition is one of the keys to solving IPs.

Example 2.7
In Example 1.7 we have seen two possibilities to formulate the Uncapacitated Facility Location Problem (UFL):

$$\begin{aligned}\min \sum_{j=1}^{n} f_j y_j + \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} \\ x \in P_1 \\ x \in \mathbb{B}^{nm}, y \in \mathbb{B}^m\end{aligned} \qquad \begin{aligned}\min \sum_{j=1}^{n} f_j y_j + \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ij}x_{ij} \\ x \in P_2 \\ x \in \mathbb{B}^{nm}, y \in \mathbb{B}^m,\end{aligned}$$

where

$$P_1 = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} : \begin{array}{l} \sum_{j=1}^m x_{ij} = 1 \quad \text{for } i = 1, \dots, n \\ x_{ij} \le y_j \quad \text{for } i = 1, \dots, n \text{ and } j = 1, \dots, m \end{array} \right\}$$

$$P_2 = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} : \begin{array}{l} \sum_{j=1}^m x_{ij} = 1 \quad \text{for } i = 1, \dots, n \\ \sum_{i=1}^n x_{ij} \le n y_j \quad \text{for } j = 1, \dots, m \end{array} \right\}$$

We claim that P_1 is a better formulation than P_2. If $x \in P_1$, then $x_{ij} \le y_j$ for all i and j. Summing these constraints over i gives us $\sum_{i=1}^n x_{ij} \le n y_j$, so $x \in P_2$. Hence we have $P_1 \subseteq P_2$. We now show that $P_1 \ne P_2$ thus proving that P_1 is a better formulation.

We assume for simplicity that $n/m = k$ is an integer. The argument can be extended to the case that m does not divide n by some technicalities. We partition the clients into m groups, each of which contains exactly k clients. The first group will be served by a (fractional) facility at y_1, the second group by a (fractional) facility at y_2 and so on. More precisely, we set

$$x_{ij} = 1 \text{ for } i = k(j-1)+1, \dots, k(j-1)+k \text{ and } j = 1, \dots, m$$

and $x_{ij} = 0$ otherwise. We also set $y_j = k/n$ for $j = 1, \dots, m$.

Fix j. By construction $\sum_{i=1}^n x_{ij} = k = n\frac{k}{n} = n y_j$. Hence, the point (x, y) just constructed is contained in P_2. On the other hand, $(x, y) \notin P_1$. $\triangleleft$

2.4 Linear Programming

We briefly recall the following fundamental results from Linear Programming which we will use in these notes. For proofs, we refer to standard books about Linear Programming such as [Sch86, Chv83].

Theorem 2.8 (Duality Theorem of Linear Programming) *Let A be an $m \times n$-matrix, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. Define the polyhedra $P = \{x : Ax \le b\}$ and $Q = \{y : A^T y = c, y \ge 0\}$.*

(i) If $x \in P$ and $y \in Q$ then $c^T x \le b^T y$. (weak duality)

(ii) In fact, we have

$$\max\{\, c^T x : x \in P \,\} = \min\{\, b^T y : y \in Q \,\}, \tag{2.3}$$

provided that both sets P and Q are nonempty. (strong duality) $\square$

Theorem 2.9 (Complementary Slackness Theorem) *Let* x^* *be a feasible solution of* $\max\{ c^T x : Ax \leq b \}$ *and* y^* *be a feasible solution of* $\min\{ b^T y : A^T y = c, y \geq 0 \}$*. Then* x^* *and* y^* *are optimal solutions for the maximization problem and minimization problem, respectively, if and only if they satisfy the* complementary slackness conditions*:*

(2.4) *for each* $i = 1, \ldots, m$*, either* $y_i^* = 0$ *or* $A_{i,\cdot} x_i^* = b_i$*.*

□

Theorem 2.10 (Farkas' Lemma) *The set* $\{x : Ax = b, x \geq 0\}$ *is nonempty if and only if there is no vector* y *such that* $A^T y \geq 0$ *and* $b^T y < 0$*.* □

2.5 Agenda

These lecture notes are consist of two main parts. The goal of Part I are as follows:

1. Prove that for any rational polyhedron $P(A, b) = \{x : Ax \leq b\}$ and $X = P \cap \mathbb{Z}^n$ the set $\text{conv}(X)$ is again a rational polyhedron.
2. Use the fact $\max\left\{c^T x : x \in X\right\} = \max\left\{c^T x : x \in \text{conv}(X)\right\}$ (see Observation 2.2) and 1 to show that the latter problem can be solved by means of Linear Programming by showing that an optimum solution will always be found at an extreme point of the polyhedron $\text{conv}(X)$ (which we show to be a point in X).
3. Give tools to derive good formulations.

Part I

Polyhedral Theory

3

Polyhedra and Integer Programs

3.1 Valid Inequalities and Faces of Polyhedra

Definition 3.1 (Valid Inequality)
Let $w \in \mathbb{R}^n$ and $t \in \mathbb{R}$. We say that the inequality $w^T x \le t$ is valid *for a set $S \subseteq \mathbb{R}^n$ if*

$$S \subseteq \{ x : w^T x \le t \}.$$

We usually write briefly $\binom{w}{t}$ for the inequality $w^T x \le t$. The set

$$S^\gamma := \left\{ \begin{pmatrix} w \\ t \end{pmatrix} : w^T x \le t \text{ is valid for } S \right\}$$

is called ***γ-polar*** *of S.*

Definition 3.2 (Face)
Let $P \subseteq \mathbb{R}^n$ be a polyhedron. The set $F \subseteq P$ is called a ***face*** *of P, if there is a be a valid inequality $\binom{w}{t}$ for P such that*

$$F = \left\{ x \in P : w^T x = t \right\}.$$

If $F \neq \varnothing$ we say that $\binom{w}{t}$ ***supports*** *the face F and call $\left\{ x : w^T x = t \right\}$ the corresponding* ***supporting hyperplane****. If $F \neq \varnothing$ and $F \neq P$, then we call F a* ***nontrivial*** *or* ***proper face****.*

Observe that any face of $P(A, b)$ has the form

$$F = \left\{ x : Ax \le b, w^T x \le t, -w^T x \le -t \right\}$$

which shows that any face of a polyhedron is again a polyhedron.

Example 3.3
We consider the polyhedron $P \subseteq \mathbb{R}^2$, which is defined by the inequalities

$$x_1 + x_2 \le 2 \tag{3.1a}$$
$$x_1 \le 1 \tag{3.1b}$$
$$x_1, x_2 \ge 0. \tag{3.1c}$$

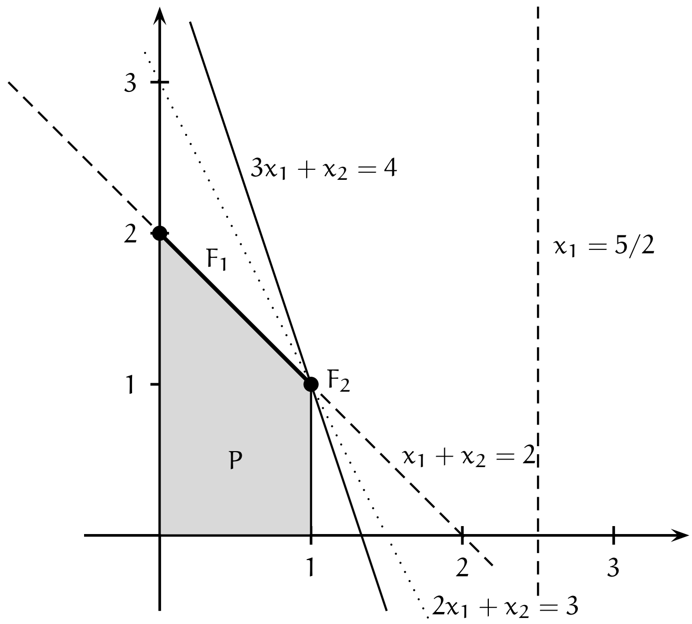


Figure 3.1: Polyhedron for Example 3.3

We have $P = P(A, b)$ with

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad \text{und} \quad b = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

The line segment F_1 from $\binom{0}{2}$ to $\binom{1}{1}$ is a face of P, since $x_1 + x_2 \leq 2$ is a valid inequality and

$$F_1 = P \cap \left\{x \in \mathbb{R}^2 : x_1 + x_2 = 2\right\}.$$

The singleton $F_2 = \{\binom{1}{1}\}$ is another face of P, since

$$F_2 = P \cap \left\{x \in \mathbb{R}^2 : 2x_1 + x_2 = 3\right\}$$
$$F_2 = P \cap \left\{x \in \mathbb{R}^2 : 3x_1 + x_2 = 4\right\}.$$

Both inequalities $2x_1 + x_2 \leq 3$ and $3x_1 + x_2 \leq 4$ induce the same face of P. In particular, this shows that the same face can be induced by completely different inequalities.

The inequalities $x_1 + x_2 \leq 2$, $2x_1 + x_2 \leq 3$ and $3x_1 + x_2 \leq 4$ induce nonempty faces of P. They support P In contrast, the valid inequality $x_1 \leq 5/2$ has

$$F_3 = P \cap \left\{x \in \mathbb{R}^2 : x_1 = 5/2\right\} = \emptyset,$$

and thus $x_1 = 5/2$ is not a supporting hyperplane of P. ◁

Remark 3.4 (i) Any polyhedron $P \subseteq \mathbb{R}^n$ is a face of itself, since $P = P \cap \left\{x \in \mathbb{R}^n : 0^T x = 0\right\}$.

(ii) $\emptyset$ is a face of any polyhedron $P \subseteq \mathbb{R}^n$, since $\emptyset = P \cap \left\{x \in \mathbb{R}^n : 0^T x = 1\right\}$.

(iii) If $F = P \cap \left\{x \in \mathbb{R}^n : c^T x = \gamma\right\}$ is a nontrivial face of $P \subseteq \mathbb{R}^n$, then $c \neq 0$, since otherwise we are either in case (i) or (ii) above.

Let us consider Example 3.3 once more. Face F_1 can be obtained by turning inequality (3.1a)) into an equality: machen:

$$F_1 = \left\{ x \in \mathbb{R}^2 : \begin{array}{lcl} x_1 + x_2 & = & 2 \\ x_1 & \leq & 1 \\ x_1, x_2 & \geq & 0 \end{array} \right\}$$

Likewise F_2 can be obtained by making (3.1a) and (3.1b) equalities

$$F_2 = \left\{ x \in \mathbb{R}^2 : \begin{array}{lcl} x_1 + x_2 & = & 2 \\ x_1 & = & 1 \\ x_1, x_2 & \geq & 0 \end{array} \right\}$$

Let $P = P(A, b) \subseteq \mathbb{R}^n$ be a polyhedron and M be the index set of the rows of A. For a subset $I \subseteq M$ we consider the set

$$\text{(3.2)} \qquad \operatorname{fa}(I) := \{x \in P : A_{I,\cdot}x = b_I\}.$$

Since any $x \in P$ satisfies $A_{I,\cdot}x \leq b_I$, we get by summing up the rows of (3.2) for

$$c^T := \sum_{i \in I} A_{I,\cdot} \quad \text{and} \quad \gamma := \sum_{i \in I} b_i$$

a valid inequality $c^T x \leq \gamma$ for P. For all $x \in P \setminus \operatorname{fa}(I)$ there is at least one $i \in I$, such that $A_{i,\cdot}x < b_i$. Thus $c^T x < \gamma$ for all $x \in P \setminus F$ and

$$\operatorname{fa}(I) = \left\{x \in P : c^T x = \gamma\right\}$$

is a face of P.

Definition 3.5 (Face induced by index set)
The set $\operatorname{fa}(I)$ *defined in (3.2) is called the* face of P induced by I.

In Example 3.3 we have $F_1 = \operatorname{fa}(\{1\})$ and $F_2 = \operatorname{fa}(\{1, 2\})$. The following theorem shows that in fact all faces of a polyhedron can be obtained this way.

Theorem 3.6 *Let $P = P(A, b) \subseteq \mathbb{R}^n$ be a nonempty polyhedron and M be the index set of the rows of A. The set $F \subseteq \mathbb{R}^n$ with $F \neq \emptyset$ is a face of P if and only if $F = \operatorname{fa}(I) = \{x \in P : A_{I,\cdot}x = b_I\}$ for a subset $I \subseteq M$.*

Proof: We have already seen that $\operatorname{fa}(I)$ is a face of P for any $I \subseteq M$. Assume conversely that $F = P \cap \left\{x \in \mathbb{R}^n : c^T x = t\right\}$ is a face of P. Then, F is precisely the set of optimal solutions of the Linear Program

$$\text{(3.3)} \qquad \max\left\{c^T x : Ax \leq b\right\}.$$

(here we need the assumption that $P \neq \emptyset$). By the Duality Theorem of Linear Programming (Theorem 2.8), the dual Linear Program for (3.3)

$$\min\left\{b^T y : A^T y = c, y \geq 0\right\}$$

also has an optimal solution y^* which satisfies $b^T y^* = t$. Let $I := \{i : y_i^* > 0\}$. The by complementary slackness (Theorem 2.9) the optimal solutions of (3.3) are precisely those $x \in P$ with $A_{i,\cdot}x = b_i$ for $i \in I$. This gives us $F = \operatorname{fa}(I)$. $\square$

This result implies the following consequence:

Corollary 3.7 *Every polyhedron has only a finite number of faces.*

Proof: There is only a finite number of subsets $I \subseteq M = \{1, \dots, m\}$. □

We can also look at the binding equations for subsets of polyhedra.

Definition 3.8 (Equality set) *Let $P = P(A, b) \subseteq \mathbb{R}^n$ be a polyhedron. For $S \subseteq P$ we call*

$$\mathrm{eq}(S) := \{i \in M : A_{i,\cdot}x = b_i \text{ for all } x \in S\},$$

*the **equality set** of S.*

Clearly, for subsets S, S' of a polyhedron $P = P(A, b)$ with $S \subseteq S'$ we have $\mathrm{eq}(S) \supseteq \mathrm{eq}(S')$. Thus, if $S \subseteq P$ is a nonempty subset of S, then any face F of P which contains S must satisfy $\mathrm{eq}(F) \subseteq \mathrm{eq}(S)$. On the other hand, $\mathrm{fa}(\mathrm{eq}(S))$ is a face of P containing S. Thus, we have the following observation:

Observation 3.9 *Let $P = P(A, b) \subseteq \mathbb{R}^n$ be a polyhedron and $S \subseteq P$ be a nonempty subset of P. The smallest face of P which contains S is $\mathrm{fa}(\mathrm{eq}(S))$.*

Corollary 3.10 *(i) The polyhedron $P = P(A, b)$ does not have any proper face if and only if $\mathrm{eq}(P) = M$, that is, if and only if P is an affine subspace $P = \{x : Ax = b\}$.*

(ii) If $A\bar{x} < b$, then $\bar{x}$ is not contained in any proper face of P.

Proof:

(i) Immediately from the characterization of faces in Theorem 3.6.

(ii) If $A\bar{x} < b$, then $\mathrm{eq}(\{\bar{x}\}) = \emptyset$ and $\mathrm{fa}(\emptyset) = P$.

□

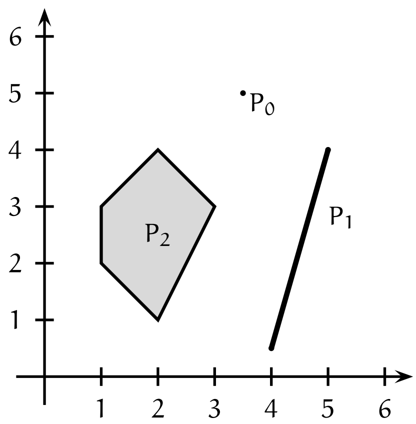
3.2 Dimension

Intuitively the notion of dimension seems clear by considering the degrees of freedom we have in moving within a given polyhedron (cf. Figure 3.2).

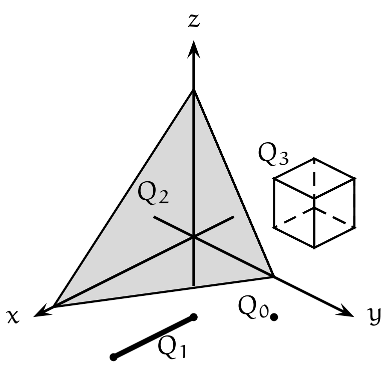
Definition 3.11 (Affine Combination, affine independence, affine hull)
*An **affine combination** of the vectors $v^1, \dots, v^k \in \mathbb{R}^n$ is a linear combination $x = \sum_{i=1}^k \lambda_i v^i$ such that $\sum_{i=1}^k \lambda_i = 1$.*
*Given a set $X \subseteq \mathbb{R}^n$, the **affine hull** of X, denoted by $\mathrm{aff}(X)$ is defined to be the set of all affine combinations of vectors from X, that is*

$$\mathrm{aff}(X) := \{\, x = \sum_{i=1}^k \lambda_i v_i : \sum_{i=1}^k \lambda_i = 1 \text{ and } v_1, \dots, v_k \in X\}$$

*The vectors $v^1, \dots, v^k \in \mathbb{R}^n$ are called **affinely independent**, if $\sum_{i=1}^k \lambda_i v^i = 0$ and $\sum_{i=1}^k \lambda_i = 0$ implies that $\lambda_1 = \lambda_2 = \dots = \lambda_k = 0$.*



(a) Polyhedra in $\mathbb{R}^2$



(b) Polyhedra in $\mathbb{R}^3$

Figure 3.2: Examples of polyhedra with various dimensions

Lemma 3.12 *The following statements are equivalent*

(i) The vectors $v^1, \dots, v^k \in \mathbb{R}^n$ *are affinely independent.*

(ii) The vectors $v^2 - v^1, \dots, v^k - v^1 \in \mathbb{R}^n$ *are linearly independent.*

(iii) The vectors $\binom{v^1}{1}, \dots, \binom{v^k}{1} \in \mathbb{R}^{n+1}$ *are linearly independent.*

Proof:

(i)⇔(ii) If $\sum_{i=2}^k \lambda_i (v^i - v^1) = 0$ and we set $\lambda_1 := -\sum_{i=2}^k \lambda_i$, this gives us $\sum_{i=1}^k \lambda_i v^i = 0$ and $\sum_{i=1}^k \lambda_i = 0$. Thus, from the affine independence it follows that $\lambda_1 = \cdots = \lambda_k = 0$.

Assume conversely that $v^2 - v^1, \dots, v^k - v^1$ are linearly independent and $\sum_{i=1}^k \lambda_i v^i = 0$ with $\sum_{i=1}^k \lambda_i = 0$. Then $\lambda_1 = -\sum_{i=2}^k \lambda_i$ which gives $\sum_{i=2}^k \lambda_i (v^i - v^1) = 0$. The linear independence of $v^2 - v^1, \dots, v^k - v^1$ implies $\lambda_2 = \cdots = \lambda_k = 0$ which in turn also gives $\lambda_1 = 0$.

(ii)⇔(iii) This follows immediately from

$$\sum_{i=1}^k \lambda_i \begin{pmatrix} v^i \\ 1 \end{pmatrix} = 0 \Leftrightarrow \left\{ \begin{array}{c} \sum_{i=1}^k \lambda_i v^i = 0 \\ \sum_{i=1}^k \lambda_i = 0 \end{array} \right\}$$

□

Definition 3.13 (Dimension of a polyhedron, full-dimensional polyhedron) *The* ***dimension*** $\dim P$ *of a polyhedron* $P \subseteq \mathbb{R}^n$ *is one less than the maximum number of affinely independent vectors in* P. *We set* $\dim \emptyset = -1$. *If* $\dim P = n$, *then we call* P ***full-dimensional****.*

Example 3.14
Consider the polyhedron $P \subseteq \mathbb{R}^2$ defined by the following inequalities (see

Figure 3.3):

$$x \leq 2 \tag{3.4a}$$
$$x + y \leq 4 \tag{3.4b}$$
$$x + 2y \leq 10 \tag{3.4c}$$
$$x + 2y \leq 6 \tag{3.4d}$$
$$x + y \geq 2 \tag{3.4e}$$
$$x, y \geq 0 \tag{3.4f}$$

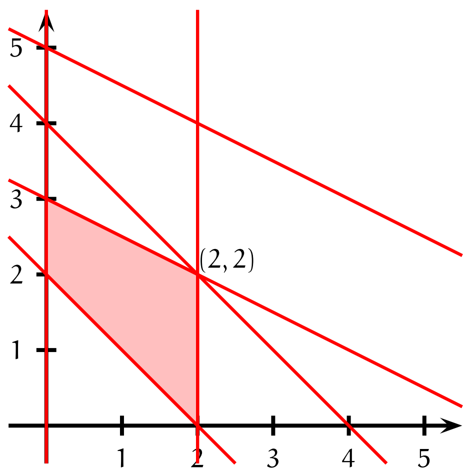


Figure 3.3: A fulldimensional polyhedron in $\mathbb{R}^2$.

The polyhedron P is full dimensional, since $(2, 0)$, $(1, 1)$ and $(2, 2)$ are three affinely independent vectors. $\triangleleft$

Example 3.15
A *stable set* (or *independent set*) in an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ of the vertices such that none of the vertices in S are joined by an edge. We can formulate the problem of finding a stable set of maximum cardinality as an IP:

$$\max \sum_{v \in V} x_v \tag{3.5a}$$
$$x_u + x_v \leq 1 \quad \text{for all edges } (u, v) \in E \tag{3.5b}$$
$$x_v \geq 0 \quad \text{for all vertices } v \in V \tag{3.5c}$$
$$x_v \leq 1 \quad \text{for all vertices } v \in V \tag{3.5d}$$
$$x_v \in \mathbb{Z} \quad \text{for all vertices } v \in V \tag{3.5e}$$

Let P be the polytope determined by the inequalities in (3.5). We claim that P is full dimensional. To see this, consider the n unit vectors $e_i = (0, \dots, 1, 0, \dots, 0)^T$, $i = 1, \dots, n$ and $e_0 := (0, \dots, 0)^T$. Then $e_0, e_1, \dots, e_n$ are affinely independent and thus $\dim P = n$. $\triangleleft$

Definition 3.16 (Inner point, interior point)
*The vector $\bar{x} \in P = P(A, b)$ is called an **inner point**, if it is not contained in any proper face of P. We call $\bar{x} \in P$ an **interior point**, if $A\bar{x} < b$.*

By Corollary 3.10(ii) an interior point $\bar{x}$ is not contained in any proper face.

Lemma 3.17 *Let* F *be a face of the polyhedron* $P(A, b)$ *and* $\bar{x} \in F$*. Then* $\bar{x}$ *is an inner point of* F *if and only if* $\text{eq}(\{\bar{x}\}) = \text{eq}(F)$*.*

Proof: Let G be an inclusionwise smallest face of F containing $\bar{x}$. Then, $\bar{x}$ is an inner point of F if and only if $F = G$. By Observation 3.9 we have $G = \text{fa}(\text{eq}(\{\bar{x}\}))$. And thus, $\bar{x}$ is an inner point of F if and only if $\text{fa}(\text{eq}(\{\bar{x}\})) = F$ as claimed. □

Thus, Definition 3.16 can be restated equivalently as: $\bar{x} \in P = P(A, b)$ is an innner point of P if $\text{eq}(\{\bar{x}\}) = \text{eq}(P)$.

Lemma 3.18 *Let* $P = P(A, b)$ *be a nonempty polyhedron. Then, the set of inner points of* P *is nonempty.*

Proof: Let $M = \{1, \dots, m\}$ be the index set of the rows of A, $I := \text{eq}(P)$ and $J := M \setminus I$. If $J = \emptyset$, that is, if $I = M$, then by Corollary 3.10(i) the polyhedron P does not have any proper face and any point in P is an inner point.

If $J \neq \emptyset$, then for any $j \in J$ we can find an $x^j \in P$ such that $Ax^j \leq b$ and $A_{j,\cdot}x^j < b_j$. Since P is convex, the vector y, defined as

$$y := \frac{1}{|J|} \sum_{j \in J} x^j$$

(which is a convex combination of the x^j, $j \in J$) is contained in P. Then, $A_{J,\cdot}y < b_J$ and $A_{I,\cdot}y = b_I$. So, $\text{eq}(\{y\}) = \text{eq}(P)$ and the claim follows. □

Theorem 3.19 (Dimension Theorem) *Let* $F \neq \emptyset$ *be a face of the polyhedron* $P(A, b) \subseteq \mathbb{R}^n$*. Then we have*

$$\dim F = n - \operatorname{rank} A_{\text{eq}(F),\cdot}.$$

Proof: By Linear Algebra we know that

$$\dim \mathbb{R}^n = n = \operatorname{rank} A_{\text{eq}(F),\cdot} + \dim \operatorname{kern} A_{\text{eq}(F),\cdot}.$$

Thus, the theorem follows, if we can show that $\dim \operatorname{kern}(A_{\text{eq}(F),\cdot}) = \dim F$. We abbreviate $I := \text{eq}(F)$ and set $r := \dim \operatorname{kern} A_{I,\cdot}$, $s := \dim F$.

"$r \geq s$": Select $s + 1$ affinely independent vectors $x^0, x^1, \dots, x^s \in F$. Then, by Lemma 3.12 $x^1 - x^0, \dots, x^s - x^0$ are linearly independent vectors and $A_{I,\cdot}(x^j - x^0) = b_I - b_I = 0$ for $j = 1, \dots, s$. Thus, the dimension of $\operatorname{kern} A_{I,\cdot}$ is at least s.

"$s \geq r$": Since we have assumed that $F \neq \emptyset$, we have $s = \dim F \geq 0$. Thus, in the sequel we can assume that $r \geq 0$ since otherwise there is nothing left to prove.

By Lemma 3.18 there exists an inner point $\bar{x}$ of F which by Lemma 3.17 satisfies $\text{eq}(\{\bar{x}\}) = \text{eq}(F) = I$. Thus, for $J := M \setminus I$ we have

$$A_{I,\cdot}\bar{x} = b_I \quad \text{and} \quad A_{J,\cdot}\bar{x} < b_J.$$

Let $\{x^1, \dots, x^r\}$ be a basis of kern $A_{I,\cdot}$. Then, since $A_{J,\cdot}\bar{x} < b_J$ we can find $\varepsilon > 0$ such that $A_{J,\cdot}(\bar{x} + \varepsilon x^k) < b_J$ and $A_{I,\cdot}(\bar{x} + \varepsilon x^k) = b_I$ for $k = 1, \dots, r$. Thus, $\bar{x} + \varepsilon x^k \in F$ for $k = 1, \dots, r$.

The vectors $\varepsilon x^1, \dots, \varepsilon x^r$ are linearly independent and, by Lemma 3.12 $\bar{x}, \varepsilon x^1 + \bar{x}, \dots, \varepsilon x^r + \bar{x}$ form a set of $r+1$ affinely independent vectors in F which implies $\dim F \geq r$.

□

Corollary 3.20 *Let $P = P(A, b) \subseteq \mathbb{R}^n$ be a nonempty polyhedron. Then:*

(i) $\dim P = n - \operatorname{rank} A_{\mathrm{eq}(P),\cdot}$

(ii) P *is full dimensional if* $\mathrm{eq}(P) = \emptyset$.

(iii) P *is full dimensional if and only if* P *contains an interior point.*

(iv) If F *is a proper face of* P, *then* $\dim F \leq \dim P - 1$.

□

Proof:

(i) Use Theorem 3.19 with $F = P$.

(ii) Immediate from (i).

(iii) P has an interior point if and only if $\mathrm{eq}(P) = \emptyset$.

(iv) Let $I := \mathrm{eq}(P)$ and $j \in \mathrm{eq}(F) \setminus I$ and $J := \mathrm{eq}(P) \cup \{j\}$. We show that $A_{j,\cdot}$ is linearly independent of the rows in $A_{I,\cdot}$. This shows that $\operatorname{rank} A_{\mathrm{eq}(F),\cdot} \geq \operatorname{rank} A_{J,\cdot} > \operatorname{rank} A_{I,\cdot}$ and by the Dimension Theorem we have $\dim F \leq \dim P - 1$.

Assume that $A_{j,\cdot} = \sum_{i \in I} \lambda_i A_{i,\cdot}$. Take $\bar{x} \in F$ arbitrary, then

$$b_j = A_{j,\cdot}\bar{x} = \sum_{i \in I} \lambda_i A_{i,\cdot} = \sum_{i \in I} \lambda_i b_i.$$

Since $j \notin \mathrm{eq}(P)$, there is $x \in P$ such that $A_{j,\cdot}x < b_j$. But by the above we have

$$b_j > A_{j,\cdot}x = \sum_{i \in I} \lambda_i A_{i,\cdot}x = \sum_{i \in I} \lambda_i b_i = b_j,$$

which is a contradiction.

□

Example 3.21
Let $G = (V, R)$ be a directed graph and $s, t \in V$ be two distinct vertices. We call a subset $A \subseteq R$ of the arcs of R an s-t-*connector* if the subgraph (V, A) contains an s-t-path. It is easy to see that A is an s-t-connector if and only if $A \cap \delta^+(S) \neq \emptyset$ for each s-t-cut (S, T), that is for each partition $V = S \cup T$, $S \cap T = \emptyset$ of the vertex set V such that $s \in S$ and $t \in T$ (cf. [KN05, Satz 3.19]). Here, we denote by $\delta^+(S)$ the subset of the arcs $(u, v) \in R$ such that $u \in S$ and $v \in T$.

Thus, the s-t-connectors are precisely the solutions of the following system:

(3.6a) $\sum_{r\in\delta^+(S)} x_r \geq 1$ for all s-t-cuts (S, T)

(3.6b) $x_r \leq 1$ for all arcs $r \in R$

(3.6c) $x_r \geq 0$ for all arcs $r \in R$

(3.6d) $x_r \in \mathbb{Z}$ for all arcs $r \in R$.

Let P be the polyhedron determined by the inequalities in (3.6). It can be shown (we will do this later, but you can also find a proof in [Sch03]) that P is in fact the convex hull of the s-t-connectors in G. For the moment, we will not need this result.

Let $R' \subseteq R$ be the set of arcs $r \in R$ such that there is an s-t-path in $G - r$ (that is, there is an s-t-path which does not use r). We claim that $\dim P = |R'|$. By the Dimension Theorem this is equivalent to showing that $\operatorname{rank} A_{\mathrm{eq}(P),\cdot} = |R| - |R'|$.

None of the inequalities $x_r \geq 0$ is in $\mathrm{eq}(P)$, since any superset of an s-t-connector is again an s-t-connector, so it can not be the case that $\chi^A_r = 0$ for all s-t-connectors A with incidence vector $\chi^A \in \mathbb{R}^R$ (which are a subset of P). Thus we note:

- None of the inequalities $x_r \geq 0$ is in $\mathrm{eq}(P)$.

Now consider the inequalities $x_r \leq 1$. If $r \notin R'$, then any s-t-path must use r, so we find an (S, T)-cut with $\delta^+(S) = \{r\}$ (choose S to be all vertices reachable from s in $G - r$ and $T := V \setminus S$). By (3.6a) we have $\sum_{r\in\delta^+(S)} x_r \geq 1$ for all $x \in P$. Since $\delta^+(S) = \{r\}$, we have $x_r = 1$ for any $x \in P$. On the other hand, if $r \in R'$, there is an s-t-path which misses r and thus there is an s-t-connector A (formed by the arc set of this path) with $\chi^A_r = 0$. Thus, we have

- The inequality $x_r \leq 1$ is in $\mathrm{eq}(P)$ if and only if $r \in R \setminus R'$.

Finally, let us look at the inequalities (3.6a). Assume that that there exists and s-t-cut (S, T) such that $\sum_{r\in\delta^+(S)} x_r = 1$ for all $x \in P$. Then, this equality must also hold for all incidence vectors of s-t-connectors. Then, it follows that $|\delta^+(S)| = 1$ (since any superset of an s-t-connector is again an s-t-connector). This implies that $r \in R \setminus R'$. Conversely, as we have seen above, if $|\delta^+(S)| = 1$ for an s-t-cut, the corresponding inequality (3.6a) must hold with equality.

- The inequality $\sum_{r\in\delta^+(S)} x_r \geq 1$ is in $\mathrm{eq}(P)$ if and only if $\delta^+(S) = \{r\}$ for some $r \in R \setminus R'$ in which case it collapses to $x_r \geq 1$.

Thus, the rows corresponding to the inequalities $x_r \leq 1$, $r \in R \setminus R'$ are a maximum size set of linearly vectors with indices in $\mathrm{eq}(P)$. Thus, $\operatorname{rank} A_{\mathrm{eq}(P),\cdot} = |R \setminus R'| = |R| - |R'|$ as needed. $\triangleleft$

We derive another important consequence of the Dimesion Theorem about the facial structure of polyhedra:

Theorem 3.22 (Hoffman and Kruskal) *Let $P = P(A, b) \subseteq \mathbb{R}^n$ be a polyhedron. Then a nonempty set $F \subseteq P$ is an inclusionwise minimal face of P if and only if $F = \{x : A_{I,\cdot}x = b_I\}$ for some index set $I \subseteq M$ and* $\operatorname{rank} A_{I,\cdot} = \operatorname{rank} A$.

Proof: "$\Rightarrow$": Let F be a minimal nonempty face of P. Then, by Theorem 3.6 and Observation 3.9 we have $F = \mathrm{fa}(I)$, where $I = \mathrm{eq}(F)$. Thus, for $J := M \setminus I$ we have

$$F = \{x : A_{I,\cdot}x = b_I, A_{J,\cdot}x \leq b_J\}. \tag{3.7}$$

We claim that $F = G$, where

$$G = \{x : A_{I,\cdot}x = b_I\}. \tag{3.8}$$

By (3.7) we have $F \subseteq G$. Suppose that there exists $y \in G \setminus F$. Then, there exists $j \in J$

$$A_{I,\cdot}y = b_I, A_{j,\cdot}y > b_j. \tag{3.9}$$

Let $\bar{x}$ be any inner point of F which exists by Lemma 3.18. We consider for $\tau \in \mathbb{R}$ the point

$$z(\tau) = \bar{x} + \tau(y - \bar{x}) = (1 - \tau)\bar{x} + \tau y.$$

Observe that $A_{I,\cdot}z(\tau) = (1 - \tau)A_{I,\cdot}\bar{x} + \tau A_{I,\cdot}y = (1 - \tau)b_I + \tau b_I = b_I$, since $\bar{x} \in F$ and y satisfies (3.9). Moreover, $A_{J,\cdot}z(0) = A_{J,\cdot}\bar{x} < b_J$, since $J \subseteq M \setminus I$.

Since $A_{j,\cdot}y > b_j$ we can find $\tau \in \mathbb{R}$ and $j_0 \in J$ such that $A_{j_0,\cdot}z(\tau) = b_{j_0}$ and $A_{J,\cdot}z(\tau) \leq b_J$. Then, $\tau \neq 0$ and

$$F' := \{x \in P : A_{I,\cdot}x = b_I, A_{j_0,\cdot}x = b_{j_0}\}$$

is a face which is properly contained in F (note that $\bar{x} \in F \setminus F'$). This contradicts the choice of F as inclusionwise minimal. Hence, we have that F can be represented as (3.8).

It remains to prove that $\operatorname{rank} A_{I,\cdot} = \operatorname{rank} A$. If $\operatorname{rank} A_{I,\cdot} < \operatorname{rank} A$, then there exists an index $j \in J = M \setminus I$, such that $A_{j,\cdot}$ is not a linear combination of the rows in $A_{I,\cdot}$. Then, we can find a vector $w \neq 0$ such that $A_{I,\cdot}w = 0$ and $A_{j,\cdot}w > 0$. For $\theta > 0$ appropriately chosen the vector $y := \bar{x} + \theta w$ satisfies (3.9) and as above we can construct a proper face F' of F contradicting the minimality of F.

"$\Leftarrow$": If $F = \{x : A_{I,\cdot} = b_I\}$, then F is an affine subspace and Corollary 3.10(i) shows that F does not have any proper face. By assumption $F \subseteq P$ and thus $F = \{x : A_{I,\cdot} = b_I, A_{J,\cdot}x \leq b_J\}$ is a minimal face of P. □

Corollary 3.23 *All minimal nonempty faces of a polyhedron* $P = P(A, b)$ *have the same dimension, namely* $n - \operatorname{rank} A$. □

Corollary 3.24 *Let* $P = P(A, b) \subseteq \mathbb{R}^n$ *be a nonempty polyhedron and* $\operatorname{rank}(A) = n - k$. *Then* P *has a face of dimension* k *and does not have a proper face of lower dimension.*

Proof: Let F be any nonempty face of P. Then, $\operatorname{rank} A_{\mathrm{eq}(F),\cdot} \leq \operatorname{rank} A = n - k$ and thus by the Dimension Theorem (Theorem 3.19) it follows that $\dim(F) \geq n - (n - k) = k$. Thus, any nonempty face of P has dimension at least k.

On the other hand, by Corollary 3.23 any inclusionwise minimal nonempty face of P has dimension $n - \operatorname{rank} A = n - (n - k) = k$. Thus, P has in fact faces of dimension k. □

There will be certain types of faces which are of particular interest:

- extreme points (vertices),
- extreme rays, and
- facets.

In the next section we discuss extreme points and their meaning for optimization. Section 3.4 deals with facets and their importance in describing polyhedra by means of inequalities. Section 3.5 shows how we can describe polyhedra by their extreme points and extreme rays. The two descriptions of polyhedra will be important later on.

3.3 Extreme Points

Definition 3.25 (Extreme point, pointed polyhedron)
The point $\bar{x} \in P = P(A, b)$ *is called an* ***extreme point*** *of* P, *if* $\bar{x} = \lambda x + (1-\lambda)y$ *for some* $x, y \in P$ *and* $0 < \lambda < 1$ *implies that* $x = y = \bar{x}$.

A polyhedron $P = P(A, b)$ *is* ***pointed****, if it has at least one extreme point.*

Example 3.26
Consider the polyhedron from Example 3.14. The point $(2, 2)$ is an extreme point of the polyhedron. ◁

Theorem 3.27 (Characterization of extreme points) *Let* $P = P(A, b) \subseteq \mathbb{R}^n$ *be a polyhedron and* $\bar{x} \in P$. *Then, the following statements are equivalent:*

(i) $\{\bar{x}\}$ *is a zero-dimensional face of* P.

(ii) There exists a vector $c \in \mathbb{R}^n$ *such that* $\bar{x}$ *is the unique optimal solution of the Linear Program* $\max\left\{c^T x : x \in P\right\}$.

(iii) $\bar{x}$ *is an extreme point of* P.

(iv) $\operatorname{rank} A_{\mathrm{eq}(\{\bar{x}\}),\cdot} = n$.

Proof: "(i)⇒(ii)": Since $\{\bar{x}\}$ is a face of P, there exists a valid inequality $w^T x \leq t$ such that $\{x\} = \left\{x \in P : w^T x = t\right\}$. Thus, $\bar{x}$ is the unique optimum of the Linear Program with objective $c := w$.

"(ii)⇒(iii)": Let $\bar{x}$ be the unique optimum solution of $\max\left\{c^T x : x \in P\right\}$. If $\bar{x} = \lambda x + (1-\lambda)y$ for some $x, y \in P$ and $0 < \lambda < 1$, then we have

$$c^T \bar{x} = \lambda c^T x + (1-\lambda) c^T y \leq \lambda c^T \bar{x} + (1-\lambda) c^T \bar{x} = c^T \bar{x}.$$

Thus, we can conclude that $c^T \bar{x} = c^T x = c^T y$ which contradicts the uniqueness of $\bar{x}$ as optimal solution.

"(iii)⇒(iv)": Let $I := \mathrm{eq}(\{x\})$. If $\operatorname{rank} A_{I,\cdot} < n$, there exists $y \in \mathbb{R}^n \setminus \{0\}$ such that $A_{I,\cdot} y = 0$. Then, for sufficiently small $\varepsilon > 0$ we have $x := \bar{x} + \varepsilon y \in P$ and $y := \bar{x} - \varepsilon y \in P$ (since $A_{j,\cdot}\bar{x} < b_j$ for all $j \notin I$). But then, $\bar{x} = \frac{1}{2}x + \frac{1}{2}y$ which contradicts the assumption that $\bar{x}$ is an extreme point.

"(iv)⇒(i)": Let $I := \mathrm{eq}(\{\bar{x}\})$. By (iv), the system $A_{I,\cdot} x = b_I$ has a unique solution which must be $\bar{x}$ (since $A_{I,\cdot}\bar{x} = b_I$ by construction of I). Hence

$$\{\bar{x}\} = \{x : A_{I,\cdot} x = b_I\} = \{x \in P : A_{I,\cdot} x = b_I\}$$

and by Theorem 3.6 $\{\bar{x}\}$ is a zero-dimensional face of P. □

The result of the previous theorem has interesting consequences for optimization. Consider the Linear Program

$$\max\left\{c^T x : x \in P\right\}, \tag{3.10}$$

where P is a pointed polyhedron (that is, it has extreme points). Since by Corollary 3.23 on page 28 all minimal proper faces of P have the same dimension, it follows that the minimal proper faces of P are of the form $\{\bar{x}\}$, where $\bar{x}$ is an extreme point of P. Suppose that $P \neq \emptyset$ and $c^T x$ is bounded on P. We know that there exists an optimal solution $x^* \in P$. The set of optimal solutions of (3.10) is a face

$$F = \left\{x \in P : c^T x = c^T x^*\right\}$$

which contains a minimal nonempty face $F' \subseteq F$. Thus, we have the following corollary:

Corollary 3.28 *If the polyhedron P is pointed and the Linear Program (3.10) has optimal solutions, it has an optimal solution which is also an extreme point of P.*

Another important consequence of the characterization of extreme points in Theorem 3.27 on the previous page is the following:

Corollary 3.29 *Every polyhedron has only a finite number of extreme points.*

Proof: By the preceeding theorem, every extreme point is a face. By Corollary 3.7, there is only a finite number of faces. □

Let us now return to the Linear Program (3.10) which we assume to have an optimal solution. We also assume that P is pointed, so that the assumptions of Corollary 3.28 are satisfied. By the Theorem of Hoffman and Kruskal (Theorem 3.22 on page 27) every extreme point $\bar{x}$ of is the solution of a subsystem

$$A_{I,\cdot}x = b_I, \text{ where } \operatorname{rank} A_{I,\cdot} = n.$$

Thus, we could obtain an optimal solution of (3.10) by "brute force", if we simply consider all subsets $I \subseteq M$ with $|I| = n$, test if $\operatorname{rank} A_{I,\cdot} = n$ (this can be done by Gaussian elimination) and solve $A_{I,\cdot}x = b_I$. We then choose the best of the feasible solutions obtained this way. This gives us a finite algorithm for (3.10). Of course, the Simplex Method provides a more sophisticated way to or solving (3.10).

Let us now derive conditions which ensure that a given polyhedron is pointed.

Corollary 3.30 *A nonempty polyhedron $P = P(A, b) \subseteq \mathbb{R}^n$ is pointed if and only if* $\operatorname{rank} A = n$.

Proof: By Corollary 3.23 the minimal nonempty faces of P are of dimension 0 if and only if $\operatorname{rank} A = n$. □

Corollary 3.31 *Any nonempty polytope is pointed.*

Proof: Let $P = P(A, b)$ and $\bar{x} \in P$ be arbitrary. By Corollary 3.30 it suffices to show that $\operatorname{rank} A = n$. If $\operatorname{rank} A < n$, then we can find $y \in \mathbb{R}^n$ with $y \neq 0$ such that $Ay = 0$. But then $x + \theta y \in P$ for all $\theta \in \mathbb{R}$ which contradicts the assumption that P is bounded. □

Corollary 3.32 *Any nonempty polyhedron* $P \subseteq \mathbb{R}^n_+$ *is pointed.*

Proof: If $P = P(A, b) \subseteq \mathbb{R}^n_+$, we can write P alternatively as

$$P = \left\{ x : \begin{pmatrix} A \\ -I \end{pmatrix} x \leq \begin{pmatrix} b \\ 0 \end{pmatrix} \right\} = P(\bar{A}, \bar{b}).$$

Since $\operatorname{rank} \bar{A} = \operatorname{rank} \begin{pmatrix} A \\ -I \end{pmatrix} = n$, we see again that the minimal faces of P are extreme points. $\square$

On the other hand, Theorem 3.27(ii) is is a formal statement of the intuition that by optimizing with the help of a suitable vector over a polyhedron we can "single out" every extreme point. We now derive a stronger result for *rational polyhedra*:

Theorem 3.33 *Let* $P = P(A, b)$ *be a rational polyhedron and let* $\bar{x} \in P$ *be an extreme point of* P. *There exists an integral vector* $c \in \mathbb{Z}^n$ *such that* $\bar{x}$ *is the unique solution of* $\max \left\{ c^T x : x \in P \right\}$.

Proof: Let $I := \operatorname{eq}(\{\bar{x}\})$ and $M := \{1, \dots, m\}$ be the index set of the rows of A. Consider the vector $\bar{c} = \sum_{i \in M} A_{i,\cdot}$. Since all the $A_{i,\cdot}$ are rational, we can find a $\theta > 0$ such that $c := \theta \bar{c} \in \mathbb{Z}^n$ is integral. Since $\operatorname{fa}(I) = \{\bar{x}\}$ (cf. Observation 3.9), for every $x \in P$ with $x \neq \bar{x}$ there is at least one $i \in I$ such that $A_{i,\cdot} x < b_i$. Thus, for all $x \in P \setminus \{\bar{x}\}$ we have

$$c^T x = \theta \sum_{i \in M} A_{i,\cdot}^T x < \theta \sum_{i \in M} b_i = \theta c^T x^0.$$

This proves the claim. $\square$

Consider the polyhedron

$$P^=(A, b) := \{x : Ax = b, x \geq 0\},$$

where A is an $m \times n$ matrix. A *basis* of A is an index set $B \subseteq \{1, \dots, n\}$ with $|B| = m$ such that the square matrix $A_{\cdot,B}$ formed by the columns from B is nonsingular. The *basic solution* corresponding to B is the vector (x_B, x_N) with $x_B = A_{\cdot,B}^{-1} b$, $x_N = 0$. The basic solution is called *feasible*, if it is contained in $P^=(A, b)$.

The following theorem is a well-known result from Linear Programming:

Theorem 3.34 *Let* $P = P^=(A, b) = \{x : Ax = b, x \geq 0\}$ *and* $\bar{x} \in P$, *where* A *is an* $m \times n$ *matrix of rank* m. *Then,* $\bar{x}$ *is an extreme point of* P *if and only if* $\bar{x}$ *is a basic feasible solution for some basis* B.

Proof: Suppose that $\bar{x}$ is a basic solution for B and $\bar{x} = \lambda x + (1 - \lambda) y$ for some $x, y \in P$. It follows that $x_N = y_N = 0$. Thus $x_B = y_b = A_{\cdot,B}^{-1} b = \bar{x}$. Thus, $\bar{x}$ is an extreme point of P.

Assume now conversely that $\bar{x}$ is an extreme point of P. Let $B := \{i : \nu_i > 0\}$. We claim that the matrix $A_{\cdot,B}$ consists of linearly independent colums. Indeed, if $A_{\cdot,B} y_B = 0$ for some $y_B \neq 0$, then for small $\varepsilon > 0$ we have $x_B \pm \varepsilon y_B \geq 0$. Hence, if we set $N := \{1, \dots, m\} \setminus B$ and $y = (y_B, y_B)$ we have $\bar{x} \pm \varepsilon y \in P$

and hence we can write x as a convex combination $x = \frac{1}{2}(\bar{x} + \varepsilon y) + \frac{1}{2}(\bar{x} - \varepsilon y)$ contradicting the fact that $\bar{x}$ is an extreme point.

Since A_B has linearly independent columns, it follows that $|B| \leq m$. Since $\operatorname{rank} A = m$ we can augment B to a basis B'. Then, $\bar{x}$ is the basic solution for B'. □

We close this section by deriving structural results for polytopes. We need one auxiliary result:

Lemma 3.35 *Let $X \subset \mathbb{R}^n$ be a finite set and $v \in \mathbb{R}^n \setminus \operatorname{conv}(X)$. There exists an inequality that separates v from $\operatorname{conv}(X)$, that is, there exist $w \in \mathbb{R}^n$ and $t \in \mathbb{R}$ such that $w^T x \leq t$ for all $x \in \operatorname{conv}(X)$ and $w^T v > t$.*

Proof: Let $X = \{x_1, \ldots, x_k\}$. Since $v \notin \operatorname{conv}(X)$, the system

$$\begin{aligned} \sum_{i=1}^{k} \lambda_k x_k &= v \\ \sum_{i=1}^{k} \lambda_k &= 1 \\ \lambda_i &\geq 0 \quad \text{for } i = 1, \ldots, k \end{aligned}$$

does not have a solution. By Farkas' Lemma (Theorem 2.10 on page 16), there exists a vector $\binom{y}{z} \in \mathbb{R}^{n+1}$ such that

$$\begin{aligned} y^T x_i + z &\leq 0, \quad \text{for } i = 1, \ldots, k \\ y^T v + z &> 0. \end{aligned}$$

If we choose $w := -y$ and $t := -z$ we have $w^T x_i \leq t$ for $i = 1, \ldots, k$.

If $x = \sum_{i=1}^{k} \lambda_i x_i$ is a convex combination of the x_i, then as in Section 2.2 we have:

$$w^T x = \sum_{i=1}^{k} \lambda_i w^T x_i \leq \max\{w^T x_i : i = 1, \ldots, k\} \leq t.$$

Thus, $w^T x \leq t$ for all $x \in \operatorname{conv}(X)$. □

Theorem 3.36 *A polytope is equal to the convex hull of its extreme points.*

Proof: The claim is trivial, if the polytope is empty. Thus, let $P = P(A, b)$ be a nonempty polytope. Let $X = \{x_1, \ldots, x_k\}$ be the extreme points of P (which exist by Corollary 3.31 on page 30 and whose number is finite by Corollary 3.29). Since P is convex and $x_1, \ldots, x_k \in P$, we have $\operatorname{conv}(X) \subseteq P$. We must show that $\operatorname{conv}(X) = P$. Assume that there exists $v \in P \setminus \operatorname{conv}(X)$. Then, by Lemma (3.35) we can find an inequality $w^T x \leq t$ such that $w^T x \leq t$ for all $x \in \operatorname{conv}(X)$ but $w^T v > t$. Since P is bounded and nonempty, the Linear Program $\max\left\{w^T x : x \in P\right\}$ has a finite solution value $t^* \in \mathbb{R}$. Since $v \in P$ we have $t^* > t$. Thus, none of the extreme points of P is an optimal solution, which is impossible by Corollary 3.28 on page 30. □

Theorem 3.37 *A set $P \subseteq \mathbb{R}^n$ is a polytope if and only if $P = \operatorname{conv}(X)$ for a finite set $X \subseteq \mathbb{R}^n$.*

Proof: By Theorem 3.36 for any polytope P, we have $P = \operatorname{conv}(X)$, where X is the finite set of extreme points. Thus, we only need to prove the other direction.

Let $X = \{x_1, \ldots, x_k\} \subseteq \mathbb{R}^n$ be a finite set and $P = \operatorname{conv}(X)$. We define the set $Q \subseteq \mathbb{R}^{n+1}$ by

$$Q := \left\{ \begin{pmatrix} a \\ t \end{pmatrix} : a \in [-1,1]^n, t \in [-1,1], a^T x \le t \text{ for all } x \in X. \right\}$$

Since Q is bounded by construction, Q is a polytope. Let $A := \{\binom{a_1}{t_1}, \ldots, \binom{a_p}{t_p}\}$ be the set of extreme points of Q. By Theorem 3.36 we have $Q = \operatorname{conv}(A)$. Set

$$P' := \left\{ x \in \mathbb{R}^n : a_j^T x \le t_j, j = 1, \ldots, p \right\}.$$

We show that $P = P'$ which completes the proof.

"$P \subseteq P'$": Let $\bar{x} \in P = \operatorname{conv}(X)$, $\bar{x} = \sum_{i=1}^k \lambda_i x_i$ be a convex combination of the points in X. Fix $j \in \{1, \ldots, p\}$. Since $\binom{a_j}{t_j} \in Q$ we have $a_j^T x_i \le t_j$ for all i and thus

$$a_j^T \bar{x} = \sum_{i=1}^k \lambda_i \underbrace{a_j^T x_i}_{\le t_j} \le \sum_{i=1}^k \lambda_i t_j = t_j.$$

So $a_t^T \bar{x} \le t_j$ for all j and $\bar{x} \in P'$. This shows $P \subseteq P'$.

"$P' \subseteq P$": Assume that there exists a vector $v \in P' \setminus P$. Then, by Lemma 3.35 there exists an inequality $w^T x \le t$ such that $w^T x \le t$ for all $x \in P$ but $w^T v > t$. Let $\theta > 0$ be such that $\bar{w} := w/\theta \in [-1,1]^n$ and $\bar{t} := t/\theta \in [-1,1]$. Then, still $\bar{w}^T v > t$ and $\bar{w}^T x \le \bar{t}$ for all $x \in P$, and thus $\binom{\bar{w}}{\bar{t}} \in Q$.

Since Q is the convex hull of its extreme points, we can represent $\binom{\bar{w}}{\bar{t}}$ as a convex combination $\binom{\bar{w}}{\bar{t}} = \sum_{j=1}^p \lambda_j \binom{a_j}{t_j}$ of the the extreme points of Q. Since $v \in P'$ we have $a_j^T \le t_j$ for all j. This gives us

$$\bar{w}^T v = \sum_{j=1}^p \lambda_j a_j^T v \le \sum_{j=1}^p \lambda_j t_j = t,$$

which is a contradiction to the assumption that $\bar{w}^T v > t$. □

Example 3.38
As an application of Theorem 3.37 we consider the so-called *stable-set polytope* $\operatorname{STAB}(G)$, which is defined as the convex hull of the incidence vectors of stable sets in an undirected graph G (cf. Example 3.15):
(3.11)
$$\operatorname{STAB}(G) = \operatorname{conv}(\left\{ x \in \mathbb{B}^V : \ x \text{ is an incidence vector of a stable set in } G \right\}).$$

By Theorem 3.37, $\operatorname{STAB}(G)$ is a polytope whose extreme points are all (incidence vectors of) stable sets in G.

The n unit vectors $e_i = (0, \ldots, 1, 0, \ldots, 0)^T$, $i = 1, \ldots, n$ and the vector $e_0 := (0, \ldots, 0)^T$ are all contained in $\operatorname{STAB}(G)$. Thus, $\dim \operatorname{STAB}(G) = n$ and the polytope is full-dimensional. ◁

The result of Theorem 3.37 is one of the major driving forces behind polyhedral combinatorics. Let $X \subseteq \mathbb{R}^n$ be a nonempty finite set, for instance, let X be the set of incidence vectors of stable sets of a given graph G as in the above example. Then, by the preceeding theorem we can represent $\operatorname{conv}(X)$ as a pointed polytope:

$$\operatorname{conv}(X) = P = P(A, b) = \{x : Ax \leq b\}.$$

Since P is bounded and nonempty, for any given $c \in \mathbb{R}^n$ the Linear Program

$$\max\left\{c^T x : x \in P\right\} = \max\left\{c^T x : x \in \operatorname{conv}(X)\right\} \tag{3.12}$$

has a finite value which by Observation 2.2 coincides with $\max\left\{c^T x : x \in X\right\}$. By Corollary 3.28 an optimal solution of (3.12) will always be obtained at an extreme point of P, which must be a point in X itself. So, if we solve the Linear Program (3.12) we can also solve the problem of maximizing $c^T x$ over the discrete set X.

3.4 Facets

In the preceeding section we proved that for a finite set $X \subseteq \mathbb{R}^n$ its convex hull $\operatorname{conv}(X)$ is always a polytope and thus has a representation

$$\operatorname{conv}(X) = P(A, b) = \{x : Ax \leq b\}.$$

This motivates the questions which inequalities are actually needed in order to describe a polytope, or more general, to describe a polyhedron.

Definition 3.39 (Facet)
*A nontrivial face F of the polyhedron $P = P(A, b)$ is called a **facet** of P, if F is not strictly contained in any proper face of P.*

Example 3.40
Consider again the polyhedron from Example 3.14. The inequality $x \leq 3$ is valid for P. Of course, also all inequalities from (3.4) are also valid. Moreover, the inequality $x + 2y \leq 6$ defines a facet, since $(3, 3)$ and $(2, 2)$ are affinely independent. On the other hand, the inequality $x + y \leq 4$ defines a face that consists only of the point $(2, 2)$. $\lhd$

Theorem 3.41 (Characterization of facets) *Let $P = P(A, b) \subseteq \mathbb{R}^n$ be a polyhedron and F be a face of P. Then, the following statements are equivalent:*

(i) F *is a facet of* P.

(ii) $\operatorname{rank} A_{\mathrm{eq}(F),\cdot} = \operatorname{rank} A_{\mathrm{eq}(P),\cdot} + 1$

(iii) $\dim F = \dim P - 1$.

Proof: The equivalence of (ii) and (iii) is an immediate consequence of the Dimension Theorem (Theorem 3.19).

"(i)⇒(iii)": Suppose that F is a facet but $k = \dim F < \dim P - 1$. By the equivalence of (ii) and (iii) we have $\operatorname{rank} A_{I,\cdot} > A_{\mathrm{eq}(P),\cdot} + 1$, where $I = \mathrm{eq}(F)$. Chose $i \in I$ such that for $J := I \setminus \{i\}$ we have $\operatorname{rank} A_{J,\cdot} = \operatorname{rank} A_{I,\cdot} - 1$. Then $\mathrm{fa}(J)$ is a face which contains F and which has dimension $k + 1 \leq \dim P - 1$. So $\mathrm{fa}(J)$ is a proper face of P containing F which contradicts the maximality of F.

"(iii)⇒(i)": Suppose that G is any proper face of P which strictly contains F. Then F is a proper face of G and by Corollary 3.20(iv) applied to F and $P' = G$ we get $\dim F \leq \dim G - 1$ which together with $\dim F = \dim P - 1$ gives $\dim G = \dim P$. But then, again by Corollary 3.20(iv), G can not be a proper face of P. $\square$

Example 3.42
As an application of Theorem 3.41 we consider again the stable-set polytope, which we have seen to bee full-dimensional in Example 3.38.

For any $v \in V$, the inequality $x_v \geq 0$ defines a facet of $\mathrm{STAB}(G)$, since the $n-1$ unit vectors with ones at places other than position v and the zero vector form a set of n affinely independent vectors from $\mathrm{STAB}(G)$ which all satisfy the inequality as equality. $\lhd$

As a consequence of the previous theorem we show that for any facet of a polyhedron $P = P(A, b)$ there is at least one inequality in $Ax \leq b$ inducing the facet:

Corollary 3.43 *Let $P = P(A, b) \subseteq \mathbb{R}^n$ be a polyhedron and F be a facet of P. Then, there exists an $j \in M \setminus \mathrm{eq}(P)$ such that*

$$F = \{x \in P : A_{j,\cdot}x = b_j\}. \tag{3.13}$$

Proof: Let $I = \mathrm{eq}(P)$. Choose any $j \in \mathrm{eq}(F) \setminus I$ and set $J := I \cup \{j\}$. Still $J \subseteq \mathrm{eq}(F)$, since $I \subseteq \mathrm{eq}(F)$ and $j \in \mathrm{eq}(F)$. Thus, $F \subseteq \mathrm{fa}(J) \subset P$ (we have $\mathrm{fa}(J) \neq P$ since any inner point $\bar{x}$ of P has $A_{j,\cdot}\bar{x} < b_j$ since $j \in \mathrm{eq}(F) \setminus I$) and by the maximality of F we have $F = \mathrm{fa}(J)$. So,

$$\begin{aligned} F = \mathrm{fa}(J) &= \{x \in P : A_{J,\cdot}x \leq b_J\} \\ &= \{x \in P : A_{I,\cdot}x = b_I, A_{j,\cdot}x = b_j\} \\ &= \{x \in P : A_{j,\cdot}x = b_j\}, \end{aligned}$$

where the last equality follows from $I = \mathrm{eq}(P)$. $\square$

The above corollary shows that, if for a polyhedron P we know A and b such that $P = P(A, b)$, then all facets of P are of the form (3.13).

Definition 3.44 (Redundant constraint, irredundant system)
Let $P = P(A, b)$ be a polyhedron and $I = \mathrm{eq}(P)$. The constraint $A_{i,\cdot} \leq b_i$ is called ***redundant*** *with respect to $Ax \leq b$, if $P(A, b) = P(A_{M\setminus\{i\},\cdot}, b_{M\setminus\{i\}})$, that is, if we can remove the inequality without changing the solution set.*

We call $Ax \leq b$ ***irredundant*** *or* ***minimal****, if it does not contain a redundant constraint.*

Observe that removing a redundant constraint may make other redundant constraints irredundant.

The following theorem shows that in order to describe a polyhedron we need an inequality for each of its facets and that, conversely, a list of all facet defining inequalities suffices.

Theorem 3.45 (Facets are necessary and sufficient to describe a polyhedron)
Let $P = P(A, b)$ be a polyhedron with equality set $I = \mathrm{eq}(P)$ and $J := M \setminus I$. Suppose that no inequality in $A_{J,\cdot}x \leq b_J$ is redundant. Then, there is a one-to-one correspondence between the facets of P and the inequalities in $A_{J,\cdot}x \leq b_J$:

For each row $A_{j,\cdot}$ of $A_{J,\cdot}$ the inequality $A_{j,\cdot}x \leq b_j$ defines a distinct facet of P. Conversely, for each facet F of P there exists exactly one inequality in $A_{J,\cdot}x \leq b_J$ which induces F.

Proof: Let F be a facet of P. Then, by Corollary 3.43 on the previous page there exists $j \in J$ such that

$$F = \{x \in P : A_{j,\cdot}x = b_j\}. \tag{3.14}$$

Thus, each facet is represented by an inequality in $A_{J,\cdot}x \leq b_J$.

Moreover, if F_1 and F_2 are facets induced by rows $j_1 \in J$ and $j_2 \in J$ with $j_1 \neq j_2$, then we must have $F_1 \neq F_2$, since $\mathrm{eq}(F_i) = \mathrm{eq}(P) \cup \{j_i\}$ for $i = 1, 2$ by Corollary 3.43. Thus, each facet is induced by exactly one row of $A_{J,\cdot}$.

Conversely, consider any inequality $A_{j,\cdot}x \leq b_j$ where $j \in J$. We must show that the face F given in (3.14) is a facet. Clearly, $F \neq P$, since $j \in \mathrm{eq}(F) \setminus \mathrm{eq}(P)$. So $\dim F \leq \dim P - 1$. We are done, if we can show that $\mathrm{eq}(F) = \mathrm{eq}(P) \cup \{j\}$, since then $\operatorname{rank} A_{\mathrm{eq}(F),\cdot} \leq A_{\mathrm{eq}(P),\cdot} + 1$ which gives $\dim F \geq \dim P - 1$ and Theorem 3.41 proves that F is a facet.

Take any inner point $\bar{x}$ of P. This point satisfies

$$A_{I,\cdot}\bar{x} = b_I \text{ and } A_{J,\cdot}\bar{x} < b_I.$$

Let $J' := J \setminus \{j\}$. Since $A_{j,\cdot}x \leq b_j$ is not redundant in $Ax \leq b$, there exists y such that

$$A_{I,\cdot}y = b_I, A_{J',\cdot}y \leq b_{J'} \text{ and } A_{j,\cdot}y > b_j.$$

Consider $z = \lambda y + (1-\lambda)\bar{x}$. Then for an appropriate choice of $\lambda \in (0, 1)$ we have

$$A_{I,\cdot}z = b_I, A_{J',\cdot}z < b_{J'}, \text{ and } A_{j,\cdot}z = b_j.$$

Thus, $z \in F$ and $\mathrm{eq}(F) = \mathrm{eq}(P) \cup \{j\}$ as required. □

Corollary 3.46 *Each face of a polyhedron P, except for P itself, is the intersection of facets of P.*

Proof: Let $K := \mathrm{eq}(P)$. By Theorem 3.6, for each face F, there is an $I \subseteq M$ such that

$$\begin{aligned} F = \{x \in P : A_{I,\cdot}x = b_I\} &= \left\{x \in P : A_{I\setminus K,\cdot}x = b_{I\setminus K}\right\} \\ &= \bigcap_{j \in I\setminus K} \{x \in P : A_{j,\cdot}x = b_j\}, \end{aligned}$$

where by Theorem 3.45 on the preceding page each of the sets in the intersection above defines a facet. □

Corollary 3.47 *Any defining system for a polyhedron must contain a distinct facet-inducing inequality for each of its facets.* □

Lemma 3.48 *Let $P = P(A, b)$ with $I = \mathrm{eq}(P)$ and let $F = \left\{x \in P : w^T x = t\right\}$ be a proper face of P. Then, the following statements are equivalent:*

(i) F is a facet of P.

(ii) If $c^T x = \gamma$ for all $x \in F$, then c^T is a linear combination of w^T and the rows in $A_{I,\cdot}$.

Proof: "(i)⇒(ii)": We can write $F = \{x \in P : w^T x = t, c^T x = \gamma\}$, so we have for $J := \mathrm{eq}(F)$ by Theorem 3.41 and the Dimension Theorem

$$\dim P = n - \operatorname{rank} A_{I,\cdot} = 1 + \dim F \le n - \operatorname{rank} \begin{pmatrix} A_{I,\cdot} \\ w^T \\ c^T \end{pmatrix}.$$

Thus

$$\operatorname{rank} \begin{pmatrix} A_{I,\cdot} \\ w^T \\ c^T \end{pmatrix} \le \operatorname{rank} A_{I,\cdot} + 1.$$

Since F is a proper face, we have $\operatorname{rank} \binom{A_{I,\cdot}}{w^T} = \operatorname{rank} A_{I,\cdot} + 1$ which means that $\operatorname{rank} \begin{pmatrix} A_{I,\cdot} \\ w^T \\ c^T \end{pmatrix} = \operatorname{rank} \binom{A_{I,\cdot}}{w^T}$. So, c is a linear combination of w^T and the vectors in $A_{I,\cdot}$.

"(ii)⇒(i)": Let $J = \mathrm{eq}(F)$. By assumption, $\operatorname{rank} A_{J,\cdot} = \operatorname{rank} \binom{A_{I,\cdot}}{w^T} = \operatorname{rank} A_{I,\cdot} + 1$. So $\dim F = \dim P - 1$ by the Dimension Theorem and by Theorem 3.41 we get that F is a facet. □

Suppose that the polyhedron $P = P(A, b)$ is of full dimension. Then, for $I := \mathrm{eq}(P)$ we have $\operatorname{rank} A_{I,\cdot} = 0$ and we obtain the following corollary:

Corollary 3.49 *Let $P = P(A, b)$ be full-dimensional let $F = \{x \in P : w^T x = t\}$ be a proper face of P. Then, the following statements are equivalent:*

(i) F is a facet of P.

(ii) If $c^T x = \gamma$ for all $x \in F$, then $\binom{c}{\gamma}$ is a scalar multiple of $\binom{w}{t}$.

Proof: The fact that (ii) implies (i) is trivial. Conversely, if F is a facet, then by Lemma 3.48 above, c^T is a "linear combination" of w^T, that is, $c = \lambda w$ is a scalar multiple of w. Now, since for all $x \in F$ we have $w^T x = t$ and $\gamma = c^T x = \lambda w^T x = \lambda t$, the claim follows. □

Example 3.50
In Example 3.42 we saw that each inequality $x_v \ge 0$ defines a facet of the stable set polytope $\mathrm{STAB}(G)$. We now use Corollary 3.49 to provide an alternative proof.

Let $F = \{x \in \mathrm{STAB}(G) : x_v = 0\}$. We want to prove that F is a facet of $\mathrm{STAB}(G)$. Assume that $c^T x = \gamma$ for all $x \in F$. Since $(0, \dots, 0)^T \in F$, we conclude that $\gamma = 0$. Using the $n - 1$ unit vectors with ones at position other than at v we obtain that $c_u = 0$ for all $u \neq v$. Thus, $c^T = (0, \dots, \lambda, 0, \dots, 0)^T = \lambda(0, \dots, 1, 0, \dots, 0)^T$ and by Corollary 3.49, F is a facet. ◁

Corollary 3.51 *A full-dimensional polyhedron has a unique (up to positive scalar multiples) irredundant defining system.*

Proof: Let $Ax \le b$ be an irredundant defining system. Since P is full-dimensional, we have $A_{\mathrm{eq}(P),\cdot} = 0$. By Theorem 3.45 there is a one-to-one correspondence between the inequalities in $Ax \le b$ and the facets of P. By Corollary 3.49 two valid inequalities for P which induce the same facet are scalar multiples of each other. □

3.5 Minkowski's Theorem

In the previous section we have learned that each polyhedron can be represented by its facets. In this section we learn another representation of a polyhedron which is via its extreme points and extreme rays.

Definition 3.52 (Characteristic cone, (extreme) ray)
Let P *be a polyhedron. Then, its* ***characteristic cone*** *or* ***recession cone*** $\text{char. cone}(P)$ *is defined to be:*

$$\text{char. cone}(P) := \{r : x + r \in P \text{ for all } x \in P\}.$$

We call any $r \in \text{char. cone}(P)$ *a* ***ray*** *of* P*. A ray* r *of* P *is called an* ***extreme ray*** *if there do not exist rays* r^1, r^2 *of* P*,* $r^1 \neq \theta r^2$ *for any* $\theta \in \mathbb{R}_+$ *such that* $r = \lambda r^1 + (1-\lambda) r^2$ *for some* $\lambda \in [0, 1]$*.*

In other words, $\text{char. cone}(P)$ is the set of all directions y in which we can go from all $x \in P$ without leaving P. This justifies the name "ray" for all vectors in $\text{char. cone}(P)$. Since $P \neq 0$ implies that $0 \in \text{char. cone}(P)$ and $P = \emptyset$ implies $\text{char. cone}(P) = \emptyset$, we have that $\text{char. cone}(P) = \emptyset$ if and only if $P \neq \emptyset$.

Lemma 3.53 *Let* $P = P(A, b)$ *be a nonempty polyhedron. Then*

$$\text{char. cone}(P) = \{x : Ax \leq 0\}.$$

Proof: If $Ay \leq 0$, then for all $x \in P$ we have $A(x + y) = Ax + Ay \leq Ax \leq b$, so $x + y \in P$. Thus $y \in \text{char. cone}(P)$.

Conversely, if $y \in \text{char. cone}(P)$ we have $A_{i,\cdot}y \leq 0$ if there exists an $x \in P$ such that $A_{i,\cdot}x = b_i$. Let $J := \{j : A_{j,\cdot}x < b_j \text{ for all } x \in P\}$ be the set of all other indices. We are done, if we can show that $A_{J,\cdot}y \leq 0_J$.

If $A_{j,\cdot}y > 0$ for some $j \in J$, take an interior point $\bar{x} \in P$ and consider $z = \bar{x} + \lambda y$ for $\lambda \geq 0$. Then, by choosing $\lambda > 0$ appropriately, we can find $j' \in J$ such that $A_{j',\cdot}z = b_{j'}$, $A_{J,\cdot}z \leq b_J$ and $A_{I,\cdot}z \leq b_I$ which contradicts the fact that $j' \in J$. $\Box$

The characteristic cone of a polyhedron $P(A, b)$ is itself a polyhedron $\text{char. cone}(P) = P(A, 0)$, albeit a very special one. For instance, $\text{char. cone}(P)$ has at most one extreme point, namely the vector 0. To see this, assume that $r \neq 0$ is an extreme point of $\text{char. cone}(P)$. Then, $Ar \leq 0$ and from $r \neq 0$ we have we have $r \neq \frac{1}{2}r \in \text{char. cone}(P)$ and $r \neq \frac{3}{2}r \in \text{char. cone}(P)$. But then $r = \frac{1}{2}(\frac{1}{2}r) + \frac{1}{2}(\frac{3}{2}r)$ is a convex combination of two distinct points in $\text{char. cone}(P)$ contradicting the assumption that r an extreme point of $\text{char. cone}(P)$.

Together with Corollary 3.30 on page 30 we have:

Observation 3.54 *Let* $P = P(A, b) \neq \emptyset$*. Then,* $0 \in \text{char. cone}(P)$ *and* 0 *is the only potential extreme point of* $\text{char. cone}(P)$*. The zero vector is an extreme point of* $\text{char. cone}(P)$ *if and only if* $\operatorname{rank} A = n$*.*

Theorem 3.55 (Characterization of extreme rays) *points] Let* $P = P(A, b) \subseteq \mathbb{R}^n$ *be a nonempty polyhedron. Then, the following statements are equivalent:*

(i) r *is an extreme ray of* P*.*

(ii) $\{\theta r : \theta \in \mathbb{R}_+\}$ *is a one-dimensional face of* $\operatorname{char.cone}(P) = \{x : Ax \leq 0\}$.

(iii) $r \in \operatorname{char.cone}(P) \setminus \{0\}$ *and for* $I := \{i : A_{i,\cdot}r = 0\}$ *we have* $\operatorname{rank} A_{I,\cdot} = n - 1$.

Proof: Let $I := \{i : A_{i,\cdot}r = 0\}$.

"(i)⇒(ii)": Let F be the smallest face of $\operatorname{char.cone}(P)$ containing the set $\{\theta r : \theta \in \mathbb{R}_+\}$. By Observation 3.9 on page 22 we have

$$F = \{x \in \operatorname{char.cone}(P) : A_{I,\cdot}x = 0_I\}$$

and $\operatorname{eq}(F) = I$. If $\dim F > 1$, then the Dimension Theorem tells us that $\operatorname{rank} A_{I,\cdot} < n - 1$. Thus, the solution set of $A_{I,\cdot}x = 0_I$ contains a vector r^1 which is linearly independent from r. For sufficently small $\varepsilon > 0$ we have $r \pm \varepsilon r^1 \in \operatorname{char.cone}(P)$, since $A_{I,\cdot}r = 0_I$, $A_{I,\cdot}r^1 = 0_I$ and $A_{M\setminus I,\cdot}r < 0$. But then $r = \frac{1}{2}(r + \varepsilon r^1) + \frac{1}{2}(r - \varepsilon r^1)$ contradicting the fact that r is an extreme ray.

So $\dim F = 1$. Since $r \neq 0$, the unbounded set $\{\theta r : \theta \in \mathbb{R}_+\}$ which is contained in F has also dimension 1, thence $F = \{\theta r : \theta \in \mathbb{R}_+\}$ is a one-dimensional face of $\operatorname{char.cone}(P)$.

"(ii)⇒(iii)": The Dimension Theorem applied to $\operatorname{char.cone}(P)$ implies that $\operatorname{rank} A_{I,\cdot} = n - 1$. Since $\{\theta r : \theta \in \mathbb{R}_+\}$ has dimension 1 it follows that $r \neq 0$.

"(iii)⇒(i)": By Linear Algebra, the solution set of $A_{I,\cdot}x = 0_I$ is one-dimensional. Since $A_{I,\cdot}r = 0_I$ and $r \neq 0$, for any y with $A_{I,\cdot}y = 0_I$ we have $y = \theta r$ for some $\theta \in \mathbb{R}$.

Suppose that $r = \lambda r^1 + (1 - \lambda) r^2$ for some $r^1, r^2 \in \operatorname{char.cone}(P)$. Then

$$0_I = A_{I,\cdot}r = \lambda \underbrace{A_{I,\cdot}r^1}_{\leq 0_I} + (1 - \lambda) \underbrace{A_{I,\cdot}r^2}_{\leq 0_I} \leq 0_I$$

implies that $A_{I,\cdot}r^j = 0$ for $j = 1, 2$. So $r^j = \theta_j r_j$ for appropriate scalars, and both rays r^1 and r^2 must be scalar multiples of each other. □

Corollary 3.56 *Every polyhedron has only a finite number of extreme rays.*

Proof: By the preceeding theorem, every extreme point is a face. By Corollary 3.7, there is only a finite number of faces. □

Combining this result with Corollary 3.29 on page 30 we have:

Corollary 3.57 *Every polyhedron has only a finite number of extreme points and rays.*

Consider once more the Linear Program

$$\max\left\{c^T x : x \in P\right\}, \tag{3.15}$$

where P is a pointed polyhedron. We showed in Corollary 3.28 on page 30, that if the Linear Program (3.10) has optimal solutions, it has an optimal solution which is also an extreme point of P. What happens, if the Linear Program (3.15) is unbounded?

Theorem 3.58 *Let* $P = P(A, b)$ *be a pointed nonempty polyhedron.*

(i) If the Linear Program (3.15) has optimal solutions, it has an optimal solution which is also an extreme point of P.

(ii) If the Linear Program (3.15) is unbounded, then there exists an extreme ray r *of* P *such that* $c^T r > 0$.

Proof: Statement (i) is a restatement of Corollary 3.28 on page 30. So, we only need to prove (ii). By the Duality Theorem of Linear Programming (Theorem 2.8 on page 15) the set

$$\{y : A^T y = c, y \geq 0\}$$

(which is the feasible set for the dual to (3.10)) must be empty. By Farkas' Lemma (Theorem 2.10 on page 16), there exists r such that $Ar \geq 0$ and $c^T r < 0$. Let $r' := -r$, then $Ar' \leq 0$ and $c^T r' > 0$. In particular, r is a ray of P.

We now consider the Linear Program

$$\max\{c^T x : Ax \leq 0, c^T x \leq 1\} = \max\{c^T x : x \in P'\}. \tag{3.16}$$

Since $\operatorname{rank} A = n$, it follows that P' is pointed. Moreover, $P' \neq \emptyset$ since $r'/c^T r' \in P$. Finally, the constraint $c^T x \leq 1$ ensures that (3.16) is bounded. In fact, the optimum value of (3.16) is 1, since this value is achieved for the vector $r'/c^T r'$.

By (i) an optimal solution of (3.16) is attained at an extreme point of $P' \subseteq \operatorname{char.cone}(P)$, say at $r^* \in \operatorname{char.cone}(P)$. So

$$c^T r^* = 1. \tag{3.17}$$

Let $I = \{i : A_{i,\cdot} r^* = 0\}$. By the Dimension Theorem we have $\operatorname{rank}\begin{pmatrix} A_{I,\cdot} \\ c^T \end{pmatrix} = n$ (since $\{r^*\}$ is a zero-dimensional face of the polyhedron P' by Theorem 3.27 on page 29).

If $\operatorname{rank} A_{I,\cdot} = n - 1$, then by Theorem 3.55 on page 38 we have that r^* is an extreme ray of P as needed. If $\operatorname{rank} A_{I,\cdot} = n$, then $r^* \neq 0$ is an extreme point of $\operatorname{char.cone}(P)$ by Theorem 3.27 on page 29 which is a contradiction to Observation 3.54 on page 38. □

The following theorem states a fundamental result on the representation of polyhedra:

Theorem 3.59 (Minkowski's Theorem) *Let* $P = P(A, b)$ *be nonempty and* $\operatorname{rank}(A) = n$ *(observe that by Corollary 3.30 on page 30 this implies that the polyhedron* P *is pointed). Then*

$$P = \left\{ x \in \mathbb{R}^n : x = \sum_{k \in K} \lambda_k x^k + \sum_{j \in J} \mu_j r^j, \sum_{k \in K} \lambda_k = 1, \lambda_k \geq 0 \text{ for } k \in K, \mu_j \geq 0 \text{ for } j \in J \right\}, \tag{3.18}$$

where x^k, $k \in K$ *are the extreme points of* P *and* r^j, $j \in J$ *are the extreme rays of* P.

Proof: Let Q be the set on the right hand side of (3.18). Since $x^k \in P$ for $k \in K$ and P is convex, we have that any convex combination $x = \sum_{k \in K} \lambda_k x^k$ of the extreme points of P is also in P. Moreover, since the r^j are rays of P, we have that $x + \sum_{j \in J} \mu_j r^j \in P$ for any $\mu_j \geq 0$. Thus, $Q \subseteq P$.

Assume for the sake of a contradiction that there exists $v \in P \setminus Q$. By assumption, there is no λ, μ solving the following linear system:

$$\sum_{k\in K} \lambda_k x^k + \sum_{j\in J} \mu_j r^j = v \tag{3.19a}$$

$$-\sum_{k\in K} \lambda_k = -1 \tag{3.19b}$$

$$\lambda, \mu \geq 0 \tag{3.19c}$$

We can write the solution set of (3.19) in the form $\left\{x : \bar{A}\binom{\lambda}{\mu} = \bar{b}, \binom{\lambda}{\mu} \geq 0\right\}$, where

$$\bar{A} = \begin{pmatrix} x^1 & x^2 & \dots & r_1 & r_2 & \dots \\ -1 & -1 & \dots & 0 & 0 & \dots \end{pmatrix}, \quad \bar{b} = \begin{pmatrix} v \\ -1 \end{pmatrix}$$

By Farkas' Lemma (see Theorem 2.10 on page 16), there exists $(y, t) \in \mathbb{R}^{n+1}$ such that $\bar{A}^T\binom{y}{t} \leq 0$ and $\bar{b}^T\binom{y}{t} > 0$. This is equivalent to:

$$y^T x^k - t \leq 0 \qquad \text{for all } k \in K \tag{3.20a}$$

$$y^T r^j \leq 0 \qquad \text{for all } j \in J \tag{3.20b}$$

$$y^T v - t > 0. \tag{3.20c}$$

Consider the Linear Program

$$\max\left\{y^T x : x \in P\right\}. \tag{3.21}$$

Recall that, since $\operatorname{rank} A = n$ we have that P is pointed.

If (3.21) has an optimal solution, then by Theorem 3.58 on page 39(i) there exists an optimal solution of (3.21) which is also an extreme point. However, $y^T x^k \leq t$ and $y^T v > t$ for the vector $v \in P \setminus Q$ by (3.20a) and (3.20c)which is a contradiction to this fact.

On the other hand, if (3.21) is unbounded, then by Theorem 3.58 on page 39(ii), there must be an extreme ray r^j with $y^T r^j > 0$ which is a contradiction to (3.20b). $\square$

The above theorem tells us that we can represent each polyhedron by its extreme points and extreme rays.

Now let $P = P(A, b)$ be a rational pointed polyhedron where A and b are already choosen to have rational entries. By Theorem 3.27 the extreme points of P are the 0-dimensional faces. By the Theorem of Hoffmann and Kruskal (Theorem 3.22 on page 27) each extreme point $\{\bar{x}\}$ is the unique solution of a subsystem $A_{I,\cdot}x = b_I$. Since A and b are rational, it follows that each extreme point has also rational entries (Gaussian elimination applied to the linear system $A_{I,\cdot}x = b_I$ does not leave the rationals). Similarly, by Theorem 3.55 an extreme ray r is determined by a system $A_{I,\cdot}r = 0$, where $\operatorname{rank} A = n - 1$. So again, r must be rational. This gives us the following observation:

Observation 3.60 *The extreme points and extreme rays of a rational polyhedron are rational vectors.*

3.6 Most IPs are Linear Programs

This section is dedicated to establishing the important fact that if

$$X = \{x \in \mathbb{R} : Ax \leq b, x \geq 0\} \cap \mathbb{Z}^n$$

is nonempty, then $\operatorname{conv}(X)$ is a rational polyhedron. In order to do so, we need a few auxiliary results.

Consider the two Linear Programs

$$\begin{array}{ll} \max\left\{c^T x : x \in P\right\} & \min\left\{b^T y : y \in Q\right\} \\ P = \{x \in \mathbb{R}^n : Ax \leq b\} & Q = \left\{y \in \mathbb{R}^m : A^T y = c, y \geq 0\right\}. \end{array}$$

Observe that Q is pointed (see Corollary 3.32 on page 31). From Linear Programming duality we know that if P and Q are nonempty, then the maximum and the minimum both exist and their values coincide.

Lemma 3.61 *Let P and Q be defined as above. Then $P \neq \varnothing$ if and only if $b^T v^t \geq 0$ for all $t \in T$, where v^t, $t \in T$ are the extreme rays of Q.*

Proof: We have

$$P = \{x : Ax \leq b\} = \varnothing \iff \left\{(x^+, x^-, s) \in \mathbb{R}_+^{2n+m} : Ax^+ - Ax^- + s = b\right\} = \varnothing.$$

By Farkas' Lemma (Theorem 2.10 on page 16) the set on the right hand side above is nonempty, if and only if for all y such that $\begin{pmatrix} A^T \\ -A^T \\ I \end{pmatrix} y \geq 0$ we have $b^T y \geq 0$. In other words, $P \neq \varnothing$ if and only if for all $y \geq 0$ such that $A^T y = 0$ we have $b^T y \geq 0$.

Observe that

$$\operatorname{char.cone}(Q) = \left\{ y : \begin{pmatrix} A^T \\ -A^T \\ -I \end{pmatrix} y \leq 0 \right\} = \left\{y : A^T y = 0, y \geq 0\right\}.$$

So we have that $P \neq \varnothing$ if and only if $b^T v \geq 0$ for all $v \in \operatorname{char.cone}(Q)$.

In particular, if $P \neq \varnothing$, then $b^T v \geq 0$ for all extreme rays v of Q (since each extreme ray of Q is a vector in $\operatorname{char.cone}(Q)$). Conversely, since any ray in $\operatorname{char.cone}(Q)$ is a convex combination of extreme rays, the condition $b^T v \geq 0$ for all extreme rays v of Q implies that $b^T v \geq 0$ for all $v \in \operatorname{char.cone}(Q)$ which implies that $P \neq \varnothing$. □

Definition 3.62 (Projection of a polyhedron)
Given a polyhedron $Q \subseteq \mathbb{R}^{n+k}$ we define the ***projection*** *of Q onto the subspace $\mathbb{R}^n$ as:*

$$\operatorname{proj}_x Q := \{ x \in \mathbb{R}^n : (x, w) \in Q \text{ for some } w \in \mathbb{R}^k \}.$$

Theorem 3.63 *Let $P = \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^p : Ax + Gy \leq b\}$ and v^t, $t \in T$ be the extreme rays of $Q = \left\{y \in \mathbb{R}_+^m : G^T y = 0\right\}$. Then*

$$\operatorname{proj}_x(P) = \left\{x \in \mathbb{R}^n : (v^t)^T (b - Ax) \geq 0 \text{ for all } t \in T\right\}$$

Proof: We have that $\text{proj}_x(P) = \bigcup_{x \in \mathbb{R}^n : M_x \neq \varnothing} \{x\}$ where

$$M_x := \{y \in \mathbb{R}^p : Gy \leq b - Ax\}. \tag{3.22}$$

We apply Lemma 3.61 to the polyhedron M_x from (3.22). The lemma shows that $M_x \neq \varnothing$ if and only if $(v^t)^T(b - Ax) \geq 0$ for all extreme rays v^t of $Q = \left\{v \in \mathbb{R}^m_+ : G^T v = 0\right\}$. Hence, we have

$$\text{proj}_x(P) = \left\{x \in \mathbb{R}^n : (v^t)^T(b - Ax) \geq 0 \text{ for all } t \in T\right\}$$

as claimed. □

We immediately obtain the following corollary:

Corollary 3.64 *The projection of a polyhedron is a polyhedron.*

Another important consequence of Theorem 3.63 is the following converse of Minkowski's Theorem:

Theorem 3.65 (Weyl's Theorem) *If A is a rational $m_1 \times n$ matrix, B is a rational $m_2 \times n$ matrix and*

$$P = \left\{x \in \mathbb{R}^n_+ : x = A^T y + B^T z, \sum_{k=1}^{m_1} y_k = 1, y \in \mathbb{R}^{m_1}_+, z \in \mathbb{R}^{m_2}_+\right\},$$

then P is a rational polyhedron.

Proof: Observe that $P = \text{proj}_x Q$, where

$$Q = \left\{(x, y, z) \in \mathbb{R}^n \times \mathbb{R}^{m_1}_+ \times \mathbb{R}^{m_2}_+ : x - A^T y - B^T z = 0, \sum_{k=1}^{m_1} y_k = 1\right\}.$$

We now apply Theorem 3.63 to get another description of $P = \text{proj}_x Q$. Observe that we can write $Q = \left\{(x, \bar{y}) : \bar{A}x + \bar{G}\bar{y} \leq \bar{b}\right\}$, where $\bar{A}$, $\bar{G}$ are rational matrices, $\bar{b}$ is a rational vector and $\bar{y} = (y, z) \in \mathbb{R}^{m_1} \times \mathbb{R}^{m_2}$. By Theorem 3.63 we have

$$\text{proj}_x Q = \left\{x \in \mathbb{R}^n : (v^t)^T(\bar{b} - \bar{A}x) \geq 0 \text{ for all } t \in T\right\}, \tag{3.23}$$

where v^t are the finitely many extreme rays of the rational polyhedron $\left\{\bar{y} : G^T\bar{y} = 0\right\}$. Since all of the extreme rays have rational coordinates it follows that (3.23) is a rational polyhedron. □

Suppose now that we are given $P = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$, where A, b are integral and let $X = P \cap \mathbb{Z}^n$. If P is bounded, then X contains a finite number of points, say $X = \{a^1, \ldots, a^k\}$. We already know by Theorem 3.37 on page 32 that $\text{conv}(X)$ is a polytope. We now know more!

Any point v in $\text{conv}(X)$ is of the form $v = \sum_{i=1}^k y_i a^i$ with $y_i \geq 0$ and $\sum_{i=1}^k y_i = 1$. Thus, if A^T is the matrix whose columns are the a^i, then $\text{conv}(X)$ is of the form

$$\text{conv}(X) = \left\{x \in \mathbb{R}^n_+ : x = A^T y, \sum_{i=1}^k y_i = 1, y \in \mathbb{R}^k_+\right\}.$$

Thus, by Weyl's theorem $\mathrm{conv}(X)$ is a rational polyhedron . In the remainder of this section we are going to show the analogous result, when X contains an infinite number of points. The idea behind the proof is to find a finite set $Q \subseteq X$ and to show that every point in X can be generated by taking a point in Q plus a nonnegative integer linear combination of the extreme rays of P.

Lemma 3.66 *Let $P = P(A, b) \neq \varnothing$ and $X = P \cap \mathbb{Z}^n$ where A, b are integral. There exists a finite set of points $q^l \in X$, $l \in L$ and a finite set of rays r^j, $j \in J$ of P such that*

$$X = \left\{ x \in \mathbb{R}^n_+ : x = q^l + \sum_{j \in J} \beta_j r^j, l \in L, \beta \in \mathbb{Z}^j_+ \right\}$$

Proof: Let x^k, $k \in K$ be the extreme points and r^j, $j \in J$ be the extreme rays of P. Since P is a rational polyhedron, all of these vectors have rational coordinates. By Minkowski's Theorem we have:

$$P = \left\{ x \in \mathbb{R}^n_+ : x = \sum_{k \in K} \lambda_k x^k + \sum_{j \in J} \mu_j r^j, \sum_{k \in K} \lambda^k = 1, \lambda_k \geq 0 \text{ for } k \in K, \mu_j \geq 0 \text{ for } j \in J \right\},$$

Thus, without loss of generality we can assume that the r^j have integral entries. We define the set $Q \subseteq S$ by

$$Q = \left\{ x \in \mathbb{Z}^n_+ : x = \sum_{k \in K} \lambda_k x^k + \sum_{j \in J} \mu_j r^j, \sum_{k \in K} \lambda_k = 1, \lambda_k \geq 0 \text{ for } k \in K, 0 \leq \mu_j < 1 \text{ for } j \in J \right\}.$$

Then, Q is a finite set, since Q is bounded and contains only integral points. Suppose that $Q = \left\{ q^l : l \in L \right\} \subset \mathbb{Z}^n$. Observe that we can write any point $x^i \in P$ as

$$x^i = \underbrace{\left(\sum_{k \in K} \lambda_k x^k + \sum_{j \in J} (\mu_j - \lfloor \mu_j \rfloor) r^j \right)}_{=: q^{l_i} \in Q \text{ if } x^i \in \mathbb{Z}^n} + \sum_{j \in J} \lfloor \mu_j \rfloor r^j, \tag{3.24}$$

where $\sum_{k \in K} \lambda^k = 1$, $\lambda_k \geq 0$ for $k \in K$ and $\mu_j \geq 0$ for $j \in J$. Hence, $x^i \in X$ if any only $x^i \in \mathbb{Z}^n_+$ and x^i is of the form (3.24). Observe that, if $x^i \in \mathbb{Z}^n$, then the first term in (3.24) is a point $q^{l_i} \in Q$ (it is integral, since the second term $\sum_{j \in J} \lfloor \mu_j \rfloor r^j$ is integral). Hence, we have have

$$x^i = q^{l_i} + \sum_{j \in J} \beta^i_j r^j, \quad \beta^i_j = \lfloor \mu^i_j \rfloor \text{ for } j \in J \tag{3.25}$$

where $\beta_j = \lfloor \mu_j \rfloor$ is integral. Thus,

$$\begin{aligned} X &= \left\{ x \in \mathbb{Z}^n_+ : x = q + \sum_{j \in J} \beta_j r^j, \beta \in \mathbb{Z}^l_+, \text{ and } q \in L \right\} \\ &= \left\{ x \in \mathbb{R}^n_+ : x = q + \sum_{j \in J} \beta_j r^j, \beta \in \mathbb{Z}^l_+, \text{ and } q \in L \right\}, \end{aligned}$$

where the equality of the two sets follows from the fact that we use only integral linear combinations of integral vectors. This shows the claim. □

We are now ready to establish the main result of this chapter.

Theorem 3.67 (Most IPs are LPs) *Let* $P = \left\{x \in \mathbb{R}^n_+ : Ax \leq b\right\}$ *with integral* A *and* b *and* $X = P \cap \mathbb{Z}^n$*. Then* $\text{conv}(X)$ *is a rational polyhedron.*

Proof: In the proof of the previous lemma we have shown that any point $x^i \in X$ can be written in the form (3.25). Thus, any point $x \in \text{conv}(X)$ can be written as a convex combination of points of the form (3.25):

$$
\begin{aligned}
x &= \sum_{i \in I} \lambda_i x^i \\
&= \sum_{i \in I} \lambda_i \left(q^{l_i} + \sum_{j \in J} \beta_j r^j \right) \\
&= \sum_{l \in L} \left(\sum_{i \in I : l_i = l} \lambda_i \right) q^l + \sum_{j \in J} \left(\sum_{i \in I} \lambda_i \beta^i_j \right) r^j \\
&= \sum_{l \in L} \alpha_l q^l + \sum_{j \in J} \beta_j r^j,
\end{aligned}
$$

where $\alpha_l = \sum_{i \in I : l_i = l} \lambda_i$ and $\beta_j = \sum_{i \in I} \lambda_i \beta^i_j$. Observe that

$$\sum_{l \in L} \alpha_l = \sum_{i \in I} \lambda_i = 1$$

and

$$\beta_j = \sum_{i \in I} \underbrace{\lambda_i}_{\geq 0} \underbrace{\beta^i_j}_{\geq 0} \geq 0.$$

This shows that

$$\text{conv}(X) = \left\{ x \in \mathbb{R}^n : x = \sum_{l \in L} \alpha_l q^l + \sum_{j \in J} \beta_j r^j, \sum_{l \in L} \alpha_l = 1, \alpha_l, \beta_j \geq 0 \text{ for } l \in L, j \in J \right\},$$

where the q^l and the r^j are all integral. By Weyl's Theorem, it now follows that $\text{conv}(X)$ is a rational polyhedron. □

It should be noted that the above result can be extended rather easily to mixed integer programs. Moreover, as a byproduct of the proof we obtain the following observation:

Observation 3.68 *Let* $P = \left\{x \in \mathbb{R}^n_+ : Ax \leq b\right\}$ *with integral* A *and* b *and* $X = P \cap \mathbb{Z}^n$*. If* $X \neq \emptyset$*, then the extreme rays of* P *and* $\text{conv}(X)$ *coincide.*

Theorem 3.69 *Let* $P = \left\{x \in \mathbb{R}^n_+ : Ax \leq b\right\}$ *and* $X = P \cap \mathbb{Z}^n$ *where* $X \neq \emptyset$*. Let* $c \in \mathbb{R}^n$ *be arbitrary. We consider the two optimization problems:*

$$
\begin{aligned}
(IP) \quad & z^{IP} = \max\left\{c^T x : x \in X\right\} \\
(LP) \quad & z^{LP} = \max\left\{c^T x : x \in \text{conv}(X)\right\}.
\end{aligned}
$$

Then, the following statements hold:

(i) The objective value of IP is bounded from above if and only if the objective value of LP is bounded from above.

(ii) If LP has a bounded optimal value, then it has an optimal solution (namely, an extreme point of $\operatorname{conv}(X)$*), that is an optimal solution to IP.*

(iii) If x^* *is an optimal solution to IP, then* x^* *is also an optimal solution to LP.*

Proof: Since $X \subseteq \operatorname{conv}(X)$ it trivially follows that $z^{\text{IP}} \geq z^{\text{IP}}$.

(i) If $z^{\text{IP}} = +\infty$ it follows that $z^{\text{LP}} = +\infty$. on the other hand, if $z^{\text{LP}} = +\infty$, there is an integral extreme point $x^0 \in \operatorname{conv}(X)$ and an integral extreme ray r of $\operatorname{conv}(X)$ such that $c^T x^0 + \mu r \in \operatorname{conv}(X)$ for all $\mu \geq 0$ and $c^T r > 0$. Thus, $x^0 + \mu r \in X$ for all $\mu \in \mathbb{N}$. Thus, we also have $z^{\text{IP}} = +\infty$.

(ii) By Theorem 3.67 we know that $\operatorname{conv}(X)$ is a rational polyhedron. Hence, if LP has an optimal solution, there exists also an optimal solution which is an extreme point of $\operatorname{conv}(X)$, say x^0. But then $x^0 \in X$ and $z^{\text{IP}} \geq c^T x^0 = z^{\text{LP}} \geq z^{\text{IP}}$. Hence, x^0 is also an optimal solution for IP.

(iii) Since $x^* \in X \subseteq \operatorname{conv}(X)$, the point x^* is also feasible for LP. The claim now follow from (ii).

$\square$

Theorems 3.67 and 3.69 are particularly interesting in conjunction with the polynomial time equivalence of the separation and optimization (see Theorem 5.24 on page 78 later on). A general method for showing that an Integer Linear Program $\max\left\{c^T x : x \in X\right\}$ with $X = P(A, b) \cap \mathbb{Z}^n$ can be solved in polynomial time is as follows:

1. Find a description of $\operatorname{conv}(X)$, that is, $\operatorname{conv}(X) = P' = \{x : A'x \leq b'\}$.
2. Give a polynomial time separation algorithm for P'.
3. Apply Theorem 5.24.

Although this procedure does usually not yield algorithms that appear to be the most efficient in practice, the equivalence of optimization and separation should be viewed as a guide for searching for more efficient algorithms. In fact, for the vast majority of problems that were first shown to be solvable in polynomial time by the method outlined above, later algorithms were developed that are faster both in theory and practice.

4

Integrality of Polyhedra

In this chapter we study properties of polyhedra P which ensure that the Linear Program $\max\left\{c^T x : x \in P\right\}$ has optimal integral solutions.

Definition 4.1 (Integral Polyhedron)
*A polyhedron P is called **integral** if every face of P contains an integral point.*

Informally speaking, if we are optimizing over an integral polyhedron we get integrality for free: the set of optimal solutions of $z = \max\left\{c^T x : x \in P\right\}$ is a face $F = \left\{x \in P : c^T x = z\right\}$ of P, and, if each face contains an integral point, then there is also an optimal solution which is also integral. In other words, for integral polyhedra we have

$$\max\left\{c^T x : x \in P\right\} = \max\left\{c^T x : x \in P \cap \mathbb{Z}^n\right\}. \tag{4.1}$$

Thus, the IP on the right hand side of (4.1) can be solved by solving the Linear Program on the left hand side of (4.1).

A large part of the study of polyhedral methods for combinatorial optimization problems was motivated by a theorem of Edmonds on matchings in graphs. A matching in an undirected graph $G = (V, E)$ is a set $M \subseteq E$ of edges such that none of the edges in M share a common endpoint. Given a matching M we say that a vertex $v \in V$ is M-*covered* if some edge in M is incident with v. Otherwise, we call v M-*exposed*. Observe that the number of M-exposed nodes is precisely $|V| - 2|M|$. We define:

$$\mathrm{PM}(G) := \left\{\chi^M \in \mathbb{B}^E : \text{ M is a perfect matching in } G\right\} \tag{4.2}$$

to be the set of incidence vectors of perfect matchings in G.

We will show in the next section that a polyhedron P is integral if and only if $P = \mathrm{conv}(P \cap \mathbb{Z}^n)$. Edmonds' Theorem can be stated as follows:

Theorem 4.2 (Perfect Matching Polytope Theorem) *For any graph $G = (V, E)$, the convex hull $\mathrm{conv}(\mathrm{PM}(G))$ of the perfect matchings in G is identical to the set of solutions of the following linear system:*

(4.3a)	$x(\delta(v)) = 1$	*for all* $v \in V$
(4.3b)	$x(\delta(S)) \geq 1$	*for all* $S \subseteq V$, $\lvert S\rvert \geq 3$ *odd*
(4.3c)	$x_e \geq 0$	*for all* $e \in E$.

Proof: See Theorem 4.23 on page 59. □

Observe that any integral solution of (4.3) is a perfect matching. Thus, if P denotes the polyhedron defined by (4.3), then by the equivalence shown in the next section the Perfect Matching Polytope Theorem states that P is integral and $P = \text{conv}(\text{PM}(G))$. Edmond's results is very strong, since it gives us an explicit description of $\text{conv}(\text{PM}(G))$.

4.1 Equivalent Definitions of Integrality

We are now going to give some equivalent definitions of integrality which will turn out to be quite useful later.

Theorem 4.3 *Let $P = P(A, b)$ be a pointed rational polyhedron. Then, the following statements are equivalent:*

(i) P *is an integral polyhedron.*

(ii) The LP $\max \left\{c^T x : x \in P\right\}$ *has an optimal integral solution for all* $c \in \mathbb{R}^n$ *where the value is finite.*

(iii) The LP $\max \left\{c^T x : x \in P\right\}$ *has an optimal integral solution for all* $c \in \mathbb{Z}^n$ *where the value is finite.*

(iv) The value $z^{LP} = \max \left\{c^T x : x \in P\right\}$ *is integral for all* $c \in \mathbb{Z}^n$ *where the value is finite.*

(v) $P = \text{conv}(P \cap \mathbb{Z}^n)$.

Proof: We first show the equivalence of statements (i)-(iv):

(i)⇒(ii) The set of optimal solutions of the LP is a face of P. Since every face contains an integral point, there is an integral optimal solution.

(ii)⇒(iii) trivial.

(iii)⇒(iv) trivial.

(iv)⇒(i) Suppose that (i) is false and let x^0 be an extreme point which by assumption is not integral, say component x^0_j is fractional. By Theorem 3.33 there exists a vector $c \in \mathbb{Z}^n$ such that x^0 is the unique solution of $\max \left\{c^T x : x \in P\right\}$. Since x^0 is the unique solution, we can find a large $\omega \in \mathbb{N}$ such that x^0 is also optimal for the objective vector $\bar{c} := c + \frac{1}{\omega} e_j$, where e_j is the jth unit vector. Clearly, x^0 must then also be optimal for the objective vector $\tilde{c} := \omega\bar{c} = \omega c + e_j$. Now we have

$$\tilde{c}^T x^0 - \omega c^T x^0 = (\omega c^T x^0 + e_j^T x^0) - \omega c^T x^0 = e_j^T x^0 = x^0_j.$$

Hence, at least one of the two values $\tilde{c}^T x^0$ and $c^T x^0$ must be fractional, which contradicts (iv).

We complete the proof of the theorem by showing two implications:

(i)⇒(v) Since P is convex, we have $\text{conv}(P \cap \mathbb{Z}^n) \subseteq P$. Thus, the claim follows if we can show that $P \subseteq \text{conv}(P \cap \mathbb{Z}^n)$. Let $v \in P$, then $v = \sum_{k \in K} \lambda_k x^k + \sum_{j \in J} \mu_j r^j$, where the x^k are the extreme points of P and the r^j are the extreme rays of P. By (i) every x^k is integral, thus $\sum_{k \in K} \lambda_k x^k \in \text{conv}(P \cap \mathbb{Z}^n)$. Since by Observation 3.68 the extreme rays of P and $\text{conv}(P \cap \mathbb{Z}^n)$ are the same, we get that $v \in \text{conv}(P \cap \mathbb{Z}^n)$.

(v)⇒(iv) Let $c \in \mathbb{Z}^n$ be an integral vector. Since by assumption $\text{conv}(P \cap \mathbb{Z}^n) = P$, the LP $\max \{c^T x : x \in P\}$ has an optimal solution in $P \cap \mathbb{Z}^n$ (If $x = \sum_i x^i \in \text{conv}(P \cap \mathbb{Z}^n)$ is a convex combination of points in $P \cap \mathbb{Z}^n$, then $c^T x \leq \max_i c^T x^i$ (cf. Observation 2.2)). Thus, the LP has an integral value for every integral $c \in \mathbb{Z}^n$ where the value is finite.

This shows the theorem. □

Recall that each minimal nonempty face of $P(A, b)$ is an extreme point if and only if $\text{rank}(A) = n$ (Corollary 3.23 on page 28). Thus, we have the following result:

Observation 4.4 *A nonempty polyhedron $P = P(A, b)$ with $\text{rank}(A) = n$ is integral if and only if all of its extreme points are integral.* □

Moreover, if $P(A, b) \subseteq \mathbb{R}^n_+$ is nonempty, then $\text{rank}(A) = n$. Hence, we also have the following corollary:

Corollary 4.5 *A nonempty polyhedron $P \subseteq \mathbb{R}^n_+$ is integral if and only if all of its extreme points are integral.* □

4.2 Matchings and Integral Polyhedra I

As mentioned before, a lot of the interest about integral polyhedra and their applications in combinatorial optimization was fueled by results on the matching polytope. As a warmup we are going to prove a weaker form of the perfect matching polytope due to Birkhoff.

A graph $G = (V, E)$ is called *bipartite*, if there is a partition $V = A \cup B$, $A \cap B = \emptyset$ of the vertex set such that every edge e is of the form $e = (a, b)$ with $a \in A$ and $b \in B$.

Lemma 4.6 *A graph $G = (V, E)$ is bipartite if and only if it does not contain an odd cycle.*

Proof: Let $G = (V, E)$ be bipartite with bipartition $V = A \cup B$. Assume for the sake of a contradiction that $C = (v_1, v_2, \ldots, v_{2k-1}, v_{2k} = v_1)$ is an odd cycle in G. We can assume that $v_1 \in A$. Then $(v_1, v_2) \in E$ implies that $v_2 \in B$. Now $(v_2, v_3) \in E$ implies $v_3 \in A$. Continuing we get that $v_{2i-1} \in A$ and $v_{2i} \in B$ for $i = 1, 2, \ldots$. But since $v_1 = v_{2k}$ we have $v_1 \in A \cap B = \emptyset$, which is a contradiction.

Assume conversely that $G = (V, E)$ does not contain an odd cycle. Since it suffices to show that any connected component of G is bipartite, we can assume without loss of generality that G is connected.

Choose $r \in V$ arbitrary. Since G is connected, the shortest path distances from v to all $v \in V$ are finite. We let

$$A = \{v \in V : d(v) \text{ is even}\}$$
$$B = \{v \in V : d(v) \text{ is odd}\}$$

This gives us a partition of V with $r \in A$. We claim that all edges are between A and B. Let $(u, v) \in E$ and suppose that $u, v \in A$. Clearly, $|d(u) - d(v)| \leq 1$

which gives us that $d(u) = d(v) = 2k$. Let $p = r, v_1, \dots, v_{2k} = v$ and $q = r, u_1, \dots, u_{2k} = u$ be shortest paths from r to v and u, respectively. The paths might share some common parts. Let v_i and u_j be maximal with the property that $v_i = u_j$ and the paths $v_{i+1}, \dots, v$ and $u_{j+1}, \dots, u$ are node disjoint. Observe that we must have that $i = j$ since otherwise one of the paths could not be shortest. But then $v_i, v_{i+1}, \dots, v_{2k} = v, u = u_{2k}, u_{2k-1}, \dots, u_i = v_i$ is a cycle of odd length, which is a contradiction. $\square$

Theorem 4.7 (Birkhoff's Theorem) *Let G be a bipartite graph. Then,* $\operatorname{conv}(\mathrm{PM}(G)) = P$, *where P is the polytope described by the following linear system:*

$$x(\delta(v)) = 1 \quad \textit{for all } v \in V \tag{4.4a}$$

$$x_e \geq 0 \quad \textit{for all } e \in E. \tag{4.4b}$$

In particular, P is integral.

Proof: Clearly $\operatorname{conv}(\mathrm{PM}(G)) \subseteq P$. To show that $\operatorname{conv}(\mathrm{PM}(G)) = P$ let x be any extreme point of P. Assume for the sake of a contradiction that x is fractional. Define $\tilde{E} := \{e \in E : 0 < x_e < 1\}$ to be set of "fractional edges". Since $x(\delta(v)) = 1$ for any $v \in V$, we can conclude that any vertex that has an edge from $\tilde{E}$ incident with it, in fact is incident to at least two such edges from $\tilde{E}$. Thus, $\tilde{E}$ contains an even cycle C (by Lemma 4.6 the graph G does not contain any odd cycle). Let y be a vector which is alternatingly ± 1 for the edges in C and zero for all other edges. For small $\varepsilon > 0$ we have $x \pm \varepsilon y \in P$. But then, x can not be an extreme point. $\square$

Observe that we can view the assignment problem (see Example 1.6 on page 5) as the problem of finding a minimum cost perfect matching on a complete bipartite graph. Thus, Birkhoff's theorem shows that we can solve the assignment problem by solving a Linear Program.

Remark 4.8 The concept of total unimodularity derived in the next section will enable us to give an alternative proof of Birkhoff's Theorem.

4.3 Total Unimodularity

Proving that a given polyhedron is integral is usually a difficult task. In this section we derive some conditions under which the polyhedron

$$P^{=}(A, b) = \{x : Ax = b, x \geq 0\}$$

is integral for every integral right hand side b.

As a motivation for the following definition of total unimodularity, consider the Linear Program

$$(\mathrm{LP}) \max \left\{ c^T x : Ax = b, x \geq 0 \right\}, \tag{4.5}$$

where $\operatorname{rank} A = m$. From Linear Programming theory, we know that if (4.5) has a feasible (optimal) solution, it also has a feasible (optimal) basic solution, that is, a solution of the form $x = (x_B, x_N)$, where $x_B = A_{\cdot,B}^{-1} b$ and $x_N = 0$ and $A_{\cdot,B}$ is an $m \times m$ nonsingular submatrix of A indexed by the columns in $B \subseteq \{1, \dots, n\}$, $|B| = m$. Here, $N = \{1, \dots, n\} \setminus B$.

Given such a basic solution $x = (x_B, x_N)$ we have by Cramer's rule:

$$x_i = \frac{\det(B_i)}{\det(A_B)} \quad \text{for } i \in B,$$

where B_i is the matrix obtained from A_B by replacing the ith column by the vector b. Hence, we conclude that if $\det(A_B) = \pm 1$, then each entry of x_B will be integral (provided b is integral as well).

Definition 4.9 (Unimodular matrix, total unimodular matrix)
*Let A be an $m \times n$-matrix with full row rank. The matrix A is called **unimodular** if all entries of A are integral and each nonsingular $m \times m$-submatrix of A has determinant ± 1. The matrix A is called **totally unimodular**, if each square submatrix of A has determinant ± 1 or 0.*

Since every entry of a matrix forms itself a square submatrix, it follows that for a totally unimodular matrix A every entry must be either ± 1 or 0.

Observation 4.10 *(i) A is totally unimodular, if and only if A^T is totally unimodular.*

(ii) A is totally unimodular, if and only if (A, I) is unimodular.

(iii) A is totally unimodular, if and only if $\begin{pmatrix} A \\ -A \\ I \\ -I \end{pmatrix}$ *is totally unimodular.*

We now show that a Linear Program with a (totally) unimodular matrix has always an integral optimal solution provided the optimum is finite. Thus, by Theorem 4.3 we get that the corresponding polyhedron must be integral.

Theorem 4.11 *Let A be an $m \times n$ matrix with integer entries and linearly independent rows. The polyhedron $\{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ is integral for all $b \in \mathbb{Z}^m$ if and only if A is unimodular.*

Proof: Suppose that A is unimodular and $b \in \mathbb{Z}^m$ be an integral vector. By Corollary 4.5 it suffices to show that all extreme points of $\{x : Ax = b, x \geq 0\}$ are integral. Let $\bar{x}$ be such an extreme point. Since A has full row rank, there exists a basis $B \subseteq \{1, \ldots, n\}$, $|B| = m$ such that $\bar{x}_B = A_{\cdot,B}^{-1} b$ and $\bar{x}_N = 0$. Since A is unimodular, we have $\det(A_{\cdot,B}) = \pm 1$ and by Cramer's rule we can conclude that $\bar{x}$ is integral.

Assume conversely that $\{x : Ax = b, x \geq 0\}$ is integral for every integral vector b. Let B be a basis of A. We must show that $\det(A_{\cdot,B}) = \pm 1$. Let $\bar{x}$ be the extreme point corresponding to the basis B. By assumption $\bar{x}_B = A_{\cdot,B}^{-1} b$ is integral for all integral b. In particular we can choose b to be the unit vectors $e_i = (0, \ldots, 1, 0, \ldots, 0)$. Then, we get that $A_{\cdot,B}^{-1}$ must be integral. Thus, it follows that $\det(A_{\cdot,B}^{-1}) = 1/\det(A_{\cdot,B})$ is integral. On the other hand, also $\det(A_{\cdot,B})$ is also integral by the integrality of A. Hence, $\det(A_{\cdot,B}) = \pm 1$ as required. $\square$

We use the result of the previous theorem to show the corresponding result for the polyhedron $\{x : Ax \leq b, x \geq 0\}$.

Corollary 4.12 (Integrality-Theorem of Hoffmann and Kruskal) *Let A be an $m \times n$ matrix with integer entries. The matrix A is totally unimodular if and only if the polyhedron $\{x : Ax \le b, x \ge 0\}$ is integral for all $b \in \mathbb{Z}^m$.*

Proof: From Observation 4.10 we know that A is totally unimodular if and only if (A, I) is unimodular. Moreover, the polyhedron $\{x : Ax \le b, x \ge 0\}$ is integral if and only if the polyhedron $\{z : (A, I)z = b, z \ge 0\}$ is integral. The result now follows from Theorem 4.11. □

The Integrality-Theorem of Hoffmann and Kruskal in conjunction with Observation 4.10 yields more characterizations of totally unimodular matrices.

Corollary 4.13 *Let A be an integral matrix. Then the following statements hold:*

(a) A is totally unimodular, if and only if the polyhedron $\{x : a \le Ax \le b, l \le x \le u\}$ is integral for all integral a, b, l, u.

(b) A is totally unimodular, if and only if the polyhedron $\{x : Ax = b, 0 \le x \le u\}$ is integral for all integral b, u.

□

4.4 Conditions for Total Unimodularity

In this section we derive sufficient conditions for a matrix to be totally unimodular.

Theorem 4.14 *Let A be any $m \times n$ matrix with entries taken from $\{0, +1, -1\}$ with the property that any column contains at most two nonzero entries. Suppose also that there exists a partition $M_1 \cup M_2 = \{1, \dots, m\}$ of the rows of A such that every column j with two nonzero entries satisfies: $\sum_{i \in M_1} a_{ij} = \sum_{i \in M_2} a_{ij}$. Then, A is totally unimodular.*

Proof: Suppose for the sake of a contradiction that A is not totally unimodular. Let B be a smallest square submatrix such that $\det(B) \notin \{0, +1, -1\}$. Obviously, B can not contain any column with at most one nonzero entry, since otherwise B would not be smallest. Thus, any column of B contains exactly two nonzero entries. By the assumptions of the theorem, adding the rows in B that are in M_1 and subtracting those that are in M_2 gives the zero vector, thus $\det(B) = 0$, a contradiction! □

Example 4.15
Consider the LP-relaxation of the assignment problem.

$$\text{(4.6a)} \qquad \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{(4.6b)} \qquad \sum_{i=1}^{n} x_{ij} = 1 \qquad \text{for } j = 1, \dots, n$$

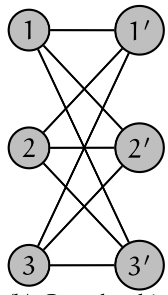
$$\text{(4.6c)} \qquad \sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for } i = 1, \dots, n$$

$$\text{(4.6d)} \qquad 0 \le x \le 1,$$

We can write the constraints (4.6b) and (4.6c) as $Ax = 1$, where A is the node-edge incidence matrix of the complete bipartite graph $G = (V, E)$ (Figure 4.1 shows the situation for $n = 3$). The rows of A correspond to the vertices and the columns to the edges. The row corresponding to edge (u, v) has exactly two ones, one at the row for u and one at the row for v. The fact that G is bipartite $V = A \cup B$, gives us a partition $A \cup B$ of the rows such that the conditions of Theorem 4.14 are satisfied. Hence, A is totally unimodular.

$$\left(\begin{array}{ccccccccc} 1 & 1 & 1 & & & & & & \\ & & & 1 & 1 & 1 & & & \\ & & & & & & 1 & 1 & 1 \\ \hline 1 & & & 1 & & & 1 & & \\ & 1 & & & 1 & & & 1 & \\ & & 1 & & & 1 & & & 1 \end{array}\right) \cdot \begin{pmatrix} x_{11'} \\ x_{12'} \\ x_{13'} \\ x_{21'} \\ x_{22'} \\ x_{23'} \\ x_{31'} \\ x_{32'} \\ x_{33'} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

(a) Constraint system



(b) Complete bipartite graph

Figure 4.1: The matrix of the assignment problem as the node-edge incidence matrix of a complete bipartite graph.

$\triangleleft$

We derive some other useful consequences of Theorem 4.14:

Theorem 4.16 *Let A be any $m \times n$ matrix with entries taken from $\{0, +1, -1\}$ with the property that any column contains at most one $+1$ and at most one -1. Then A is totally unimodular.*

Proof: First, assume that A contains *exactly* two nonzero entries per column. The fact that A is totally unimodular for this case follows from Theorem 4.14 with $M_1 = \{1, \dots, m\}$ and $M_2 = \emptyset$. For the general case, observe that a column with at most one nonzero from $\{-1, +1\}$ can not destroy unimodularity, since we can develop the determinant (of a square submatrix) by that column. $\square$

The node-arc incidence matrix of a directed network $G = (V, A)$ is the $n \times m$-Matrix $M(A) = (m_{xy})$ such that

$$m_{xy} = \begin{cases} +1 & \text{if } a = (i, j) \text{ and } x = j \\ -1 & \text{if } a = (i, j) \text{ and } y = i \\ 0 & \text{otherwise} \end{cases}$$

The minimum cost flow problem can be stated as the following Linear Program:

$$\min \sum_{(i,j) \in A} c(i, j) x(i, j) \tag{4.7a}$$

$$\sum_{j:(j,i) \in A} x(j, i) - \sum_{j:(i,j) \in A} x(i, j) = b(i) \qquad \text{for all } i \in V \tag{4.7b}$$

$$0 \le x(i, j) \le u(i, j) \qquad \text{for all } (i, j) \in A \tag{4.7c}$$

By using the node-arc incidence matrix $M = M(A)$, we can rewrite (4.7) as:

$$\min\left\{c^T x : Mx = b, 0 \leq x \leq u\right\}, \tag{4.8}$$

where b is the vector of all required demands.

Corollary 4.17 *The node-arc incidence matrix of a directed network is totally unimodular.*

Proof: The claim follows immediately from Theorem 4.16 and Corollary 4.13. □

We close this section by one more sufficient condition for total unimodularity.

Theorem 4.18 (Consecutive ones Theorem) *Let A be any $m \times n$-matrix with entries from $\{0, 1\}$ and the property that the rows of A can be permutated in such a way that all 1s appear consecutively. Then, A is totally unimodular.*

Proof: Let B be a square submatrix of A. Without loss of generality we can assume that the rows of A (and thus also of B) are already permuted in such a way that the ones appear consecutively. Let $b_1^T, \dots, b_k^T$ be the rows of B. Consider the matrix B' with rows $b_1^T - b_2^T, b_2^T - b_3^T, \dots, b_{k-1}^T - b_k^T, b_k$. The determinant of B' is the same as of B.

Any column of B' contains at most two nonzero entries, one of which is a -1 (one before the row where the ones in this column start) and a $+1$ (at the row where the ones in this column end). By Theorem 4.16, B' is totally unimodular, in particular $\det(B') = \det(B) \in \{0, +1, -1\}$. □

4.5 Applications of Unimodularity: Network Flows

We have seen above that if M is the node-arc incidence matrix of a directed graph, then the polyhedron $\{x : Mx = b, 0 \leq x \leq u\}$ is integral for all integral b and u. In particular, for integral b and u we have strong duality between the IP

$$\max\left\{c^T x : Mx = b, 0 \leq x \leq u, x \in \mathbb{Z}^n\right\} \tag{4.9}$$

and the dual of the LP-relaxation

$$\min\left\{b^T z + u^T y : M^T z + y \geq c, y \geq 0\right\}. \tag{4.10}$$

Moreover, if the vector c is integral, then by total unimodularity the LP (4.10) has always an integral optimal solution value.

4.5.1 The Max-Flow-Min-Cut-Theorem

As an application of the strong duality of the problems (4.9) and (4.10) we will establish the Max-Flow-Min-Cut-Theorem.

Definition 4.19 (Cut in a directed graph, forward and backward part) *Let $G = (V, A)$ be a directed graph and $S \cup T = V$ a partition of the node set V. We call (S, T) the **cut induced by** S **and** T. We also denote by*

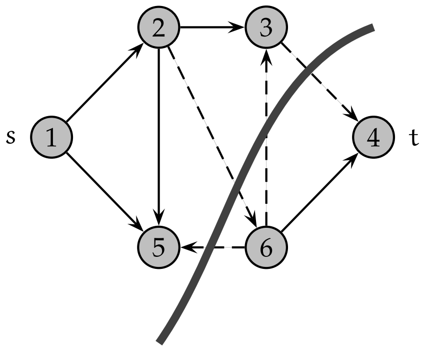
$$\delta^+(S := \{\, (i, j) \in A : i \in S \textit{ und } j \in T \,\}$$
$$\delta^-(S) := \{\, (j, i) \in A : j \in T \textit{ und } i \in S \,\}$$

*the **forward part** and the **backward part** of the cut. The cut* (S, T) *is an* (s, t)*-cut if* $s \in S$ *and* $t \in T$.

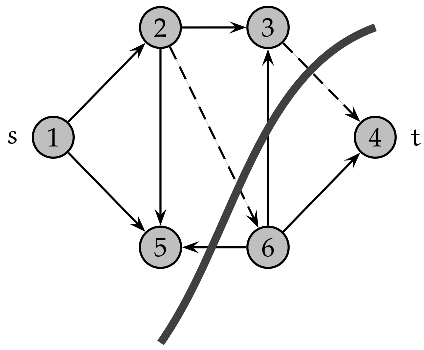
If $u \colon A \to \mathbb{R}_{0 \geq 0}$ *is a capacity function defined on the arcs of the network* $G = (V, A)$ *and* (S, T) *is a cut, then the **capacity of the cut** is defined to be the sum of the capacities of its forward part:*

$$u(\delta^+(S)) := \sum_{(u,v) \in (S,T)} u(u, v).$$

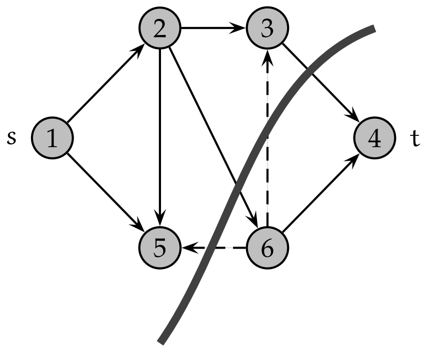
Figure 4.2 shows an example of a cut and its forward and backward part.



(a) An (s, t)-cut (S, T) in a directed graph. The arcs in $\delta^+(S) \cup \delta^-(S)$ are shown as dashed arcs.



(b) The forward part $\delta^+(S)$ of the cut: Arcs in $\delta^+(S)$ are shown as dashed arcs.



(c) The backward part $\delta^-(S)$ of the cut: arcs in $\delta^-(S)$ are shown as dashed arcs.

Figure 4.2: A cut (S, T) in a directed graph and its forward part $\delta^+(S)$ and backward part $\delta^-(S)$.

Let f be an (s, t)-flow and $[S, T]$ be an (s, t)-cut in G. For a node $i \in V$ we define by

(4.11) $$\text{excess}_f(i) := \sum_{a \in \delta^-(v)} f(a) - \sum_{a \in \delta^+(v)} f(a)$$

the *excess* of i with respect to f. The first term in (4.11) corresponds to the inflow into i, the second term is the outflow out of i. Then we have:

$$\begin{aligned} \mathrm{val}(f) &= -\mathrm{excess}_f(s) = -\sum_{i \in S} \mathrm{excess}_f(i) \\ &= \sum_{i \in S} \left(\sum_{(i,j) \in A} f(i,j) - \sum_{(j,i) \in A} f(j,i) \right). \end{aligned} \tag{4.12}$$

If for an arc (x, y) both nodes x and y are contained in S, then the term $f(x, y)$ appears twice in the sum (4.12), once with a positive and once with a negative sign. Hence, (4.12) reduces to

$$\mathrm{val}(f) = \sum_{a \in \delta^+(S)} f(a) - \sum_{(a) \in \delta^-(S)} f(a). \tag{4.13}$$

Using that f is feasible, that is, $0 \leq f(i,j) \leq u(i,j)$ for all arcs (i,j), we get from (4.13):

$$\mathrm{val}(f) = \sum_{(a) \in \delta^+(S)} f(a) - \sum_{a \in \delta^-(S)} f(a) \leq \sum_{a \in \delta^+(S)} u(a) = u(\delta^+(S)).$$

Thus, the value $\mathrm{val}(f)$ of the flow is bounded from above by the capacity $u(\delta^+(S)))$ of the cut. We have proved the following lemma:

Lemma 4.20 *Let f be an (s,t)-flow and $[S,T]$ an (s,t)-cut. Then:*

$$\mathit{val}(f) \leq u(\delta^+(S)).$$

Since f and $[S,T]$ are arbitrary we deduce that:

$$\max_{f \text{ is an } (s,t)\text{-flow in } G} \mathit{val}(f) \leq \min_{(S,T) \text{ is an } (s,t)\text{-cut in } G} u(\delta^+(S)). \tag{4.14}$$

□

We are now ready to prove the famous Max-Flow-Min-Cut-Theorem of Ford and Fulkerson:

Theorem 4.21 (Max-Flow-Min-Cut-Theorem) *Let $G = (V, A)$ be a network with capacities $u\colon A \to \mathbb{R}_+$, then the value of a maximum (s,t)-flow equals the minimum capacity of an (s,t)-cut.*

Proof: We add a backward arc (t, s) to G. Call the resulting graph $G' = (V, A')$, where $A' = A \cup \{(t, s)\}$. Then, we can write the maximum problem as the Linear Program

$$z = \max\{x_{ts} : Mx = 0, 0 \leq x \leq u\}, \tag{4.15}$$

where M is the node-arc incidence matrix of G' and $u(t,s) = +\infty$. We know that M is totally unimodular from Corollary 4.17. So, (4.15) has an optimal integral solution value for all integral capacities u. By Linear Programming duality we have:

$$\max\{x_{ts} : Mx = 0, 0 \leq x \leq u\} = \min\left\{u^T y : M^T z + y \geq \chi^{(t,s)}, y \geq 0\right\},$$

where $\chi^{(t,s)}$ is the vector in $\mathbb{R}^A$ which has a one at entry (t,s) and zero at all other entries. We unfold the dual which gives:

$$w = \min \sum_{(i,j)\in A} u_{ij} y_{ij} \tag{4.16a}$$

$$z_i - z_j + y_{ij} \geq 0 \qquad \text{for all } (i,j) \in A \tag{4.16b}$$

$$z_t - z_s \geq 1 \tag{4.16c}$$

$$y_{ij} \geq 0 \qquad \text{for all } (i,j) \in A \tag{4.16d}$$

There are various ways to see that (4.16) has an optimum solution which is also integral, for instance:

- The constraint matrix of (4.16) is of the form $(M^T I)$ and, from the total unimodularity of M it follows that $(M^T I)$ is also totally unimodular. In particular, (4.16) has an integral optimal solution for every integral right hand side (and our right hand side is integral!).
- The polyhedron of the LP (4.15) is integral by total unimodularity. Thus, it has an optimum integer value for all integral capacities (the objective is also integral). Hence, by LP-duality (4.16) has an optimum integral value for all integral objectives (which are the capacities). Hence, by Theorem 4.3 the polyhedron of (4.16) is integral and has an optimum integer solution.

Let (y^*, z^*) be such an integral optimal solution of (4.16). Observe that replacing z^* by $z^* - \alpha$ for some $\alpha \in \mathbb{R}$ does not change anything, so we may assume without loss of generality that $z_s^* = 0$.

Since (y^*, z^*) is integral, the sets S and T defined by

$$S := \{v \in V : z_v^* \leq 0\}$$
$$T := \{v \in V : z_v^* \geq 1\}$$

induce an (S,T)-cut. Then,

$$w = \sum_{(i,j)\in A} u_{ij} y_{ij}^* \geq \sum_{(i,j)\in\delta^+(S)} u_{ij} y_{ij}^* \geq \sum_{(i,j)\in\delta^+(S)} u_{ij} (\underbrace{z_j^*}_{\geq 1} - \underbrace{z_i^*}_{\leq 0}) \geq u(\delta^(S)).$$

Thus, the optimum value w of the dual (4.16) which by strong duality equals the maximum flow value is at least the capacity $u(\delta^+(S))$ of the cut (S,T). By Lemma 4.20 it now follows that (S,T) must be a minimum cut and the claim of the theorem is proved. □

4.6 Matchings and Integral Polyhedra II

Birkhoff's theorem provided a complete description of conv(PM(G)) in the case where the graph $G = (V,E)$ was bipartite. In general, the conditions in (4.4) do not suffice to ensure integrality of every extreme point of the corresponding polytope. Let FPM(G) (the *fractional matching polytope*) denote the polytope defined by (4.4). Consider the case where the graph G contains an odd cycle of length 3 (cf. Figure 4.3).

The vector $\tilde{x}$ with $\tilde{x}_{e_1} = \tilde{x}_{e_2} = \tilde{x}_{e_3} = \tilde{x}_{e_5} = \tilde{x}_{e_6} = \tilde{x}_{e_7} = 1/2$ and $\tilde{x}_{e_4} = 0$ is contained in FPM(G). However, $\tilde{x}$ is not a convex combination of incidence

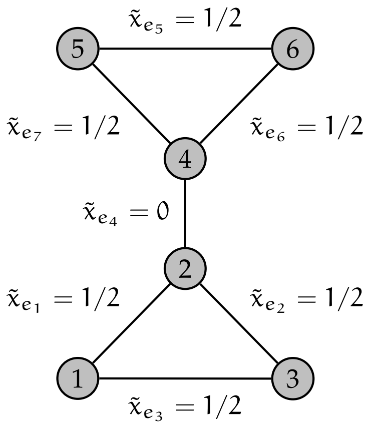


Figure 4.3: In an odd cycle, the blossom inequalities are necessary to ensure integrality of all extreme points.

vectors of perfect matchings of G, since $\{e_3, e_4, e_5\}$ is the only perfect matching in G. However, the fractional matching polytope FPM(G) still has an interesting structure, as the following theorem shows:

Theorem 4.22 (Fractional Matching Polytope Theorem) *Let* $G = (V, E)$ *be a graph and* $x \in \text{FPM}(G)$. *Then,* x *is an extreme point of* $\text{FPM}(G)$ *if and only if* $x_e \in \{0, 1/2, 1\}$ *for all* $e \in E$ *and the edges* e *for which* $x_e = 1/2$ *form node disjoint odd cycles.*

Proof: Suppose that $\tilde{x}$ is a half-integral solution satisfying the conditions stated in the theorem. Define the vector $w \in \mathbb{R}^n$ by $w_e = -1$ if $\tilde{x}_e = 0$ and $w_e = 0$ if $\tilde{x}_e > 0$. Consider the face $F = \left\{x \in \text{FPM}(G) : w^T x = 0\right\}$. Clearly, $\tilde{x} \in F$. We claim that $F = \{\tilde{x}\}$ which shows that $\tilde{x}$ is an extreme point.

For every $x \in F$ we have

$$0 = w^T x = -\sum_{e \in E: \tilde{x}_e = 0} \underbrace{x_e}_{\geq 0}.$$

Thus $x_e = 0$ for all edges such that $\tilde{x}_e = 0$. Now consider an edge e where $\tilde{x}_e = 1/2$. By assumption, this edge lies on an odd cycle C. It is now easy to see that the values of x on the cycle must be alternatingly θ and $1 - \theta$ since $x(\delta(v)) = 1$ for all $v \in V$ (see Figure 4.4). The only chance that $x \in \text{FPM}(G)$ is $\theta = 1/2$ and thus $x = \tilde{x}$.

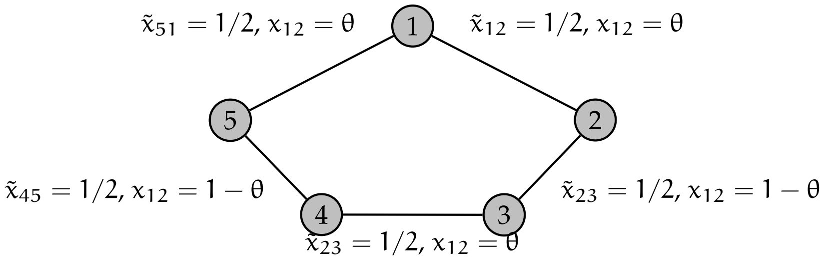
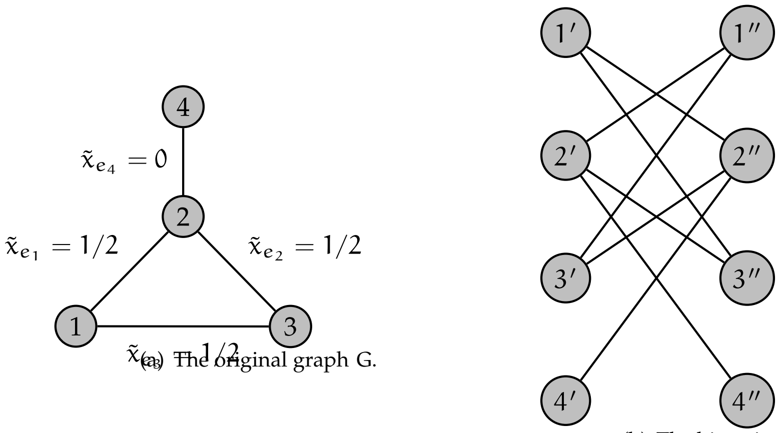


Figure 4.4: If $\tilde{x}$ satisfies the conditions of Theorem 4.22 it is the only member of the face $F = \left\{x \in \text{FPM}(G) : w^T x = 0\right\}$.

Assume conversely that $\tilde{x}$ is an extreme point of $\mathrm{FPM}(G)$. We first show that $\tilde{x}$ is half-integral. By Theorem 3.33 there is an integral vector c such that $\tilde{x}$ is the unique solution of $\max\left\{c^T x : x \in \mathrm{FPM}(G)\right\}$.

Construct a bipartite graph $H = (V_H, E_H)$ from G by replacing each node $v \in V$ by two nodes v', v'' and replacing each edge $e = (u, v)$ by two edges $e' = (u', v'')$ and $e'' = (v', u'')$ (see Figure 4.5 for an illustration). We extend the weight function $c\colon E \to \mathbb{R}$ to E_H by setting $c(u', v'') = c(v', u'') = c(u, v)$.



(a) The original graph G.

(b) The bipartite graph H.

Figure 4.5: Construction of the bipartite graph G in the proof of Theorem 4.22.

Observe that, if $x \in \mathrm{FPM}(G)$, then x' defined by $x'_{u',v''} := x'_{v',u''} := x_{uv}$ is a vector in $\mathrm{FPM}(H)$ of twice the objective function value of x. Conversely, if $x' \in \mathrm{FPM}(H)$, then $x_{uv} = \frac{1}{2}(x'_{u'v''} + x'_{u''v'})$ is a vector in $\mathrm{FPM}(G)$ of half of the objective function value of x'.

By Birkhoff's Theorem (Theorem 4.7 on page 50), the problem

$$\max\left\{c^T x_H : x_H \in \mathrm{FPM}(H)\right\}$$

has an integral optimal solution x^*_H. Using the correspondence $x_{uv} = \frac{1}{2}(x^*_{u'v''} + x^*_{u''v'})$ we obtain a half-integral optimal solution to

$$\max\left\{c^T x : x \in \mathrm{FPM}(G)\right\}.$$

Since $\tilde{x}$ was the unique optimal solution to this problem, it follows that $\tilde{x}$ must be half-integral.

If $\tilde{x}$ is half-integral, it follows that the edges $\{e : \tilde{x}_e = 1/2\}$ must form node disjoint cycles (every node that meets a half-integral edge, meets exactly two of them). As in the proof of Birkhoff's Theorem, none of these cycles can be even, since otherwise $\tilde{x}$ is no extreme point. □

With the help of the previous result, we can now prove the Perfect Matching Polytope Theorem, which we restate here for convenience.

Theorem 4.23 (Perfect Matching Polytope Theorem) *For any graph* $G = (V, E)$, *the convex hull* $\mathrm{conv}(\mathrm{PM}(G))$ *of the perfect matchings in* G *is identical to the set of solutions of the following linear system:*

(4.17a)	$x(\delta(v)) = 1$	*for all* $v \in V$
(4.17b)	$x(\delta(S)) \geq 1$	*for all* $S \subseteq V$, $\lvert S\rvert \geq 3$ *odd*
(4.17c)	$x_e \geq 0$	*for all* $e \in E$.

The inequalities (4.17b) are called blossom inequalities*.*

Proof: We show the claim by induction on the number $|V|$ of vertices of the graph $G = (V, E)$. If $|V| = 2$, then the claim is trivial. So, assume that $|V| > 2$ and the claim holds for all graphs with fewer vertices.

Let P be the polyhedron defined by the inequalities (4.17) and let $x' \in P$ be any extreme point of P. Since $\operatorname{conv}(PM(G)) \subseteq P$, the claim of the theorem follows if we can show that $x' \in PM(G)$. Since $\{x'\}$ is a minimal face of $\operatorname{conv}(PM(G))$, by Theorem 3.6 there exist a subset $E' \subseteq E$ of the edges and a family $\mathcal{S}'$ of odd subsets $S \subseteq V$ such that x' is the unique solution to:

(4.18a)	$x(\delta(v)) = 1$	for all $v \in V$
(4.18b)	$x(\delta(S)) = 1$	for all $S \in \mathcal{S}'$
(4.18c)	$x_e = 0$	for all $e \in E'$.

Case 1: $\mathcal{S}' = \emptyset$.
In this case, x' is a vertex of $FPM(G)$. By Theorem 4.22, x' is half-integral and the fractional edges form node-disjoint odd cycles. On the other hand, x' satisfies the blossom inequalities (4.17b) which is a contradiction.

Case 2: $\mathcal{S}' \neq \emptyset$.
Fix $S \in \mathcal{S}'$, by definition we have $x'(\delta(S)) = 1$. Notice that $|S|$ is odd. The complement $\bar{S} := V \setminus S$ need not be of odd cardinality, but observe that, if $\bar{S}$ is of odd cardinality, then G does not contain a perfect matching (since the total number of vertices is odd in this case). Let G^S and $G^{\bar{S}}$ be the graphs obtained from G by shrinking S and $\bar{S} = V \setminus S$ to a single node (see Figure 4.6). Let x^S and $x^{\bar{S}}$ be the restriction of x' to the edges of G^S and $G^{\bar{S}}$, respectively. By construction, $x^i(\delta(S)) = x^i(\delta(\bar{S}) = 1$ for $i = S, \bar{S}$.

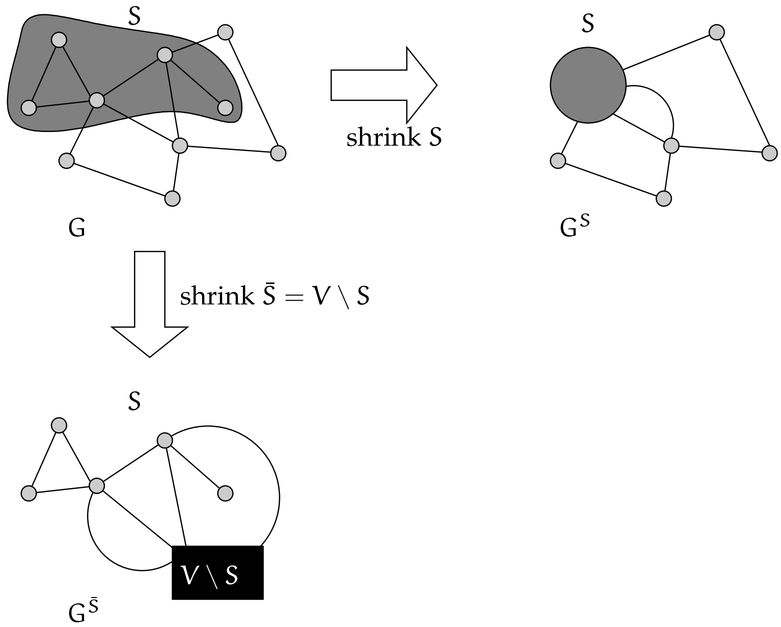


Figure 4.6: Graphs G^S and $G^{\bar{S}}$ obtained from G by shrinking the odd set S and $V \setminus S$ in the proof of Theorem 4.23.

It is easy to see that x^S and $x^{\bar{S}}$ satisfy the constraints (4.17) with respect to G^S and $G^{\bar{S}}$, respectively. Thus, by the induction hypothesis, we have

$x^i \in \text{conv}(PM(G^i))$ for $i = S, \bar{S}$. Hence, we can write x^i as convex combinations of perfect matchings of G^i:

$$x^S = \frac{1}{k} \sum_{j=1}^{k} \chi^{M_j^S} \tag{4.19}$$

$$x^{\bar{S}} = \frac{1}{k} \sum_{j=1}^{k} \chi^{M_j^{\bar{S}}} \tag{4.20}$$

Here, we have assumed without loss of generality that in both convex combinations the number of vectors used is the same, namely k. Also, we have assumed a special form of the convex combination which can be justified as follows: x' is an extreme point of P and thus is rational. This implies that x^i, $i = S, \bar{S}$ are also rational. Since all λ_j are rational, any convex combination $\sum_j \lambda_j y^j$ can be written by using common denominator k as $\sum_j \frac{\mu_j}{k} y^j$, where all μ_j are integral. Repeating vector y^j exactly μ_j times, we get the form $\frac{1}{k}\sum_j z^j$.

For $e \in \delta(S)$ the number of j such that $e \in M_j^S$ is $kx_e^S = kx'_e = kx_e^{\bar{S}}$. This is the same number of j such that $e \in M_j^{\bar{S}}$. Again: for every $e \in \delta(S)$ the number of j such that $e \in M_j^S$ is the same as the number of j with $e \in M_j^{\bar{S}}$. Thus, we can order the M_j^i so that M_j^S and $M_j^{\bar{S}}$ share an edge in $\delta(S)$ (any M_j^i, $i = S, \bar{S}$ has exactly one edge from $\delta(S)$). Then, $M_j := M_j^S \cup M_j^{\bar{S}}$ is a perfect matching of G since every vertex in G is matched and no vertex has more than one edge incident with it.

Let $M_j := M_j^S \cup M_j^{\bar{S}}$. Then we have:

$$x' = \frac{1}{k} \sum_{i=1}^{k} \chi^{M_j}. \tag{4.21}$$

Since M_j is a perfect matching of G we see from (4.21) that x' is a convex combination of perfect matchings of G. Since x' is an extreme point, it follows that x' must be a perfect matching itself. $\square$

4.7 Total Dual Integrality

Another concept for proving integrality of a polyhedron is that of *total dual integrality*.

Definition 4.24 (Totally dual integral system)
A rational linear system $Ax \le b$ is ***totally dual integral (TDI)****, if for each integral vector c such that*

$$z^{LP} = \max\left\{c^T x : Ax \le b\right\}$$

is finite, the dual

$$\min\left\{b^T y : A^T y = c, y \ge 0\right\}$$

has an integral optimal solution.

Theorem 4.25 *If $Ax \le b$ is TDI and b is integral, then the polyhedron $P = \{x : Ax \le b\}$ is integral.*

Proof: If $Ax \leq b$ is TDI and b is integral, then the dual Linear Program

$$\min \left\{ b^T y : A^T y = c, y \geq 0 \right\}$$

has an optimal integral objective value if it is finite (the optimal vector y^* is integral and b is integral by assumption, so $b^T y^*$ is also integral). By LP-duality, we see that the value

$$z^{LP} = \max \left\{ c^T x : Ax \leq b \right\}$$

is integral for all $c \in \mathbb{Z}^n$ where the value is finite. By Theorem 4.3(iv), the polyhedron P is integral. $\square$

Example 4.26
Let $G = (A \cup B, E)$ be a complete bipartite graph. Suppose we wish to solve a generalized form of STABLESET on G in which we are allowed to pick a vertex more than once. Given weights c_{ab} for the edges (a, b) we want to solve the following Linear Program:

$$\max \sum_{v \in V} w_v x_v \tag{4.22a}$$

$$x_a + x_b \leq c_{ab} \qquad \text{for all } (a, b) \in E \tag{4.22b}$$

We claim that the system of ineqalities $x_a + x_b \leq c_{ab}$ $(a, b) \in A$ is TDI. The dual of (4.22) is given by:

$$\min \sum_{(a,b) \in E} c_{ab} y_{ab} \tag{4.23a}$$

$$\sum_{b \in B} x_{ab} = w_a \qquad \text{for all } a \in A \tag{4.23b}$$

$$\sum_{a \in a} x_{ab} = w_b \qquad \text{for all } b \in B \tag{4.23c}$$

$$y_{ab} \geq 0 \qquad \text{for all } (a, b) \in A. \tag{4.23d}$$

The constraint matrix of (4.23) is the constraint matrix of the assignment problem, which we have already shown to be totally unimodular (see Example 4.15). Thus, if the weight vector w is integral, then the dual (4.23) has an optimal integral solution (if it is feasible). $\lhd$

It should be noted that the condition "and b is integral" in the previous theorem is crucial. It can be shown that for any rational system $Ax \leq b$ there is an integer ω such that $(1/\omega)Ax \leq (1/\omega)b$ is TDI. Hence, the fact that a system is TDI does not yet tell us anything useful about the structure of the corresponding polyhedron.

We have seen that if we find a TDI system with integral right hand side, then the corresponding polyhedron is integral. The next theorem shows that the converse is also true: if our polyhedron is integral, then we can also find a TDI system with integral right hand side defining it.

Theorem 4.27 *Let P be a rational polyhedron. Then, there exists a TDI system $Ax \leq b$ with A integral such that $P = P(A, b)$. Moreover, if P is an integral polyhedron, then b can be chosen to be integral.*

Proof: Let $P = \{x \in \mathbb{R}^n : Mx \leq d\}$ be a rational polyhedron. If $P = \emptyset$, then the claim is trivial. Thus, we assume from now on that P is nonempty. Since we can scale the rows of M by multiplying with arbitrary scalars, we can assume without loss of generality that M has integer entries. We also assume that the system $Mx \leq d$ does not contain any redundant rows. Thus, for any row m_i^T of M there is an $x \in P$ with $m_i^T x = d_i$.

Let $S = \left\{s \in \mathbb{Z}^n : s = M^T y, 0 \leq y \leq 1\right\}$ be the set of integral vectors which can be written as nonnegative linear combinations of the rows of M where no coefficient is larger than one. Since y comes from a bounded domain and S contains only integral points, it follows that S is finite. For $s \in S$ we define

$$z(s) := \max\left\{s^T x : x \in P\right\}.$$

Observe that, if $s \in S$, say $s = \sum_{i=1}^m y_i m_i$, and $x \in P$, then $m_i^T x \leq d_i$ which means $y_i m_i^T x \leq y_i d_i$ for $i = 1, \ldots, m$, from which we get that

$$s^T x = \sum_{i=1}^m y_i m_i^T x \leq \sum_{i=1}^m y_i d_i \leq \sum_{i=1}^m |d_i|.$$

Thus, $s^T x$ is bounded on P and $z(s) < +\infty$ for all $s \in S$. Moreover, the inequality $s^T x \leq z(s)$ is valid for P. We define the system $Ax \leq b$ to consist of all inequalities $s^T x \leq z(s)$ with $s \in S$.

Every row m_i^T of M is a vector in S (by assumption M is integral and m_i^T is a degenerated linear combination of the rows, namely with coefficient one for itself and zero for all other rows). Since $m_i^T x \leq d_i$ for all $x \in P$, the inequality $m_i^T x \leq d_i$ is contained in $Ax \leq b$. Furthermore, since we have only added valid inequalities to the system, it follows that

$$P = \{x : Ax \leq b\}. \tag{4.24}$$

If P is integral, then by Theorem 4.3 the value $z(s)$ is integral for each $s \in S$, so the system $Ax \leq b$ has an integral right hand side. The only thing that remains to show is that $Ax \leq b$ is TDI.

Let c be an integral vector such that $z^{\text{LP}} = \max\left\{c^T x : Ax \leq b\right\}$ is finite. We have to construct an optimal integral solution to the dual

$$\min\left\{b^T y : A^T y = c, y \geq 0\right\}. \tag{4.25}$$

We have

$$\begin{aligned}
z^{\text{LP}} &= \max\left\{c^T x : Ax \leq b\right\} \\
&= \max\left\{c^T x : x \in P\right\} && \text{(by (4.24))} \\
&= \max\left\{c^T x : Mx \leq d\right\} \\
&= \min\left\{d^T y : M^T y = c, y \geq 0\right\} && \text{(by LP-duality).} \qquad (4.26)
\end{aligned}$$

Let y^* be an optimal solution for the problem in (4.26) and consider the vector $\bar{s} = M^T(y^* - \lfloor y^* \rfloor)$. Observe that $y - \lfloor y^* \rfloor$ has entries in $[0, 1]$. Morever $\bar{s}$ is integral, since $\bar{s} = M^T y^* - M^T \lfloor y^* \rfloor = c - M^T \lfloor y^* \rfloor$ and c, M^T and $\lfloor y^* \rfloor$ are all integral. Thus, $\bar{s} \in S$. Now,

$$\begin{aligned}
z(\bar{s}) &= \max\left\{\bar{s}^T x : x \in P\right\} \\
&= \min\left\{d^T y : M^T y = \bar{s}, y \geq 0\right\} && \text{(by LP-duality).} \qquad (4.27)
\end{aligned}$$

The vector $y^* - \lfloor y^* \rfloor$ is feasible for (4.27) by construction. If v is feasible for (4.27), then $v + \lfloor y^* \rfloor$ is feasible for (4.26). Thus, it follows easily that $y - \lfloor y^* \rfloor$ is optimal for (4.27). Thus, $z(\bar{s}) = d^T(y^* - \lfloor y^* \rfloor)$, or

$$z^{LP} = d^T y^* = z(\bar{s}) + d^T \lfloor y^* \rfloor. \tag{4.28}$$

Consider the integral vector $\bar{y}$ defined as $\lfloor y^* \rfloor$ for the dual variables corresponding to rows in M, one for the dual variable corresponding to the constraint $\bar{s}^T x \leq z(\bar{s})$ and zero everywhere else. Clearly, $\bar{y} \geq 0$. Moreover,

$$A^T \bar{y} = \sum_{s \in S} \bar{y}_s s = M^T \lfloor y^* \rfloor + 1 \cdot \bar{s} = M^T \lfloor y^* \rfloor + M^T(y^* - \lfloor y^* \rfloor) = M^T y^* = c.$$

Hence, $\bar{y}$ is feasible for (4.25). Furthermore,

$$b^T \bar{y} = z(\bar{s}) + d^T \lfloor y^* \rfloor \stackrel{(4.28)}{=} z^{LP}.$$

Thus, $\bar{y}$ is an optimal integral solution for the dual (4.25). □

4.8 Submodularity and Matroids

In this section we apply our results about TDI systems to prove integrality for a class of important polyhedra.

Definition 4.28 (Submodular function)
*Let N be a finite set. A function $f\colon 2^N \to \mathbb{R}$ is called **submodular**, if*

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \text{ for all } A, B \subseteq N. \tag{4.29}$$

*The function is called **nondecreasing** if*

$$f(A) \leq f(B) \text{ for all } A, B \subseteq N \text{ with } A \subseteq B. \tag{4.30}$$

Usually we will not be given f "explicitly", that is, by a listing of all the $2^{|N|}$ pairs $(N, f(N))$. Rather, we will have access to f via an "oracle", that is, given N we can compute $f(N)$ by a call to the oracle.

Example 4.29

(i) The function $f(A) = |A|$ is nondecreasing and submodular.

(ii) Let $G = (V, E)$ be an undirected graph with edge weights $u\colon E \to \mathbb{R}_+$. The function $f\colon 2^V \to \mathbb{R}_+$ defined by $f(A) := \sum_{e \in \delta(A)} u(e)$ is submodular but not necessarily nondecreasing.

◁

Definition 4.30 (Submodular polyhedron, submodular optimization problem)
*Let f be submodular and nondecreasing. The **submodular polyhedron** associated with f is*

$$P(f) := \left\{ x \in \mathbb{R}^n_+ : \sum_{j \in S} x_j \leq f(S) \text{ for all } S \subseteq N. \right\} \tag{4.31}$$

*The **submodular optimization problem** is to optimize a linear objective function over $P(f)$:*

$$\max \left\{ c^T x : x \in P(f) \right\}.$$

Observe that by the polynomial time equivalence of optimization and separation (see Section 5.4) we can solve the submodular optimization problem in polynomial time if we can solve the corresponding separation problem in polynomial time: We index the polyhedra by the finite sets N, and it is easy to see that this class is proper.

We now consider the simple Greedy algorithm for the submodular optimization problem described in Algorithm 4.1. The surprising result proved in the following theorem is that the Greedy algorithm in fact solves the submodular optimization problem. But the result is even stronger:

Algorithm 4.1 Greedy algorithm for the submodular optimization problem.

GREEDY-SUBMODULAR

1. Sort the variables such that $c_1 \geq c_2 \geq \cdots \geq c_k > 0 \geq c_{k+1} \geq \cdots \geq c_n$.
2. Set $x_i := f(S^i) - f(S^{i-1})$ for $i = 1, \ldots, k$ and $x_i = 0$ for $i = k+1, \ldots, n$, where $S^i = \{1, \ldots, i\}$ and $S^0 = \emptyset$.

Theorem 4.31 *Let f be a submodular and nondecreasing function, $c\colon N \to \mathbb{R}$ be an arbitrary weight vector.*

(i) The Greedy algorithm solves the submodular optimization problem for maximizing $c^T x$ over $P(f)$.

(ii) The system (4.31) is TDI.

(iii) For integral valued f, the polyhedron $P(f)$ is integral.

Proof:

(i) Since f is nondecreasing, we have $x_i = f(S^i) - f(S^{i-1}) \geq 0$ for $i = 1, \ldots, k$. Let $S \subseteq N$. We have to show that $\sum_{j \in S} x_j \leq f(S)$. By the submodularity of f we have for $j \in S$:

$$f(\overbrace{S^j \cap S}^{=A}) + f(\overbrace{S^{j-1}}^{=B}) \geq f(\overbrace{\ S^j\ }^{=A\cup B}) + f(\overbrace{S^{j-1} \cap S}^{=A\cap B})$$

$$\Leftrightarrow f(S^j) - f(S^{j-1}) \leq f(S^j \cap S) - f(S^{j-1} \cap S) \tag{4.32}$$

Thus,

$$\begin{aligned}
\sum_{j \in S} x_j &= \sum_{j \in S \cap S^k} \left(f(S^j) - f(S^{j-1})\right) \\
&\leq \sum_{j \in S \cap S^k} \left(f(S^j \cap S) - f(S^{j-1} \cap S)\right) && \text{(by (4.32))} \\
&\leq \sum_{j \in S^k} \left(f(S^j \cap S) - f(S^{j-1} \cap S)\right) && \text{(f nondecreasing)} \\
&= f(S^k \cap S) - f(\emptyset) \\
&\leq f(S).
\end{aligned}$$

Thus, the vector x computed by the Greedy algorithm is in fact contained in $P(f)$. Its solution value is

$$c^T x = \sum_{i=1}^{k} c_i \left(f(S^i) - f(S^{i-1})\right). \tag{4.33}$$

We now consider the Linear Programming dual of the submodular optimization problem:

$$
\begin{aligned}
w^D = \min \quad & \sum_{S \subseteq N} f(S) y_S \\
& \sum_{S: j \in S} y_S \geq c_j && \text{for all } j \in N \\
& y_S \geq 0 && \text{for all } S \subseteq N.
\end{aligned}
$$

If we can show that $c^T x = w^D$, then it follows that x is optimal. Construct a vector y by $y_{S^i} = c_i - c_{i+1}$ for $i = 1, \dots, k-1$, $y_{S^k} = c_k$ and $y_S = 0$ for all other sets $S \subseteq N$. Since we have sorted the sets such that $c_1 \geq c_2 \geq c_k > 0 \geq c_{k+1} \geq \cdots \geq c_n$, it follows that y has only nonnegative entries.

For $j = 1, \dots, k$ we have

$$\sum_{S: j \in S} y_S \geq \sum_{i=j}^{k} y_{S^i} = \sum_{i=j}^{k-1} (c_i - c_{i+1}) + c_k = c_j.$$

On the other hand, for $j = k+1, \dots, n$

$$\sum_{S: j \in S} y_S \geq 0 \geq c_j.$$

Hence, y is feasible for the dual. The objective function value for y is:

$$
\begin{aligned}
\sum_{S \subseteq N} f(S) y_S = \sum_{i=1}^{k} f(S^i) y_{S^i} &= \sum_{i=1}^{k-1} f(S^i)(c_i - c_{i+1}) + f(S^k) c_k \\
&= \sum_{i=1}^{k} (f(S^i) - f(S^{i-1}) c_i \\
&= c^T x,
\end{aligned}
$$

where the last equality stems from (4.33). Thus, y must be optimal for the dual and x optimal for the primal.

(ii) The proof of statement (ii) follows from the observation that, if c is integral, then the optimal vector y constructed is integral.

(iii) Follows from (ii) and Theorem 4.25.

□

An important class of submodular optimization problems are induced by special submodular functions, namely the rank functions of matroids.

Definition 4.32 (Independence system, matroid)
*Let N be a finite set and $\mathcal{I} \subseteq 2^N$. The pair $(N, \mathcal{I})$ is called an **independence system**, if $A \in \mathcal{I}$ and $B \subseteq A$ implies that $B \in \mathcal{I}$. The sets in $\mathcal{I}$ are called **independent sets**.*

*The independence system is a **matroid** if for each $A \in \mathcal{I}$ and $B \in \mathcal{I}$ with $|B| > |A|$ there exists $a \in B \setminus A$ with $A \cup \{a\} \in \mathcal{I}$.*

*Given a matroid $(N, \mathcal{I})$, its **rank function** $r\colon 2^N \to \mathbb{N}$ is defined by*

$$r(A) := \max\{|I| : I \subseteq A \text{ and } I \in \mathcal{I}\}.$$

Observe that $r(A) \leq |A|$ for any $A \subset N$ and $r(A) = |A|$ if any only if $A \in \mathcal{I}$. Thus, we could alternatively specify a matroid $(N, \mathcal{I})$ also by (N, r).

Lemma 4.33 *The rank function of a matroid is submodular and nondecreasing.*

Proof: The fact that r is nondecreasing is trivial. Let $A, B \subseteq N$. We must prove that

$$r(A) + r(B) \geq r(A \cup B) + r(A \cap B).$$

Let $X \subseteq A \cup B$ with $|X| = r(A \cup B)$ and $Y \subseteq A \cap B$ with $|Y| = r(A \cap B)$. Let $X' := Y$. Since X' is independent and X is independent, if $|X'| < |X|$ we can add an element from X to X' without loosing independence. Continuing this procedure, we find X' with $|X'| = |X|$ and $Y \subseteq X'$ by construction. Hence, we can assume that $Y \subseteq X$. Now,

$$\begin{aligned} r(A) + r(B) &\geq |X \cap A| + |X \cap B| \\ &= |X \cap (A \cap B)| + |X \cap (A \cup B)| \\ &\geq |Y| + |X| \\ &= r(A \cap B) + r(A \cup B). \end{aligned}$$

This shows the claim. □

Example 4.34 (Matric matroid)
Let A be an $m \times n$-matrix with columns $a_1, \dots, a_n$. Set $N := \{1, \dots, n\}$ and the family $\mathcal{I}$ by the condition that $S \in \mathcal{I}$ if and only if the vectors $\{a_i : i \in S\}$ are linearly independent. Then $(N, \mathcal{I})$ is an independendence system. By Steinitz' Theorem (basis exchange) from Linear algebra, we know that $(N, \mathcal{I})$ is in fact also a matroid. ◁

Example 4.35
Let $E = \{1, 3, 5, 9, 11\}$ and $\mathcal{F} := \left\{A \subseteq E : \sum_{e \in A} e \leq 20\right\}$. Then, $(E, \mathcal{F})$ is an independence system but not a matroid.

The fact that $(E, \mathcal{F})$ is an independence system follows from the property that, if $B \subseteq A \in \mathcal{F}$, then $\sum_{e \in B} e \leq \sum_{e \in A} e \leq 20$.

Now consider $B := \{9, 11\} \in \mathcal{F}$ and $A := \{1, 3, 5, 9\} \in \mathcal{F}$ where $|B| < |A|$. However, there is no element in $A \setminus B$ that can be added to B without losing independence. ◁

Definition 4.36 (Tree, forest)
*Let $G = (V, E)$ a graph. A **forest** in G is a subgraph (V, E') which does not contain a cycle. A **tree** is a forest which is connected (i.e., which contains a path between any two vertices).*

Example 4.37 (Graphic matroid)
Let $G = (V, E)$ be a graph. We consider the pair $(E, \mathcal{F})$, where

$$\mathcal{F} := \{T \subseteq E : \ (V, T) \text{ is a forest}\}.$$

Clearly, $(E, \mathcal{F})$ is an independence system, since by deleting edges from a forest, we obtain again a forest. We show that the system is also a matroid. If (V, T) is a forest, then it follow easily by induction on $|T|$ that (V, T) has exactly $|V| - |T|$ connected components. Let $A \in \mathcal{I}$ and $B \in \mathcal{I}$ with $|B| > |A|$. Let $C_1, \dots, C_k$ be the connected components of (V, A) where $k = |V| - |A|$. Since (V, B) has fewer connected components, there must be an edge $e \in B \setminus A$ whose endpoints are in different components of (V, A). Thus $A \cup \{e\}$ is also a forest. ◁

Lemma 4.38 *Let $G = (V, E)$ be a graph and $(N, \mathcal{I})$ be the associated graphic matroid. Then the rank function r is given by $r(A) = |V| - \text{comp}(V, A)$, where $\text{comp}(V, A)$ denotes the number of connected components of the graph (V, A).*

Proof: Let $A \subseteq E$. It is easy to see that in a matroid all maximal independent subsets of A have the same cardinality. Let $C_1, \dots, C_k$ be the connected components of (V, A). For $i = 1, \dots, k$ we can find a spanning tree of C_i. The union of these spanning trees is a forest with $\sum_{i=1}^{k}(|C_i| - 1) = |V| - k$ edges, all of which are in A. Thus, $r(A) \geq |V| - k$.

Assume now that $F \subseteq A$ is an independent set. Then, F can contain only edges of the components $C_1, \dots, C_k$. Since F is a forest, the restriction of F to any C_i is also a forest, which implies that $|F \cap C_i| \leq |C_i| - 1$. Thus, we have $|F| \leq |V| - k$ and hence $r(A) \leq |V| - k$. $\square$

Definition 4.39 (Matroid optimization problem)
Given a matroid $(N, \mathcal{I})$ and a weight function $c\colon N \to \mathbb{R}$, the ***matroid optimization problem*** *is to find a set $A \in \mathcal{I}$ maximizing $c(A) = \sum_{a \in A} c(a)$.*

By Lemma 4.33 the polyhedron

$$P(N, \mathcal{I}) := \left\{ x \in \mathbb{R}^n_+ : \sum_{j \in S} x_j \leq r(S) \text{ for all } S \subseteq N \right\}$$

is a submodular polyhedron. Moreover, by the integrality of the rank function and Theorem 4.31(iii) $P(N, \mathcal{I})$ is integral. Finally, since $r(\{j\}) \leq 1$ for all $j \in N$, it follows that $0 \leq x \leq 1$ for all $x \in P(N, \mathcal{I})$. Thus, in fact we have:

$$P(N, \mathcal{I}) = \text{conv}\left(\left\{ x \in \mathbb{B}^n : \sum_{j \in S} x_j \leq r(S) \text{ for all } S \subseteq N. \right\} \right) \tag{4.34}$$

With (4.34) it is easy to see that the matroid optimization problem reduces to the submodular optimization problem. By Theorem 4.31(i) the Greedy algorithm finds an optimal (integral) solution and thus solves the matroid optimization problem.

It is worthwhile to have a closer look at the Greedy algorithm in the special case of a submodular polyhedron induced by a matroid. Let again be $S^i = \{1, \dots, i\}$, $S^0 = \emptyset$. Since $r(S^i) - r(S^{i-1}) \in \{0, 1\}$ and the algorithm works as follows:

1. Sort the variables such that $c_1 \geq c_2 \geq \dots \geq c_k > 0 \geq c_{k+1} \geq \dots \geq c_n$.
2. Start with $S = \emptyset$.
3. For $i = 1, \dots, k$, if $S \cup \{i\} \in \mathcal{I}$, then set $S := S \cup \{i\}$.

Part II

Algorithms

5

Basics about Problems and Complexity

5.1 Encoding Schemes, Problems and Instances

We briefly review the basics from the theory of computation as far as they are relevant in our context. Details can be found for instance in the book of Garey and Johnson [GJ79].

Informally, a decision problem is a problem that can be answered by "yes" or "no". As an example consider the problem of asking whether a given graph $G = (V, E)$ has a Hamiltonian Tour. Such a problem is characterized by the inputs for which the answer is "yes". To make this statement more formal we need to define exactly what we mean by "input", in particular if we speak about complexity in a few moments.

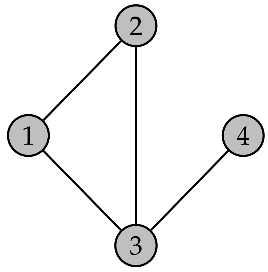


Figure 5.1: Example of an undirected graph.

Let Σ be a finite set of cardinality at least two. The set Σ is called the *alphabet*. By Σ^* we denote the set of all finite strings with letters from Σ. The *size* $|x|$ of a string $x \in \Sigma^*$ is defined to be the number of characters in it. For example, the undirected graph depicted in Figure 5.1 can be encoded as the following string over an appropriate alphabet which contains all the necessary letters:

$$(\{001, 010, 011, 100\}, \{(001, 010), (001, 011), (010, 011), (011, 100)\})$$

Here, we have encoded the vertices as binary numbers.

A *(decision) problem* is a subset $\Pi \subseteq \Sigma^*$. To *decide* Π means, given $x \in \Sigma^*$ to decide whether $x \in \Pi$ or not. The string x is called the *input* of the problem.

One speaks of an *instance* of the problem if one asks for a concrete input x whether x belongs to Π or not.

Example 5.1 (Stable Set Problem)
A *stable set* (or *independent set*) in an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ of the vertices such that none of the vertices in S are joined by an edge. The set of instances of the stable set problem (STABLESET) consists of all words (a, b) such that:

- a encodes an undirected graph $G = (V, E)$
- b encodes an integer k with $0 \leq k \leq |V|$
- G contains a stable set of size at least k.

Alternatively we could also say that an instance of STABLESET is given by an undirected graph $G(V, E)$ and an integer k and the problem is to decide wether G has a stable set of size at least k. $\lhd$

Classical complexity theory expresses the running time of an algorithm in terms of the size of the input, which is intended to measure the amount of data necessary to describe an instance of a problem. Naturally, there are many ways to encode a problem as words over an alphabet Σ. We assume the following standard encoding:

- Integers are encoded in binary. The standard binary representation of an integer n uses $\lceil \log_2 n \rceil + 1$ bits. We need an additional bit to encode the sign. Hence, the encoding length of an integer n is $\langle n \rangle = \lceil \log_2 n \rceil + 2$.
- Any rational number r has a unique representation $r = p/q$ where $p \in \mathbb{Z}$, $q \in \mathbb{N}$ and p and q do not have a common divisor other than 1. The encoding length of r is $\langle r \rangle := \langle p \rangle + \langle q \rangle$.
- The encoding length of a vector $x = (x_1, \ldots, x_n)^T \in \mathbb{Q}^n$ is $\sum_{i=1}^{n} \langle x_i \rangle$.
- The encoding length of a matrix $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$ is $\sum_{i,j} \langle a_{ij} \rangle$.
- Graphs are encoded by means of their adjacency matrices, incidence matrices or adjacency lists.

  The *adjacency matrix* of a graph $G = (V, E)$ with $n := |V|$ nodes is the $n \times n$ matrix $A(G) \in \mathbb{B}^{n \times n}$ with $a_{ij} = 1$ if $(v_i, v_j) \in E$ and $a_{ij} = 0$ if $(v_i, v_j) \notin E$. The *incidence matrix* of G is the $m \times n$ matrix $I(G)$ with rows corresponding to the vertices of G and columns representing the arcs/edges of G. The column corresponding to arc (v_i, v_j) has a -1 at row i and a $+1$ at row j. All other entries are zero. Since we have already specified the encoding lengths of matrices, the size of a graph encoding is now defined if one of the matrix represenations is used.

  The *adjacency list* representation consists of the number n of vertices and the number m of edges plus a vector Adj of n lists, one for each vertex. The list Adj[u] contains all $v \in V$ such that $(u, v) \in E$. Figure 5.2 shows an example of such a representation for a directed graph. The size of the adjacency list representation of a graph is $\langle n \rangle + \langle m \rangle + m + n$ for directed graphs and $\langle n \rangle + \langle m \rangle + 2m + n$ for undirected graphs.

  One might wonder why we have allowed different types of encodings for graphs. The answer is that for the question of polynomial time solvability it does in fact not matter which representation we use, since they are all polynomially related in size.

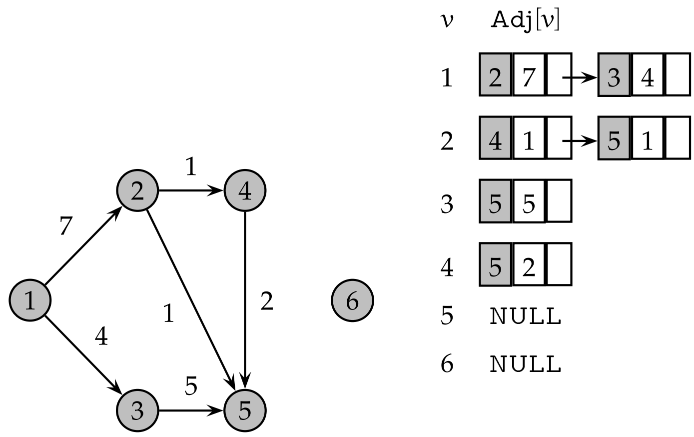


Figure 5.2: Adjacency list representation of a directed graph.

Example 5.2
An instance $(G = (V, E), k)$ of STABLESET has size $\langle n \rangle + \langle m \rangle + n + 2m + \langle k \rangle$ if we use the adjacency list representation. ◁

Example 5.3
Suppose we are given a MIP

$$
\begin{aligned}
\text{(MIP)} \quad \max \quad & c^T x \\
& Ax \le b \\
& x \ge 0 \\
& x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}.
\end{aligned}
$$

where all data A, b, c is integral. Then, the encoding size of an instance of the decision version of the MIP which asks whether there exists a feasible solution x with $c^T x \ge k$ is given by

$$\langle A \rangle + \langle b \rangle + \langle c \rangle + \langle k \rangle + n + p.$$

◁

The running time of an algorithm on a specific input is defined to be the sum of times taken by each instruction executed. The *worst case time complexity* or simply *time complexity* of an algorithm is the function $T(n)$ which is the maximum running time taken over all inputs of size n (cf. [AHU74, GJ79, GLS88]). An algorithm is a *polynomial time algorithm* if $T(n) \le p(n)$ for some polynomial p.

5.2 The Classes P and NP

Definition 5.4 (Complexity Class P)
The class P *consists of all problems which can be decided in polynomial time on a deterministic Turing machine.*

Example 5.5
The decision version of the maximum flow problem, that is, decide whether the following IP has a solution of value greater than a given flow value F,

$$\begin{aligned}
\max \sum_{(i,t)\in A} f(i,t) - \sum_{(t,j)\in A} f(t,j) & \\
\sum_{(j,i)\in A} f(j,i) - \sum_{(i,j)\in A} f(i,j) = 0 & \qquad \text{for all } i \in V \setminus \{s,t\} \\
0 \le f(i,j) \le u(i,j) & \qquad \text{for all } (i,j) \in A
\end{aligned}$$

is in P, since for instance the Edmonds-Karp-Algorithm solves it in time $\mathcal{O}(nm^2)$ which is polynomial in the input size.[1] $\triangleleft$

Definition 5.6 (Complexity Class NP)
The class NP *consists of all problems* $\Pi \subseteq \Sigma^*$ *such that there is a problem* $\Pi' \in \mathsf{P}$ *and a polynomial* p *such that for each* $x \in \Sigma^*$ *the following property holds:*

$$x \in \Pi \textit{ if and only if there exists } y \in \Sigma^* \textit{ with } |y| \le p(|x|) \textit{ and } (x,y) \in \Pi'.$$

The word y *is called a* ***certificate*** *for* x.

It is clear from the definition of NP and P that $\mathsf{P} \subseteq \mathsf{NP}$.

Example 5.7 (Stable Set Problem (continued))
STABLESET is in NP, since if a graph $G = (V, E)$ has a stable set S of size k we can simply use S as a certificate. Clearly it can be checked in polynomial time that S is in fact a stable set in G of size at least k. $\triangleleft$

To obtain a classification of “easy” and “difficult” problems we introduce the concept of a *polynomial time reduction*:

Definition 5.8 (Polynomial Time Reduction)
A ***polynomial time reduction*** *of a problem* Π *to a problem* Π' *is a polynomial time computable function* f *with the property that* $x \in \Pi$ *if and only if* $f(x) \in \Pi'$.

We say that Π *is* ***polynomial time reducible*** *to* Π' *if there exists a polynomial time reduction from* Π *to* Π. *In this case we write* $\Pi \propto \Pi'$.

Intuitively, if $\Pi \propto \Pi'$, then Π could be called “easier” than Π', since we can reduce the solvability of Π to that of Π'. In fact, we have the following important observation:

Observation 5.9 *If* $\Pi \propto \Pi'$ *and* $\Pi' \in \mathsf{P}$, *then* $\Pi \in \mathsf{P}$.

Definition 5.10 (Vertex Cover)
Let $G = (V, E)$ *be an undirected graph. A* ***vertex cover*** *in* G *is a subset* $C \subseteq V$ *of the vertices of* G *such that for each edge* $e \in M$ *at least one endpoint is in* C.

[1] There are actually faster algorithms such as the FIFO-Preflow-Push Algorithm of Goldberg and Tarjan which runs in time $\mathcal{O}(n^3)$. This time can even be reduced to $\mathcal{O}\left(nm\log\frac{n^2}{m}\right)$ by the use of sophisticated data structures.

Example 5.11
An instance of the *vertex cover problem (*VC*)* consists of a graph G and an integer k. The question posed is whether there exists a vertex cover of size at most k in G.

We claim that VC $\propto$ STABLESET. To see this observe that C is a vertex cover in G if and only if $V \setminus C$ is a stable set. This immediately leads to a polynomial time reduction. $\lhd$

Example 5.12
A *clique* C in a graph $G = (V, E)$ is a subset of the nodes such that every pair of nodes in C is connected by an edge. The *clique problem (*CLIQUE*)* asks whether a given graph G contains a clique of size at least a given number k.

Since C is a clique in G if and only if C is a stable set in the complement graph $\bar{G}$, a polynomial time reduction from CLIQUE to STABLESET is obtained by computing $\bar{G}$. $\lhd$

We are now ready to define "hard" problems:

Definition 5.13 (NP-complete problem)
A problem $\Pi' \in \mathsf{NP}$ is called NP***-complete*** *if $\Pi \propto \Pi'$ for every problem $\Pi \in \mathsf{NP}$.*

By the fact that the composition of polynomial time reductions is again a polynomial time reduction, we have the following useful observation:

Observation 5.14 *Suppose that Π is* NP*-complete. Let $\Pi' \in \mathsf{NP}$ with the property that $\Pi \propto \Pi'$. Then, Π' is also* NP*-complete.*

From Observation 5.9 we also have the following important fact:

Observation 5.15 *Suppose that Π is* NP*-complete and also $\Pi \in \mathsf{P}$. Then, $\mathsf{P} = \mathsf{NP}$.*

Most problems encountered in these lecture notes are optimization problems rather than decision problems. Clearly, to any minimization problem (maximization problem) we can associate a corresponding decision problem that asks whether there exists a feasible solution of cost at most (at least) a given value.

Definition 5.16 (NP-hard optimiziation problem)
An optimization problem whose corresponding decision problem is NP*-complete is called* NP***-hard****.*

Definition 5.13 raises the question whether there exist NP-complete problems. This question was settled in the affirmative in the seminal work by Cook [Coo71] and Karp [Kar72]. An instance of the *Satisfiability Problem (*SAT*)* is given by a finite number n of Boolean variables $X_1, \dots, X_n$ and a finite number m of *clauses*

$$C_j = L_{i_1} \vee L_{i_2} \vee \cdots \vee L_{i_j},$$

where $L_{i_l} \in \{X_{i_l}, \bar{X_{i_l}}\}$ is a *literal*, that is, a variable or its negation. Given an instance of SAT the question posed is whether there exists an assignment of truth values to the variables such that all clauses are satisfied.

Theorem 5.17 (Cook [Coo71]) SAT *is* NP*-complete.*

Karp [Kar72] showed a number of elementary graph problems to be NP-complete, among them STABLESET, CLIQUE, VC and the TSP. Since then the list of NP-complete problems has been growing enormously (see the book Garey and Johnson [GJ79] for an extensive list of NP-complete problems).

Theorem 5.18 (Karp [Kar72]) *The following problems are all* NP*-complete:*

- STABLESET
- CLIQUE
- VC
- TSP
- KNAPSACK

□

We have already remarked above that $\mathsf{P} \subseteq \mathsf{NP}$. By Observation 5.15 we would have $\mathsf{P} = \mathsf{NP}$ if we find a polynomial time algorithm for a single NP-complete problem. Despite great efforts to date no one has succeeded in doing so. It is widely believed that NP-completeness, or, more general, NP-hardness of a problem is a certificate of intractability.

5.3 The Complexity of Integer Programming

We now consider the complexity of Integer Linear Programming. To this end we first address the case of binary variables.

$$
\begin{aligned}
\text{(BIP)} \quad \max \quad & c^T x \\
& Ax \leq b \\
& x \geq 0 \\
& x \in \mathbb{B}^n
\end{aligned}
$$

Theorem 5.19 *Binary Integer Programming (BIP) is* NP*-hard.*

Proof: In order to show that the decision version of BIP (here also called BIP for simplicity) is NP-complete we have to prove two things:

1. BIP is in NP.
2. There is an NP-complete problem Π such that Π is polynomial time reducable to BIP.

The fact that BIP is contained in NP is easy to see. Suppose that the instance (A, b, c, k) has a feasible solution x^* with $c^T x \geq k$. The encoding size of $x^* \in \mathbb{B}^n$ is n (since x^* is a binary vector with n entries). Since we can check in polynomial time that in fact x^* is a feasible solution by checking all the contraints and the objective function value, this gives us a certificate that (A, b, c, k) is a "yes"-instance.

In order to prove the completeness of BIP, we reduce the satisfiability problem SAT to BIP. Suppose that we are given an instance $(X_1, \dots, X_n, C_1, \dots, C_m)$ of

SAT. For a clause C_j we denote by C_j^+ and C_j^- the index sets of positive and negative literals in C_j, respectively. We now formulate the following BIP:

$$
\begin{aligned}
&\max 0 \\
(5.1) \qquad &\sum_{i \in C_j^+} x_i + \sum_{i \in C_j^-} (1 - x_i) \geq 1 \\
&x_i \in \{0, 1\} \qquad\qquad \text{for } i = 1, \ldots, n
\end{aligned}
$$

It is easy to see that the BIP can be set up in polynomial time given the instance of SAT. Consider a clause C_j. If we have a truth assignment to the variables in C_j that satisfies C_j, then the corresponding setting of binary values to the variables in the BIP will satisfy the constraint (5.1) and vice versa. Hence it follows that the BIP has a feasible solution if and only if the given instance of SAT is satisfiable. □

Integer Linear Programming and Mixed Integer Programming are generalizations of Binary Integer Programming. Hence, Binary Integer Programming can be reduced to both problems in polynomial time. Can we conclude now that both IP and MIP are NP-complete? We can not, since we still have to show that in case of a "yes"-instance there exists a certificate of polynomial size. This was not a problem for the BIP since the certificate just involved n binary values, but for general IP and MIP we have to bound the entries of a solution vector x by a polynomial in $\langle A \rangle$, $\langle b \rangle$ and $\langle c \rangle$. This is somewhat technical and we refer the reader to [Sch86, NW99].

Theorem 5.20 *Solving MIP and IP is* NP*-hard.* □

Thus, solving *general* Integer Linear Programs is a hard job. Nevertheless, proving a problem to be hard does not make the problem disappear. Thus, in the remainder of these notes we seek to explore the structure of specific problems which sometimes makes it possible to solve them in polynomial time or at least much more efficiently than brute-force enumeration.

5.4 Optimization and Separation

We have seen a number of problems which had an exponential number of constraints (e.g. the formulation of the TSP in Example 1.8, the formulation of the minimum spanning tree problem in Example 1.10). Having such a large number of constraints in a Linear Programming problem does not defeat the existence of a polynomial time algorithm, at least not if we do not write down the LP explicitly. We will now make this statement precise.

To this end, we need a framework for describing linear systems that may be very large (relative to the size of the combinatorial problem that we want to solve). For instance, we do not want to list all the 2^n inequalities for the LP-relaxation of the MST-problem.

Definition 5.21 (Separation problem over an implicitly given polyhedron) *Given a bounded rational polyhedron $P \subset \mathbb{R}^n$ and a rational vector $v \in \mathbb{R}^n$, either conclude that $v \in P$ or, if not, find a rational vector $w \in \mathbb{R}^n$ such that $w^T x < w^T v$ for all $x \in P$.*

Definition 5.22 (Optimization problem over an implicitly given polyhedron)
Given a bounded rational polyhedron $P \subset \mathbb{R}^n$ *and a rational objective vector* c*, either find* $x^* \in P$ *maximizing* $c^T x$ *over* P *or conclude that* P *is empty.*

A famous theorem of Grötschel, Lovász and Schrijver [GLS88] says that the separation problem is polynomial time solvable if and only if the corresponding optimization problem is. To make this statement more precise, we need a bit of notation.

We consider classes of polyhedra $\mathcal{P} = \{P_o : o \in \mathcal{O}\}$, where $\mathcal{O}$ is some collection of objects. For instance, $\mathcal{O}$ can be the collection of all graphs, and for a fixed $o \in \mathcal{O}$, P_o is the convex hull of all incidence vectors of spanning trees of o. The class $\mathcal{P}$ of polyhedra is called *proper*, if for each $o \in \mathcal{O}$ we can compute in polynomial time (with respect to the size of o) positive integers n_o and s_o such that $P_o \subset \mathbb{R}^{n_o}$ and P_o can be described by a linear system of inequalities each of which has encoding size at most s_o.

Example 5.23
We consider the LP-relaxation of the IP-formulation for the MST-problem from Example 1.10, that is, the LP obtained by dropping the integrality constraints:

$$\text{(5.2a)} \qquad \min \sum_{e \in E} c_e x_e$$

$$\text{(5.2b)} \qquad \sum_{e \in \delta(S)} x_e \geq 1 \qquad \text{for all } \varnothing \subset S \subset V$$

$$\text{(5.2c)} \qquad 0 \leq x \leq 1$$

The class of polyhedra associated with (5.2) MST-problem is proper. Given the graph (object) $o = G = (V, E)$, the dimension where the polytope P_o lives in is $m = |E|$, the number of edges of G. Clearly, m can be computed in polynomial time. Moreover each of the inequalities (5.2b) has encoding size at most $m + 1$, since we need to specify at most m variables to sum up. Each of the inequalities (5.2c) has also size at most $m + 1$. $\lhd$

We say that the separation problem is *polynomial time solvable* for a proper class of polyhedra $\mathcal{P}$, if there exists an algorithm that solves the separation problem for each $P_o \in \mathcal{P}$ in time polynomial in the size of o and the given rational vector $v \in \mathbb{R}^{n_o}$. The optimiziation problem over $\mathcal{P}$ is *polynomial time solvable*, if there exists a polynomial time algorithm for solving any instance (P_o, c) of the optimization problem, where $o \in \mathcal{O}$ and c is a rational vector in $\mathbb{R}^{n_o}$.

The exact statement of the theorem of Grötschel, Lovász and Schrijver [GLS88] is as follows:

Theorem 5.24 *For any proper class of polyhedra, the optimization problem is polynomial time solvable if and only if the separation problem is polynomial time solvable.*

The proof is beyond the scope of these lecture notes, and we refer the reader to [GLS88, NW99] for details. We close this section with an example to give a flavor of the application of this result.

Example 5.25
Consider again the LP-relaxation of the IP-formulation for the MST-problem. We have already seen that the class of polytopes associate with this problem

is proper. We show that we can solve the separation problem in polynomial time. To this end, let $v \in \mathbb{R}^E$ be a vector. We have to check whether $x \in P_o$ and if not, find a violated inequality.

As a first step, we check the $2n$ constraints $0 \leq v_e \leq 1$ for all $e \in E$. Clearly, we can do this in polynomial time. If we find a violated inequality, we are already done. The more difficult part is to check the $2^n - 2$ constraints $\sum_{e \in \delta(S)} v_e \geq 1$.

Consider the graph $G = (V, E)$ with edge weights given by v, that is, edge e has weight v_e. Then, the constraints $\sum_{e \in \delta(S)} v_e \geq 1$ for $\emptyset \subset S \subset V$ say that with respect to this weighing, G must not have a cut of weight less than 1. Thus, we solve a minimum cut problem in G with edge weights v. If the minimum cut has weight at least 1, we know that v satisfies all constraints. In the other case, if S is one side of the minimum cut, which has weight less than 1, then $\sum_{e \in \delta(S)} v_e \geq 1$ is a violated inequality. Since we can solve the minimum cut problem in polynomial time, we can now solve the separation problem in polynomial time, too.

By the result in Theorem 5.24, it now follows that we can solve the optimization problem (5.2) in polynomial time. $\triangleleft$

6

Relaxations and Bounds

6.1 Optimality and Relaxations

Suppose that we are given an IP

$$z = \max\left\{c^T x : x \in X\right\},$$

where $X = \{x : Ax \leq b, x \in \mathbb{Z}^n\}$ and a vector $x^* \in X$ which is a candidate for optimality. Is there a way we can prove that x^* is optimal? In the realm of Linear Programming the Duality Theorem provides a nice characterization of optimality. Similarly, the concept of complementary slackness is useful for (continuous) Linear Programs.

Unfortunately, there a no such nice characterizations of optimality for Integer Linear Programs. However, there is a simple concept that sometimes suffices to prove optimality, namely the concept of bounding with the help of upper and lower bounds. If we are given a lower bound $\underline{z} \leq z$ and an upper bound $z \leq \bar{z}$ and we know that $\bar{z} - \underline{z} \leq \varepsilon$ for some $\varepsilon > 0$, then we have enclosed the optimal function value within an accuracy of ε. Moreover, if $\bar{z} = \underline{z}$, then we have found an optimal solution.

In particular, consider the case mentioned at the beginning of this section, we had a candidate x^* for optimality. Then, $\underline{z} = c^T x^*$ is a lower bound for the optimal solution value z. If we are able to find an upper bound $\bar{z}$ such that $c^T x^* = \bar{z}$, we have shown that x^* is an optimal solution.

In the case of maximization (minimization) problems lower (upper) bounds are usually called *primal bounds*, whereas upper (lower) bounds are referred to as *dual bounds*.

In the sequel we consider the case of a maximization problem

$$\max\left\{c^T x : x \in X \subseteq \mathbb{R}^n\right\},$$

the case of a minimization problem is analogous.

Primal bounds Every feasible solution $x \in X$ provides a lower bound $\underline{z} = c^T x$ for the optimal objective function value z. This is the only known way to establish primal bounds and the reason behind the name: every primal bound goes with a feasible solution for the (primal) problem.

Sometimes, it is hard to find a feasible solution. However, sometimes we can find a feasible solution (and thus a lower bound) almost trivially. For instance, in the traveling salesman problem (see Example 1.8), any permutation of the cities gives a feasible solution.

Dual bounds In order to find upper bounds one usually replaces the optimization problem by a simpler problem, whose optimization value is at least as large as z. For the "easier" problem one either optimizes over a larger set or one replaces the objective function by a version with larger value everywhere.

The most useful concept for proving dual bounds is that of a relaxation:

Definition 6.1 (Relaxation)
Consider the following two optimization problems:

$$\begin{aligned} &\textit{(IP)} \quad z = \max\left\{c^T x : x \in X \subseteq \mathbb{R}^n\right\} \\ &\textit{(RP)} \quad z^{RP} = \max\{f(x) : x \in T \subseteq \mathbb{R}^n\} \end{aligned}$$

The problem RP is called a ***relaxation*** *of IP if* $X \subseteq T$ *and* $f(x) \geq c^T x$ *for all* $x \in X$.

Clearly, if RP is a relaxation of IP, then $z^{RP} \geq z$. We collect this property and a few more easy but useful observations:

Observation 6.2 *Let RP be a relaxation of IP. Then, the following properties hold:*

(i) $z^{RP} \geq z$

(ii) If RP is infeasible, then IP is also infeasible.

(iii) Suppose that x^* *is a feasible solution to RP. If* $x^* \in X$ *and* $f(x^*) = c^T x^*$, *then* x^* *is also optimal for IP.*

One of the most useful relaxations of integer programs are the *linear programming relaxations*:

Definition 6.3 (Linear Programming relaxation)
The ***Linear Programming relaxation*** *of the IP* $\max\left\{c^T x : x \in P \cap \mathbb{Z}\right\}$ *with formulation* $P = \{x : Ax \leq b\}$ *is the Linear Program* $z^{LP} = \max\left\{c^T x : x \in P\right\}$.

Example 6.4
Consider the following slight modification of the IP from Example 1.2 (the only modification is in the objective function):

$$\begin{aligned} z = \max \quad & -2x_1 + 5x_2 \\ & 2x_2 - 3x_1 \leq 2 \\ & x_1 + x_2 \leq 5 \\ & 1 \leq x_1 \leq 3 \\ & 1 \leq x_2 \leq 3 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

A primal bound is obtained by observing that $(2, 3)$ is feasible for the IP which gives $\underline{z} = -2 \cdot 2 + 5 \cdot 3 = 11 \leq z$. We obtain a dual bound by solving the LP relaxation. This relaxation has an optimal solution $(4/3, 3)$ with value $z^{LP} = 37/3$. Hence, we have an upper bound $\bar{z} = 37/3 \geq z$. But we can improve this upper bound slightly. Observe that for integral x_1, x_2 the objective function value is also integral. Hence, if $z \leq 37/3$ we also have $z \leq \lfloor 37/3 \rfloor = 12$ (in fact we have $z = 11$). $\triangleleft$

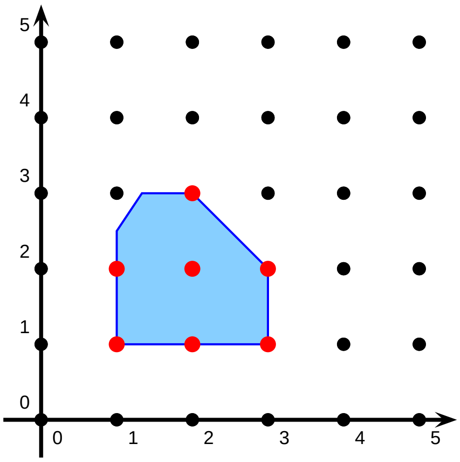


Figure 6.1: Feasible region for the LP-relaxation in Example 6.4 and feasible set of the IP.

Example 6.5 (Knapsack Problem)
Consider the Linear Programming relaxation of the Knapsack Problem (KNAPSACK) from Example 1.3.

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} c_i x_i \\ & \sum_{i=1}^{n} a_i x_i \leq b \\ & 0 \leq x_i \leq 1 \quad \text{for } i = 1, \dots, n. \end{aligned}$$

In this relaxation we are not required to either pack an item i or not pack it, but we are allowed to pack an arbitrary fraction $0 \leq x_i \leq 1$ of the item into the knapsack. Although we could solve the LP-relaxation by standard Linear Programming methods, this is in fact overkill! Suppose that we sort the items into nonincreasing order according to their "bang-for-the-buck-ratio" c_i/a_i. Without loss of generality we assume now that $c_1/a_1 \geq c_2/a_2 \geq \cdots \geq c_n/a_n$. Let i_0 be the largest index such that $\sum_{i=1}^{i_0} a_i \leq b$. We pack all items $1, \dots, i_0$ completely ($x_i = 1$ for $i = 1, \dots, i_0$) and fill the remaining empty space $s = b - \sum_{i=1}^{i_0} a_i$ with s/a_{i_0+1} units of item $i_0 + 1$ ($x_{i_0+1} = s/a_{i_0+1}$). All other items stay completely out of the knapsack. It is easy to see that this in fact yields an optimal (fractional) packing. $\lhd$

We turn to the relation between formulations and the quality of Linear Programming relaxations:

Lemma 6.6 *Let P_1 and P_2 be formulations for the set $X \subseteq \mathbb{R}^n$, where P_1 is better than P_2. Consider the integer program $z^{IP} = \max \left\{c^T x : x \in X\right\}$ and denote by $z_i^{LP} = \max \left\{c^T x : x \in P_i\right\}$ for $i = 1, 2$ the values of the associated Linear Programming relaxations. Then we have*

$$z^{IP} \leq z_1^{LP} \leq z_2^{LP}$$

for all vectors $c \in \mathbb{R}^n$.

Proof: The result immediately follows from the fact that $X \subseteq P_1 \subseteq P_2$. □

Example 6.7
By using the formulation given in Example 6.4 we obtained an upper bound of 12 for the optimal value of the IP. If we use the ideal formulation

$$\begin{aligned} z = \max \quad & 2x_1 + 5x_2 \\ & x_1 + x_2 \leq 5 \\ & -x_1 + x_2 \leq 1 \\ & 1 \leq x_1 \leq 3 \\ & 1 \leq x_2 \leq 3 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

then we obtain the optimal solution for the relaxation $(2,3)$ with objective function value 11. Thus, $(2,3)$ is an optimal solution for the original IP. ◁

6.2 Combinatorial Relaxations

Sometimes the relaxation of a problem is a combinatorial optimization problem. In this case we speak of a *combinatorial relaxation*. We have already seen an example of such a problem in Example 6.5, where the LP-relaxation of KNAPSACK turned out to be solvable by combinatorial methods. Combinatorial relaxations are particularly nice if we can solve them in polynomial time.

Example 6.8 (Traveling Salesman Problem)
Consider the TSP from Example 1.8.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} & \\ & \sum_{j:j \neq i} x_{ij} = 1 & \text{for } i = 1, \dots, n \\ & \sum_{i:i \neq j} x_{ij} = 1 & \text{for } j = 1, \dots, n \\ & \sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 & \text{for all } \varnothing \subset S \subset V. \qquad (6.1) \\ & x \in \mathbb{B}^{n(n-1)} & \end{aligned}$$

Observe that if we drop the subtour elimination constraints (6.1) we have in fact an assignment problem. An *assignment* in a directed graph $G = (V, A)$ is a subset $T \subseteq A$ of the arcs such that for each vertex $v \in V$ we have either exactly one outgoing arc or exactly one incoming arc. If we interprete the TSP in the graph theoretic setting as we already did in Example 1.8 we get:

$$\begin{aligned} z^{\text{TSP}} &= \min_{T \subseteq A} \left\{ \sum_{(i,j) \in T} c_{ij} : T \text{ forms a tour} \right\} \\ &\geq \min_{T \subseteq A} \left\{ \sum_{(i,j) \in T} c_{ij} : T \text{ forms an assignment} \right\} \end{aligned}$$

◁

Assignments in directed graphs are an analogon to matchings in undirected graphs. Matchings will play an important role througout our lecture notes, one reason being that a theorem by Edmonds about perfect matching polyhedra sparked the interest in polyhedral combinatorics.

Definition 6.9 (Matching, perfect matching)
Let $G = (V, E)$ *be an undirected graph. A* ***matching*** *in* G *is a subset* $M \subseteq E$ *of the edges of* G *such that no two edges in* M *share an endpoint. A matching is called* ***perfect****, if for each* $v \in V$ *there is one edge in* M *that is incident to* v*.*

We remark here that we can decide in polynomial time whether a given graph has a perfect matching. We can also compute a perfect matching of minimum weight in polynomial time.

Example 6.10 (Symmetric Traveling Salesman Problem)
If in the TSP all distances are symmetric, that is $c_{ij} = c_{ji}$ for all i, j, one speaks of a *symmetric traveling salesman problem*. This problem is best modelled on a complete undirected graph $G = (V, E)$. Each tour corresponds to a Hamiltonian cycle in G, that is, a cycle which touches each vertex exactly once.

Observe that every TSP-tour (i.e., a Hamiltonian cycle) also contains a spanning tree: If we remove one edge of the tour we obtain a path, a somewhat degenerated spanning tree. Hence, the problem of finding a minimum spanning tree in $G = (V, E)$ with edge weights c_{ij} is a relaxation of the symmetric TSP:

$$z^{\text{TSP}} \geq z^{\text{MST}}.$$

Recall that a minimum spanning tree can be computed in polynomial time for instance by Kruskal's algorithm. ◁

The relaxation in the previous example can actually be used to prove an improved lower bound on the length of the optimal TSP-tour. Suppose that we are given an instance of the symmetric TSP where the distances satisfy the *triangle inequality*, that is, we have

$$c_{ij} \leq c_{ik} + c_{kj} \quad \text{for all } i, j, k.$$

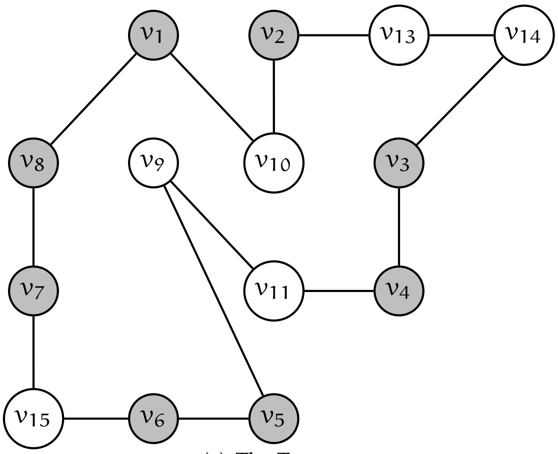
Let T be a minimum spanning tree in $G = (V, E)$ and let O denote the subset of the vertices which have an odd degree in T. Observe that O contains an even number of vertices, since the sum of all degrees in T sums up to $2(n-1)$ which is even. Build a complete auxiliary graph $H = (O, E_O)$ where the weight of an edge (u, v) coincides with the corresponding edgeweight in G. Since H contains an even number of vertices and is complete, there exists a perfect matching M in H. We can compute a perfect matching with minimum weight in H in polynomial time.

Lemma 6.11 *The total weight* $c(M) = \sum_{e \in M} c_e$ *of the minimum weight perfect matching in* H *is at most* $1/2 z^{\text{TSP}}$*.*

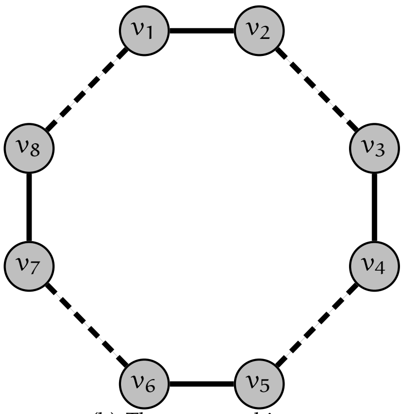
Proof: Let $O = (v_1, \dots, v_{2k})$ be the sequence of odd degree vertices in T in the order as they are visited by the optimum TSP-tour T^*. Consider the following two perfect matchings in H:

$$M_1 = \{(v_1, v_2), (v_3, v_4), \dots, (v_{2k-1}, v_{2k})\}$$
$$M_2 = \{(v_2, v_3), (v_4, v_5), \dots, (v_{2k}, v_1)\}$$

Figure 6.2 shows an illustration of the matchings M_1 and M_2. By the triangle inequality we have $c(T^*) \geq c(M_1) + c(M_2)$, so that at least one of the two matchings has weight at most $1/2c(T^*)$. Thus, the minimum weight perfect matching must have weight at most $1/2c(T^*)$. □



(a) The TSP-tour.



(b) The two matchings

Figure 6.2: The two matchings constructed in Lemma 6.11 from the optimal TSP-tour. Matching M_1 contains all the solid edges, M_2 all the dashed ones.

Consider the graph $\bar{G} = (V, T \cup M)$ obtained by adding the edges of the minimum weight perfect matching M to the spanning tree T (see Figure 6.3(c)). Then, by construction any node in $\bar{G}$ has even degree. Since $\bar{G}$ is also connected (by the fact that it contains a spanning tree), it follows that $\bar{G}$ is Eulerian (see e.g. [Har72, AMO93]), that is, it contains a cycle W which traverses every edge exactly once. We have that

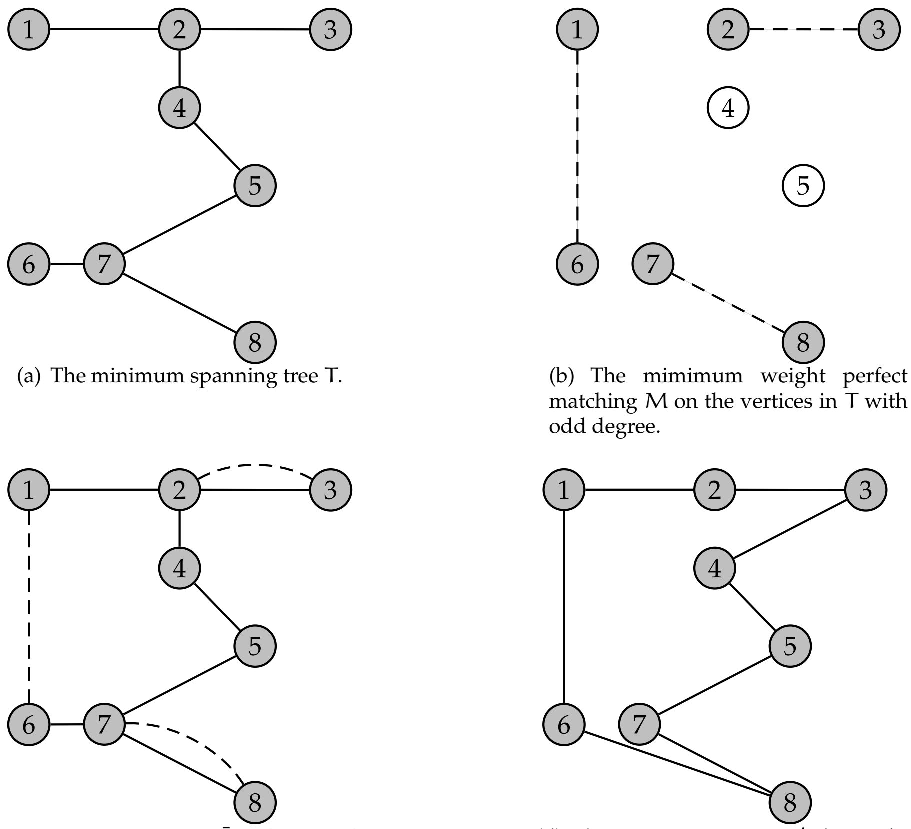
$$c(W) = \sum_{w \in W} c(e) = c(T) + c(M) \leq z^{\mathrm{TSP}} + \frac{1}{2} z^{\mathrm{TSP}} = \frac{3}{2} z^{\mathrm{TSP}}. \tag{6.2}$$

In other words, $\frac{2}{3} c(W) \leq z^{\mathrm{TSP}}$ is a lower bound for the optimal TSP-tour. Moreover, we can convert the cycle W into a feasible tour T' by "shortcutting" the Eulerian cycle: we start to follow W at node 1. If we have reached some node i, we continue to the first subsequent node in W which we have not visited yet. Due to the triangle inequality, the tour obtained this way has weight at most that of W which by (6.2) is at most $3/2z^{\text{TSP}}$. The algorithm we just described is known as *Christofides' algorithm*. Figure 6.3 shows an example of the execution. Let T' be the tour found by the algorithm. By (6.2) we know that

$$\frac{2}{3}c(T') \leq z^{\text{TSP}} \leq c(T').$$

We summarize our results:

Theorem 6.12 *Given any instance of the symmetric* TSP *with triangle inequality, Christofides' algorithm returns a tour of length at most* $3/2z^{\text{TSP}}$. □



(a) The minimum spanning tree T.

(b) The mimimum weight perfect matching M on the vertices in T with odd degree.

(c) The Eulerian graph $\bar{G} = (V, T \cup M)$.

(d) The approximate tour T' obtained by shortcutting.

Figure 6.3: Example of the execution of Christofides' algorithm for the symmetric TSP with triangle inequality.

Christofides' algorithm falls in the class of *approximation algorithms*. Those are algorithms with provable worst-case performance guarantees.

Example 6.13 (1-Tree relaxation for the symmetric TSP)
Another relaxation of the symmetric TSP is obtained by the following observation: Every tour consists of two edges incident to node 1 and a path through nodes $\{2, \ldots, n\}$ (in some order). Observe that every path is also a tree. Call a subgraph T of $G = (V, E)$ a *1-tree*, if T consists of two edges incident on node 1

and the edges of a tree on nodes $\{2, \ldots, n\}$. We get:

$$\begin{aligned} z^{\mathrm{TSP}} &= \min \left\{ \sum_{e \in T} c_e : T \text{ is a tour} \right\} \\ &\geq \min \left\{ \sum_{e \in T} c_e : T \text{ is a 1-tree} \right\} \\ &=: z^{\text{1-Tree}} \end{aligned} \tag{6.3}$$

The problem in (6.3) is called the *1-tree relaxation* of the (symmetric) TSP. Observe that we can find an optimal 1-tree easily in polynomial time: Let $e, f \in E$ with $e \neq f$ be the two lightest edges incident with 1 and let $H = G \setminus \{1\}$ be the graph obtained from G by removing vertex 1. Then, the weight of an optimal 1-tree is $c_e + c_f + \mathrm{MST}(H)$, where $\mathrm{MST}(H)$ is the weight of a minimum spanning tree in H. $\triangleleft$

6.3 Lagrangian Relaxation

The shortest path problem consists of finding a path of minimum weight between given vertices s and t in a directed (or undirected) graph G. Here, we consider the directed version, that is, we are given a directed graph $G = (V, A)$ with arc weights $c\colon A \to \mathbb{R}_+$. We set up binary variables $x(i, j)$ for $(i, j) \in A$ where $x(i, j) = 1$ if the path from s to t uses arc (i, j). Then, the shortest path problem (which is a special case of the minimum cost flow problem) can be written as the following IP:

$$\begin{aligned} &\min \sum_{(i,j) \in A} c(i,j) x(i,j) \\ &\sum_{j:(j,i) \in A} x(j,i) - \sum_{j:(i,j) \in A} x(i,j) = \begin{cases} 1 & \text{if } i = t \\ -1 & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \qquad (6.4) \\ &x \in \mathbb{B}^A \end{aligned}$$

The mass balance constraints (6.4) ensure that any feasible solution contains a path from s to t. Now, the shortest path problem can be solved very efficiently by combinatorial algorithms. For instance, Dijkstra's algorithm implemented with Fibonacci-heaps runs in time $\mathcal{O}(m + n \log n)$, where $n = |V|$ and $m = |A|$. In other words, the above IP is "easy" to solve.

Now, consider the addition of the following constraint to the IP: in addition to the arc weights $c\colon A \to \mathbb{R}_+$ we are also given a second set of arc costs $d\colon A \to \mathbb{R}_+$. Given a budget B we wish to find a shortest (s, t)-path with respect to the c-weights subject to the constraint that the d-cost of the path is at most B. This problem is known as the *resource constrained shortest path problem (*RCSP*)*.

Lemma 6.14 *The* RCSP *is* NP*-hard to solve.*

Proof: We show the claim by providing a polynomial time reduction from KNAPSACK, which is known to be NP-complete to solve. Given an instance of KNAPSACK with items $\{1, \ldots, n\}$ we construct a directed acyclic graph for the RCSP as depicted in Figure 6.4. Let $Z := \max\{c_i : i = 1, \dots, n\}$. For each $i = 1, \dots, n$ there are two arcs ending at node i. One represents the action that item i is packed and has c-weight $Z - c_i \geq 0$ and d-weight a_i. The other arc corresponds to the case when item i is not packed into the knapsack and has c-weight Z and d-weight 0. We set the budget in the RCSP to be $D := b$, where b is the bound specified in the instance of KNAPSACK. It is now easy to see that there is a path P from s to t of c-weight at most $nZ - u$ and d-weight at most b, if and only if we can pack items of profit at least u into the knapsack of size b.

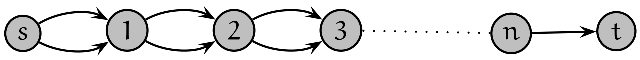


Figure 6.4: Graph used in the proof of NP-completeness of the RCSP

□

Apparently, the addition of the constraint $\sum_{(i,j) \in A} d(i,j)x(i,j) \leq B$ makes our life much harder! Essentially, we are in the situation, where we are given an integer program of the following form:

(6.5a) $z = \max c^T x$

(6.5b) $Dx \leq d$

(6.5c) $x \in X,$

with $X = \{Ax \leq b, x \in \mathbb{Z}^n\}$ and $Dx \leq d$ are some "complicating constraints" (D is a $k \times n$ matrix and $d \in \mathbb{R}^k$).

Of course, we can easily obtain a relaxation of (6.5) by simply dropping the complicating constraints $Dx \leq d$. However, this might give us very bad bounds. Consider for instance the RCSP depicted in Figure 6.5. If we respect the budget constraint $\sum_{a \in A} d_a x_a \leq B$, then the length of the shortest path from s to t is Ω. Dropping the constraint allows the shortest path consisting of arc a_1 which has length 1.

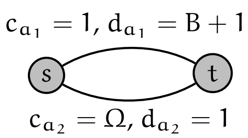


Figure 6.5: A simple RCSP where dropping the constraint $\sum_{a \in A} d_a x_a \leq B$ leads to a weak lower bound.

An alternative to dropping the constraints is to incorporate them into the objective function with the help of Lagrange multipliers:

Definition 6.15 (Langrangean Relaxation)
For a vector $u \in \mathbb{R}^m_+$ *the* ***Lagrangean relaxation*** $IP(u)$ *for the IP (6.5) is defined as the following optimization problem:*

$$\text{(6.6a)} \qquad IP(u) \quad z(u) = \max \quad c^T x + u^T(d - Dx)$$
$$\text{(6.6b)} \qquad x \in X$$

We note that there is no "unique" Lagrangean relaxation, since we are free to include or exclude some constraints by the definition of the set X. The relaxation IP(u) as defined above is usually called the relaxation obtained by *relaxing the constraints* $Dx \leq d$. The vector $u \geq 0$ is referred to as the *price, dual variable or Lagrangian multiplier* associated with the constraints $Dx \leq d$.

Lemma 6.16 *For any* $u \geq 0$*, the problem* $IP(u)$ *is a relaxation of the IP (6.5).*

Proof: The fact that any feasible solution to the IP is also feasible to IP(u) is trivial: $X \supseteq \{x : Dx \leq d, x \in X\}$. If x is feasible for the IP, then $Dx \leq d$ or $d - Dx \geq 0$. Since $u \geq 0$ we have $u^T(d - Dx) \geq 0$ and thus $c^T x + u^T(d - Dx) \geq c^T x$ as required. □

Consider again the RCSP. What happens, if we relax the budget constraint $\sum_{(i,j)\in A} d(i,j)x(i,j) \leq B$? For $u \in \mathbb{R}_+$ the Lagrangean relaxation is

$$\begin{aligned}
\min \sum_{(i,j)\in A} c(i,j)x(i,j) &+ u\left(\sum_{(i,j)\in A} d(i,j)x(i,j) - B\right) \\
\sum_{j:(j,i)\in A} x(j,i) - \sum_{j:(i,j)\in A} x(i,j) &= \begin{cases} 1 & \text{if } i = t \\ -1 & \text{if } i = s \\ 0 & \text{otherwise} \end{cases} \\
x &\in \mathbb{B}^A.
\end{aligned}$$

If we rearrange the terms in the objective function, we obtain

$$\sum_{(i,j)\in A} (c(i,j) + ud(i,j))x(i,j) - uB.$$

In other words, IP(u) asks for a path P from s to t minimizing $\sum_{a\in P}(c_a + ud_a) - uB$. If $u \geq 0$ is fixed, this is the same as minimizing $\sum_{a\in P}(c_a + ud_a)$. Hence, IP(u) is a (standard) shortest path problem with weights $(c + ud)$ on the arcs.

We have seen that IP(u) is a relaxation of IP for every $u \geq 0$, in particular $z(u) \geq z^{IP}$. In order to get the best upper bound for z^{IP} we can optimize over u, that is, we solve the *Lagrangean Dual Problem:*

$$\text{(6.7)} \qquad \text{(LD)} \quad w_{LD} = \min\{z(u) : u \geq 0\}.$$

Lemma 6.17 *Let* $u \geq 0$ *and* $x(u)$ *be an optimal solution for* $IP(u)$*. Suppose that the following two conditions hold:*

(i) $Dx(u) \leq d$ *(that is,* x *is* feasible *for IP);*

(ii) $(Dx(u))_i = d_i$ *if* $u_i > 0$ *(that is,* x *is* complementary *to* u*).*

Then, $x(u)$ *is an optimal solution for the original IP.*

Proof: Since $x(u)$ is feasible for IP, we have $c^T x(u) \leq z$. On the other hand

$$\begin{aligned} z &\leq c^T x(u) + u^T(d - Dx(u)) && \text{(by Lemma 6.16)} \\ &= c^T x(u) + \sum_{i=1}^{m} \underbrace{u_i (Dx(u) - d)_i}_{=0} && \text{(by assumption (ii))} \\ &= c^T x(u). \end{aligned}$$

Thus, $c^T x(u) = z$ and $x(u)$ is optimal as claimed. □

Observe that if we dualize some equality constraints $Dx = d$, then the Lagrange multiplier u is unrestricted in sign. In this case, the Lagrangean dual becomes:

$$\text{(LD)} \quad w_{\text{LD}} = \min \{z(u) : u \in \mathbb{R}^m\}. \tag{6.8}$$

Moreover, in this case, assumption (ii) of Lemma 6.17 is automatically satisfied. Hence, if $x(u)$ is feasible for the IP, then $x(u)$ is also optimal.

What kind of bounds can we expect from LD? The following theorem addresses this question:

Theorem 6.18 *Let* $X = \{x : Ax \leq b, x \in \mathbb{Z}^n\}$ *where* A *and* b *are rational, and let* z *denote the optimal value of the IP (6.5):*

$$z = \max \left\{c^T x : Dx \leq x, x \in X\right\}$$

Then for the value of the Lagrangean dual as defined in (6.8) we have:

$$w_{LD} = \max \left\{c^T x : Dx \leq d, x \in \text{conv}(X)\right\} \geq z.$$

Proof: We have:

$$\begin{aligned} w_{\text{LD}} &= \min\{z(u) : u \geq 0\} \\ &= \min_{u \geq 0} \max \left\{c^T x + u^T(d - Dx) : x \in X\right\} \\ &= \min_{u \geq 0} \max \left\{(c - D^T u)^T x + u^T d : x \in X\right\} \\ &= \min_{u \geq 0} \max \left\{c^T x + u^T(d - Dx) : x \in \text{conv}(X)\right\}, \end{aligned}$$

where the last equality follows from the fact that the objective function is linear. If $X = \varnothing$, then $\text{conv}(X) = \varnothing$ and $w_{\text{LD}} = -\infty$ since the inner maximum is taken over the empty set for every u. Since $z = -\infty$, the result holds in this case.

If $X \neq \varnothing$ by Theorem 3.67 $\text{conv}(X)$ is a rational polyhedron. Let x^k, $k \in K$ be the extreme points and r^j, $j \in J$ be the extreme rays of $\text{conv}(X)$. Fix $u \geq 0$. We have

$$\begin{aligned} z(u) &= \max \left\{c^T x + u^T(d - Dx) : x \in \text{conv}(X)\right\} \\ &= \begin{cases} +\infty & \text{if } (c^T - u^T D) r^j > 0 \text{ for some } j \in J \\ c^T x^k + u^T(d - Dx^k) & \text{for some } k \in K \text{ otherwise.} \end{cases} \end{aligned}$$

Since for the minimization of $z(u)$ it suffices to consider the case of finite $z(u)$, we have

$$w_{LD} = \min_{u \geq 0 : (c^T - u^T D) r^j \leq 0 \text{ for } j \in J} \max_{k \in K} c^T x^k + u^T (d - D x^k)$$

We can restate the last problem as follows:

$$w_{LD} = \min t \tag{6.9a}$$

$$t + (Dx^k - d)^T u \geq c^T x^k \text{ for } k \in K \tag{6.9b}$$

$$(Dr^j)^T u \geq c^T r^j \text{ for } j \in J \tag{6.9c}$$

$$u \in \mathbb{R}^m_+, t \in \mathbb{R} \tag{6.9d}$$

Observe that the problem (6.9) is a Linear Program. Hence by Linear Programming duality (cf. Theorem 2.8) we get:

$$\begin{aligned} w_{LD} = \max & \sum_{k \in K} \alpha_k c^T x^k + \sum_{j \in J} \beta_j c^T r^j \\ & \sum_{k \in K} \alpha_k = 1 \\ & \sum_{k \in K} \alpha_k (Dx^k - d) + \sum_{j \in J} \beta_j r^j \leq 0 \\ & \alpha_k, \beta_j \geq 0, \text{ for } k \in K, j \in J \end{aligned}$$

Rearranging terms this leads to:

$$\begin{aligned} w_{LD} = \max & \, c^T \left(\sum_{k \in K} \alpha^k x^k + \sum_{j \in J} \beta_j r^j \right) \\ & \sum_{k \in K} \alpha_k = 1 \\ & D \left(\sum_{k \in K} \alpha^k x^k + \sum_{j \in J} \beta_j r^j \right) \leq d (\sum_{k \in K} \alpha_k) \\ & \alpha_k, \beta_j \geq 0, \text{ for } k \in K, j \in J \end{aligned}$$

Since any point of the form $\sum_{k \in K} \alpha^k x^k + \sum_{j \in J} \beta_j r^j$ is in $\text{conv}(X)$ (cf. Minkowski's Theorem), this gives us:

$$w_{LD} = \max \left\{ c^T x : x \in \text{conv}(X), Dx \leq d \right\}.$$

This completes the proof. $\square$

Theorem 6.18 tells us exactly how strong the Langrangean dual is. Under some circumstances, the bounds provided are no better than that of the LP-relaxation: More precisely, if $\text{conv}(X) = \{x : Ax \leq b\}$, then by Theorem 6.18: $w_{LD} = \max \left\{ c^T x : Ax \leq b, Dx \leq d \right\}$ and w_{LD} is exactly the value of the LP-relaxation of (6.5):

$$z^{LP} = \max \left\{ c^T x : Ax \leq b, Dx \leq d \right\}$$

However, since

$$\text{conv}(X) = \text{conv} \{ x \in \mathbb{Z}^n : Ax \leq b \} \subseteq \{ x \in \mathbb{R}^n : Ax \leq b \},$$

we always have

$$z \le w_{LD} \le z^{LP},$$

and in most cases we will have $w_{LD} < z^{LP}$. In Chapter 11 we will learn more about Lagrangean duality. In particular, we will be concerned with the question how to solve the Lagrangean dual.

Example 6.19 (Lagrangean relaxation for the symmetric TSP)
The symmetric TSP can be restated as the following Integer Linear Program:

$$z^{\text{TSP}} = \min \sum_{e \in E} c_e x_e \tag{6.10a}$$

$$\sum_{e \in \delta(i)} x_e = 2, \text{for all } i \in V \tag{6.10b}$$

$$\sum_{e \in E(S)} x_e \le |S| - 1, \text{ for all } 2 \le |S| \le |V| - 1 \tag{6.10c}$$

$$x \in \mathbb{B}^E \tag{6.10d}$$

Here $\delta(i)$ denotes the set of edges incident with node i and $E(S)$ is the set of edges that have both endpoints in S.

Let x be any feasible solution to the LP-relaxation of (6.10). Let $S \subset V$ with $2 \le |S| \le |V| - 1$ and $\bar{S} = V \setminus S$. We denote by $(S, \bar{S})$ the set of edges which have one endpoint in S and the other one in $\bar{S}$. Due to the constraints (6.10b) we have:

$$|S| = \frac{1}{2} \sum_{i \in S} \sum_{j \in \delta(i)} x_e.$$

Hence,

$$\begin{aligned} |S| - \sum_{e \in E(S)} x_e &= \frac{1}{2} \sum_{i \in S} \sum_{j \in \delta(i)} x_e - \sum_{e \in E(S)} x_e \\ &= \frac{1}{2} \sum_{e \in (S, \bar{S})} x_e. \end{aligned} \tag{6.11}$$

By the analogous calculation we get

$$|\bar{S}| - \sum_{e \in E(\bar{S})} x_e = \frac{1}{2} \sum_{e \in (S, \bar{S})} x_e. \tag{6.12}$$

From (6.11) and (6.12) we conclude that $\sum_{e \in E(S)} x_e \le |S| - 1$ if and only if $\sum_{e \in E(\bar{S})} x_e \le |\bar{S}| - 1$. This calculation shows that in fact half of the subtour elimination constraints (6.10b) are redundant. As a consequence, we can drop all subtour constraints with $1 \in S$. This gives us the equivalent integer program:

$$z^{\text{TSP}} = \min \sum_{e \in E} c_e x_e \tag{6.13a}$$

$$\sum_{e \in \delta(i)} x_e = 2, \text{for all } i \in V \tag{6.13b}$$

$$\sum_{e \in E(S)} x_e \le |S| - 1, \text{ for all } 2 \le |S| \le |V| - 1, 1 \notin S \tag{6.13c}$$

$$x \in \mathbb{B}^E \tag{6.13d}$$

If we sum up all the constraints (6.13b) we get $2\sum_{e\in E} x_e = 2n$, since every edge is counted exactly twice. Now comes our Lagrangean relaxation: We dualize all degree constraints (6.13b) in (6.13) but leave the degree constraint for node 1. We also add the constraint $\sum_{e\in E} x_e = n$. This gives us the following relaxation:

$$z(u) = \min \sum_{(i,j)\in E} (c_{ij} - u_i - u_j)x_e + 2\sum_{i\in V} u_i \tag{6.14a}$$
$$\sum_{e\in\delta(1)} x_e = 2 \tag{6.14b}$$
$$\sum_{e\in E(S)} x_e \leq |S| - 1, \text{ for all } 2 \leq |S| \leq |V| - 1, 1 \notin S \tag{6.14c}$$
$$\sum_{e\in E} x_e = n \tag{6.14d}$$
$$x \in \mathbb{B}^E \tag{6.14e}$$

Let us have a closer look at the relaxation (6.14). Every feasible solution T has two edges incident with node 1 by constraint (6.14b). By constraint (6.14d), the solution must consist of exactly n edges and hence has a cycle. By constraints (6.14c), T can not contain a cycle that does not contain node 1. Thus, if we remove one edge incident to 1, then T is cycle free and must contain a spanning tree on the vertices $\{2, \dots, n\}$. To summarize, the feasible solutions for (6.14) are exactly the 1-trees. We have already seen in Example 6.13 that we can compute a minimum weight 1-tree in polynomial time. Hence, the relaxation (6.14) is particularly interesting. $\lhd$

6.4 Duality

As mentioned at the beginning of Chapter 6, for Linear Programs the Duality Theorem is a convenient way of proving optimality or, more general, for proving upper and lower bounds for an optimization problem. We are now going to introduce the concept of duality in a more general sense.

Definition 6.20 ((Weak) dual pair of optimization problems)
The two optimization problems:

$$\begin{aligned} (IP) \quad & z = \max\{c(x) : x \in X\} \\ (D) \quad & w = \min\{\omega(y) : y \in Y\} \end{aligned}$$

for a ***(weak-) dual pair****, if* $c(x) \leq \omega(y)$ *for all* $x \in X$ *and all* $y \in Y$*. If* $z = w$*, then we say that the problem form a* ***strong-dual pair****.*

By the Duality Theorem of Linear Programming, the both problems

$$\begin{aligned} &(P) \max\{\, c^T x : x \in Ax \leq b \,\} \\ &(D) \min\{\, b^T y : y \in Q \,\} \end{aligned}$$

form a strong-dual pair.

Example 6.21 (Matching and vertex cover)
Given an undirected graph $G = (V, E)$, the following two problems form a weak-dual pair:

$$\max\{|M| : M \subseteq E \text{ is a matching in } G\}$$
$$\min\{|C| : C \subseteq V \text{ is a vertex cover in } G\}$$

In fact, let $M \subseteq E$ be a matching in G and $C \subseteq V$ a vertex cover. Since M is a matching, the edges in M do not share a common endpoint. Thus, since C must contain at least one endpoint for each $e \in M$ (and all of the $2|M|$ endpoints are different as we have just seen), we get that $|C| \geq |M|$.

Do the two problems form a strong-dual pair? The answer is no: Figure 6.6 shows an example of a graph, where the maximum size matching has cardinality 1, but the minimum vertex cover has size 2. After a moment's thought, it would be very surprising if the two problems formed a strong-dual pair: the vertex-cover problem is NP-hard to solve, whereas the matching problem is solvable in polynomial time. $\lhd$

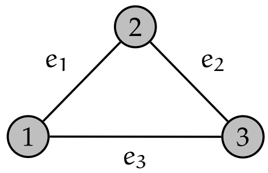


Figure 6.6: A graph where the maximum matching has size 1, but the minimum vertex cover has size 2.

Lemma 6.22 *The Integer Linear Program* $\max\left\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\right\}$ *and the Linear Program* $\min\left\{b^T y : A^T y \geq c, y \in \mathbb{R}^m_+\right\}$ *form a weak dual pair. Moreover, also the two Integer Linear Programs*

$$\max\left\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\right\}$$
$$\min\left\{b^T y : A^T y \geq c, y \in \mathbb{Z}^m_+\right\}$$

form a weak dual pair.

Proof: We have

$$\begin{aligned}\max\left\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\right\} &\leq \max\left\{c^T x : Ax \leq b, x \in \mathbb{R}^n\right\}\\ &= \min\left\{b^T y : A^T y \geq c, y \in \mathbb{R}^m_+\right\}\\ &\leq \min\left\{b^T y : A^T y \geq c, y \in \mathbb{Z}^m_+\right\},\end{aligned}$$

where the equality follows from the Duality Theorem of Linear Programming. $\Box$

Example 6.23 (Matching and vertex cover (continued))
We formulate the maximum cardinality matching problem as an integer program:

$$z = \max \sum_{e \in E} x_e \tag{6.15}$$

$$\sum_{e : e \in \delta(v)} x_e \leq 1, \text{ for all } v \in V \tag{6.16}$$

$$x \in \mathbb{B}^E \tag{6.17}$$

Let z^{LP} be the value of the LP-relaxation of (6.15) obtained by dropping the integrality constraints. The dual of the LP-relaxation is

$$w^{LP} = \min \sum_{v \in V} y_v \tag{6.18}$$

$$x_u + x_v \geq 1, \text{ for all } e = (u, v) \in E \tag{6.19}$$

$$x \geq 0. \tag{6.20}$$

If we put integrality constraints on (6.18) we obtain the vertex cover problem. Let w denote the optimal value of this problem. We then have $z \leq z^{LP} = w^{LP} \leq w$. This is an alternative proof for the fact that the matching problem and the vertex cover problem form a weakly dual pair.

Consider again the graph depicted in Figure 6.6. We have already seen that for this graph $z = 1$ and $w = 2$. Moreover, the Linear Programming relaxation is

$$\begin{aligned}
\max x_{e_1} + x_{e_2} + x_{e_3} \\
x_{e_1} + x_{e_2} \leq 1 \\
x_{e_1} + x_{e_3} \leq 1 \\
x_{e_2} + x_{e_3} \leq 1 \\
x_{e_1}, x_{e_2}, x_{e_3} \geq 0
\end{aligned}$$

The vector $x_{e_1} = x_{e_2} = x_{e_3} = 1/2$ is feasible for this relaxation with objective function value 3/2. The dual of the LP is

$$\begin{aligned}
\min y_1 + y_2 + y_3 \\
y_1 + y_2 \geq 1 \\
y_2 + y_3 \geq 1 \\
y_1 + y_3 \geq 1 \\
y_1, y_2, y_3 \geq 0
\end{aligned}$$

so that $y_1 = y_2 = y_3 = 1/2$ is feasible for the dual. Hence we have $z^{LP} = w^{LP} = 3/2$. $\triangleleft$

7 Dynamic Programming

7.1 Shortest Paths Revisited

Suppose that we are given a directed graph $G = (V, A)$ with arc lengths $c\colon A \to \mathbb{R}_+$ and we wish to compute a shortest paths from a distinguished source node s to all other vertices $v \in V$. Let $d(v)$ denote the shortest path distance from s to v with respect to c.

Suppose the shortest path from s to v passes through some intermediate node u (see Figure 7.1 for an illustration). What can we say about the paths P_{su} and P_{uv} from s to u and from u to v? Clearly, P_{su} must be a shortest (s, u)-path since otherwise we could replace it by a shorter one which together with P_{uv} would form a shorter path from s to v. The same arguments show that P_{uv} must be a shortest (u, v)-path.

The above observations are known as the *Bellman principle of optimality*: any subpath (any partial solution) of a shortest path (of an optimal solution) must be a shortest path (an optimal solution of a subproblem) itself.

Let $v \in V \setminus \{s\}$. If we apply the Bellman principle to all predecessors u of v, that is, to all u such that $(u, v) \in A$ we get the following recurrences for the shortest path distances:

$$d(v) = \min_{u:(u,v)\in A} d(u) + c(u, v). \tag{7.1}$$

Equation (7.1) states that, if we know the shortest path distance $d(u)$ from s to all predecessors u of v, then we can easily compute the shortest path distance $d(v)$. Unfortunately, it may be the case that in order to compute $d(u)$ we

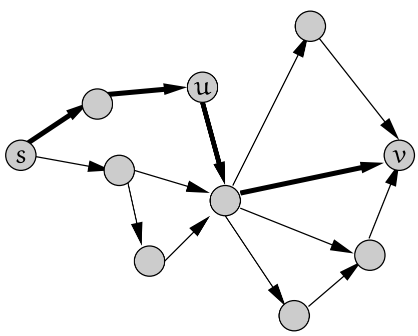


Figure 7.1: A shortest path from s to v that passes through the intermediate node u. The subpath from s to u must be a shortest (s, u)-path.

also need $d(v)$, because v in turn is a predecessor of u. So, for general graphs, (7.1) does not yet give us a method for solving the shortest path problem.

There is, however, one important class of graphs, where (7.1) can be used such as to obtain a very efficient algorithm.

Definition 7.1 (Directed acyclic graph (DAG))
*A **directed acyclic graph (DAG)** is a graph that does not contain a directed cycle.*

DAGs allow a special ordering of the vertices defined below:

Definition 7.2 (Topological sorting)
*Let $G = (V, A)$ be a directed graph. A bijection $f\colon V \to \{1, \dots, n\}$ is called a **topological sorting** of G if for all $(u, v) \in A$ we have $f(u) < f(v)$.*

The proof of the following fact is left as an exercise:

Theorem 7.3 *Let $G = (V, A)$ be a directed graph. Then, G has a topological sorting if and only if G is acyclic. If G is acyclic, the topological sorting can be computed in linear time.*

Proof: Exercise. □

Suppose that G is a DAG and that f is a topological sorting (which by the previous theorem can be obtained in linear time $\mathcal{O}(n + m)$, where $n = |V|$ and $m = |A|$). For the sake of a simpler notation we assume that the vertices are already numbered $v_1, \dots, v_n$ according to the topological sorting, that is, $f(v_i) = i$. We can now use (7.1) by computing $d(v_i) = 0$ in order of increasing $i = 1, \dots, n$. In order to compute $d(v_i)$ we need values $d(v_j)$ where $j < i$ and these have already been computed. Thus, in case of a DAG (7.1) leads to an algorithm which can be seen to run in total time $\mathcal{O}(n + m)$.

In our algorithm above we have calculated the solution value of a problem recursively from the optimal values of slightly different problems. This method is known as *dynamic programming*. We will see in this chapter that this method can be used to obtain optimal solutions to many problems. As a first step we go back to the shortest path problem but we now consider the case of general graphs. As we have already remarked above, the recursion (7.1) is not directly useful in this case. We need to somehow enforce an ordering on the vertices such as to get a usable recursion.

Let $d_k(v)$ be the length of a shortest path from s to v using at most k arcs. Then, we have

$$d_k(v) = \min\left\{ d_{k-1}(v), \min_{u:(u,v)\in A} d_{k-1}(u) + c(u, v) \right\}. \tag{7.2}$$

Now, the recurrence (7.2) gives us a way to compute the shortest path distances in time $\mathcal{O}((n + m)m)$. We first compute $d_1(v)$ for all v which is easy. Once we know all values d_{k-1}, we can use (7.2) to compute $d_k(v)$ for all $v \in V$ in time $\mathcal{O}(n+m)$ (every arc has to be considered exactly once). Since the range of k is from 1 to $n-1$ (every path with at least n arcs contains a cycle and thus can not be shortest), we get the promised running time of $\mathcal{O}(n(n + m)) = \mathcal{O}(n^2 + nm)$.

The general tune of a dynamic programming algorithm is the following: We compute a solution value recursively from the values of modified problems.

The problems that we compute solutions for are referred to as *states*, the ordering in which we compute these are usually called *stages*. We can imagine a dynamic programming algorithm as filling in the values in a table which is indexed by the states. In a stage we compute one table entry from other table entries which have been computed earlier.

7.2 Knapsack Problems

Consider the KNAPSACK problem (see Example 1.3 on page 2):

$$z = \max \sum_{i=1}^{n} c_i x_i \tag{7.3a}$$

$$\sum_{i=1}^{n} a_i x_i \leq b \tag{7.3b}$$

$$x \in \mathbb{B}^n \tag{7.3c}$$

Let us set up a dynamic programming algorithm for KNAPSACK. The states are subproblems $P_k(\lambda)$ of the form

$$(P_k(\lambda)) \quad f_k(\lambda) = \max \sum_{i=1}^{k} c_i x_i \tag{7.4a}$$

$$\sum_{i=1}^{k} a_i x_i \leq \lambda \tag{7.4b}$$

$$x \in \mathbb{B}^k \tag{7.4c}$$

More verbosely, the problem $P_k(\lambda)$ consists of getting the maximum profit from items $1, \ldots, k$ where the size of the knapsack is λ. We have $z = f_n(b)$, thus we get the optimal value once all $f_k(\lambda)$ for $k = 1, \ldots, n$, $\lambda = 0, \ldots, b$ are known.

We need a recursion to calculate the $f_k(\lambda)$. Consider an optimal solution x^* for $P_k(\lambda)$.

- If it does not use item k, then it $(x_1^*, \ldots, x_{k-1}^*)$ is an optimal solution for $P_{k-1}(\lambda)$, that is, $f_k(\lambda) = f_{k-1}(\lambda)$.
- If the solution uses k, then it uses weight at most $\lambda - a_k$ from items $1, \ldots, k-1$. By the arguments used for the shortest path problem, we get that $(x_1^*, \ldots, x_{k-1}^*)$ must be an optimal solution for $P_{k-1}(\lambda - a_k)$, that is, $f_k(\lambda) = f_{k-1}(\lambda - a_k) + c_k$.

Combining the above two cases yiels the recursion:

$$f_k(\lambda) = \max\{f_{k-1}(\lambda), f_{k-1}(\lambda - a_k) + c_k\}. \tag{7.5}$$

Thus, once we know all values f_{k-1} we can compute each value $f_k(\lambda)$ in constant time by just inspecting two states. Figure 7.2 illustrates the computation of the values $f_k(\lambda)$ is they are imagined to be stored in a $b \times n$-table. The kth column of the table is computed with the help of the entries in the $(k-1)$st column.

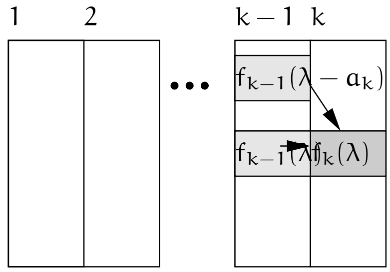


Figure 7.2: Computation of the values $f_k(\lambda)$ in the dynamic programming algorithm for KNAPSACK.

The recursion (7.5) gives a dynamic programming algorithm for KNAPSACK which runs in total time $\mathcal{O}(nb)$ (it is easy to actually set the binary variables according to the results of the recusion). Observe that this running time is *not polynomial*, since the encoding length of an instance of KNAPSACK is only $\mathcal{O}(n + n \log a_{\max} + n \log c_{\max} + \log b)$.

Let us now consider a generalized version of KNAPSACK, where we are allowed to pack more than one copy of an item. The *Integer Knapsack Problem (*INTEGERKNAPSACK*)* is:

$$z = \max \sum_{i=1}^{n} c_i x_i \tag{7.6a}$$

$$\sum_{i=1}^{n} a_i x_i \leq b \tag{7.6b}$$

$$x \in \mathbb{Z}_+^n \tag{7.6c}$$

Analogous to KNAPSACK we can define

$$(P_k(\lambda)) \quad g_k(\lambda) = \max \sum_{i=1}^{k} c_i x_i \tag{7.7a}$$

$$\sum_{i=1}^{k} a_i x_i \leq \lambda \tag{7.7b}$$

$$x \in \mathbb{Z}_+^k \tag{7.7c}$$

As before, $g_n(b)$ gives us the optimal solution for the whole problem. We need a recursion to compute the $g_k(\lambda)$, $k = 1, \ldots, n$, $\lambda = 0, \ldots, b$.

Let x^* be an optimal solution for $P_k(\lambda)$. Suppose that $x_k^* = t$ for some integer $t \in \mathbb{Z}_+$. Then, x^* uses space at most $\lambda - ta_k$ for the items $1, \ldots, k-1$. It follows that $(x_1^*, \ldots, x_{k-1}^*)$ must be an optimal solution for $P_k(\lambda - ta_k)$. These observations give us the following recursion:

$$g_k(\lambda) = \max_{t=0,\ldots,\lfloor \lambda/a_k \rfloor} \{c_k t + g_{k-1}(\lambda - ta_k)\}. \tag{7.8}$$

Notice that $\lfloor \lambda/a_k \rfloor \leq b$, since a_k is an integer. Thus (7.8) shows how to compute $g_k(\lambda)$ in time $\mathcal{O}(b)$. This yields an overall time of $\mathcal{O}(nb^2)$ for the nb values $g_k(\lambda)$.

Let us now be a bit smarter and get a dynamic programming algorithm with a better running time. The key is to accomplish the computation of $g_k(\lambda)$ by inspecting only a constant (namely two) previously computed values instead of $\Theta(b)$ as in (7.8).

Again, let x^* be an optimal solution for $P_k(\lambda)$.

- If $x_k^* = 0$, then $g_k(\lambda) = g_{k-1}(\lambda)$.
- If $x_k^* = t + 1$ for some $t \in \mathbb{Z}_+$, then $(x_1^*, \ldots, x_k^* - 1)$ must be an optimal solution for $P_k(\lambda - a_k)$.

The above two cases can be combined into the new recursion

$$g_k(\lambda) = \max\{g_{k-1}(\lambda), g_k(\lambda - a_k) + c_k\}. \tag{7.9}$$

Observe that the recursion allows us to compute all $g_k(\lambda)$ if we compute them in the order $k = 1, \ldots, n$ and $\lambda = 0, \ldots, b$. The total running time is $\mathcal{O}(nb)$ which is the same as for KNAPSACK.

7.3 Problems on Trees

Problems defined on tree structures are natural candidates for dynamic programming algorithms. Suppose that we are given a tree $T = (V, E)$ with root r (cf. Figure 7.3). The ubiquitous theme for applying dynamic programming algorithms on tree structured problems is to solve the problem on T by solving problems on T_{v_i}, $i = 1, \ldots, k$, where v_i are the children of r and T_{v_i} is the subtree of T rooted at v_i.

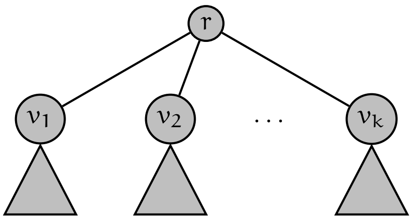


Figure 7.3: Recursive solution of a problem on a rooted tree.

We illustrate the construction by considering a specific problem. Suppose that we are given a tree $T = (V, E)$ with root r profits/costs $p\colon V \to \mathbb{R}$ for the vertices. The goal is to find a (possibly empty) subtree T' rooted at r (which may correspond to some distribution center) such as to maximize the profit, that is, maximize

$$f(T') := \sum_{v \in V(T')} p(v)$$

For a node $v \in V$ let $g(v)$ denote the optimal net profit of a subtree of T_v (the subtree rooted at v). Thus, the optimal solution (value) is obtained from $g(r)$.

We now show how to build a recursion in order to compute $g(v)$ for all vertices $v \in V$. The recursion scheme is typical for dynamic programming algorithms on trees. We compute the values $g(v)$ "bottom up" starting from the leaves. If v is a leaf, then

$$g(v) = \max\{0, p(v)\}. \tag{7.10}$$

Suppose now that v has the children $v_1, \ldots, v_k$ and that we have already computed $g(v_i)$, $i = 1, \ldots, k$. Any subtree T' of T_v is composed of (possibly empty) subtrees of T_{v_i}. Consider an optimal subtree T^* of T_v. Any subtree of T_{v_i} included in T^* must be an optimal subtree of T_{v_i} (by Bellman's principle). Thus, we get

$$g(v) = \max\left\{0, p(v) + \sum_{i=1}^{k} g(v_i)\right\}, \tag{7.11}$$

where the 0 in the maximum covers the case that the optimal subtree of T_v is empty. Equations (7.10) and (7.11) now give us a way to compute all the $g(v)$ in time $\mathcal{O}(n)$, where $n = |V|$ is the number of vertices in T.

We now consider a generalization of the problem. In addition to the profits/costs on the vertices, we are also given costs $c\colon E \to \mathbb{R}_+$ on the edges (the vertices correspond to customers which may be connected by a network that has to be built at some cost). The problem is now to select a subtree which maximizes the net profit:

$$h(T') := \sum_{v \in V(T')} p(v) - \sum_{e \in E(T')} c(e).$$

Again, we define $h(v)$ to be the optimal profit obtainable in the subtree rooted at $v \in V$. The problem once more easy for the leaves:

$$h(v) = \max\{0, p(v)\}. \tag{7.12}$$

Now consider a non-leaf v with children $v_1, \dots, v_k$. An optimal subtree T^* of T_v is once more composed of optimal subtrees of the T_{v_i}. But now the subtree T_{v_i} may only be nonempty if the edge (v, v_i) is contained in T^*. This gives us the recursion:

$$h(v) = \max\left\{0, \max_{x \in \mathbb{B}^k} \sum_{i=1}^{k} (h(v_i) - c(v, v_i))x_i\right\}. \tag{7.13}$$

Here, $x_i \in \mathbb{B}$ is a decision variable which indicates whether we should include the optimal subtree of T_{v_i} in the solution for T_v. It seems like (7.13) forces us to evaluate 2^k values in order to compute $h(v)$, if v has k children. However, the problem

$$\max_{x \in \mathbb{B}^k} \sum_{i=1}^{k} (h(v_i) - c(v, v_i))x_i$$

can be solved easily in time $\mathcal{O}(k)$: Set $x_i = 1$, if $h(v_i) - c(v, v_i) > 0$ and $x_i = 0$ otherwise. Let $\deg^+(v)$ denote the number of children of v in T. Then, $\sum_{v \in V} \deg^+(v) = n - 1$, since every edge is counted exactly once. Thus, the dynamic programming algorithm for the generalized problem runs in time $\mathcal{O}(n \sum_{v \in V} \deg^+(v)) = \mathcal{O}(n^2)$.

8

Branch and Bound

8.1 Divide and Conquer

Suppose that we would like to solve the problem

$$z = \max\left\{c^T x : x \in S\right\}.$$

If the above problem is hard to solve, we might try to break it into smaller problems which are easier to solve.

Observation 8.1 *If* $S = S_1 \cup \cdots \cup S_k$ *and* $z^i = \max\left\{c^T x : x \in S^i\right\}$ *for* $i = 1, \ldots, k$ *then* $z = \max\left\{z^i : i = 1, \ldots, k\right\}$.

Although Observation 8.1 is (almost) trivial, it is the key to branch and bound methods which turn out to be effective for many integer programming problems.

The recursive decomposition of S into smaller problems can be represented by an *enumeration tree*. Suppose that $S \subseteq \mathbb{B}^3$. We first divide S into S_0 and S_1, where

$$\begin{aligned} S_0 &= \{x \in S : x_1 = 0\} \\ S_1 &= \{x \in S : x_1 = 1\} \end{aligned}$$

The sets S_0 and S_1 will be recursively divided into smaller sets as shown in the enumeration tree in Figure 8.1.

As a second example consider the tours of a TSP on four cities. Let S denote the set of all tours. We first divide S into three pieces, S_{12}, S_{13} and S_{14}, where $S_{1i} \subset S$ is the set of all tours which from city 1 go directly to city i. Set S_{1i} in turn is divided into two pieces $S_{1i,ij}$, $j \neq 1, i$, where $S_{1i,ij}$ is the set of all tours in S_{i1} which leave city i for city j. The recursive division is depicted in the enumeration tree in Figure 8.2. The leaves of the tree correspond to the $(4-1)! = 3! = 6$ possible permutations of $\{1, \ldots, 4\}$ which have 1 at the first place.

8.2 Pruning Enumeration Trees

It is clear that a complete enumeration soon becomes hopeless even for problems of medium size. For instance, in the TSP the complete enumeration tree

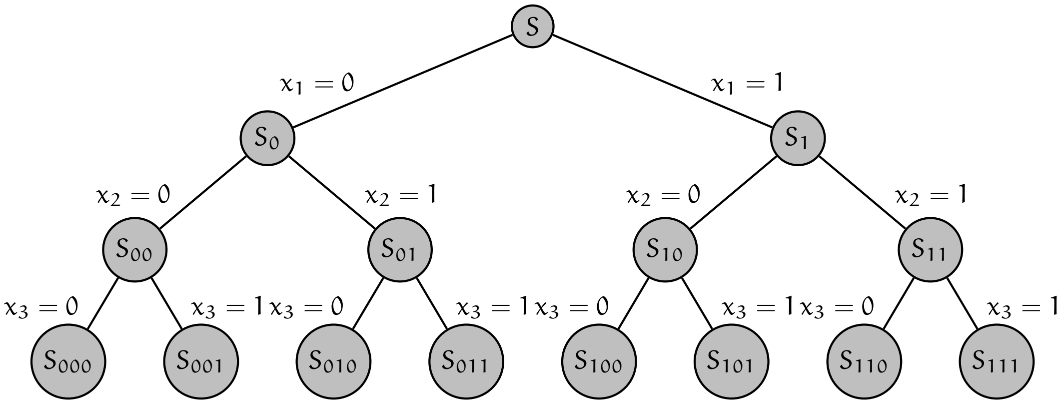


Figure 8.1: Enumeration tree for a set $S \subseteq \mathbb{B}^4$.

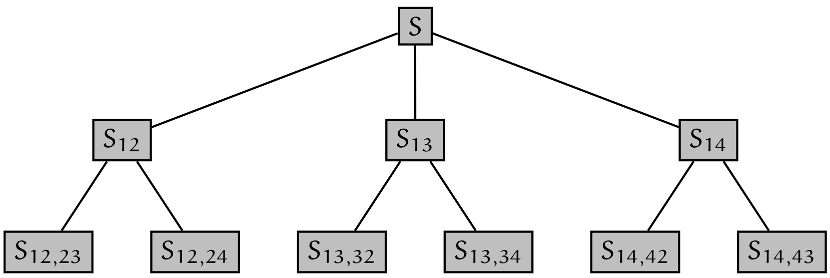


Figure 8.2: Enumeration tree for the TSP on four cities.

would have $(n-1)!$ leaves and thus for $n = 60$ would be larger than the estimated number of atoms in the universe[1].

The key to efficient algorithms is to "cut off useless parts" of an enumeration tree. This is where bounds (cf. Chapter 6) come in. The overall scheme obtained is also referred to as *implicit enumeration*, since most of the time not all of the enumeration tree is completely unfolded.

Observation 8.2 *Let* $S = S_1 \cup \dots S_k$ *and* $z^i = \max\left\{c^T x : x \in S_i\right\}$ *for* $i = 1, \dots, k$ *then* $z = \max\left\{z^i : i = 1, \dots, k\right\}$. *Suppose that we are given* $\underline{z}^i$ *and* $\bar{z}^i$ *for* $i = 1, \dots, k$ *with*

$$\underline{z}^i \leq z^i \leq \bar{z}^i \text{ for } i = 1, \dots, k.$$

Then for $z = \max\left\{c^T x : x \in S\right\}$ *it is true that:*

$$\max\left\{\underline{z}^i : i = 1, \dots, k\right\} \leq z \leq \max\left\{\bar{z}^i : i = 1, \dots, k\right\}$$

The above observation enables us to deduce rules how to *prune the enumeration tree*.

Observation 8.3 (Pruning by optimality) *If* $\underline{z}^i = \bar{z}^i$ *for some* i, *then the optimal solution value for the* i*th subproblem* $z^i = \max\left\{c^T x : x \in S_i\right\}$ *is known and there is no need to subdivide* S_i *into smaller pieces.*

As an example consider the situation depicted in Figure 8.3. The set S is divided into S_1 and S_2 with the upper and lower bounds as shown.

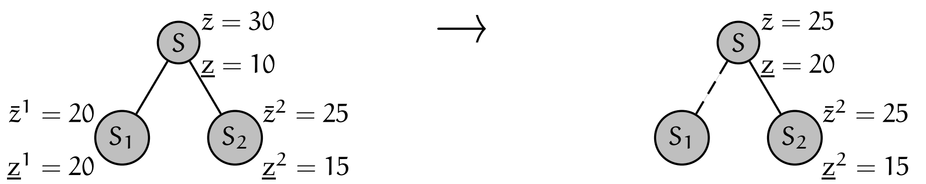


Figure 8.3: Pruning by optimality.

Since $\underline{z}^1 = \bar{z}^1$, there is no need to further explore the branch rooted at S_1. So, this branch can be pruned by optimality. Moreover, Observation 8.2 allows us to get new lower and upper bounds for z as shown in the figure.

Observation 8.4 (Pruning by bound) *If* $\bar{z}^i < \underline{z}^j$ *for some* $i \neq j$, *then the optimal solution value for the* i*th subproblem* $z^i = \max\left\{c^T x : x \in S_i\right\}$ *can never be an optimal solution of the whole problem. Thus, there is no need to subdivide* S_i *into smaller pieces.*

Consider the example in Figure 8.4. Since $\bar{z}^1 = 20 < 21 = \underline{z}^2$, we can conclude that the branch rooted at S_1 does not contain an optimal solution: any solution in this branch has objective function value at most 20, whereas the branch rooted at S_2 contains solutions of value at least 21. Again, we can stop to explore the branch rooted at S_1.

We list one more generic reason which makes the exploration of a branch unnecessary.

Observation 8.5 (Pruning by infeasibility) *If* $S_i = \emptyset$, *then there is no need to subdivide* S_i *into smaller pieces, either.*

[1] The estimated number of atoms in the universe is 10^{80}.

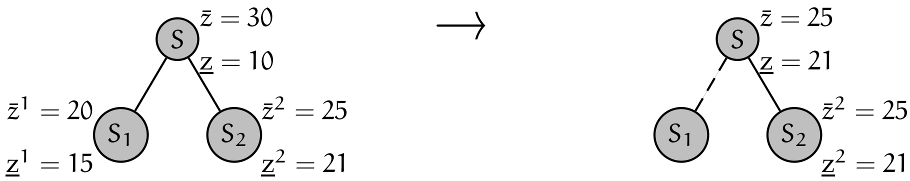


Figure 8.4: Pruning by bound.

The next section will make clear how the case of infeasibility can occur in a branch and bound system. We close this section by showing an example where no pruning is possible. Consider the partial enumeration tree in Figure 8.5, where again we have divided the set S into the sets S_1 and S_2 with the bounds as shown.

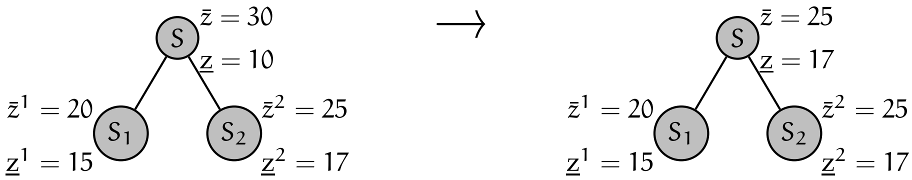


Figure 8.5: No pruning possible.

Although the lower and upper bounds for the subproblems allow us to get better bounds for the whole problem, we still have to explore both subtrees.

8.3 LP-Based Branch and Bound: An Example

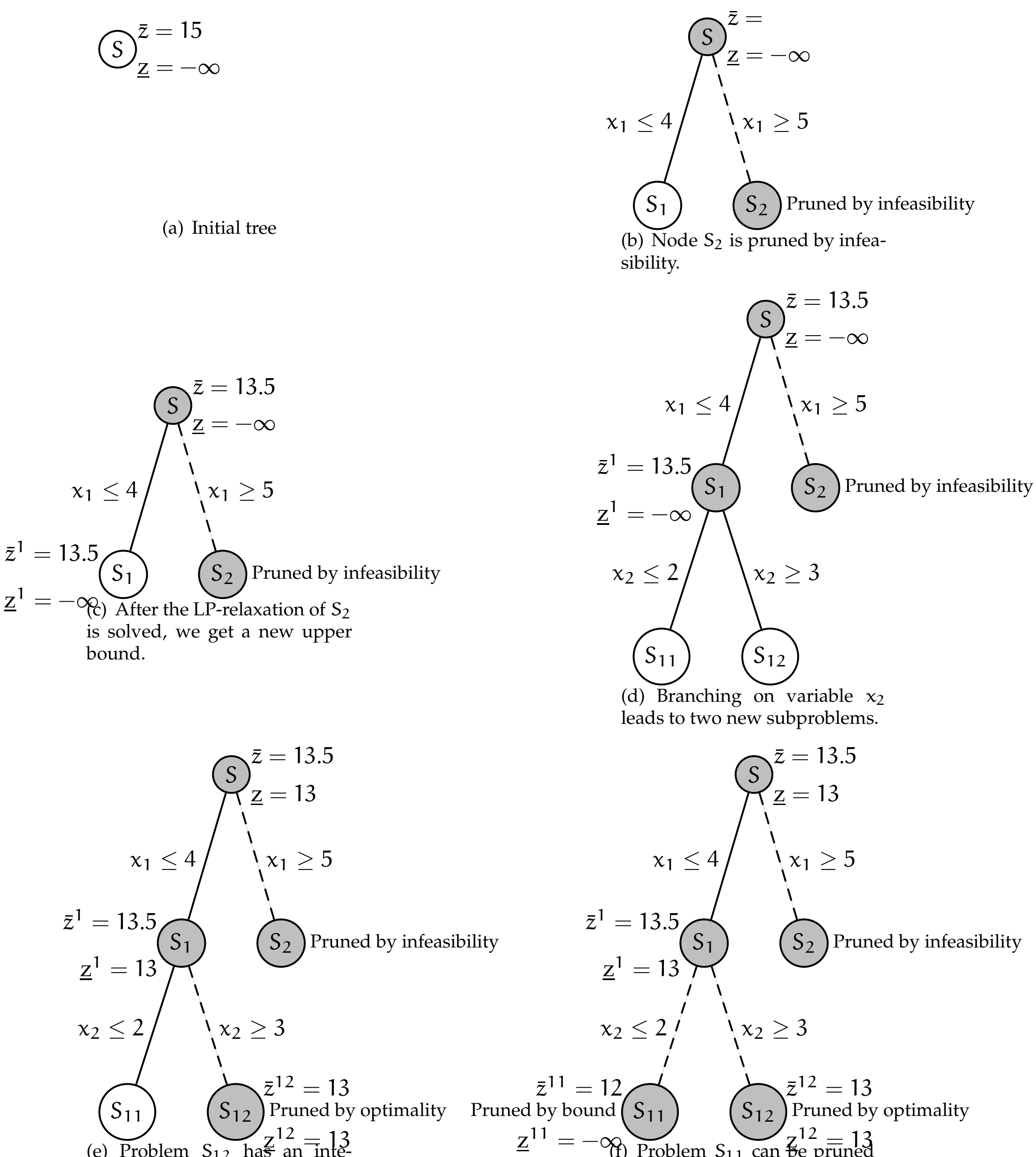
The most common way to solve integer programs is to use an implicit enumeration scheme based on Linear Programming. Lower bounds are provided by LP-relaxations and branching is done by adding new constraints. We illustrate this procedure for a small example. Figure 8.6 shows the evolution of the corresponding branch-and-bound-tree.

Consider the integer program

$$z = \max 4x_1 - x_2 \tag{8.1a}$$
$$3x_1 - 2x_2 + x_3 = 14 \tag{8.1b}$$
$$x_2 + x_4 = 3 \tag{8.1c}$$
$$2x_1 - 2x_2 + x_5 = 3 \tag{8.1d}$$
$$x \in \mathbb{Z}^5_+ \tag{8.1e}$$

Bounding The first step is to get upper and lower bounds for the optimal value z. An upper bound is obtained by solving the LP-relaxation of (8.1). This LP has the optimal solution $\tilde{x} = (4.5, 3, 6.5, 0, 0)$. Hence, we get the upper bound $\bar{z} = 15$. Since we do not have any feasible solution yet, by convention we use $\underline{z} = -\infty$ as a lower bound.

Branching In the current situation (see Figure 8.6(a)) we have $\underline{z} < \bar{z}$, so we have to branch, that is, we have to split the feasible region S. A common way to achieve this is to take an integer variable x_j that is basic but not integral and



(a) Initial tree

(b) Node S_2 is pruned by infeasibility.

(c) After the LP-relaxation of S_2 is solved, we get a new upper bound.

(d) Branching on variable x_2 leads to two new subproblems.

(e) Problem S_{12} has an integral optimal solution, so we can prune it by optimality. This also gives the first finite lower bound.

(f) Problem S_{11} can be pruned by bound, since $\bar{z}^{11} = 12 < 13 = \underline{z}$.

Figure 8.6: Evolution of the branch-and-bound-tree for the example. The active nodes are shown in white.

set:

$$S_1 = S \cap \{x : x_j \leq \lfloor \tilde{x}_j \rfloor\}$$
$$S_2 = S \cap \{x : x_j \geq \lceil \tilde{x}_j \rceil\}$$

Clearly, $S = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$. Observe that due to the above choice of branching, the current optimal basic solution $\tilde{x}$ is not feasible for either subproblem. If there is no degeneracy (i.e., multiple optimal LP solutions), then we get $\max\left\{\bar{z}^1, \bar{z}^2\right\} < \bar{z}$, which means that our upper bound decreases (improves).

In the concrete example we choose variable x_1 where $\tilde{x}_1 = 4.5$ and set $S_1 = S \cap \{x : x_1 \leq 4\}$ and $S_2 = S \cap \{x : x_1 \geq 5\}$.

Choosing an active node The list of active problems (nodes) to be examined is now S_1, S_2. Suppose we choose S_2 to be explored. Again, the first step is to obtain an upper bound by solving the LP-relaxation:

$$\begin{aligned}
&(8.2a) && z = \max 4x_1 - x_2 \\
&(8.2b) && 3x_1 - 2x_2 + x_3 = 14 \\
&(8.2c) && x_2 + x_4 = 3 \\
&(8.2d) && 2x_1 - 2x_2 + x_5 = 3 \\
&(8.2e) && x_1 \geq 5 \\
&(8.2f) && x \geq 0
\end{aligned}$$

It turns out that (8.2) is infeasible, so S_2 can be pruned by infeasibility (see Figure 8.6(b)).

Choosing an active node The only active node at the moment is S_1. So, we choose S_1 to be explored. We obtain an upper bound $\bar{z}^1$ by solving the LP-relaxation

$$\begin{aligned}
&(8.3a) && z = \max 4x_1 - x_2 \\
&(8.3b) && 3x_1 - 2x_2 + x_3 = 14 \\
&(8.3c) && x_2 + x_4 = 3 \\
&(8.3d) && 2x_1 - 2x_2 + x_5 = 3 \\
&(8.3e) && x_1 \leq 4 \\
&(8.3f) && x \geq 0.
\end{aligned}$$

An optimal solution for (8.3) is $\tilde{x}^2 = (4, 2.5, 7, 4, 0)$ with objective function value 13.5. This allows us to update our bounds as shown in Figure 8.6(c).

This time we choose to branch on variable x_2, since $\tilde{x}_2^2 = 2.5$. The two subproblems are defined on the sets $S_{11} = S_1 \cap \{x : x_2 \leq 2\}$ and $S_{12} = S_1 \cap \{x : x_2 \geq 3\}$, see Figure 8.6(d).

Choosing an active node The list of active nodes is S_{12} and S_{22}. We choose S_{22} and solve the LP-relaxation

$$\begin{align*}
z = \max 4x_1 - x_2 \tag{8.4a}\\
3x_1 - 2x_2 + x_3 = 14 \tag{8.4b}\\
x_2 + x_4 = 3 \tag{8.4c}\\
2x_1 - 2x_2 + x_5 = 3 \tag{8.4d}\\
x_1 \leq 4 \tag{8.4e}\\
x_2 \geq 3 \tag{8.4f}\\
x \geq 0. \tag{8.4g}
\end{align*}$$

An optimal solution of (8.4) is $\tilde{x}^{12} = (4, 3, 8, 1, 0)$ with objective function value 13. As the solution is integer, we can prune S_{12} by optimality. Moreover, we can also update our bounds as shown in Figure 8.6(e).

Choosing an active node At the moment we only have one active node: S_{11}. The corresponding LP-relaxation is

$$\begin{align*}
z = \max 4x_1 - x_2 \tag{8.5a}\\
3x_1 - 2x_2 + x_3 = 14 \tag{8.5b}\\
x_2 + x_4 = 3 \tag{8.5c}\\
2x_1 - 2x_2 + x_5 = 3 \tag{8.5d}\\
x_1 \leq 4 \tag{8.5e}\\
x_2 \leq 2 \tag{8.5f}\\
x \geq 0, \tag{8.5g}
\end{align*}$$

which has $\tilde{x}^{11} = (3.5, 2, 7.5, 1, 0)$ as an optimal solution. Hence, $\bar{z}^{11} = 12$ which is strictly smaller than $\underline{z} = 13$. This means that S_{11} can be pruned by bound.

8.4 Techniques for LP-based Branch and Bound

The previous section contained an example for an execution of an LP-based branch and bound algorithm. While the basic outline of such an algorithm should be clear, there are a few issues that need to be taken into account in order to obtain the best possible efficiency.

8.4.1 Reoptimization

Consider the situation in the example from the previous section when we wanted to explore node S_1. We had already solved the initial LP for S. The only difference between LP(S) and LP(S_1) is the presence of the constraint $x_1 \leq 4$ in LP(S_1). How can we solve LP(S_1) without starting from scratch?

The problem LP(S) is of the form $\max \left\{c^T x : Ax = b, x \geq 0\right\}$. An optimal basis for this LP is $B = \{x_1, x_2, x_3\}$ with solution $x_B^* = (4.5, 3, 6.5)$ and $x_N^* = (0, 0)$. We can parametrize any solution $x = (x_B, x_N)$ for LP(S) by the nonbasic variables:

$$x_B := A_B^{-1}b - A_B^{-1}A_N x_N = x_B^* - A_B^{-1}A_N x_N$$

Let $\bar{A}_N := A_B^{-1} A_N$, $\bar{c}_N^T := c_B^T A_B^{-1} A_N - c_N$ and $\bar{b} := A_B^{-1} b$. Multiplying the constraints $Ax = b$ by A_B^{-1} gives the optimal basis representation of the LP:

$$\max c_B^T x_B^* - \bar{c}_N^T x_N \tag{8.6a}$$
$$x_B + \bar{A}_N x_N = \bar{b} \tag{8.6b}$$
$$x \geq 0 \tag{8.6c}$$

Recall that a basis B is called *dual feasible*, if $\bar{c}_N \geq 0$. The optimal primal basis is also dual feasible. We refer to [Sch86, NW99] for details on Linear Programming.

In the branch and bound scheme, we add a new constraint $x_i \leq t$ (or $x_i \geq t$) for some basic variable $x_i \in B$. Let us make this constraint an equality constraint by adding a slack variable $s \geq 0$. The new constraints are $x_i + s = t$, $s \geq 0$. By (8.6) we can express x_i in terms of the nonbasic variables: $x_i = (\bar{b} - \bar{A}_N x_N)_i$. Hence $x_i + s = t$ becomes

$$(-\bar{A}_N x_N)_i + s = t - \bar{b}_i.$$

Adding this constraint to (8.6) gives us the following new LP that we must solve:

$$\max c_B^T x_B^* - \bar{c}_N^T x_N \tag{8.7a}$$
$$x_B + \bar{A}_N x_N = \bar{b} \tag{8.7b}$$
$$(-\bar{A}_N x_N)_i + s = t - \bar{b}_i \tag{8.7c}$$
$$x \geq 0, s \geq 0 \tag{8.7d}$$

The set $B \cup \{s\}$ forms a dual feasible basis for (8.7), so we can start to solve (8.7) by the dual Simplex method starting with the situation as given.

Let us illustrate the method for the concrete sitation of our example. We have

$$A_B = \begin{pmatrix} 3 & -2 & 1 \\ 0 & 1 & 0 \\ 2 & -2 & 0 \end{pmatrix}$$
$$A_B^{-1} = \begin{pmatrix} 0 & 1 & 1/2 \\ 0 & 1 & 0 \\ 1 & -1 & -3/2 \end{pmatrix}$$

and thus (8.6) amounts to

$$\begin{array}{llllll} \max 15 - 3x_4 - 2x_5 & & & & & \\ x_1 & & & +x_4 & +\frac{1}{2}x_5 & = \frac{9}{2} \\ & x_2 & & +x_4 & & = 3 \\ & & x_3 & -x_4 & -\frac{3}{2}x_5 & = \frac{13}{2} \\ x_1, x_2, x_3, x_4, x_5 \geq 0 & & & & & \end{array}$$

If we add a slack variable s, then the new constraint $x_1 \leq 4$ becomes $x_1 + s = 4$, $s \geq 0$. Since $x_1 = \frac{9}{2} - x_4 - \frac{1}{2}x_5$, we can rewrite this constraint in terms of the nonbasic variables as:

$$-x_4 - \frac{1}{2}x_5 + s = -\frac{1}{2}. \tag{8.8}$$

This gives us the dual feasible representation of $LP(S_1)$:

$$
\begin{array}{lllllcl}
\max 15 - 3x_4 - 2x_5 & & & & & & \\
x_1 & & & +x_4 & +\frac{1}{2}x_5 & & = & \frac{9}{2} \\
& x_2 & & +x_4 & & & = & 3 \\
& & x_3 & -x_4 & -\frac{3}{2}x_5 & & = & \frac{13}{2} \\
& & & -x_4 & -\frac{1}{2}x_5 & +s & = & -\frac{1}{2} \\
x_1, x_2, x_3, x_4, x_5, s \geq 0 & & & & & & &
\end{array}
$$

We can now do dual Simplex pivots to find the optimal solution of $LP(S_1)$.

8.4.2 Other Issues

Storing the tree In practice we do not need to store the whole branch and bound tree. It suffices to store the active nodes.

Bounding Upper bounds are obtained by means of the LP-relaxations. Cutting-planes (see the following chapter) can be used to strengthen bounds. Lower bounds can be obtained by means of heuristics and approximation algorithms.

Branching In our example we branched on a variable that was fractional in the optimal solution for the LP-relaxation. In general, there will be more than one such variable. A choice that has proved to be efficient in practice is to branch on the *most fractional variable*, that is, to choose a variable which maximizes $\max_j \min\{f_j, 1 - f_j\}$, where $f_j = x_j - \lfloor x_j \rfloor$. There are other rules based on estimating the cost of a variable to become integer. We refer to [NW99] for more details.

Choosing an active node In our example we just chose an arbitrary active node to be explored. In practice there are two antagonistic arguments how to choose an active node:

- The tree can only be pruned significantly if there is a good primal feasible solution available which gives a good lower bound. This suggests to apply a *depth-first search strategy*. Doing so, in each step a single new constraint is added and we can reoptimize easily as shown in Section 8.4.1.
- On the other hand, one would like to minimize the total number of nodes evaluated in the tree. This suggests to choose an active node with the best upper bound. Such a strategy is known as *best-first search*.

In practice, one usually employs a mixture of depth-first search and best-first search. After an initial depth-first search which leads to a feasible solution one switches to a combined best-first and depth-first strategy.

Preprocessing An important technique for obtaining efficient algorithms is to use preprocessing which includes

- tightening of bounds (e.g. by cutting-planes, see the following chapter)
- removing of redundant variables
- variable fixing

We demonstrate preprocessing techniques for a small example. Consider the LP

$$\begin{aligned} \max \quad & 2x_1 + x_2 - x_3 \\ & 5x_1 - 2x_2 + 8x_3 \leq 15 \\ & 8x_1 + 3x_2 - x_3 \geq 9 \\ & x_1 + x_2 + x_3 \leq 6 \\ & 0 \leq x_1 \leq 3 \\ & 0 \leq x_2 \leq 1 \\ & 1 \leq x_3 \end{aligned}$$

Tightening of constraints

Suppose we take the first constraint and isolate variable x_1. Then we get

$$\begin{aligned} 5x_1 &\leq 15 + 2x_2 - 8x_3 \\ &\leq 15 + 2 \cdot 1 - 8 \cdot 1 \\ &= 9, \end{aligned}$$

where we have used the bounds on the variables x_2 and x_3. This gives us the bound

$$x_1 \leq \frac{9}{5}.$$

Similarly, taking the first constraint and isolating variable x_3 results in:

$$\begin{aligned} 8x_3 &\leq 15 + 2x_2 - 5x_1 \\ &\leq 15 + 2 \cdot 1 - 5 \cdot 0 \\ &= 17. \end{aligned}$$

Our new bound for x_3 is:

$$x_3 \leq \frac{17}{8}.$$

By similar operations we get the new bound

$$x_1 \geq \frac{7}{8}$$

and now using all the new bounds for x_3 in the first inequality gives:

$$x_3 \leq \frac{101}{64}.$$

Plugging the new bounds into the third constraint gives:

$$x_1 + x_2 + x_3 \leq \frac{9}{5} + 1 + \frac{101}{64} < 6.$$

So, the third constraint is superfluous and can be dropped. Our LP has reduced to the following problem:

$$\begin{aligned} \max \quad & 2x_1 + x_2 - x_3 \\ & 5x_1 - 2x_2 + 8x_3 \leq 15 \\ & 8x_1 + 3x_2 - x_3 \geq 9 \\ & \frac{7}{8} \leq x_1 \leq \frac{9}{5} \\ & 0 \leq x_2 \leq 1 \\ & 1 \leq x_3 \leq \frac{101}{64} \end{aligned}$$

Variable Fixing

Consider variable x_2. Increasing variable x_2 makes all constraints less tight. Since x_2 has a positive coefficient in the objective function it will be as large as possible in an optimal solution, that is, it will be equal to its upper bound of 1:

$$x_2 = 1.$$

The same conclusion could be obtained by considering the LP dual. One can see that the dual variable corresponding to the constraint $x_2 \leq 1$ must be positive in order to achieve feasibility. By complementary slackness this means that $x_2 = 1$ in any optimal solution.

Decreasing the value of x_3 makes all constraints less tight, too. The coefficient of x_3 in the objective is negative, so x_3 can be set to its lower bound.

After all the preprocessing, our initial problem has reduced to the following simple LP:

$$\max 2x_1$$
$$\frac{7}{8} \leq x_1 \leq \frac{9}{5}$$

We formalize the ideas from our example above:

Observation 8.6 *Consider the set*

$$S = \left\{ x : a_0 x_0 + \sum_{j=1}^{n} a_j x_j \leq b, l_j \leq x_j \leq u_j, \textit{ for } j = 1, \dots, n \right\}.$$

The following statements hold:

Bounds on variables *If* $a_0 > 0$*, then*

$$x_0 \leq \left(b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j \right) / a_0.$$

and, if $a_0 < 0$*, then*

$$x_0 \geq \left(b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j \right) / a_0.$$

Redundancy *The constraint* $a_0 x_0 + \sum_{j=1}^{n} a_j x_j \leq b$ *is redundant, if*

$$\sum_{j:a_j>0} a_j u_j + \sum_{j:a_j<0} a_j l_j \leq b.$$

Infeasibility *The set* S *is empty, if*

$$\sum_{j:a_j>0} a_j l_j + \sum_{j:a_j<0} a_j u_j > b.$$

Variable Fixing *For a maximization problem of the form* $\max \left\{ c^T x : Ax \leq b, l \leq x \leq u \right\}$, *if* $a_{ij} \geq 0$ *for all* $i = 1, \dots, m$ *and* $c_j < 0$*, then* $x_j = l_j$ *in an optimal solution. Conversely, if* $a_{ij} \leq 0$ *for all* $i = 1, \dots, m$ *and* $c_j > 0$*, then* $x_j = u_j$ *in an optimal solution.*

For integer programming problems, the preprocessing can go further. If $x_j \in \mathbb{Z}$ and the bounds $l_j \leq x_j \leq u_j$ are not integer, then we can tighten them to

$$\lceil l_j \rceil x_j \leq \lfloor u_j \rfloor.$$

We will explore this fact in greater depth in the following chapter on cutting-planes.

9

Cutting Planes

Let $P = \left\{x \in \mathbb{R}^n_+ : Ax \leq b\right\}$ be a ratioinal polyhedron and $P_I = \operatorname{conv}(P \cap \mathbb{Z}^n)$. We have seen in Section 3.6 that P_I is a rational polyhedron and that we can solve the integer program

$$\max\left\{c^T x : x \in P \cap \mathbb{Z}^n\right\} \tag{9.1}$$

by solving the Linear Program

$$\max\left\{c^T x : x \in P_I\right\}. \tag{9.2}$$

In this chapter we will be concerned with the question how to find a linear description of P_I (or an adequate superset of P_I) which enables us to solve (9.2) and (9.1).

Recall that by Theorem 3.45 on page 35 in order to describe a polyhedron we need exactly its facets.

9.1 Cutting-Plane Proofs

Suppose that we are about to solve an integer program

$$\max\left\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\right\}.$$

Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ and $X = P \cap \mathbb{Z}^n$. If we want to establish optimality of a solution (or at least provide an upper bound) this task is equivalent to proving that $c^T x \leq t$ is valid for all points in X. Without the integrality constraints we could prove the validity of the inequality by means of a variant of Farkas' Lemma (cf. Theorem 2.10):

Lemma 9.1 (Farkas' Lemma (Variant)) *Let* $P = \{x \in \mathbb{R}^n : Ax \leq b\} \neq \emptyset$*. The following statements are equivalent:*

(i) The inequality $c^T x \leq t$ *is valid for* P.

(iii) There exists $y \geq 0$ *such that* $A^T y = c$ *and* $b^T y \leq t$.

Proof: By Linear Programming duality, $\max\left\{c^T x : x \in P\right\} \leq t$ if and only if $\min\left\{b^T y : A^T y = c, y \geq 0\right\} \leq t$. So, $c^T x \leq t$ is valid for $P \neq \emptyset$ if and only if there exists $y \geq 0$ such that $A^T y = c$ and $b^T y \leq t$. □

As a consequence of the above lemma, if an inequality $c^T x \leq t$ is valid for $P = \{x : Ax \leq b\}$, then we can derive the validity of the inequality. Namely, we can find $y \geq 0$ such that for $c = A^T y$ and $t' = y^T b$ the inequality $c^T x \leq t'$ is valid for P and $t' \leq t$. This clearly implies that $c^T x \leq t$ is valid.

How can we prove validity in the presence of integrality constraints? Let us start with an example. Consider the following linear system

$$2x_1 + 3x_2 \leq 27 \tag{9.3a}$$
$$2x_1 - 2x_2 \leq 7 \tag{9.3b}$$
$$-6x_1 - 2x_2 \leq -9 \tag{9.3c}$$
$$-2x_1 - 6x_2 \leq -11 \tag{9.3d}$$
$$-6x_1 + 8x_2 \leq 21 \tag{9.3e}$$

Figure 9.1 shows the polytope P defined by the inequalities in (9.3) together with the convex hull of the points from $X = P \cap \mathbb{Z}^2$.

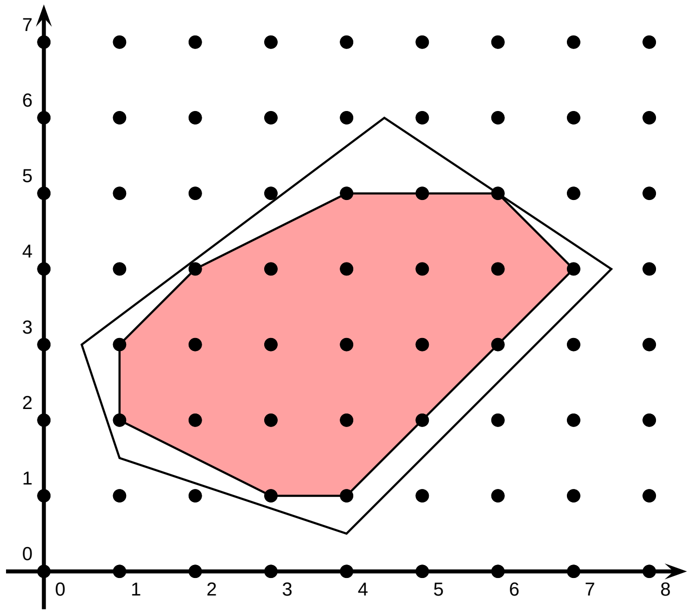


Figure 9.1: Example of a polytope and its integer hull.

As can be seen from Figure 9.1, the inequality $x_2 \leq 5$ is valid for $X = P \cap \mathbb{Z}^2$. However, we can not use Farkas' Lemma to prove this fact from the linear system (9.3), since the point $(9/2, 6) \in P$ has second coordinate 6.

Suppose we multiply the last inequality (9.3e) of the system (9.3) by 1/2. This gives us the valid inequality

$$-3x_1 + 4x_2 \leq 21/2. \tag{9.4}$$

For any integral vector $(x_1, x_2) \in X$ the left hand side of (9.4) will be integral, so we can round down the right hand side of (9.4) to obtain the valid inequality (for X):

$$-3x_1 + 4x_2 \leq 10. \tag{9.5}$$

We now multiply the first inequality (9.3a) by 3 and (9.5) by 2 and add those inequalities. This gives us a new valid inequality:

$$17x_2 \leq 101.$$

Dividing this inequality by 17 and rounding down the resulting right hand side gives us the valid inequality $x_2 \leq 5$.

The procedure used in the example above is called a *cutting plane proof*. Suppose that our system $Ax \leq b$ is formed by the inequalities

$$a_i^T x \leq b_i, \quad i = 1, \dots, m \tag{9.6}$$

and let $P = \{x : Ax \leq b\}$. Let $y \in \mathbb{R}^m_+$ and set

$$c := (A^T y) = \sum_{i=1}^{m} y_i a_i$$

$$t := b^T y = \sum_{i=1}^{m} y_i b_i.$$

As we have already seen, every point in P satisfies $c^T x \leq t$. But we can say more. If c is integral, then for every integral vector in P the quantity $c^T x$ is integral, so it satisfies the stronger inequality

$$c^T x \leq \lfloor t \rfloor. \tag{9.7}$$

The inequality (9.7) is called a *Gomory-ChvÃ¡tal cutting plane*. The term "cutting plane" stems from the fact that (9.7) cuts off part of the polyhedron P but not any of the integral vectors in P.

Definition 9.2 (Cutting-Plane Proof)
Let $Ax \leq b$ be a system of linear inequalities and $c^T x \leq t$ be an inequality. A sequence of linear inequalities

$$c_1^T x \leq t_1, c_2^T x \leq t_2, \dots, c_k^T x \leq t_k$$

*is called a **cutting-plane proof of** $c^T x \leq t$ (from $Ax \leq b$), if each of the vectors $c_1, \dots, c_k$ is integral, $c_k = c$, $t_k = t$, and if for each $i = 1, \dots, k$ the following statement holds: $c_i^T x \leq t_i'$ is a nonnegative linear combination of the inequalities $Ax \leq b$, $c_1^T x \leq t_1, \dots, c_{i-1}^T x \leq t_{i-1}$ for some t_i' with $\lfloor t_i' \rfloor \leq t_i$.*

Clearly, if $c^T x \leq t$ has a cutting-plane proof from $Ax \leq b$, then $c^T x \leq t$ is valid for each integral solution of $Ax \leq b$. Moreover, a cutting plane proof is a clean way to show that the inequality $c^T x \leq t$ is valid for all integral vectors in a polyhedron.

Example 9.3 (Matching Polytope)
The matching polytope $M(G)$ of a graph $G = (V, E)$ is defined as the convex hull of all incidence vectors of matchings in G. It is equal to the set of solutions of

$$x(\delta(v)) \leq 1 \quad \text{for all } v \in V$$
$$x \in \mathbb{B}^E$$

Alternatively, if we let P denote the polytope obtained by replacing $x \in \mathbb{B}^E$ by $0 \leq x$, then $M(G) = P_I$.

Let $T \subseteq V$ be a set of nodes of odd cardinality. As the edges of a matching do not share an endpoint, the number of edges of a matching having both endpoints in T is at most $\frac{|T|-1}{2}$. Thus,

$$x(\gamma(T)) \leq \frac{|T|-1}{2} \tag{9.8}$$

is a valid inequality for $M(G) = P_I$. Here, $\gamma(T)$ denotes the set of edges which have both endpoints in T. We now give a cutting-plane proof of (9.8).

For $v \in T$ take the inequality $x(\delta(v)) \leq 1$ with weight $1/2$ and sum up the resulting $|T|$ inequalities. This gives:

$$x(\gamma(E)) + \frac{1}{2}x(\delta(E)) \leq \frac{|T|}{2}. \tag{9.9}$$

For each $e \in \delta(E)$ we take the inequality $-x_e \leq 0$ with weight $1/2$ and add it to (9.9). This gives us:

$$x(\gamma(E)) \leq \frac{|T|}{2}. \tag{9.10}$$

Rounding down the right hand side of (9.10) yields the desired result (9.8). $\lhd$

In the sequel we are going to show that cutting-plane proofs are always possible, provided P is a polytope.

Theorem 9.4 *Let* $P = \{x : Ax \leq b\}$ *be a rational polytope and let* $c^T x \leq t$ *be a valid inequality for* $X = P \cap \mathbb{Z}^n$, *where* c *is integral. Then, there exists a cutting-plane proof of* $c^T x \leq t'$ *from* $Ax \leq b$ *for some* $t' \leq t$.

We will prove Theorem 9.4 by means of a special case (Theorem 9.6). We need another useful equivalent form of Farkas' Lemma:

Theorem 9.5 (Farkas' Lemma for inequalities) *The sytem* $Ax \leq b$ *has a solution* x *if and only if there is no vector* $y \geq 0$ *such that* $y^T A = 0$ *and* $y^T b < 0$.

Proof: See standard textbooks about Linear Programming, e.g. [Sch86]. $\square$

From this variant of Farkas' Lemma we see that $P = \{x : Ax \leq b\}$ is empty if and only if we can derive a contradiction $0^T x \leq -1$ from the system $Ax \leq b$ by means of taking a nonnegative linear combination of the inequalities. The following theorem gives the analogous statement for integral systems:

Theorem 9.6 *Let* $P = \{x : Ax \leq b\}$ *be a rational polytope and* $X = P \cap \mathbb{Z}^n$ *be empty:* $X = \emptyset$. *Then there exists a cutting-plane proof of* $0^T x \leq -1$ *from* $Ax \leq b$.

Before we embark upon the proofs (with the help of a technical lemma) let us derive another look at Gomory-ChvÃ¡tal cutting-planes. By Farkas' Lemma, we can derive any valid inequality $c^T x \leq t$ (or a stronger version) for a polytope $P = \{x : Ax \leq b\}$ by using a nonnegative linear combination of the inequalities. In view of this fact, we can define Gomory-ChvÃ¡tal cutting-planes also directly in terms of the polyhedron P: we just take a valid inequality $c^T x \leq t$ for P with c integral which induces a nonempty face and round down t to obtain the cutting plane $c^T x \leq \lfloor t \rfloor$.

The proof of Theorems 9.4 and 9.6 is via induction on the dimension of the polytope. The following lemma allows us to translate a cutting-plane proof on a face F to a proof on the entire polytope P.

Lemma 9.7 *Let* $P = \{x : Ax \leq b\}$ *be a rational polytope and* F *be a face of* P*. If* $c^T x \leq \lfloor t \rfloor$ *is a Gomory-ChvÃ¡tal cutting-plane for* F*, then there exists a Gomory-ChvÃ¡tal cutting-plane* $\bar{c}^T x \leq \lfloor \bar{t} \rfloor$ *for* P *such that*

$$F \cap \left\{x : \bar{c}^T x \leq \lfloor \bar{t} \rfloor\right\} = F \cap \left\{x : c^T x \leq \lfloor t \rfloor\right\}. \tag{9.11}$$

Proof: By Theorem 3.6 we can write $P = \{x : A'x \leq b', A''x \leq b'\}$ and $F = \{x : A'x \leq b', A''x = b''\}$, where A'' and b'' are integral. Let $t^* = \max\left\{c^T x : x \in F\right\}$. Since $c^T x \leq t$ is valid for F we must have $t \geq t^*$. So, the following system does not have a solution:

$$\begin{aligned} A'x &\leq b' \\ A''x &\leq b'' \\ -A''x &\leq -b'' \\ c^T x &> t \end{aligned}$$

By the Farkas' Lemma there exist vectors $y' \geq 0$, y'' such that

$$\begin{aligned} (y')^T A' + (y'')^T A'' &= c^T \\ (y')^T b' + (y'')^T b'' &= t. \end{aligned}$$

This looks like a Gomory-ChvÃ¡tal cutting-plane $c^T x \leq \lfloor t^* \rfloor$ for P with the exception that y'' is not necessarily nonnegative. However, the vector $y'' - \lfloor y'' \rfloor$ is nonnegative and it turns out that replacing y'' by this vector will work. Let

$$\begin{aligned} \bar{c}^T &:= (y')^T A' + (y'' - \lfloor y'' \rfloor)^T A'' = c - (\lfloor y'' \rfloor)^T A'' \\ \bar{t} &:= (y')^T b' + (y'' - \lfloor y'' \rfloor)^T b'' = t - (\lfloor y'' \rfloor)^T b''. \end{aligned}$$

Observe that $\bar{c}$ is integral, since c is integral, A'' is integral and $\lfloor y'' \rfloor$ is integral. The inequality $\bar{c}^T x \leq \bar{t}$ is a valid inequality for P, since we have taken a nonnegative linear combination of the constraints. Now, we have

$$\lfloor t \rfloor = \lfloor y' + (\lfloor y'' \rfloor)^T b'' \rfloor = \lfloor \bar{t} \rfloor + (\lfloor y'' \rfloor)^T b'', \tag{9.12}$$

where the last equality follows from the fact that $\lfloor y'' \rfloor$ and b'' are integral. This gives us:

$$\begin{aligned} &F \cap \left\{x : \bar{c}^T x \leq \lfloor \bar{t} \rfloor\right\} \\ =&F \cap \left\{x : \bar{c}^T x \leq \lfloor \bar{t} \rfloor, A''x = b''\right\} \\ =&F \cap \left\{x : \bar{c}^T x \leq \lfloor \bar{t} \rfloor, (\lfloor y'' \rfloor)^T A''x = (\lfloor y'' \rfloor)^T b''\right\} \\ =&F \cap \left\{x : c^T x \leq \lfloor \bar{t} \rfloor + (\lfloor y'' \rfloor)^T b'', (\lfloor y'' \rfloor)^T A''x = (\lfloor y'' \rfloor)^T b''\right\} \\ =&F \cap \left\{x : c^T x \leq \lfloor t \rfloor\right\}. \end{aligned}$$

This completes the proof. □

Proof of Theorem 9.6 We use induction on the dimension of P. If $\dim(P) = 0$, then the claim obviously holds. So, let us assume that $\dim(P) \geq 1$ and that the claim holds for all polytopes of smaller dimension.

Let $c^T x \leq \delta$ with c integral be an inequality which induces a proper face of P. Then, by Farkas' Lemma, we can derive the inequality $c^T x \leq \delta$ from $Ax \leq b$ and $c^T x \leq \lfloor \delta \rfloor$ is a Gomory-ChvÃ¡tal cutting-plane for P. Let

$$\bar{P} := \left\{x \in P : c^T x \leq \lfloor \delta \rfloor\right\}$$

be the polytope obtained from P by applying the Gomory-ChvÃ¡tal cut $c^T x \leq \lfloor \delta \rfloor$.

Case 1: $\bar{P} = \emptyset$:

By Farkas' Lemma we can derive the inequality $0^T x \leq -1$ from the inequality system $Ax \leq b$, $c^T x \leq \lfloor \delta \rfloor$ which defines $\bar{P}$. Since $c^T x \leq \lfloor \delta \rfloor$ was a Gomory-ChvÃ¡tal cutting-plane for P (and thus was derived itself from $Ax \leq b$) this means, we can derive the contradiction from $Ax \leq b$.

Case 2: $\bar{P} \neq \emptyset$:

Define the face F of $\bar{P}$ by

$$F := \left\{x \in \bar{P} : c^T x = \lfloor \delta \rfloor\right\} = \left\{x \in P : c^T x = \lfloor \delta \rfloor\right\}.$$

If δ is integral, then F is a proper face of P, so $\dim(F) < \dim(P)$ in this case. If δ is not integral, then P contains points which do not satisfy $c^T x = \lfloor \delta \rfloor$ and so also in this case we have $\dim(F) < \dim(P)$.

By the induction hypothesis, there is a cutting-plane proof of $0^T x \leq -1$ for F, that is, from the system $Ax \leq b$, $c^T x = \lfloor \delta \rfloor$. By Lemma 9.7 there is a cutting-plane proof from $Ax \leq b$, $c^T x \leq \lfloor \delta \rfloor$ for an inequality $w^T x \leq d$ such that

$$\emptyset = F \cap \left\{x : 0^T x \leq -1\right\} = F \cap \left\{x : w^T x \leq \lfloor d \rfloor\right\}.$$

We have

$$\emptyset = F \cap \left\{x : w^T x \leq \lfloor d \rfloor\right\} = \bar{P} \cap \left\{x : c^T x = \lfloor \delta \rfloor, w^T x \leq \lfloor d \rfloor\right\}. \tag{9.13}$$

Let us restate our result so far: We have shown that there is a cutting plane proof from $Ax \leq b$, $c^T x \leq \lfloor \delta \rfloor$ for an inequality $w^T x \leq d$ which satisfies (9.13).

Thus, the following linear system does not have a solution:

$$Ax \leq b \tag{9.14a}$$
$$c^T x \leq \lfloor \delta \rfloor \tag{9.14b}$$
$$-c^T x \leq -\lfloor \delta \rfloor \tag{9.14c}$$
$$w^T x \leq \lfloor d \rfloor. \tag{9.14d}$$

By Farkas' Lemma for inequalities there exist $y, \lambda_1, \lambda_2, \mu \geq 0$ such that

$$y^T A + \lambda_1 c^T - \lambda_2 c^T + \mu w^T = 0 \tag{9.15a}$$
$$y^T b + \lambda_1 \lfloor \delta \rfloor - \lambda_2 \lfloor \delta \rfloor + \mu \lfloor d \rfloor < 0. \tag{9.15b}$$

If $\lambda_2 = 0$, then (9.15) means that already the system obtained from (9.15) by dropping $c^T x \geq \lfloor \delta \rfloor$ does not have a solution, that is

$$\emptyset = \left\{x : Ax \leq b, c^T x \leq \lfloor \delta \rfloor, w^T x \leq \lfloor d \rfloor\right\}.$$

So, by Farkas' Lemma we can derive $0^T x \leq -1$ from this system which consists completely of Gomory-ChvÃ¡tal cutting-planes for P.

So, it suffices to handle the case that $lambda_2 > 0$. In this case, we can divide both lines in (9.15) by λ_2 and get that there exist $y' \geq 0$, $\lambda' \geq 0$ and $\mu \geq 0$ such that

$$(y')^T A + (\lambda')c^T + (\mu')w^T = c^T \tag{9.16a}$$
$$(y')^T b + (\lambda')\lfloor \delta \rfloor + (\mu')\lfloor d \rfloor = \theta < \lfloor \delta \rfloor. \tag{9.16b}$$

Now, (9.16) states that we can derive an inequality $c^T x \leq \theta$ from $Ax \leq b$, $c^T x \leq \lfloor \delta \rfloor$, $w^T x \leq \lfloor d \rfloor$ with $\theta < \lfloor d \rfloor$. Since all the inequalities in the system were Gomory-ChvÃ¡tal cutting-planes this implies that

$$c^T x \leq \lfloor \delta \rfloor - \tau \quad \text{for some } \tau \in \mathbb{Z}, \tau \geq 1 \tag{9.17}$$

is a Gomory-ChvÃ¡tal cutting-plane for P.

Since P is bounded, the value $z = \min \{c^T x : x \in P\}$ is finite. If we continue as above, starting with $\bar{P} = \{x \in P : c^T x \leq \lfloor \delta \rfloor - \tau\}$, at some point we will obtain a cutting-plane proof of some $c^T x \leq t$ where $t < z$ so that $P \cap \{x : c^T x \leq t\} = \varnothing$. Then, by Farkas' Lemma we will be able to derive $0^T x \leq -1$ from $Ax \leq b$, $c^T x \leq t$. □

Proof of Theorem 9.4 Case 1: $P \cap \mathbb{Z}^n = \varnothing$

By Theorem 9.6 there is a cutting-plane proof of $0^T x \leq -1$ from $Ax \leq b$. Since P is bounded, $\ell := \max \{c^T x : x \in P\}$ is finite. By Farkas' Lemma, we can derive $c^T x \leq \ell$ and thus we have the Gomory-ChvÃ¡tal cutting plane $c^T x \leq \lfloor \ell \rfloor$. Adding an appropriate multiple of $0^T x \leq -1$ to $c^T x \leq \lfloor \ell \rfloor$ gives an inequality $c^T x \leq t'$ for some $t' \leq t$ which yields the required cutting-plane proof.

Case 2: $P \cap \mathbb{Z}^n \neq \varnothing$

Again, let $\ell := \max \{c^T x : x \in P\}$ which is finite, and define $\bar{P} := \{x \in P : c^T x \leq \lfloor \ell \rfloor\}$, that is, $\bar{P}$ is the polytope obtained by applying the Gomory-ChvÃ¡tal cutting-plane $c^T x \leq \lfloor \ell \rfloor$ to P.

If $\lfloor \ell \rfloor \leq t$ we already have a cutting-plane proof of an inequality with the desired properties. So, assume that $\lfloor \ell \rfloor > t$. Consider the face

$$F = \{x \in \bar{P} : c^T x = \lfloor \ell \rfloor\}$$

of $\bar{P}$. Since $c^T x \leq t$ is valid for all integral points in P and by assumption $t < \lfloor \ell \rfloor$, the face F can not contain any integral point. By Theorem 9.6 there is a cutting-plane proof of $0^T x \leq -1$ from $Ax \leq b$, $c^T x = \lfloor \ell \rfloor$. We now use Lemma 9.7 as in the proof of Theorem 9.6. The lemma shows that there exists a cutting plane proof of some inequality $w^T x \leq \lfloor d \rfloor$ from $Ax \leq b$, $c^T x \leq \lfloor \ell \rfloor$ such that $\bar{P} \cap \{x : c^T x = \lfloor \ell \rfloor, w^T x \leq \lfloor d \rfloor\} = \varnothing$.

By using the same arguments as in the proof of Theorem 9.6 it follows that there is a cutting-plane proof of an inequality $c^T x \leq \lfloor \ell \rfloor - \tau$ for some $\tau \in \mathbb{Z}$, $\tau \geq 1$ from $Ax \leq b$. Contiuning this way, we finally get an inequality $c^T x \leq t'$ with $t' \leq t$. □

9.2 A Geometric Approach to Cutting Planes: The ChvÃ¡tal Rank

Let $P = \{x : Ax \leq b\}$ be a rational polyhedron and $P_I := \text{conv}(P \cap \mathbb{Z}^n)$. Suppose we want to find a linear description of P_I. One approach is to add valid inequalities step by step, obtaining tighter and tighter approximations of P_I.

We have already seen that, if $c^T x \leq \delta$ is a valid inequality for P with c integral, then $c^T x \leq \lfloor \delta \rfloor$ is valid for P_I. If $c^T x = \delta$ was a supporting hyperplane of P, that is, $P \cap \{x : c^T x = \delta\}$ is a proper face of P, then $c^T x \leq \lfloor \delta \rfloor$ is a Gomory-ChvÃ¡tal cutting-plane. Otherwise, the inequality $c^T x \leq \delta$ is dominated by that of a supporting hyperplane. Anyway, we have

$$P_I \subseteq \{x \in \mathbb{R}^n : c^T x \leq \lfloor \delta \rfloor\} \tag{9.18}$$

for any valid inequality $c^T x \leq \delta$ for P where c is integral. This suggests to take the intersection of all sets of the form (9.18) as an approximation to P.

Definition 9.8 *Let P be a rational polyhedron. Then, P' is defined as*

$$P' := \bigcap_{\substack{c \text{ is integral} \\ \text{and} \\ c^T x \leq \delta \text{ is valid for } P}} \{x \in \mathbb{R}^n : c^T x \leq \lfloor \delta \rfloor\}. \tag{9.19}$$

Observe that (9.19) is the same as taking the intersection over all Gomory-ChvÃ¡tal cutting-planes for P. It is not a priori clear that P' is a polyhedron, since there is an infinite number of cuts.

Theorem 9.9 *Let P be a rational polyhedron. Then P' as defined in (9.19) is also a rational polyhedron.*

Proof: If $P = \varnothing$ the claim is trivial. So let $P \neq \varnothing$. By Theorem 4.27 there is a TDI-system $Ax \leq b$ with integral A such that $P = \{x : Ax \leq b\}$. We claim that

$$P' = \{x \in \mathbb{R}^n : Ax \leq \lfloor b \rfloor\}. \tag{9.20}$$

From this the claim follows, since the set on the right hand side of (9.20) is a rational polyhedron (A and $\lfloor b \rfloor$ are integral, and there are only finitely many constraints).

Since every row $a_i^T x \leq b_i$ of $Ax \leq b$ is a valid inequality for P it follows that $P' \subseteq \{x \in \mathbb{R}^n : Ax \leq \lfloor b \rfloor\}$. So, it suffices to show that the set on the right hand side of (9.20) is contained in P'.

Let $c^T x = \delta$ be a supporting hyperplane of P with c integral, $P \subseteq \{x : c^T x \leq \delta\}$. By Linear Programming duality we have

$$\delta = \max\{c^T x : x \in P\} = \min\{b^T y : A^T y = c, y \geq 0\}. \tag{9.21}$$

Since the system $Ax \leq b$ is TDI and c is integral, the minimization problem in (9.21) has an optimal solution y^* which is integral.

Let $x \in \{x : Ax \leq \lfloor b \rfloor\}$.

$$\begin{aligned}
c^T x &= (A^T y^*)^T x && \text{(since } y^* \text{ is feasible for the problem in (9.21))} \\
&= (y^*)^T (Ax) \\
&\leq (y^*)^T \lfloor b \rfloor && \text{(since } Ax \leq \lfloor b \rfloor \text{ and } y^* \geq 0) \\
&= \lfloor (y^*)^T \lfloor b \rfloor \rfloor && \text{(since } y^* \text{ and } \lfloor b \rfloor \text{ are integral)} \\
&\leq \lfloor (y^*)^T b \rfloor && \text{(since } \lfloor b \rfloor \leq b \text{ and } y^* \geq 0) \\
&= \lfloor \delta \rfloor && \text{(by the optimality of } y^* \text{ for (9.21)).}
\end{aligned}$$

Thus, we have

$$\{x : Ax \leq \lfloor b \rfloor\} \subseteq \{x : c^T x \leq \lfloor \delta \rfloor\}.$$

Since $c^T x = \delta$ was an arbitrary supporting hyperplane, we get that

$$\{x : Ax \leq \lfloor b \rfloor\} \subseteq \bigcap_{c,\delta} \left\{x : c^T x \leq \lfloor \delta \rfloor\right\} = P'$$

as required. $\square$

We have obtained P' from P by taking all Gomory-ChvÃ¡tal cuts for P as a first wave. Given that P' is a rational polyhedron, we can take as a second wave all Gomory-ChvÃ¡tal cuts for P'. Continuing this procedure gives us better and better approximations of P_I. We let

$$\begin{aligned} P^{(0)} &:= P \\ P^{(i)} &:= \left(P^{(i-1)}\right)' \quad \text{for } i \geq 1. \end{aligned}$$

This gives us a sequence of polyhedra

$$P = P^{(0)} \supset P^{(1)} \supset P^{(2)} \supset \cdots \supset P_I,$$

which are generated by the waves of cuts.

We know that P_I is a rational polyhedron (given that P is one) and by Theorem 9.4 every valid inequality for P_I will be generated by the waves of Gomory-ChvÃ¡tal cuts. Thus, we can restate the result Theorem 9.4 in terms of the polyhedra $P^{(i)}$ as follows:

Theorem 9.10 *Let P be a rational polytope. Then we have $P^{(k)} = P_I$ for some $k \in \mathbb{N}$.* $\square$

Definition 9.11 (ChvÃ¡tal rank)
*Let P be a rational polytope. The **ChvÃ¡tal rank of** P is defined to be the smallest integer k such that $P^{(k)} = P_I$.*

9.3 Cutting-Plane Algorithms

Cutting-plane proofs are usually employed to prove the validity of some classes of inequalities. These valid inequalities can then be used in a cutting-plane algorithm.

Suppose that we want to solve the integer program

$$z^* = \max\left\{c^T x : x \in P \cap \mathbb{Z}^n\right\}$$

and that we know a family $\mathcal{F}$ of valid inequalities for $P_I = \text{conv}(P \cap \mathbb{Z}^n)$. Usually, $\mathcal{F}$ will not contain a complete description of P_I since either such a description is not known or we do not know how to separate over $\mathcal{F}$ efficiently. The general idea of a cutting-plane algorithm is as follows:

- We find an optimal solution x^* for the Linear Program $\max\left\{c^T x : x \in P\right\}$. This can be done by any Linear Programming algorithm (possibly a solver that is available only as a black-bock).
- If x^* is integral, we already have an optimal solution to the IP and we can terminate.

- Otherwise, we search our family (or families) of valid inequalities for inequalities which are violated by x^*, that is, $w^T x^* > d$ where $w^T x \leq d$ is valid for P_I.

- We add the inequalities found to our LP-relaxation and resolve to find a new optimal solution x^{**} of the improved formulation. This procedure is contiued.

- If we are fortunate (or if $\mathcal{F}$ contains a complete description of P_I), we terminate with an optimal integral solution. We say "fortunate", since if $\mathcal{F}$ is not a complete description of P_I, this depends on the objective function and our family $\mathcal{F}$.

- If we are not so lucky, we still have gained something. Namely, we have found a new formulation for our initial problem which is better than the original one (since we have cut off some non-integral points). The formulation obtained upon termination gives an upper bound $\bar{z}$ for the optimal objective function value z^* which is no worse than the initial one (and usually is much better). We can now use $\bar{z}$ in a branch and bound algorithm.

Algorithm 9.1 gives a generic cutting-plane algorithm along the lines of the above discussion. The technique of using improved upper bounds from a cutting-plane algorithm in a branch and bound system is usually referred to as *branch-and-cut* (cf. the comments about preprocessing in Section 8.4.2).

Algorithm 9.1 Generic cutting-plane algorithm

GENERIC-CUTTING-PLANE
Input: An integer program $\max \{c^T x : x \in P, x \in \mathbb{Z}^n\}$; a family $\mathcal{F}$ of valid inequalities for $P_I = \text{conv}(P \cap \mathbb{Z}^n)$

repeat
  Solve the Linear Program $\max \{c^T x : x \in P\}$. Let x^* be an optimal solution.
  if x^* is integral **then**
    An optimal solution to the integer program has been found. **stop**.
  else
    Solve the separation problem for $\mathcal{F}$, that is, try to find an inequality $w^T x \leq d$ in $\mathcal{F}$ such that $w^T x^* > d$.
    if such an inequality $w^T x \leq d$ cutting off x^* was found **then**
      Add the inequality to the system, that is, set $P := P \cap \{x : w^T x \leq d\}$.
    else
      We do not have an optimal solution yet. However, we have a better formulation for the original problem. **stop**.
    end if
  end if
until forever

It is clear that the efficiency of a cutting-plane algorithm depends on the availability of constraints that give good upper bounds. In view of Theorem 3.45 the only inequalities (or cuts) we need are those that induce facets. Thus, one is usually interested in finding (by means of mathematical methods) as many facet-inducing inequalities as possible.

9.4 Gomory's Cutting-Plane Algorithm

In this section we assume that the integer program which we want to solve is given in the following form:

$$\max c^T x \tag{9.22a}$$
$$Ax = b \tag{9.22b}$$
$$x \geq 0 \tag{9.22c}$$
$$x \in \mathbb{Z}^n \tag{9.22d}$$

where A is an integral $m \times n$-matrix and b is an integral vector in $\mathbb{Z}^m$. As usual we let $P = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}$ and $P_I = \text{conv}(P \cap \mathbb{Z}^n)$.

Any integer program with rational data can be brought into this form by elementary transformations (see textbooks about Linear Programming [Sch86, Lue84, NW99] where those methods are used for Linear Programs): if x_j is not sign restricted, we replace x_j by two new variables $x_j = x_j^+ - x_j^-$ where $x_j^+, x_j^- \geq 0$. Any inequality $a_i^T x \leq b_i$ can be transformed into an equality by introducing a slack variable $s \geq 0$ which yields $a_i^T x + s = b_i$. Since A and b are integral, the new slack variable will also be an integral variable.

Suppose that we solve the LP-relaxation

$$\max \quad c^T x \tag{9.23a}$$
$$Ax = b \tag{9.23b}$$
$$x \geq 0 \tag{9.23c}$$

of (9.22) by means of the Simplex method.[1]

Recall that a *basis* for (9.23) is an index set $B \subseteq 1, \dots, n$ with $|B| = m$ such that the corresponding submatrix A_B of A is nonsingular. The basis is termed *feasible* if $x_B := A_B^{-1} b \geq 0$. Clearly, in this case (x_B, x_N) with $x_N := 0$ is a feasible solution of (9.23). It is a well known fact that (9.23) has an optimal solution if and only if there is an optimal basic solution [Lue84, CC$^+$98, Sch86].

Suppose that we are given an optimal basis B and a corresponding optimal basic solution x^* for (9.23). As in Section 8.4.1 we can parametrize x^* by the nonbasic variables:

$$x_B^* = A_B^{-1} b - A_B^{-1} A_N x_N^* =: \bar{b} - \bar{A}_N x_N^* x_N^* \tag{9.24}$$
$$x_N^* = 0. \tag{9.25}$$

This gives the equivalent statement of the problem (9.23) in the *basis representation*:

$$\max c_B^T x_B^* - \bar{c}_N^T x_N \tag{9.26a}$$
$$x_B + \bar{A}_N x_N = \bar{b} \tag{9.26b}$$
$$x \geq 0 \tag{9.26c}$$

If x^* is integral, then x^* is an optimal solution for our integer program (9.22). Otherwise, there is a basic variable x_i^* which has a fractional value, that is, $x_i^* = \bar{b}_i \notin \mathbb{Z}$. We will now use the equation in (9.24) which defines x_i^* to derive a valid inequality for P_I. We will then show that the inequality derived is in fact a Gomory-ChvÃ¡tal cutting-plane.

[1] Basically, one could also use an interior point method. The key point is that in the sequel we need an optimal basis.

Let $\bar{A} = (\bar{a}_{lk})$. Any feasible solution x of the integer program (9.22) satisfies (9.24). So, we have

$$
\begin{aligned}
&(9.27) \qquad & x_i = \quad & \bar{b}_i & - \sum_{j \in N} \bar{a}_{ij} x_j &\in \mathbb{Z} \\
&(9.28) \qquad & & -\lfloor \bar{b}_i \rfloor & &\in \mathbb{Z} \\
&(9.29) \qquad & & & \sum_{j \in N} \lfloor \bar{a}_{ij} \rfloor x_j &\in \mathbb{Z}.
\end{aligned}
$$

Adding (9.27), (9.28) and (9.29) results in:

$$
(9.30) \qquad \underbrace{(\bar{b}_i - \lfloor \bar{b}_i \rfloor)}_{\in (0,1)} - \underbrace{\sum_{j \in N} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j}_{\geq 0} \in \mathbb{Z}.
$$

Since $0 < (\bar{b}_i - \lfloor \bar{b}_i \rfloor) < 1$ and $\sum_{j \in N} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \geq 0$, the value on the left hand side of (9.30) can only be integral, if it is nonpositive, that is, we must have

$$
(\bar{b}_i - \lfloor \bar{b}_i \rfloor) - \sum_{j \in N} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \leq 0
$$

for every $x \in P_I$. Thus, the following inequality is valid for P_I:

$$
(9.31) \qquad \sum_{j \in N} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \geq (\bar{b}_i - \lfloor \bar{b}_i \rfloor).
$$

Moreover, the inequality (9.31) is violated by the current basic solution x^*, since $x^*_N = 0$ (which means that the left hand side of (9.31) is zero) and $x^*_i = \bar{b}_i \notin Z$, so that $(\bar{b}_i - \lfloor \bar{b}_i \rfloor) = (x^*_i - \lfloor x^*_i \rfloor) > 0$.

As promised, we are now going to show that (9.31) is in fact a Gomory-ChvÃ¡tal cutting-plane. By (9.24) the inequality

$$
(9.32) \qquad x_i + \sum_{j \in N} \bar{a}_{ij} x_j \leq \bar{b}_i
$$

is valid for P. Since $P \subseteq \mathbb{R}^n_+$, we have $\sum_{j \in N} \lfloor \bar{a}_{ij} \rfloor x_j \leq \sum_{j \in N} \bar{a}_{ij} x_j$ and the inequality

$$
(9.33) \qquad x_i + \sum_{j \in N} \lfloor \bar{a}_{ij} \rfloor x_j \leq \bar{b}_i
$$

must also be valid for P. In fact, since the basic solution x^* for the basis B satisfies (9.32) and (9.33) with equality, the inequalities (9.32) and (9.33) both induce supporting hyperplanes. Observe that all coefficients in (9.33) are integral. Thus,

$$
(9.34) \qquad x_i + \sum_{j \in N} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor \bar{b}_i \rfloor,
$$

is a Gomory-ChvÃ¡tal cutting-plane. Â We can now use (9.24) to rewrite (9.34), that is, to eliminate x_i (this corresponds to taking a nonnegative linear combination of (9.34) and the appropriate inequality stemming from the equality (9.24)). This yields (9.31), so (9.31) is (a scalar multiple of) a Gomory-ChvÃ¡tal cutting-plane. It is important to notice that the difference between the left-hand side and the right-hand side of the Gomory-ChvÃ¡tal cutting-plane (9.34), hence also of (9.31) is integral, when x is integral. Thus, if (9.31) is rewritten using a slack variable $s \geq 0$ as

$$\sum_{j \in N} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j - s = (\bar{b}_i - \lfloor \bar{b}_i \rfloor), \tag{9.35}$$

then this slack variable s will also be a nonnegative integer variable. Gomory's cutting plane algorithm is summarized in Algorithm 9.2.

Algorithm 9.2 Gomory's cutting-plane algorithm

GOMORY-CUTTING-PLANE

Input: An integer program $\max \left\{ c^T x : Ax = b, x \geq 0, x \in \mathbb{Z}^n \right\}$

repeat
  Solve the current LP-relaxation $\max \left\{ c^T x : Ax = b, x \geq 0 \right\}$. Let x^* be an optimal basic solution.
  if x^* is integral **then**
    An optimal solution to the integer program has been found. **stop**.
  else
    Choose one of the basis integer variables which is fractional in the optimal LP-solution, say $x_i = \bar{b}_i$. This variable is parametrized as follows:

$$x_i = \bar{b}_i - \sum_{j \in N} \bar{a}_{ij} x_j$$

    Generate the Gomory-ChvÃ¡tal cut

$$\sum_{j \in N} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \geq (\bar{b}_i - \lfloor \bar{b}_i \rfloor)$$

    and add it to the LP-formulation by means of a new nonnegative integer slack variable s:

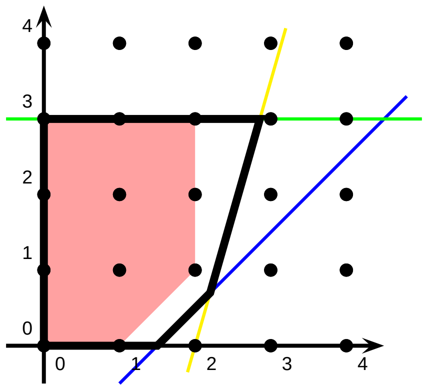
$$\begin{aligned} \sum_{j \in N} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j - s &= (\bar{b}_i - \lfloor \bar{b}_i \rfloor) \\ s &\geq 0 \\ s &\in \mathbb{Z} \end{aligned}$$

  end if
until forever

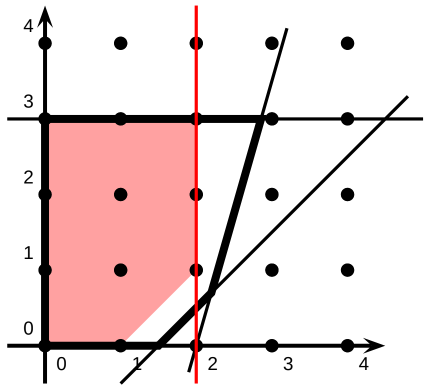
Example 9.12
We consider the following integer program:

$$\begin{array}{llrcl} \max & 4x_1 & -x_2 & & \\ & 7x_1 & -2x_2 & \leq & 14 \\ & & x_2 & \leq & 3 \\ & 2x_1 & -2x_2 & \leq & 3 \\ & \multicolumn{2}{c}{x_1, x_2 \geq 0} & & \\ & \multicolumn{2}{c}{x_1, x_2 \in \mathbb{Z}} & & \end{array}$$

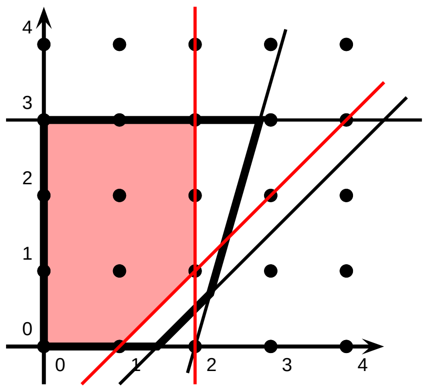
The feasible points and the polyhedron P described by the inequalities above are depicted in Figure 9.2 (a).



(a) Inequalities leading to the polytope shown with thick lines together with the convex hull of the integral points.



(b) The first cut produced by Gomory's algorithm is $x_1 \leq 2$ (shown in red)



(c) The next cut produced is $x_1 - x_2 \leq 1$ (shown in red). After this cut the algorithm terminates with the optimum solution $(2, 1)$.

Figure 9.2: Example problem for Gomory's algorithm. Thick solid lines indicate the polytope described by the inequalities, the pink shaded region is the convex hull of the integral points (red) which are feasible.

Adding slack variables gives the following LP-problem:

$$\begin{array}{llllllll}
\max & 4x_1 & -x_2 & & & & & \\
 & 7x_1 & -2x_2 & +x_3 & & & = & 14 \\
 & & x_2 & & +x_4 & & = & 3 \\
 & 2x_1 & -2x_2 & & & +x_5 & = & 3 \\
 & & \multicolumn{6}{l}{x_1, x_2, x_3, x_4, x_5 \geq 0} \\
 & & \multicolumn{6}{l}{x_1, x_2, x_3, x_4, x_5 \in \mathbb{Z}}
\end{array}$$

An optimal basis for the corresponding LP-relaxation is $B = \{1, 2, 5\}$ with

$$A_B = \begin{pmatrix} 7 & -2 & 0 \\ 0 & 1 & 0 \\ 2 & -2 & 1 \end{pmatrix} \qquad x_B^* = \begin{pmatrix} 20/7 \\ 3 \\ 0 \\ 0 \\ 23/7 \end{pmatrix}$$

The optimal basis representation (9.26) is given by:

$$\begin{array}{lllllllll}
\max 59/7 & & & -4/7x_3 & -1/7x_4 & & & \\
 & x_1 & & +1/7x_3 & +2/7x_4 & & = & 20/7 \\
 & & x_2 & & +x_4 & & = & 3 \\
 & & & -2/7x_3 & +10/7x_4 & +x_5 & = & 23/7
\end{array} \tag{9.36a}$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0 \tag{9.36b}$$

$$x_1, x_2, x_3, x_4, x_5 \in \mathbb{Z} \tag{9.36c}$$

The variable x_1^* is fractional, $x_1^* = 20/7$. From the first line of (9.36) we have:

$$x_1 + \frac{1}{7}x_3 + \frac{2}{7}x_4 = \frac{20}{7}.$$

The Gomory-ChvÃ¡tal cut (9.31) generated for x_1 is

$$\begin{aligned}
&(\frac{1}{7} - \lfloor\frac{1}{7}\rfloor)x_3 + (\frac{2}{7} - \lfloor\frac{2}{7}\rfloor)x_4 \geq (\frac{20}{7} - \lfloor\frac{20}{7}\rfloor) \\
\Leftrightarrow &\frac{1}{7}x_3 + \frac{2}{7}x_4 \geq \frac{6}{7}
\end{aligned}$$

Thus, the following new constraint will be added to the LP-formulation:

$$\frac{1}{7}x_3 + \frac{2}{7}x_4 - s = \frac{6}{7},$$

where $s \geq 0$, $s \in \mathbb{Z}$ is a new integer nonnegative slack variable. Before we continue, let us look at the inequality in terms of the original variables, that is, without the slack variables. We have $x_3 = 14 - 7x_1 + 2x_2$ and $x_4 = 3 - x_2$. Substituting we get the cutting-plane

$$\begin{aligned}
&\frac{1}{7}(14 - 7x_1 + 2x_2) + \frac{2}{7}(3 - x_2) \geq \frac{6}{7} \\
\Leftrightarrow &x_1 \leq 2.
\end{aligned}$$

The cutting-plane $x_1 \leq 2$ is shown in Figure 9.2(b).

Reoptimization of the new LP leads to the following optimal basis representation:

$$\begin{array}{llllllll}
\max 15/2 & & & & & -1/5x_5 & -3s & \\
& x_1 & & & & & +s & = & 2 \\
& & x_2 & & & -1/2x_5 & +s & = & 1/2 \\
& & & x_3 & & -x_5 & -s & = & 1 \\
& & & & x_4 & +1/2x_5 & +6s & = & 5/2
\end{array}$$

$$x_1, x_2, x_3, x_4, x_5, s \geq 0$$
$$x_1, x_2, x_3, x_4, x_5, s \in \mathbb{Z}$$

The optimal solution $x^* = (2, \frac{1}{2}, 1, \frac{5}{2}, 0)$ is still fractional. We choose basic variable x_2 which is fractional to generate a new cut. We have

$$x_2 - \frac{1}{2}x_5 + s = \frac{1}{2},$$

and so the new cut is

$$\begin{aligned}
&(-\frac{1}{2} - \lfloor -\frac{1}{2} \rfloor)x_5 \geq (\frac{1}{2} - \lfloor \frac{1}{2} \rfloor) \\
\Leftrightarrow &\frac{1}{2}x_5 \geq \frac{1}{2}
\end{aligned}$$

(observe that $(-\frac{1}{2} - \lfloor -\frac{1}{2} \rfloor) = \frac{1}{2}$, since $\lfloor -\frac{1}{2} \rfloor = -1$). We introduce a new slack variable $t \geq 0$, $t \in \mathbb{Z}$ and add the following constraint:

$$\frac{1}{2}x_5 - t = \frac{1}{2}.$$

Again, we can translate the new cut $\frac{1}{2}x_5 \geq \frac{1}{2}$ in terms of the original variables. It amounts to

$$\begin{aligned}
&\frac{1}{2}(2x_1 - 2x_2) \geq \frac{1}{2} \\
\Leftrightarrow &x_1 - x_2 \leq 1.
\end{aligned}$$

The new cutting-plane $x_1 - x_2 \leq 1$ is shown in Figure 9.2(c).

After reoptimization we obtain the following situation:

$$\begin{array}{lllllllll}
\max 7 & & & & & & -3s & -t & \\
& x_1 & & & & & +s & & = & 2 \\
& & x_2 & & & & +s & -t & = & 1 \\
& & & x_3 & & & -5s & -2t & = & 2 \\
& & & & x_4 & & +6s & +t & = & 2 \\
& & & & & x_5 & & -t & = & 1
\end{array}$$

$$x_1, x_2, x_3, x_4, x_5, s, t \geq 0$$
$$x_1, x_2, x_3, x_4, x_5, s, t \in \mathbb{Z}$$

The optimal basic solution is integral, thus it is also an optimal solution for the original integer program: $(x_1, x_2) = (2, 1)$ constitutes an optimal solution of our original integer program. $\triangleleft$

One can show that Gomory's cutting-plane algorithm always terminates after a finite number of steps, provided the cuts are chosen appropriately. The proof of the following theorem is beyond the scope of these lecture notes. We refer the reader to [Sch86, NW99].

Theorem 9.13 *Suppose that Gomory's algorithm is implemented in the following way:*

(i) We use the lexicographic Simplex algorithm for solving the LPs.

(ii) We always derive the Gomory-ChvÃ¡tal cut from the first Simplex row in which the basic variable is fractional.

Then, Gomory's cutting-plane algorithm terminates after a finite number of steps with an optimal integer solution. □

9.5 Mixed Integer Cuts

In this section we consider the situation of mixed integer programs

$$\text{(9.37a)} \qquad \text{(MIP)} \quad \max \quad c^T x$$
$$\text{(9.37b)} \qquad Ax = b$$
$$\text{(9.37c)} \qquad x \geq 0$$
$$\text{(9.37d)} \qquad x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}.$$

In this case, the approach taken so far does not work: In a nutshell, the basis of the Gomory-ChvÃ¡tal cuts was the fact that, if $X = \{y \in \mathbb{Z} : y \leq b\}$, then $y \leq \lfloor b \rfloor$ is valid for X. More precisely, we saw that all Gomory-ChvÃ¡tal cuts for $P_I = P \cap \mathbb{Z}^n$ are of the form $c^T x \leq \lfloor \delta \rfloor$ where $c^T x \leq \delta$ is a supporting hyperplane of P with integral c. If x is not required to be integral, we may not round down the right hand side of $c^T x \leq \delta$ to obtain a valid inequality for P_I.

The approach taken in Gomory's cutting-plane algorithm from Section 9.4 does not work either, since for instance

$$\frac{1}{3} + \frac{1}{3}x_1 - 2x_2 \in \mathbb{Z}$$

with $x_1 \in \mathbb{Z}_+$ and $x_2 \in \mathbb{R}_+$ has a larger solution set than

$$\frac{1}{3} + \frac{1}{3}x_1 \in \mathbb{Z}.$$

Thus, we can not derive the validity of (9.31) (since we can not assume that the coefficients of the fractional variables are nonnegative) which forms the basis of Gomory's algorithm.

The key to obtaining cuts for mixed integer programs is the following *disjunctive argument*:

Lemma 9.14 *Let* P_1 *and* P_2 *be polyhedra in* $\mathbb{R}^n_+$ *and* $(a^{(i)})^T x \leq \alpha_i$ *be valid for* P_i, $i = 1, 2$. *Then, for any vector* $c \in \mathbb{R}^n$ *satisfying* $c \leq \min(a^{(1)}, a^{(2)})$ *componentwise and* $\delta \geq \max(\alpha_1, \alpha_2)$ *the inequality*

$$c^T x \leq \delta$$

is valid for $X = P_1 \cup P_2$ *and* $\operatorname{conv}(X)$.

Proof: Let $x \in X$, then $x \in P_1$ or $x \in P_2$. If $x \in P_i$, then

$$c^T x = \sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} a_j^{(i)} x_j \leq \alpha_i \leq \delta,$$

where the first inequality follows from $c \leq a^{(i)}$ and $x \geq 0$. □

Let us go back to the situation in Gomory's algorithm. We solve the LP-relaxation

$$
\begin{aligned}
&(9.38a) && \text{(LP)} \quad \max \quad c^T x \\
&(9.38b) && \qquad\qquad\quad Ax = b \\
&(9.38c) && \qquad\qquad\quad x \geq 0
\end{aligned}
$$

of (9.37) and obtain an optimal basic solution x^*. As in Section 9.4 we parametrize the solutions of (9.38) by means of the nonbasic variables:

$$x_B^* = A_B^{-1} b - A_B^{-1} A_N x_N^* =: \bar{b} - \bar{A}_N x_N^*$$

Let x_i be an integer variable. Then, any feasible solution to the MIP (9.37) satisfies:

$$\bar{b}_i - \sum_{j \in N} \bar{a}_{ij} x_j \in \mathbb{Z}, \tag{9.39}$$

since the quantity on the left hand side of (9.39) is the value of variable x_i. Let $N^+ := \{j \in N : \bar{a}_{ij} \geq 0\}$, $N^- := N \setminus N^+$. Also, for a shorter notation we set $f_0 := (\bar{b}_i - \lfloor \bar{b}_i \rfloor) \in [0, 1)$. Equation (9.39) is equivalent to

$$\sum_{j \in N} \bar{a}_{ij} x_j = f_0 + k \quad \text{for some } k \in \mathbb{Z}. \tag{9.40}$$

If, $k \geq 0$, then the quantity on the left hand side of (9.40) is at least f_0, if $k \leq -1$, then it is at most $f_0 - 1$. Accordingly, we distinguish between two cases:

Case 1: $\sum_{j \in N} \bar{a}_{ij} x_j \geq f_0 \geq 0$: In this case, we get from $\sum_{j \in N^+} \bar{a}_{ij} x_j \geq \sum_{j \in N} \bar{a}_{ij} x_j$ the inequality:

$$\sum_{j \in N^+} \bar{a}_{ij} x_j \geq f_0. \tag{9.41}$$

Case 2: $\sum_{j \in N} \bar{a}_{ij} x_j \leq f_0 - 1 < 0$: Then, $\sum_{j \in N^-} \bar{a}_{ij} x_j \leq \sum_{j \in N} \bar{a}_{ij} x_j \leq f_0 - 1$ which is equivalent to

$$-\frac{f_0}{1 - f_0} \sum_{j \in N^-} \bar{a}_{ij} x_j \geq f_0. \tag{9.42}$$

We split P_I into two parts, $P_I = P_1 \cup P_2$, where

$$
\begin{aligned}
P_1 &:= P_I \cap \left\{ x : \sum_{j \in N} \bar{a}_{ij} x_j \geq 0 \right\} \\
P_2 &:= P_I \cap \left\{ x : \sum_{j \in N} \bar{a}_{ij} x_j < 0 \right\}.
\end{aligned}
$$

Inequality (9.41) is valid for P_1, while inequality (9.42) is valid for P_2. We apply Lemma 9.14 to get an inequality $\pi^T x \geq \pi_0$. If $j \in N^+$, the coefficient for x_j is $\pi_j = \max\{\bar{a}_{ij}, 0\} = \bar{a}_{ij}$. If $j \in N^-$, the coefficient for x_j is $\pi_j = \max\left\{0, -\frac{f_0}{1-f_0} \bar{a}_{ij}\right\} = -\frac{f_0}{1-f_0} \bar{a}_{ij}$. Thus, we obtain the following inequality which is valid for P_I:

$$\sum_{j \in N^+} \bar{a}_{ij} x_j - \frac{f_0}{1 - f_0} \sum_{j \in N^-} \bar{a}_{ij} x_j \geq f_0. \tag{9.43}$$

We can strengthen (9.43) by the following technique: The derivation of (9.43) remains valid even if we add integer multiples of integer variables to the left hand side of (9.40): if π is an integral vector, then

$$\sum_{j:x_j \text{ is an integer variable}} \pi_j x_j + \sum_{j \in N} \bar{a}_{ij} x_j = f_0 + k \quad \text{for some } k \in \mathbb{Z}. \tag{9.40'}$$

Thus, we can achieve that every integer variable is in one of the two sets $M^+ = \{j : \bar{a}_{ij} + \pi_j \geq 0\}$ or $M^- := \{j : \bar{a}_{ij} + \pi_j < 0\}$. If $j \in M^+$, then the coefficient π_j of x_j in the new version $\pi^T x \geq \pi_0$ of (9.43) is $\bar{a}_{ij} + \pi_j$, so the best we can achieve is $\pi_j = f_j := (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor)$. If $j \in M^-$, then the coefficient π_j is $-\frac{f_0}{1-f_0}(\bar{a}_{ij} + \pi_j)$ and the smallest value we can achieve is $-\frac{f_0}{1-f_0}(f_j - 1) = \frac{f_0(1-f_j)}{1-f_0}$. In summary, the smallest coefficient we can achieve for an integer variable is

$$\min\left(f_j, \frac{f_0(1-f_j)}{1-f_0}\right). \tag{9.44}$$

The minimum in (9.44) is f_j if and only if $f_j \leq f_0$. This leads to *Gomory's mixed integer cut*:

$$\begin{aligned} &\sum_{\substack{j: f_j \leq f_0 \\ x_j \text{ integer variable}}} f_j x_j + \sum_{\substack{j: f_j > f_0 \\ j:\, x_j \text{ integer variable}}} \frac{f_0(1-f_j)}{1-f_0} x_j. \\ &+ \sum_{\substack{j \in N^+ \\ x_j \text{ no integer variable}}} \bar{a}_{ij} x_j - \frac{f_0}{1-f_0} \sum_{\substack{j \in N^- \\ x_j \text{ no integer variable}}} \bar{a}_{ij} x_j \geq f_0. \end{aligned} \tag{9.45}$$

Similar to Theorem 9.13 it can be shown that an appropriate choice of cutting-planes (9.45) leads to an algorithm which solves the MIP (9.37) in a finite number of steps.

9.6 Structured Inequalities

In the previous sections of this chapter we have derived valid inequalities for general integer and mixed-integer programs. Sometimes focussing on a single constraint (or a small subset of the constraints) can reveal that a particular problem has a useful "local structure". In this section we will explore such local structures in order to derive strong inequalities.

9.6.1 Knapsack and Cover Inequalities

We consider the 0/1-Knapsack polytope

$$P_{\text{KNAPSACK}} := P_{\text{KNAPSACK}}(N, a, b) := \operatorname{conv}\left\{ x \in \mathbb{B}^N : \sum_{j \in N} a_j x_j \leq b \right\}$$

which we have seen a couple of times in these lecture notes (for instance in Example 1.3). Here, the a_i are nonnegative coefficients and $b \geq 0$. We use the general index set N instead of $N = \{1, \dots, n\}$ to emphasize that a knapsack constraint $\sum_{j \in N} a_j x_j \leq b$ might occur as a constraint in a larger integer program and might not involve *all* variables.

In all what follows, we assume that $a_j \leq b$ for $j \in N$ since $a_j > b$ implies that $x_j = 0$ for all $x \in P_{\text{KNAPSACK}}(N, a, b)$. Under this assumption $P_{\text{KNAPSACK}}(N, a, b)$ is full-dimensional, since $\chi^{\varnothing}$ and $\chi^{\{j\}}$ $(j \in N)$ form a set of $n + 1$ affinely independent vectors in $P_{\text{KNAPSACK}}(N, a, b)$.

Each inequality $x_j \geq 0$ for $j \in N$ is valid for $P_{\text{KNAPSACK}}(N, a, b)$. Moreover, each of these nonnegativity constraints defines a facet of $P_{\text{KNAPSACK}}(N, a, b)$, since $\chi^{\varnothing}$ and $\chi^{\{i\}}$ $(i \in N \setminus \{j\})$ form a set of n affinely independent vectors that satisfy the inequality at equality. In the sequel we will search for more facets of $P_{\text{KNAPSACK}}(N, a, b)$ and, less ambitious, for more valid inequalities.

Definition 9.15 (Cover, minimal cover)
A set $C \subseteq N$ *is called a* ***cover****, if*

$$\sum_{j \in C} a_j > b.$$

The cover is called a minimal cover, *if* $C \setminus \{j\}$ *is not a cover for all* $j \in C$.

Each cover C gives us a valid inequality $\sum_{j \in C} x_j \leq |C| - 1$ for P_{KNAPSACK} (if you do not see this immediately, the proof will be given in the following theorem). It turns out that this inequality is quite strong, provided that the cover is minimal.

Example 9.16
Consider the knapsack set

$$X = \left\{x \in \mathbb{B}^7 : 11x_1 + 6x_2 + 6x_3 + 5x_4 + 5x_5 + 4x_6 + x_7 \leq 19\right\}$$

Three covers are $C_1 = \{1, 2, 6\}$, $C_2 = \{3, 4, 5, 6\}$ and $C_3 = \{1, 2, 5, 6\}$ so we have the *cover inequalities*:

$$\begin{array}{cccccccc}
x_1 & +x_2 & & & & +x_6 & \leq & 2 \\
 & & x_3 & +x_4 & +x_5 & +x_6 & \leq & 3 \\
x_1 & +x_2 & & & +x_5 & +x_6 & \leq & 3
\end{array}$$

The cover C_3 is not minimal, since $C_1 \subset C_3$ is also a cover. $\triangleleft$

Theorem 9.17 *Let* $C \subseteq N$ *be a cover. Then, the* ***cover inequality***

$$\sum_{j \in C} x_j \leq |C| - 1$$

is valid for $P_{\text{KNAPSACK}}(N, a, b)$*. Moreover, if* C *is minimal, then the cover inequality defines a facet of* $P_{\text{KNAPSACK}}(C, a, b)$.

Proof: By Observation 2.2 it suffices to show that the inequality is valid for the knapsack set

$$X := \left\{ x \in \mathbb{B}^N : \sum_{j \in N} a_j x_j \leq b \right\}.$$

Suppose that $x \in X$ does not satisfy the cover inequality. We have that $x = \chi^S$ is the incidence vector of a set $S \subseteq N$. By assumption we have

$$|C| - 1 < \sum_{j \in C} x_j = \sum_{j \in C \cap S} \underbrace{x_j}_{=1} = |C \cap S|.$$

So $|C \cap S| = |C|$ and consequently $C \subseteq S$. Thus,

$$\sum_{j \in N} a_j x_j \geq \sum_{j \in C} a_j x_j = \sum_{j \in C \cap S} a_j = \sum_{j \in C} a_j > b,$$

which contradicts the fact that $x \in X$.

It remains to show that the cover inequality defines a facet of $P_{\text{KNAPSACK}}(C, a, b)$, if the cover is minimal. Suppose that there is a facet-defining inequality $c^T x \leq \delta$ such that

$$\left\{ x \in P_{\text{KNAPSACK}}(C, a, b) : \sum_{j \in C} x_j = |C| - 1 \right\} \subseteq F_c := \left\{ x \in P_{\text{KNAPSACK}}(C, a, b) : c^T x = \delta \right\}. \tag{9.46}$$

We will show that $c^T x \leq \delta$ is a nonnegative scalar multiple of the cover inequality.

For $i \in C$ consider the set $C_i := C \setminus \{i\}$. Since, C is minimal, C_i is not a cover. Consequently, each of the $|C|$ incidence vectors $\chi^{C_i} \in \mathbb{B}^C$ is contained in the set on the left hand side of (9.46), so $\chi^{C_i} \in F_c$ for $i \in C$. Thus, for $i \neq j$ we have

$$0 = c^T \chi^{C_i} - c^T \chi^{C_j} = c^T (\chi^{C_i} - \chi^{C_j}) = c_i - c_j.$$

Hence we have $c_i = \gamma$ for $i \in C$ and $c^T x \leq \delta$ is of the form

$$\gamma \sum_{j \in C} x_j \leq \delta. \tag{9.47}$$

Fix $i \in C$. Then, by (9.47) we have

$$c^T \chi^{C_i} = \gamma \sum_{j \in C_i} x_j = \delta$$

and by (9.46) we have

$$\sum_{j \in C_i} x_j = |C| - 1,$$

so $\delta = \gamma(|C| - 1)$ and $c^T x \leq \delta$ must be a nonnegative scalar multiple of the cover inequality. □

The proof technique above is a general tool to show that an inequality defines a facet of a full-dimensional polyhedron:

Observation 9.18 (Proof technique 1 for facets) *Suppose that $P = \text{conv}(X) \subseteq \mathbb{R}^n$ is a full dimensional polyhedron and $\pi^T x \leq \pi_0$ is a valid inequality for P. In order to show that $\pi^T x \leq \pi_0$ defines a facet of P, it suffices to accomplish the following steps:*

(i) Select $t \geq n$ points $x^1, \dots, x^t \in X$ with $\pi^T x^i = \pi_0$ and suppose that all these points lie on a generic hyperplane $c^T x = \delta$.

(ii) Solve the linear equation system

$$\sum_{j=1}^{n} c_j x_j^i = \delta \quad \textit{for } i = 1, \dots, t \tag{9.48}$$

in the $n + 1$ unknowns (c, δ).

(iii) If the only solution of (9.48) is $(c, \delta) = \gamma(\pi, \pi_0)$ *for some* $\gamma \neq 0$*, then the inequality* $\pi^T x \leq \pi_0$ *defines a faceet of* P.

In the proof of Theorem 9.17 we were dealing with a polytope $P_{\text{KNAPSACK}}(C, a, b) \subset \mathbb{R}^C$ and we choose $|C|$ points χ^{C_i} $(i \in C)$ satisfying the cover inequality with equality.

Let us return to the cover inequalities. We have shown that for a minimal cover C the cover inequality $\sum_{j \in C} x_j \leq |C| - 1$ defines a facet of $P_{\text{KNAPSACK}}(C, a, b)$. However, this does not necessarily mean that the inequality is also facet-defining for $P_{\text{KNAPSACK}}(N, a, b)$. Observe that there is a simple way to strengthen the basic cover inequalities:

Lemma 9.19 *Let* C *be a cover for* $X = \left\{x \in \mathbb{B}^N : \sum_{j \in N} a_j x_j \leq b\right\}$*. We define the* ***extended cover*** $E(C)$ *by*

$$E(C) := C \cup \{j \in N : a_j \geq a_i \textit{ for all } i \in C\}.$$

The ***extended cover inequality***

$$\sum_{j \in E(C)} x_j \leq |C| - 1$$

is valid for $P_{\text{KNAPSACK}}(N, a, b)$.

Proof: Along the same lines as the validity of the cover inequality in Theorem 9.17. $\square$

Example 9.20 (Continued)
In the knapsack set of Example 9.16 the extended cover inequality for $C = \{3, 4, 5, 6\}$ is

$$x_3 + x_4 + x_5 + x_6 \leq 3.$$

So, the cover inequality $x_3 + x_4 + x_5 + x_6 \leq 3$ is dominated by the extended cover inequality. On the other hand, the extended cover inequality in turn is dominatd by the inequality $2x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 3$, so it can not be facet-defining (cf. Theorem 3.45). $\lhd$

We have just seen that even extended cover inequalities might not give us a facet of the knapsack polytope. Nevertheless, under some circumstances they are facet-defining:

Theorem 9.21 *Let* $a_1 \geq a_2 \geq \cdots \geq a_n$ *and* $C = \{j_1, \ldots, j_r\}$ *with* $j_1 < j_2 < \cdots < j_r$ *be a minimal cover. Suppose that at least one of the following conditions is satisfied:*

(i) $C = N$

(ii) $E(C) = N$ *and* $(C \setminus \{j_1, j_2\}) \cup \{1\}$ *is not a cover.*

(iii) $C = E(C)$ *and* $(C \setminus \{j_1\}) \cup \{p\}$ *is a cover, where* $p = \min\{j : j \in N \setminus E(C)\}$.

(iv) $C \subset E(C) \subset N$ *and* $(C \setminus \{j_1, j_2\}) \cup \{1\}$ *is a cover and* $(C \setminus \{j_1\}) \cup \{p\}$ *is a cover, where* $p = \min\{j : j \in N \setminus E(C)\}$.

Proof: We construct n affinely independent vectors in X that satisfy the extended cover inequality at equality. Then, it follows that the proper face induced by the inequality has dimension at least $n-1$, which means that is constitutes a facet.

We use the incidence vectors of the following subsets of N:

1. the $|C|$ sets $C_i := C \setminus \{j_i\}$ for $j_i \in C$.
2. the $|E(C) \setminus C|$ sets $C'_k := (C \setminus \{j_1, j_2\}) \cup \{k\}$ for $k \in E(C) \setminus C$. Observe that $|C'_k \cap E(C)| = |C| - 1$ and that C'_k is not a cover by the assumptions of the theorem.
3. the $|N \setminus E(C)|$ sets $\bar{C}_j := C \setminus \{j_1\} \cup \{j\}$ for $j \in N \setminus E(C)$; again $|E(C) \cap \bar{C}_j| = |C| - 1$ and $\bar{C}_j$ is not a cover by the assumptions of the theorem.

It is straightforward to verify that the n vectors constructed above are in fact affinely independent. □

On the way to proving Theorem 9.21 we saw another technique to prove that an inequality is facet defining, which for obvious reasons is called the *direct method:*

Observation 9.22 (Proof technique 2 for facets) *In order to show that* $\pi^T x \leq \pi_0$ *defines a facet of* P*, it suffices to present* $\dim(P) - 1$ *affinely independent vectors in* P *that satisfy* $\pi^T x = \pi_0$ *at equality.*

Usually we are in the situation that $P = \text{conv}(X)$ and we will be able to exploit the combinatorial structure of X. Let us diverge for a moment and illustrate this one more time for the matching polytope.

Example 9.23
Let $G = (V, E)$ be an undirected graph and $M(G)$ be the convex hull of the incidence vectors of all matchings of G. Then, all vectors in $M(G)$ satisfy the following inequalities:

$$x(\delta(v)) \leq 1 \qquad \text{for all } v \in V \tag{9.49a}$$

$$x(\gamma(T)) \leq \frac{|T| - 1}{2} \qquad \text{for all } T \subseteq V, |T| \geq 3 \text{ odd} \tag{9.49b}$$

$$x_e \geq 0 \qquad \text{for all } e \in E. \tag{9.49c}$$

Inequalities (9.49a) and (9.49c) are obvious and the inequalities (9.49b) have been shown to be valid in Example 9.3.

The polytope $M(G)$ is clearly full-dimensional. Moreover, each inequality $x_{e'} \geq 0$ defines a facet of $M(G)$. To see this, take the $|E|$ incidence vectors of the matchings $\emptyset$ and $\{e\}$ where $e \in E \setminus \{e'\}$ which are clearly independent and satisfy the inequality at equality. ◁

9.6.2 Lifting of Cover Inequalities

We return from our short excursion to matchings to the extended cover inequalities. In Example 9.20 we saw that the extended cover inequality $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 3$ is dominated by the inequality $2x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 3$. How could we possibly derive the latter inequality?

Consider the cover inequality for the cover $C = \{3, 4, 5, 6\}$

$$x_3 + x_4 + x_5 + x_6 \leq 3$$

which is valid for our knapsack set

$$X = \left\{x \in \mathbb{B}^7 : 11x_1 + 6x_2 + 6x_3 + 5x_4 + 5x_5 + 4x_6 + x_7 \leq 19\right\}$$

from Example 9.16. We may also say that the cover inequality is valid for the set

$$X' := \left\{x \in \mathbb{B}^4 : 6x_3 + 5x_4 + 5x_5 + 4x_6 \leq 19\right\},$$

which is formed by the variables in C. Since the cover is minimal, by Theorem 9.17 the cover inequality defines a facet of $\text{conv}(X')$, so it is as strong as possible. We would like to transfer the inequality and its strength to the higher dimensional set $\text{conv}(X)$.

As a first step, let us determine the coefficients β_1 such that the inequality

$$\beta_1 x_1 + x_3 + x_4 + x_5 + x_6 \leq 3 \tag{9.50}$$

is valid for

$$X'' := \left\{x \in \mathbb{B}^5 : 11x_1 + 6x_3 + 5x_4 + 5x_5 + 4x_6 \leq 19\right\}.$$

In a second step, we will choose β_1 as large as possible, making the inequality as strong as possible.

For all $x \in X''$ with $x_1 = 0$, the inequality (9.50) is valid for all values of β_1. If $x_1 = 1$, then (9.50) is valid if and only if

$$\beta_1 + x_3 + x_4 + x_5 + x_6 \leq 3$$

is valid for all $x \in \mathbb{B}^4$ satisfying

$$6x_3 + 5x_4 + 5x_5 + 4x_6 \leq 19 - 11 = 8.$$

Thus, (9.50) is valid if and only if

$$\beta_1 + \max\left\{x_3 + x_4 + x_5 + x_6 : x \in \mathbb{B}^4 \text{ and } 6x_3 + 5x_4 + 5x_5 + 4x_6 \leq 8\right\} \leq 3.$$

This is equivalent to saying that $\beta_1 \leq 3 - z_1$, where

$$z_1 = \max\left\{x_3 + x_4 + x_5 + x_6 : x \in \mathbb{B}^4 \text{ and } 6x_3 + 5x_4 + 5x_5 + 4x_6 \leq 8\right\} \tag{9.51}$$

The problem in (9.51) is itself a KNAPSACK problem. However, the objective function is particularly simple and in our example we can see easily that $z_1 = 1$ (we have $z_1 \geq 2$ since $(1, 0, 0, 0)$ is feasible for the problem; on the other hand not two items fit into the knapsack of size 8). Thus, (9.50) is valid for all values $\beta \leq 3 - 1 = 2$. Setting $\beta_1 = 2$ gives the strongest inequality.

The technique that we have seen above is called *lifting*: we "lift" a lower-dimensional (facet-defining) inequality to a higher-dimensional polyhedron. The fact that this lifting is possible gives another justification for studying "local structures" in integer programs such as knapsack inequalities.

In our example we have lifted the cover inequality one dimension. In order to lift the inequality to the whole polyhedron, we need to solve a more general problem. Namely, we wish to find the best possible values β_j for $j \in N \setminus C$ such that the inequality

$$\sum_{j \in N \setminus C} \beta_j x_j + \sum_{j \in C} x_j \leq |C| - 1$$

is valid for $X = \left\{x \in \mathbb{B}^N : \sum_{j \in N} a_j x_j \leq b\right\}$. The procedure in Algorithm 9.3 accomplishes this task.

Algorithm 9.3 Algorithm to lift cover inequalities.

LIFT-COVER

Input: The data N, a, b for a knapsack set $X = \left\{x \in \mathbb{B}^N : \sum_{j \in N} a_j x_j \leq b\right\}$, a minimal cover C

Output: Values β_j for $j \in N \setminus C$ such that

$$\sum_{j \in N \setminus C} \beta_j x_j + \sum_{j \in C} x_j \leq |C| - 1$$

is valid for X

Let $j_1, \ldots, j_r$ be an ordering of $N \setminus C$.
for $t = 1, \ldots, r$ **do**
  The valid inequality

$$\sum_{i=1}^{t-1} \beta_{j_i} x_{j_i} + \sum_{j \in C} x_j \leq |C| - 1$$

  has been obtained so far.
  To calculate the largest value β_{j_t} for which

$$\beta_{j_t} x_{j_t} + \sum_{i=1}^{t-1} \beta_{j_i} x_{j_i} + \sum_{j \in C} x_j \leq |C| - 1$$

  is valid, solve the following KNAPSACK problem:

$$\begin{aligned} z_t = \max & \sum_{i=1}^{t-1} \beta_{j_i} x_{j_i} + \sum_{j \in C} x_j \\ & \sum_{i=1}^{t-1} a_{j_i} x_{j_i} + \sum_{j \in C} a_j x_j \leq b - a_{j_t} \\ & x \in \mathbb{B}^{|C|+t-1} \end{aligned}$$

  Set $\beta_{j_t} := |C| - 1 - z_t$.
end for

Example 9.24 (Continued)
We return to the knapsack set of Example 9.16 and 9.20. Take the minimal cover $C = \{3, 4, 5, 6\}$

$$x_3 + x_4 + x_5 + x_6 \leq 3$$

and set $j_1 = 1$, $j_2 = 2$ and $j_3 = 7$. We have already calculated the value $\beta_1 = 2$. For $\beta_{j_2} = \beta_2$, the coefficient for x_2 in the lifted inequality we need to solve the following instance of KNAPSACK:

$$\begin{aligned} z_2 = \max \quad & 2x_1 + x_3 + x_4 + x_6 + x_6 \\ & 11x_1 + 6x_3 + 5x_4 + 5x_5 + 4x_6 \leq 19 - 6 = 13 \\ & x \in \mathbb{B}^5 \end{aligned}$$

It is easy to see that $z_2 = 2$, so we have $\beta_{j_2} = \beta_2 = 3 - 2 = 1$.

Finally, for $\beta_{j_3} = \beta_7$, we must solve

$$\begin{aligned} z_7 = \max \quad & 2x_1 + x_2 + x_3 + x_4 + x_6 + x_6 + x_7 \\ & 11x_1 + 6x_2 + 6x_3 + 5x_4 + 5x_5 + 4x_6 \leq 19 - 1 = 18 \\ & x \in \mathbb{B}^6 \end{aligned}$$

Here, the problem gets a little bit more involved. It can be seen that $z_7 = 3$, so the coefficient β_7 for x_7 in the lifted cover inequality is $\beta_7 = 3 - 3 = 0$. We finish with the inequality:

$$2x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 3.$$

$\lhd$

We prove that the lifting technique in a more general setting provides us with a tool to derive facet-defining inequalities:

Theorem 9.25 *Suppose that* $X \subseteq \mathbb{B}^n$ *and let* $X^\delta = X \cap \{x \in \mathbb{B}^n : x_1 = \delta\}$ *for* $\delta \in \{0, 1\}$.

(i) Suppose that the inequality

$$\sum_{j=2}^{n} \pi_j x_j \leq \pi_0 \tag{9.52}$$

is valid for X^0. *If* $X^1 = \varnothing$, *then* $x_1 \leq 0$ *is valid for* X. *If* $X^1 \neq \varnothing$, *then the inequality*

$$\beta_1 x_1 + \sum_{j=2}^{n} \pi_j x_j \leq \pi_0 \tag{9.53}$$

is valid for X *if* $\beta_1 \leq \pi_0 - z$, *where*

$$z = \max \left\{ \sum_{i=2}^{n} \pi_j x_j : x \in X^1 \right\}.$$

Moreover, if $\beta_1 = \pi_0 - z$ *and (9.52) defines a face of dimension* k *of* $\operatorname{conv}(X^0)$, *then the lifted inequality (9.53) defines a face of dimension* $k+1$ *of* $\operatorname{conv}(X)$. *In particular, if (9.52) is facet-defining for* $\operatorname{conv}(X^0)$, *then (9.53) is facet-defining for* $\operatorname{conv}(X)$.

(ii) Suppose that the inequality (9.52) is valid for X^1. *If* $X^0 = \emptyset$, *then* $x_1 \geq 1$ *is valid for* X. *If* $X^0 \neq \emptyset$, *then*

$$\gamma_1 x_1 + \sum_{j=2}^{n} \pi_j x_j \leq \pi_0 + \gamma_1 \tag{9.54}$$

is valid for X *if* $\gamma_1 \geq z' - \pi_0$, *where*

$$z' = \max\left\{\sum_{i=2}^{n} \pi_j x_j : x \in X^0\right\}.$$

Moreover, if $\gamma_1 = \pi_0 - z'$ *and (9.52) defines a face of dimension* k *of* $\operatorname{conv}(X^1)$, *then the lifted inequality (9.54) defines a face of dimension* $k+1$ *of* $\operatorname{conv}(X)$. *In particular, if (9.52) is facet-defining for* $\operatorname{conv}(X^1)$, *then (9.53) is facet-defining for* $\operatorname{conv}(X)$.

Proof: We only prove the first part of the theorem. The second part can be proved along the same lines.

We first show that the lifted inequality (9.53) is valid for X for all $\beta_1 \leq \pi_0 - z$. We have $X = X^0 \cup X^1$. If $x \in X^0$, then

$$\beta_1 x_1 + \sum_{j=2}^{n} \pi_j x_j = \sum_{j=2}^{n} \pi_j x_j \leq \pi_0.$$

If $x \in X^1$, then

$$\beta_1 x_1 + \sum_{j=2}^{n} \pi_j x_j = \beta_1 + \sum_{j=2}^{n} \pi_j x_j \leq \beta_1 + z \leq (\pi_0 - z) + z = \pi_0$$

by definition of z. Thus, the validity follows.

If (9.52) defines a k-dimensional face of $\operatorname{conv}(X^0)$, then there are $k+1$ affinely independent vectors $\bar{x}^i$, $i = 1, \dots, k+1$ that satisfy (9.52) at equality. Everyone of those vectors has $x_1 = 0$ and also satisfies (9.53) at equality. Choose $x^* \in X^1$ such that $z = \sum_{j=2}^{n} \pi_j x_j^*$. If $\beta_1 = \pi_0 - z$, then x^* satsifies (9.53) also at equality. Moreover, x^* must be affinely independent from all the vectors $\bar{x}^i$, since the first component of x^* is 1 while all the vectors $\bar{x}^i$ have first component 0. Thus, we have found $k+2$ affinely independent vectors satisfying (9.53) at equality and it follows that the face induced has dimension $k+1$. □

Theorem 9.25 can be used iteratively as in our lifting procedure (Algorithm 9.3): Given $N_1 \subset N = \{1, \dots, n\}$ and an inequality $\sum_{j \in N_1} \pi_j x_j \leq \pi_0$ which is valid for

$$X \cap \{x \in \mathbb{B}^n : x_j = 0 \text{ for } j \in N \setminus N_1\}$$

we can lift one variable at a time to obtain a valid inequality

$$\sum_{j \in N \setminus N_1} \beta_j x_j + \sum_{j \in N_1} x_j \leq |C| - 1 \tag{9.55}$$

for X. The coefficients β_j in (9.55) are independent of the order in which the variables are lifted. The corresponding lifting procedure is a straightforward generalization of our lifting procedure for the cover inequalities. From Theorem 9.25 we obtain the following corollary:

Corollary 9.26 *Let* C *be a minimal cover for* $X = \left\{x \in \mathbb{B}^N : \sum_{j \in N} a_j x_j \leq b\right\}$. *The lifting procedure in Algorithm 9.3 determines a facet-defining inequality for* $\operatorname{conv}(X)$. □

9.6.3 The Set-Packing Polytope

Integer and mixed integer programs often contain inequalities that have all coefficients from $\mathbb{B} = \{0, 1\}$. In particular, many applications require logical inequalities of the form $\sum_{j\in N} x_j \leq 1$ (packing constraint: at most one of the js is chosen) or $\sum_{j\in N} x_j \geq 1$ (covering constraint: at least one of the js is picked). This motivates the study of packing, covering problems, cf. Example 1.9 on page 8:

Definition 9.27 (Set-Packing Polytope and Set Covering Polytope)
Let $A \in \mathbb{B}^{m\times n}$ be an $m \times n$-matrix with entries from $\mathbb{B} = \{0, 1\}$ and $c \in \mathbb{R}^n$. The integer problems

$$\max\left\{c^T x : Ax \leq 1, x \in \mathbb{B}^n\right\}$$
$$\max\left\{c^T x : Ax \geq 1, x \in \mathbb{B}^n\right\}$$
$$\max\left\{c^T x : Ax = 1, x \in \mathbb{B}^n\right\}$$

*are called the **set-packing problem**, the **set-covering problem** and the **set-covering partitioning problem**, respectively.*

In this section we restrict ourselves to the set-packing problem and the set-packing polytope:

$$P_{\text{PACKING}}(A) := \text{conv}\{x \in \mathbb{B}^n : Ax \leq 1\}.$$

For the set-packing problem, there is a nice graph-theoretic interpretation of feasible solutions. Given the matrix A, define an undirected graph $G(A)$ as follows: the vertices of $G(A)$ correspond to the columns of A. There is an edge between i and j if there is a common nonzero entry in columns i and j. The graph $G(A)$ is called the *conflict graph* or *intersection graph*.

Obviously, each feasible binary vector for the set-packing problem corresponds to a stable set in $G(A)$. Conversely, each stable set in $G(A)$ gives a feasible solution for the set-packing problem. Thus, we have a one-to-one correspondence and it follows that

$$P_{\text{PACKING}}(A) = \text{conv}\{x \in \mathbb{B}^n : x_i + x_j \leq 1 \text{ for all } (i,j) \in G(A)\}.$$

In other words, $P_{\text{PACKING}}(A)$ is the stable-set polytope STAB$(G(A))$ of $G(A)$. If G is a graph, then incidence vectors of the $n + 1$ sets $\varnothing$ and $\{v\}$, where $v \in V$ are all affinely independent and contained in STAB(G) whence STAB(G) has full dimension.

We know from Theorem 4.14 that the node-edge incidende matrix of a bipartite graph is totally unimodular (see also Example 4.15). Thus, if $G(A)$ is bipartite, then by the Theorem of Hoffmann and Kruskal (Corollary 4.12 on page 52) we have that $P_{\text{PACKING}}(A)$ is completely described by the linear system:

$$x_i + x_j \leq 1 \text{ for all } (i,j) \in G(A) \tag{9.56a}$$
$$x \geq 0. \tag{9.56b}$$

We also know that for a general graph, the system (9.56) does not suffice to describe the convex hull of its stable sets, here $P_{\text{PACKING}}(A)$. A graph is bipartite if and only if it does not contain an odd cycle (see Lemma 4.6). Odd cycles gives us new valid inequalities:

Theorem 9.28 *Let* C *be an odd cycle in* G. *The* ***odd-cycle inequality***

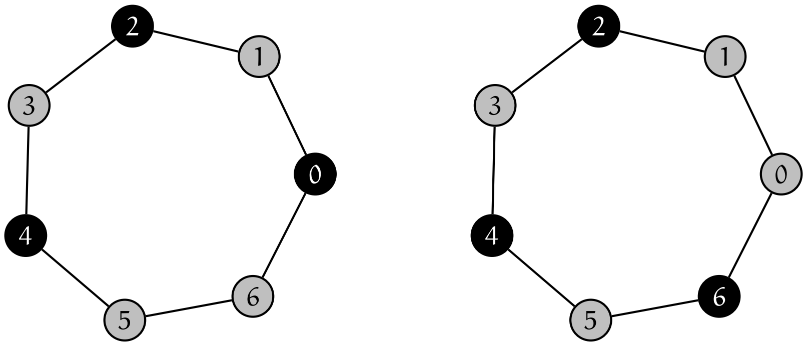
$$\sum_{i \in C} x_i \leq \frac{|C|-1}{2}$$

is valid for $\text{STAB}(G)$. *The above inequality defines a facet of* $\text{STAB}(V(C), E(C))$ *if and only if* C *is an* ***odd hole***, *that is, a cycle without chords.*

Proof: Any stable set x can contain at most every second vertex from C, thus $x(C) \leq (|C|-1)/2$ since $|C|$ is odd. So, the odd-cycle inequality is valid for $\text{STAB}(G)$.

Suppose the C is an odd hole with $V(C) = \{0, 1, \ldots, k-1\}$, $k \in \mathbb{N}$ even and let $c^T x \leq \delta$ be a facet-defining inequality with

$$\begin{aligned} F_C &= \left\{ x \in \text{STAB}(V(C), E(C)) : \sum_{i \in C} x_i = \frac{|C|-1}{2} \right\} \\ &\subseteq F_c = \left\{ x \in \text{STAB}(V(C), E(C)) : c^T x = \delta \right\} \end{aligned}$$



(a) The stable set $S_1 = \{i+2, i+4, \ldots, i-3, i\}$ in the odd cycle C.

(b) The stable set $S_2 = \{i+2, i+4, \ldots, i-3, i-1\}$ in the odd cycle C.

Figure 9.3: Construction of the stable sets S_1 and S_2 in the proof of Theorem 9.28: Here, node $i = 0$ is the anchor point of the stable sets. The stable sets are indicated by the black nodes.

Fix $i \in C$ and consider the two stable sets

$$\begin{aligned} S_1 &= \{i+2, i+4, \ldots, i-3, i\} \\ S_2 &= \{i+2, i+4, \ldots, i-3, i-1\} \end{aligned}$$

where all indices are taken modulo k (see Figure 9.3 for an illustration). Then, $\chi^{S_i} \in F_C \subseteq F_c$, so we have

$$0 = c^T \chi^{S_1} - c^T \chi^{S_2} = c^T (\chi^{S_1} - \chi^{S_2}) = c_i - c_{i-1}.$$

Since we can choose $i \in C$ arbitrarily, this implies that $c_i = \gamma$ for all $i \in C$ for some $\gamma \in \mathbb{R}$. As in the proof of Theorem 9.17 on page 134 we can now conclude that $c^T x \leq \delta$ is a positive scalar multiple of the odd-hole inequality (observe that we used proof technique 1 for facets).

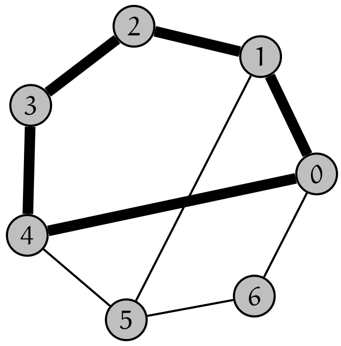


Figure 9.4: If C is an odd cycle with choords, then there is an odd hole H contained in C (indicated by the thick lines).

Finally, suppose that C is a cycle with at least one chord. We can find an odd hole H that is contained in C (see Figure 9.4). There are $|C| - |H|$ vertices in $C \setminus H$, and we can find $(|C| - |H|)/2$ edges $(i_k, v_k) \in C$ where both endpoints are in $C \setminus H$. Consider the following valid inequalities:

$$\sum_{i \in H} x_i \leq \frac{|H| - 1}{2}$$
$$x_{i_k} + x_{j_k} \leq 1 \text{ for } k = 1, \ldots, \frac{|C| - |H|}{2}.$$

Summing up those inequalities yields $\sum_{i \in C} x_i \leq \frac{|C|-1}{2}$, which is the odd-cycle inequality for C. Hence, C is redundant and can not induce a facet of $\mathrm{STAB}(V(C), E(C))$. This completes the proof. □

The final class of inequalities we consider here are the so-called *clique-inequalities*:

Theorem 9.29 *Let* Q *be a clique in* G. *The* ***clique inequality***

$$\sum_{i \in Q} x_i \leq 1$$

is valid for STAB(G). *The above inequality defines a facet of* STAB(G) *if and only if* Q *is a* ***maximal clique***, *that is, a clique which is maximal with respect to inclusion.*

Proof: The validity of the inequality is immediate. Assume that Q is maximal. We find n affinely independent vectors that satisfy $x(Q) = 1$. For $v \in Q$, we take the incidence vector of $\{v\}$. For $u \notin Q$, we choose a node $v \in Q$ which is adjacent to u. Such a node exists, since Q is maximal. We add the incidence vector of $\{u, v\}$ to our set. In total we have n vectors which satisfy $x(Q) \leq 1$ with equality. They are clearly affinely independent.

Assume conversely that Q is not maximal. So, there is a clique $Q' \supset Q$, $Q' \neq Q$. The clique inequality $x(Q') \leq 1$ dominates $x(Q) \leq 1$, so $x(Q) \leq 1$ is not necessary in the description of STAB(G) and $x(Q) \leq 1$ can not define a facet. □

10

Column Generation

One of the recurring ideas in optimization is that of *decomposition*. The idea of a decomposition method is to remove some of the variables from a problem and handle them in a *master problem*. The resulting subproblems are often easier and more efficient to solve. The decomosition method iterates back and forth between the master problem and the subproblem(s), exchanging information in order to solve the overall problem to optimality.

10.1 Dantzig-Wolfe Decomposition

We start with the Dantzig-Wolfe Decomposition for Linear Programs. Suppose that we are given the following Linear Program:

$$\max \quad c^T x \tag{10.1a}$$

$$A^1 x \leq b^1 \tag{10.1b}$$

$$A^2 x \leq b^2 \tag{10.1c}$$

where A^i is an $m_i \times n$-matrix. Consider the polyhedron

$$P^2 = \left\{x : A^2 x \leq b^2\right\}.$$

From Minkowski's Theorem (Theorem 3.59 on page 40) we know that any point $x \in P^2$ is of the form

$$x = \sum_{k \in K} \lambda_k x^k + \sum_{j \in J} \mu_j r^j \tag{10.2}$$

with $\sum_{k \in K} \lambda_k = 1$, $\lambda_k \geq 0$ for $k \in K$, $\mu_j \geq 0$ for $j \in J$. The vectors x^k and r^j are the extreme points and extreme rays of P^2. Using (10.2) in (10.1) yields:

$$\max \quad c^T \left(\sum_{k \in K} \lambda_k x^k + \sum_{j \in J} \mu_j r^j \right) \tag{10.3a}$$

$$A^1 \left(\sum_{k \in K} \lambda_k x^k + \sum_{j \in J} \mu_j r^j \right) \leq b^1 \tag{10.3b}$$

$$\sum_{k \in K} \lambda_k = 1 \tag{10.3c}$$

$$\lambda \in \mathbb{R}_+^K, \mu \in \mathbb{R}_+^J \tag{10.3d}$$

Rearranging terms, we see that (10.3) is equivalent to the following Linear Program:

$$\max \quad \sum_{k\in K}(c^T x^k)\lambda_k + \sum_{j\in J}(c^T \mu_j) r^j \tag{10.4a}$$

$$\sum_{k\in K}(A^1 x^k)\lambda_k + \sum_{j\in J}(A^1 r^j)\mu_j \le b^1 \tag{10.4b}$$

$$\sum_{k\in K}\lambda_k = 1 \tag{10.4c}$$

$$\lambda \in \mathbb{R}^K_+, \mu \in \mathbb{R}^J_+ \tag{10.4d}$$

Let us compare the initial formulation (10.1) and the equivalent one (10.4):

Formulation	number of variables	number of constraints
(10.1)	n	$m_1 + m_2$
(10.4)	$\lvert K\rvert + \lvert J\rvert$	m_1

In the transition from (10.1) to (10.4) we have decreased the number of contstraints by m_2, but we have increased the number of variables from n to $|K| + |J|$ which is usually much larger than n (as an example that $|K| + |J|$ can be exponentially larger than n consider the unit cube $\{x \in \mathbb{R}^n : 0 \le x \le 1\}$ which has $2n$ constraints and 2^n extreme points). Thus, the reformulation may seem like a bad move. The crucial point is that we can still apply the Simplex method to (10.4) without actually writing down the huge formulation.

Let us abbreviate (10.4) (after the introduction of slack variables) by

$$\max\left\{w^T\eta : D\eta = b, \eta \ge 0\right\}, \tag{10.5}$$

where $D \in \mathbb{R}^{(m_1+1)\times(|K|+|J|)}$, $d \in \mathbb{R}^{m_1+1}$.

The Simplex method iterates from basis to basis in order to find an optimal solution. A basis of (10.5) is a set $B \subseteq \{1, \ldots, |K| + |J|\}$ with $|B| = m_1 + 1$ such that the corresponding square submatrix D_B of B is nonsingular. Observe that such a basis is smaller (namely by m_2 variables) than a basis of the original formulation (10.1). In particular, the matrix $D_B \in \mathbb{R}^{(m_1+1)\times(m_1+1)}$ is *much* smaller than a basic matrix for (10.1) which is an $(m_1 + m_2) \times (m_1 + m_2)$-matrix. In any basic solution $\eta_B := D_B^{-1}d$ of (10.5) and $\eta_N := 0$ only a very small number of variables ($m_1 + 1$ out of $|K| + |J|$) can be nonzero.

It is easy to see that a basic solution (η_B, η_N) of (10.5) is optimal if and only if the vector y defined by $y^T D_B = w_B^T$ satisfies $w_N - y^T D_N \le 0$. We only outline the basics and refer to standard textbooks on Linear Programming for details, e.g. [Lue84, CC+98]. The key observation is that in the Simplex method the only operation that uses all columns of the system (10.5) is this *pricing operation*, which checks whether the reduced costs of the nonbasic variables are nonnegative.

10.1.1 A Quick Review of Simplex Pricing

Recall that the Linear Programming dual to (10.5) is given by:

$$\min\left\{b^T y : D^T y \ge w\right\}. \tag{10.6}$$

The pricing step works as follows. Given a basis B and correspondig basic solution $\eta = (\eta_B, \eta_N) = (D_B^{-1}d, 0)$ the Simplex method solves $y^T D_B = w_B^T$ to obtain y. If $w_N - y^T D_N \leq 0$, then y is feasible for the dual (10.6) and we have an optimal solution for (10.5), since

$$w^T \eta = w_B^T \eta_B = y^T D_B \eta_B = y^T b = b^T y,$$

and for any pair (η', y') of feasible solutions for (P) and (D), respectively, we have

$$w^T \eta \leq (D^T y')^T \eta' = (y')^T D \eta' = b^T y'.$$

If on the other hand, $w_i - d_i^T y > 0$ for some $i \in N$, we can improve the solution by adding variable η_i to the basis and throwing out another index. We first express the new variable in terms of the old basis, that is, we solve $D_B z = d_i$. Let $\eta(\varepsilon)$ be defined by $\eta_B(\varepsilon) := \eta_B - \varepsilon z$, $\eta_i(\varepsilon) := \varepsilon$ and zero for all other variables. Then,

$$\begin{aligned}
w_B^T \eta(\varepsilon) &= w_B^T (\eta_B - \varepsilon z) + w_i \varepsilon \\
&= w_B^T \eta_B + \varepsilon (w_i - w_B^T z) \\
&= w_B^T \eta_B + \varepsilon (w_i - y^T D_B z) \\
&= w_B^T \eta_B + \varepsilon \underbrace{(w_i - y^T d_i)}_{>0}
\end{aligned}$$

Thus, for $\varepsilon > 0$ the new solution $\eta(\varepsilon)$ is better than the old one η. The Simplex method now chooses the largest possible value of ε, such that $\eta(\varepsilon)$ is feasible, that is $\eta(\varepsilon) \geq 0$. This operation will make one of the old basic variables j in B become zero. The selection of j is usually called the *ratio test*, since j is any index in B such that $z_j > 0$ and j minimizes the ratio η_i / z_i over all $i \in B$ having $z_i > 0$.

10.1.2 Pricing in the Dantzig-Wolfe Decomposition

In our particular situation we do not want to explicitly iterate over all entries of the vector $w_N - y^T D_N$ in order to check for nonnegativity. Rather, the pricing can be accomplished by solving the following Linear Program:

$$\zeta = \max \quad (c^T - \bar{y}^T A^1) x - y_{m_1+1} \tag{10.7a}$$

$$A^2 x \leq b^2 \tag{10.7b}$$

where $\bar{y}$ is the vector composed of the first m_1 components of y and y satisfies $y^T D_B = w_B$. The following cases can occur:

Case 1: We have $\zeta > 0$ in (10.7).

Then, the problem (10.7) has an optimal solution x^* with $(c^T - \bar{y}^T A^1) x^* > \bar{y}_{m_1+1}$. In this case, $x^* = x^k$ for some $k \in K$ is an extreme point. Let $D_{k,\cdot}$ be the kth row of D. The reduced cost of x^k is given by

$$w_k - \bar{y}^T D_{\cdot,k} = c^T x^k - \bar{y}^T \begin{pmatrix} A^1 x^k \\ 1 \end{pmatrix} = c^T x^k - \bar{y}^T A_1 x^k - \bar{y}_{m_1+1} > 0.$$

Thus, x^k will be the variable entering the basis in the Simplex step (or in other words, $\binom{A^1 x^k}{1}$ will be the column entering the Simplex tableau).

Case 2: The problem (10.7) is unbounded.

Then, there is an extreme ray r^j for some $j \in J$ with $(c^T - \bar{y}^T A^1) r^j > 0$. The reduced cost of r^j is given by

$$w_{|K|+j} - \bar{y}^T D_{\cdot,|K|+j} = c^T r^j - \bar{y}^T \begin{pmatrix} A^1 r^j \\ 0 \end{pmatrix} = c^T r^j - \bar{y}^T A^1 r^j > 0.$$

So, r^j will be the variable entering the basis, or $\binom{A^1 r^j}{0}$ will be the column entering the Simplex tableau.

Case 3: We have $\zeta < 0$ in (10.7).

Then, the problem (10.7) has an optimal solution x^* with $(c^T - \bar{y}^T A^1) x^* \leq \bar{y}_{m_1+1}$. By the same arguments as in Case 1 and Case 2 it follows that $w_i - \bar{y}^T D_{\cdot,i} \leq 0$ for all $i \in K \cup J$ which shows that the current basic solution is an optimal solution solution of (10.4).

We have decomposed the original problem (10.1) into two problems: the master problem (10.4) and the pricing problem (10.7). The method works with a feasible solution for the master problem (10.4) and generates columns of the constraint matrix *on demand*. Thus, the method is also called *column generation method*.

10.2 Dantzig-Wolfe Reformulation of Integer Programs

We now turn to integer programs. Decomposition methods are particularly promising if the solution set of the IP which we want to solve has a "decomposable structure". In particular, we will be interested in integer programs where the constraints take on the following form:

$$\begin{array}{ccccccc} A^1 x^1 & +A^2 x^2 & + \dots & A^k x^k & = & b \\ D^1 x^1 & & & & \leq & d^1 \\ & D^2 x^2 & & & \leq & d^2 \\ & & \ddots & & \vdots & \vdots \\ & & & D^K x^K & \leq & d^K \end{array} \tag{10.8}$$

If the IP has the form above, then the sets

$$X^k = \left\{ x^k \in \mathbb{Z}_+^{n_k} : D^k x^k \leq d^k \right\}$$

are independent except for the *linking constraint* $\sum_{k=1}^K A^k x^k = b$. Our integer program which we want to solve is:

$$z = \max \left\{ \sum_{k=1}^K (c^k)^T x^k : \sum_{k=1}^K A^k x^k = b, x^k \in X^k \text{ for } k = 1, \dots, k \right\}. \tag{10.9}$$

In the sequel we assume that each set X^k contains a large but finite number of points:

$$X^k = \left\{ x^{k,t} : t = 1, \dots, T_k \right\}. \tag{10.10}$$

An extension of the method also works for unbounded sets X^k but the presentation is even more technical since we also need to handle extreme rays. In the

Dantzig-Wolfe decomposition in Section 10.1 we reformulated the problem to be solved by writing every point as a convex combination of its extreme points (in case of bounded sets there are no extreme rays). We will adopt this idea to the integer case.

Using (10.10) we can write:

$$X^k = \left\{ x^k \in \mathbb{R}^{n_k} : x^k = \sum_{t=1}^{T_k} \lambda_{k,t} x^{k,t}, \sum_{t=1}^{T_k} \lambda_{k,t} = 1, \lambda_{t,k} \in \{0,1\} \text{ for } t = 1, \dots, T_k \right\}.$$

Substituting into (10.9) gives an equivalent integer program, the *IP master problem*:

(10.11a)

$$\text{(IPM)} \quad z = \max \quad \sum_{k=1}^{K} \sum_{t=1}^{T_k} ((c^k)^T x^{k,t}) \lambda_{k,t}$$

$$\text{(10.11b)} \qquad \sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t}) \lambda_{k,t} = b$$

$$\text{(10.11c)} \qquad \sum_{t=1}^{T_k} \lambda_{k,t} = 1 \qquad \text{for } k = 1, \dots, K$$

$$\text{(10.11d)} \qquad \lambda_{k,t} \in \{0,1\} \qquad \text{for } k = 1, \dots, K \text{ and } t = 1, \dots, T_k$$

10.2.1 Solving the Master Linear Program

In order to solve the IP master problem (10.11) we first solve its Linear Programming relaxation which is given by

(10.12a)

$$\text{(LPM)} \quad z^{\text{LPM}} = \max \quad \sum_{k=1}^{K} \sum_{t=1}^{T_k} ((c^k)^T x^{k,t}) \lambda_{k,t}$$

$$\text{(10.12b)} \qquad \sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t}) \lambda_{k,t} = b$$

$$\text{(10.12c)} \qquad \sum_{t=1}^{T_k} \lambda_{k,t} = 1 \qquad \text{for } k = 1, \dots, K$$

$$\text{(10.12d)} \qquad \lambda_{k,t} \geq 0 \qquad \text{for } k = 1, \dots, K \text{ and } t = 1, \dots, T_k$$

The method to solve (10.12) is the same we used for the reformulation in Section 10.1: a column generation technique allows us to solve (10.12) without writing down the whole Linear Program at once. We always work with a small subset of the columns.

Observe that (10.12) has a column

$$\begin{pmatrix} c^k x \\ A^k x \\ e_k \end{pmatrix} \quad \text{with} \quad e_k = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow k$$

for every $x \in X^k$. The constraint matrix of (10.12) is of the form:

$$\begin{pmatrix} A^1x^{1,1} & \dots & A^1x^{1,T_1} & A^2x^{2,1} & \dots & A^2x^{2,T_2} & \dots & \dots & A^Kx^{K,1} & \dots & A^Kx^{1,T_K} \\ 1 & \dots & 1 & & & & & & & & \\ & & & 1 & \dots & 1 & & & & & \\ & & & & & & \ddots & & & & \\ & & & & & & & \ddots & & & \\ & & & & & & & & 1 & \dots & 1 \end{pmatrix},$$

while the right hand side has the form

$$\bar{b} = \begin{pmatrix} b \\ 1 \end{pmatrix}.$$

Let π_i, $i = 1, \dots, m$ be the dual variables associated with the linking constraints (10.12b) and μ_k, $k = 1, \dots, K$ be the dual variables corresponding to the constraints (10.12c). Constraints (10.12c) are also called *convexity constraints*. The Linear Programming dual of (10.1) is given by:

(10.13a)
$$\text{(DLPM)} \quad \min \quad \sum_{i=1}^{m} \pi_i + \sum_{k=1}^{K} \mu_k$$

(10.13b) $$\pi^T(A^kx^k) + \mu_k \geq c^k \quad \text{for all } x^k \in X^k, k = 1, \dots, K$$

Suppose that we have a subset of the columns which contains at least one column for each $k = 1, \dots, K$ such that the following *Restricted Linear Programming Master Problem* is feasible:

(10.14a) $$\text{(RLPM)} \quad \bar{z}^{\text{RLPM}} = \max \quad \bar{c}^T\bar{\lambda}$$

(10.14b) $$\bar{A}\bar{\lambda} = \bar{b}$$

(10.14c) $$\bar{\lambda} \geq 0$$

The matrix $\bar{A}$ is a submatrix of the constraint matrix, $\bar{\lambda}$ denotes the restricted set of variables and $\bar{c}$ is the corresponding restriction of the cost vector to those variables. Let $\bar{\lambda}^*$ be an optimal solution of (10.14) and $(\pi, \mu) \in \mathbb{R}^m \times \mathbb{R}^K$ be a corresponding dual solution (which is optimal for the Linear Programming dual to (10.14)).

Clearly, any feasible solution to the (RLPM) (10.14) is feasible for (LPM) (10.12), so we get

$$\bar{z}^{\text{RLPM}} = \bar{c}^T\bar{\lambda}^* = \sum_{i=1}^{m} \pi_i b_i + \sum_{k=1}^{K} \mu_k \leq z^{\text{LPM}} \leq z.$$

The pricing step is essentially the same as in Section 10.1. We need to check whether for each $x \in X^k$ the reduced cost is nonpositive, that is whether $(c^k)^T x - \pi^T A^k x - \mu_k \leq 0$ (equivalently, this means to check whether (π, μ) is feasible for the Linear Programming dual (DLPM) (10.13) of (LPM) (10.12)). Instead of going through the reduced costs one by one, we solve K optimization problems. While in Section 10.1.2 the corresponding optimization problem was a Linear Program (10.7), here we have to solve an *integer program* for each $k = 1, \dots, K$:

(10.15a) $$\zeta_k = \max \quad ((c^k)^T x - \pi^T A^k)x - \mu_k$$

(10.15b) $$x \in X^k = \left\{x^k \in \mathbb{Z}_+^{n_k} : D^k x^k \leq d^k\right\}$$

Two cases can occur, $\zeta_k \leq 0$ and $\zeta_k > 0$ (the case that (10.15) is unbounded is impossible since we have assumed that X^k is a bounded set).

Case 1: $\zeta_k > 0$ for some $k \in \{1, \ldots, K\}$

Let $\tilde{x}^k$ be the optimal solution of (10.15) for this value of k. As in Section 10.1.2 it follows that $\tilde{x}^k$ is an extreme point of X^k, say $\tilde{x}^k = x^{k,t}$. The column corresponding to the variable $x^{k,t}$ has a positive reduced price. We introduce a new column $\begin{pmatrix} c^k x^{t,k} \\ A^k x^{t,k} \\ e_k \end{pmatrix}$. This leads to a new Restricted Linear Programming Master Problem which can be reoptimized easily by Simplex steps.

Case 2: $\zeta_k \leq 0$ for $k = 1, \ldots, K$

In this case, the dual solution (π, μ) which we obtained for the dual of (RLPM) (10.14) is also feasible for the Linear Programming dual (DLPM) (10.13) of (LPM) (10.12) and we have

$$\bar{z}^{\text{LPM}} \leq \sum_{i=1}^{m} \pi_i b_i + \sum_{k=1}^{K} \mu_k = \bar{c}^T \bar{\lambda}^* = \bar{z}^{\text{RLPM}} \leq \bar{z}^{\text{LPM}}.$$

Thus $\bar{\lambda}^*$ is optimal for (LPM) (10.12) and we can terminate.

We can derive an upper bound for z^{LPM} during the run of the algorithm by using Linear Programming duality. By definition, we have

$$\zeta_k \geq ((c^k)^T - \pi^T A^k)x^k - \mu_k \quad \text{for all } x^k \in K, k = 1, \ldots, K.$$

Let $\zeta = (\zeta_1, \ldots, \zeta_K)$. Then, it follows that $(\pi, \mu + \zeta)$ is feasible for the dual (DLPM) (10.13). Thus, by Linear Programming duality we get

$$z^{\text{LPM}} \leq \sum_{i=1}^{m} \pi_i b_i + \sum_{k=1}^{K} \mu_k + \sum_{k=1}^{K} \zeta_k. \tag{10.16}$$

Finally, we note that there is an alternative stopping criterion to the one we have already derived in Case 2 above. Let $(\bar{x}^1, \ldots, \bar{x}^K)$ be the K solutions of (10.15), so that $\zeta_k = ((c^k)^T \bar{x}^k - \pi^T A^k)x^k - \mu_k$. Then

$$\sum_{k=1}^{K} (c^k)^T \bar{x}^k = \sum_{k=1}^{K} \pi^T A^k x^k + \sum_{k=1}^{K} \mu_k + \sum_{k=1}^{K} \zeta^k. \tag{10.17}$$

So, if $(\bar{x}^1, \ldots, \bar{x}^K)$ satisfies the linking constraint $\sum_{k=1}^{K} A^k x^k = b$, then we get from (10.17) that

$$\sum_{k=1}^{K} (c^k)^T \bar{x}^k = \pi^T b + \sum_{k=1}^{K} \mu_k + \sum_{k=1}^{K} \zeta^k. \tag{10.18}$$

The quantity on the right hand side of (10.18) is exactly the upper bound (10.16) for the optimal value z^{LPM} of the Linear Programming Master. Thus, we can conclude that $(\bar{x}^1, \ldots, \bar{x}^K)$ is optimal for (LPM).

10.2.2 Strength of the Master Linear Program and Relations to Lagrangean Duality

We first investigate what kind of bounds the Master Linear Program will provide us with.

Theorem 10.1 *The optimal value z^{LPM} of the Master Linear Program (10.12) satisfies:*

$$z^{LPM} = \max\left\{\sum_{k=1}^{K}(c^k)^T x^k : \sum_{k=1}^{K} A^k x^k = b, x^k \in \text{conv}(X^k) \textit{ for } k = 1, \dots, K\right\}.$$

Proof: We obtain the Master Linear Program (10.12) by substituting $x^k = \sum_{t=1}^{T_k} \lambda_{k,t} x^{k,t}$, $\sum_{t=1}^{T_k} \lambda_{k,t} = 1$ and $\lambda_{t,k} \geq 0$ for $t = 1, \dots, T_k$. This is equivalent to substituting $x^k \in \text{conv}(X^k)$. □

The form of the integer program (10.9) under study suggests an alternative approach to solving the problem. We could dualize the linking constraints to obtain the Lagrangean dual (see Section 6.3)

$$w_{LD} = \min_{u \in \mathbb{R}^m} L(u),$$

where

$$\begin{aligned} L(u) &= \max\left\{\sum_{k=1}^{K}(c^k)^T x^k + u^T(b - A^k x^k) : x^k \in X^k \; k = 1, \dots, K\right\} \\ &= \max\left\{\sum_{k=1}^{K}((c^k)^T - u^T A^k)x^k + u^T b) : x^k \in X^k \; k = 1, \dots, K\right\} \end{aligned}$$

Observe that the calculation of $L(u)$ decomposes automatically into K independent subproblems:

$$L(u) = u^T b + \sum_{k=1}^{K} \max\left\{((c^k)^T - u^T A^k)x^k : x^k \in X^k\right\}.$$

Theorem 6.18 tells us that

$$w_{LD} = \max\left\{\sum_{k=1}^{K}((c^k)^T - u^T A^k)x^k + u^T b) : x^k \in \text{conv}(X^k) \; k = 1, \dots, K\right\}.$$

Comparing this result with Theorem 10.1 gives us the following corollary:

Corollary 10.2 *The value of the Linear Programming Master and the Lagrangean dual obtained by dualizing the linking constraints coincide:*

$$z^{LPM} = w_{LD}.$$

□

10.2.3 Getting an Integral Solution

We have shown how to solve the Linear Programming Master and also shown that it provides the same bounds as a Lagrangean dual. If at the end of the column generation process the optimal solution $\bar{\lambda}^*$ of (LPM) (10.12) is integral, we have an optimal solution for the integer program (10.11) which was our original problem. However, if $\bar{\lambda}^*$ is fractional, then (10.11) is not yet solved. We know that $z^{\text{LPM}} = w_{\text{LD}} \geq z$. This gives us at least an upper bound for the optimal value and suggests using this bound in a branch-and-bound algorithm.

Recall that in Section 9.3 we combined a branch-and-bound algorithm with a cutting-plane approach. The result was a *branch-and-cut algorithm*. Similarly, we will now combine branch-and-bound methods and column generation which gives us what is known as a *branch-and-price algorithm*. In this section we restrict the presentation to 0-1-problems ("binary problems").

The integer program which we want to solve is:

$$(10.19) \qquad z = \max \left\{ \sum_{k=1}^{K} (c^k)^T x^k : \sum_{k=1}^{K} A^k x^k = b, x^k \in X^k \text{ for } k = 1, \dots, k \right\},$$

where

$$X^k = \left\{ x^k \in \mathbb{B}_+^{n_k} : D^k x^k \leq d^k \right\}$$

for $k = 1, \dots, K$. As in Section 10.2 we reformulate the problem. Observe that in the binary case each set X^k is automatically bounded, $X^k = \left\{ x^{k,t} : t = 1, \dots, T_k \right\}$. so that

$$X^k = \left\{ x^k \in \mathbb{R}^{n_k} : x^k = \sum_{t=1}^{T_k} \lambda_{k,t} x^{k,t}, \sum_{t=1}^{T_k} \lambda_{k,t} = 1, \lambda_{t,k} \in \{0,1\} \text{ for } t = 1, \dots, T_k \right\}.$$

and substituting into (10.19) gives the *IP master problem*:

$$\begin{aligned}
&(10.20a) && && \\
&(\text{IPM}) \quad z = \max && \sum_{k=1}^{K} \sum_{t=1}^{T_k} ((c^k)^T x^{k,t}) \lambda_{k,t} && \\
&(10.20b) && \sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{k,t}) \lambda_{k,t} = b && \\
&(10.20c) && \sum_{t=1}^{T_k} \lambda_{k,t} = 1 && \text{for } k = 1, \dots, K \\
&(10.20d) && \lambda_{k,t} \in \{0,1\} && \text{for } k = 1, \dots, K \text{ and } t = 1, \dots, T_k
\end{aligned}$$

Let $\bar{\lambda}$ be an optimal solution to the LP-relaxation of (10.20), so that $z^{\text{LPM}} =$

$\sum_{k=1}^{K}\sum_{t=1}^{T_k}((c^k)^T x^{k,t})\bar{\lambda}_{k,t}$, where

$$
\begin{aligned}
&(10.21a) && (LPM) \quad z^{LPM} = \max && \sum_{k=1}^{K}\sum_{t=1}^{T_k}((c^k)^T x^{k,t})\lambda_{k,t} && \\
&(10.21b) && && \sum_{k=1}^{K}\sum_{t=1}^{T_k}(A^k x^{k,t})\lambda_{k,t} = b && \\
&(10.21c) && && \sum_{t=1}^{T_k}\lambda_{k,t} = 1 && \text{for } k = 1,\dots,K \\
&(10.21d) && && \lambda_{k,t} \geq 0 && \text{for } k = 1,\dots,K \text{ and } t = 1,\dots,T_k
\end{aligned}
$$

Let $\bar{x}^k = \sum_{t=1}^{T_k}\bar{\lambda}_{k,t}x^{k,t}$ and $\bar{x} = (\bar{x}^1,\dots,\bar{x}^k)$. Since all the $x^{k,t} \in X^k$ are different binary vectors, it follows that $\bar{x}^k \in \mathbb{B}^{n_k}$ if and only if all $\bar{\lambda}_{k,t}$ are integers.

So, if $\bar{\lambda}$ is integral, then $\bar{x}$ is an optimal solution for (10.19). Otherwise, there is k_0 such that $\bar{x}^{k_0} \notin \mathbb{B}^{n_{k_0}}$, i.e, there is t_0 such that $\bar{x}^{k_0}_{t_0} \notin \mathbb{B}$. So, like in the branch-and-bound scheme in Chapter 8 we can branch on the fractional variable $\bar{x}^{k_0}_{t_0}$: We split the feasible set S of all feasible solutions into $S = S_0 \cup S_1$, where

$$
\begin{aligned}
S_0 &= S \cap \left\{x : x^{k_0}_{j_0} = 0\right\}\\
S_1 &= S \cap \left\{x : x^{k_0}_{j_0} = 1\right\}.
\end{aligned}
$$

This is illustrated in Figure 10.1(a). We could also branch on the column variables and split S into $S = S_0' \cup S_1'$ where

$$
\begin{aligned}
S_0' &= S \cap \left\{\lambda : \lambda^{k_1,t_1} = 0\right\}\\
S_1' &= S \cap \left\{\lambda : \lambda^{k_1,t_1} = 1\right\},
\end{aligned}
$$

where $\lambda^{k_1}_{t_1}$ is a fractional varialbe in the optimal solution $\bar{\lambda}$ of the Linear Programming Master (see Figure 10.1(b)). However, branching on the column variables leads to a highly unbalanced tree for the following reason: the branch with $\lambda_{k_1,t_1} = 0$ exludes just one column, namely the t_1th feasible solution of the k_1st subproblem. Hence, usually branching on the original variables is more desirable, since it leads to more balanced enumeration trees.

(a) Branching on the original variables (b) Branching on the column variables

Figure 10.1: Branching for 0-1 column generation.

We return to the situation where we branch on an original variable $x^{k_0}_{j_0}$. We have

$$x_{j_0}^{k_0} = \sum_{t=1}^{T_{k_0}} \lambda_{k_0,t} x_{j_0}^{k_0,t}.$$

Recall that $x_{j_0}^{k_0,t} \in \mathbb{B}$, since each vector $x^{k_0,t} \in \mathbb{B}^{n_{k_0}}$.

If we require that $x_{j_0}^{k_0} = 0$, this implies that $\lambda_{k_0,t} = 0$ for all t with $x_{j_0}^{k_0,t} > 0$. Put in Similarly, if we require that $x_{j_0}^{k_0} = 1$, this implies that $\lambda_{k_0,t} = 0$ for all t with $x_{j_0}^{k_0,t} > 0$.

In summary, if we require in the process of branching that $x_{j_0}^{k_0} = \delta$ for some fixed $\delta \in \{0,1\}$, then $\lambda_{k_0,t} > 0$ only for those t such that $x_j^{k_0,t} = \delta$. Thus, the Master Problem at node $S_\delta = S_0 = S \cap \left\{x : x_{j_0}^{k_0} = \delta\right\}$ is given by:

(10.22a)

$$(\text{IPM})(S_\delta) \quad \max \quad \sum_{k\neq k_0} \sum_{t=1}^{T_k} ((c^k)^T x^{k,t})\lambda_{k,t} + \sum_{t:x_{j_0}^{k_0,t}=\delta} ((c^{k_0})^T x^{k_0,t})\lambda_{k_0,t}$$

$$\sum_{k\neq k_0} \sum_{t=1}^{T_k} (A^k x^{k,t})\lambda_{k,t} + \sum_{t:x_{j_0}^{k_0,t}=\delta} (A^{k_0} x^{k_0,t})\lambda_{k_0,t} = b \tag{10.22b}$$

$$\sum_{t=1}^{T_k} \lambda_{k,t} = 1 \quad \text{for } k \neq k_0 \tag{10.22c}$$

$$\lambda_{k,t} \in \{0,1\} \quad \text{for } k = 1, \dots, K \text{ and } t = 1, \dots, T_k \tag{10.22d}$$

The new problems (10.22) have the same form as the original Master Problem (10.20). The only difference is that some columns, namely all those where $x_{j_0}^{k_0,t} \neq \delta$, are excluded. This means in particular that for $k \neq k_0$ each subproblem

$$\zeta_k = \max \quad ((c^k)^T x - \pi^T A^k)x - \mu_k \tag{10.23a}$$

$$x \in X^k = \left\{x^k \in \mathbb{B}_+^{n_k} : D^k x^k \leq d^k\right\} \tag{10.23b}$$

remains unchanged. For $k = k_0$ and $\delta = \{0,1\}$ the k_0th subproblem is

$$\zeta_{k_0} = \max \quad ((c^{k_0})^T x - \pi^T A^{k_0})x - \mu_{k_0} \tag{10.24a}$$

$$x \in X^k \cap \{x : x_j = \delta\}. \tag{10.24b}$$

11

More About Lagrangean Duality

Consider the integer program:

$$z = \max\left\{c^T x : Dx \leq d, x \in X\right\} \tag{11.1}$$

with

$$X = \{Ax \leq b, x \in \mathbb{Z}^n\} \tag{11.2}$$

and $Dx \leq d$ are some "complicating constraints" (D is a $k \times n$ matrix and $d \in \mathbb{R}^k$). In Section 6.3 we introduced the *Lagrangean Relaxation*

$$\text{IP}(u) \quad z(u) = \max\left\{c^T x + u^T(d - Dx) : x \in X\right\} \tag{11.3}$$

for fixed $u \geq 0$ and showed that IP(u) is a relaxation for (11.1) (Lemma 6.16). We then considered the *Lagrangean Dual* for (11.1)

$$(\text{LD}) \quad w_{\text{LD}} = \min\{z(u) : u \geq 0\}. \tag{11.4}$$

Clerly, $w_{\text{LD}} \geq z$, and Theorem 6.18 gave us precise information about the relation between w_{LD} and z, namely,

$$w_{\text{LD}} = \max\left\{c^T x : Dx \leq d, x \in \text{conv}(X)\right\}.$$

In this chapter we will learn more about Lagrangean duality. We will investigate how we can use information from a Lagrangean Relaxation to determine values for the variables in the original problem (variable fixing) and we will also sketch how to actually solve the Lagrangean Dual.

11.1 Convexity and Subgradient Optimization

In the proof of Theorem 6.18 we derived two Linear Programming formulation for solving the Lagrangean dual (11.4):

$$w_{\text{LD}} = \min t \tag{11.5a}$$

$$t + (Dx^k - d)^T u \geq c^T x^k \text{ for } k \in K \tag{11.5b}$$

$$(Dr^j)^T u \geq c^T r^j \text{ for } j \in J \tag{11.5c}$$

$$u \in \mathbb{R}^m_+, t \in \mathbb{R} \tag{11.5d}$$

and

$$\text{(11.6a)} \qquad w_{LD} = \max c^T \left(\sum_{k \in K} \alpha^k x^k + \sum_{j \in J} \beta_j r^j \right)$$

$$\text{(11.6b)} \qquad \sum_{k \in K} \alpha_k = 1$$

$$\text{(11.6c)} \qquad D \left(\sum_{k \in K} \alpha^k x^k + \sum_{j \in J} \beta_j r^j \right) \leq d(\sum_{k \in K} \alpha_k)$$

$$\text{(11.6d)} \qquad \alpha_k, \beta_j \geq 0, \text{ for } k \in K, j \in J.$$

Here, x^k, $k \in K$ are the extreme points and r^j, $j \in J$ are the extreme rays of $\text{conv}(X)$, where X is defined in (11.2). Formulation (11.5) contains a huge number of constraints, so a practially efficient solution method will have to use a cutting-plane algorithm. Formulation (11.6) contains a huge number of constraints, in fact, it is a Dantzig-Wolfe reformulation and we may solve it by column generation (see Chapter 10).

In this section we describe an alternative approach to solve the Lagrangean dual. This approach is comparatively simple, easy to implement and exploits the fact that the function $z(u)$, defined in (11.3) is convex and piecewise linear (we will prove this fact in a minute).

Definition 11.1 (Convex Function)
*A function $f\colon \mathbb{R}^n \to \mathbb{R}$ is **convex**, if*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

for all $x, y \in \mathbb{R}^n$ and all $\lambda \in [0,1]$.

The definition above simply states that for any point $\lambda x + (1-\lambda)y$ on the segment joining x and y the function value $f(\lambda x + (1-\lambda)y)$ lies below the segment connecting $f(x)$ and $f(y)$. Figure 11.1 provides an illustration for the one-dimensional case.

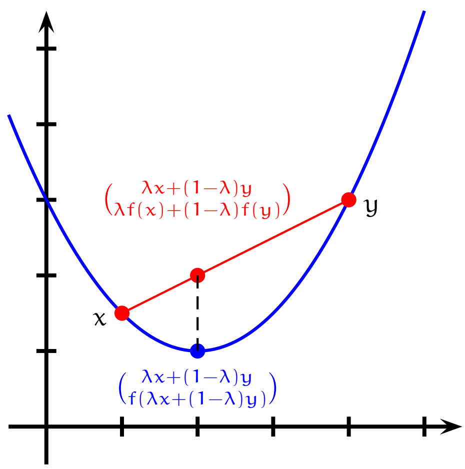


Figure 11.1: A convex function.

Convex functions have a the nice property that local minimal are also global minima:

Definition 11.2 (Local and global minimum)
The point $x^* \in \mathbb{R}^n$ *is a* ***local minimum*** *of* $f\colon \mathbb{R}^n \to \mathbb{R}$ *if there is* $\delta > 0$ *such that* $f(x) \geq f(x^*)$ *for all* $\|x - x^*\| \leq \delta$*. The point* x^* *is a* ***global minimum*** *if* $f(x) \geq f(x^*)$ *for all* $x \in \mathbb{R}^n$*.*

Lemma 11.3 *Let* x^* *be a local minimum of the convex function* $f\colon \mathbb{R}^n \to \mathbb{R}$*. Then,* x^* *is also a global minimum.*

Proof: Let $\delta > 0$ be such that $f(x) \geq f(x^*)$ for all $\|x - x^*\| \leq \delta$. Suppose that $\bar{x} \in \mathbb{R}^n$ with $f(\bar{x}) < f(x^*)$. Let $y := x^* + \delta(\bar{x} - x^*)/\|\bar{x} - x^*\|$. Then, $\|y - x^*\| = \delta$.

$$
\begin{aligned}
f(y) &= f((1 - \frac{\delta}{\|\bar{x} - \bar{x}^*\|})x^* + \frac{\delta}{\|\bar{x} - \bar{x}^*\|}\bar{x}) \\
&\leq (1 - \frac{\delta}{\|\bar{x} - \bar{x}^*\|})f(x^*) + \frac{\delta}{\|\bar{x} - \bar{x}^*\|}f(\bar{x}) \\
&< (1 - \frac{\delta}{\|\bar{x} - \bar{x}^*\|})f(x^*) + \frac{\delta}{\|\bar{x} - \bar{x}^*\|}f(x^*) \\
&= f(x^*).
\end{aligned}
$$

This contradicts the fact that x^* is a local minimum. □

It can be shown that a differentiable function $f\colon \mathbb{R}^n \to \mathbb{R}$ is convex if and only if

$$f(x) \geq f(x^*) + \nabla f(x^*)(x - x^*)$$

for all $x, x^* \in \mathbb{R}^n$, see e.g. [Lue84]. Here, $\nabla f(x^*)$ is the gradient of f at x^*. So x^* is a local (and by Lemma 11.3 also a global) minimizer of a convex differentiable function $f\colon \mathbb{R}^n \to \mathbb{R}$ if and only if $\nabla f(x^*) = 0$. The notion of a subgradient is meant as an extension to the nondifferentiable (and possibly non-smooth) case:

Definition 11.4 (Subgradient, subdifferential)
Let $f\colon \mathbb{R}^n \to \mathbb{R}$ *be convex. The vector* $s \in \mathbb{R}^n$ *is a* ***subgradient*** *of* f *at* x^* *if*

$$f(x) \geq f(x^*) + s^T(x - x^*)$$

for all $x \in \mathbb{R}^n$*. The* ***subdifferential*** $\partial f(x)$ *of* f *at* x *is the set of all subgradients of* f *at* x*.*

Lemma 11.5 *Let* $f\colon \mathbb{R}^n \to \mathbb{R}$ *be convex. Then,* x^* *is an optimal solution of* $\min\{f(x) : x \in \mathbb{R}^n\}$ *if and only if* $0 \in \partial f(x^*)$*.*

Proof: We have

$$
\begin{aligned}
0 \in \partial f(x^*) &\Leftrightarrow f(x) \geq f(x^*) + 0^T(x - x^*) \text{ for all } x \in \mathbb{R}^n \\
&\Leftrightarrow f(x) \geq f(x^*) \text{ for all } x \in \mathbb{R}^n
\end{aligned}
$$

□

Algorithm 11.1 Subgradient algorithm for minimizing a convex function $f\colon \mathbb{R}^n \to \mathbb{R}$.

SUBGRADIENT-ALG(f)

Choose a starting point $x^0 \in \mathbb{R}^n$ and set $k := 0$.
Choose any subgradient $s \in \partial f(x^k)$.
while $s \neq 0$ **do**
  Set $x^{k+1} := x^k - \theta_k s$ for some $\theta_k > 0$.
  Set $k = k + 1$.
end while
x^{k-1} is an optimal solution. **stop**.

The *subgradient algorithm* (see Algorithm 11.1) is designed to solve problems of the form

$$\min\{f(x) : x \in \mathbb{R}^n\},$$

where f is convex. At any step, it chooses an arbitrary subgradient and moves into that direction. Of course, the question arises how to get a subgradient, which subgradient to choose and how to select the steplengths θ_k.

We will not elaborate on this in the general setting and restrict ourselves to the special case of the Lagrangean dual.

11.2 Subgradient Optimization for the Lagrangean Dual

Before we start to apply subgradient descent methods to the Lagrangean dual, we will first prove that the function we wish to minimize actually possesses the desired structural properties.

Lemma 11.6 *Let* $f_1, \dots, f_m\colon \mathbb{R}^n \to \mathbb{R}$ *be convex functions and*

$$f(x) = \max\{f_i(x) : i = 1, \dots, m\}$$

be the pointwise maximum of the f_i. *Then,* f *is convex.*

Proof: Let $x, y \in \mathbb{R}^n$ and $\lambda \in [0, 1]$. For $i = 1, \dots, m$ we have from the convexity of f_i

$$f_i(\lambda x + (1-\lambda)y) \leq \lambda f_i(x) + (1-\lambda) f_i(y).$$

Thus,

$$\begin{aligned} f(\lambda x + (1-\lambda)y) &= \max\{f_i(\lambda x + (1-\lambda)y) : i = 1, \dots, m\} \\ &\leq \max\{\lambda f_i(x) + (1-\lambda) f_i(y) : i = 1, \dots, m\} \\ &\leq \lambda \max\{f_i(x) : i = 1, \dots, m\} + (1-\lambda)\max\{f_i(y) : i = 1, \dots, m\} \\ &= \lambda f(x) + (1-\lambda) f(y). \end{aligned}$$

Thus, f is convex. □

Theorem 11.7 *The function* $z(u)$ *defined in* (11.3) *is piecewise linear and convex on the domain over which it is finite.*

Proof: Let x^k, $k \in K$ be the extreme points and r^j, $j \in J$ be the extreme rays of $\operatorname{conv}(X)$. Fix $u \geq 0$. We have (see also Theorem 6.18):

$$
\begin{aligned}
z(u) &= \max\left\{c^T x + u^T(d - Dx) : x \in \operatorname{conv}(X)\right\} \\
&= \begin{cases} +\infty & \text{if } (c^T - u^T D)r^j > 0 \text{ for some } j \in J \\ c^T x^k + u^T(d - Dx^k) & \text{for some } k \in K \text{ otherwise.} \end{cases}
\end{aligned}
$$

So, $z(u)$ is finite if and only if u is contained in the polyhedron

$$Q := \left\{y \in \mathbb{R}^m_+ : u^T D r^j \geq c^T r^j \text{ for all } j \in J\right\}$$

If $u \in Q$, then

$$z(u) = u^T d + \max\left\{(c^T - u^T D)x^k : k \in K\right\}$$

is the maximum of a finite set of affine functions $f_k(u) = u^T d + (c^T - u^T D)x^k$, $(k \in K)$. Since affine functions are convex, the convexity of $z(u)$ follows from Lemma 11.6. □

The above theorem shows that the function occuring in the Lagrangean dual is a particular convex function: it is piecewise linear. This property enables us to derive subgradients easily:

Lemma 11.8 *Let $\bar{u} \geq 0$ and $x(\bar{u})$ be an optimal solution of the Lagrangean relaxation IP($\bar{u}$) given in (11.3). Then, the vector $d - Dx(\bar{u})$ is a subgradient of $z(u)$ at $\bar{u}$.*

Proof: For any $u \geq 0$ we have

$$
\begin{aligned}
z(u) &= \max\left\{c^T x + u^T(d - Dx) : x \in X\right\} \\
&\geq c^T x(\bar{u}) + u^T(d - Dx(\bar{u})) \\
&= c^T x(u) + \bar{u}^T(d - Dx(\bar{u})) + (u - \bar{u})^T(d - Dx(\bar{u})) \\
&= z(u) + (u - \bar{u})^T(d - Dx(\bar{u})) \\
&= z(u) + (d - Dx(\bar{u}))^T(u - \bar{u})
\end{aligned}
$$

This proves the claim. □

Algorithm 11.2 Subgradient algorithm for solving the Lagrangean dual.

SUBGRADIENT-LD

Choose a starting dual vector $u^0 \in \mathbb{R}^n_+$ and set $k := 0$.
repeat
  Solve the Lagrangean Problem:

$$\text{IP}(u^t) \quad z(u^k) = \max\left\{c^T x + (u^k)^T(d - Dx) : x \in X\right\}$$

  and let $x(u^k)$ be its optimal solution.
  Set $s := d - Dx(u^k)$, then s^t is a subgradient of $z(u)$ at u^k (Lemma 11.8).
  Set $u^{k+1} := \max\{u^k - \theta_k s, 0\}$ for some $\theta_k > 0$.
  Set $k := k + 1$.
until $s = 0$
u^{k-1} is an optimal solution. **stop**.

We state the following theorem without proof:

Theorem 11.9 *Let θ_t be the sequence of set lengths chosen by Algorithm 11.2.*

(i) If $\lim_{k\to\infty}\theta_k = 0$ *and* $\sum_{k=0}^{\infty}\theta_k = \infty$, *then* $\lim_{k\to\infty} z(u^k) = w_{LD}$.

(ii) If $\theta_k = \theta_0\rho^k$ *for some* $0 < \rho < 1$, *then* $\lim_{k\to\infty} z(u^k) = w_{LD}$ *if* θ_0 *and* ρ *are sufficiently large.*

(i) If $\bar{w} \geq w_{LD}$ *and* $\theta_k = \varepsilon_k \frac{z(u^k)-\bar{w}}{\|d-Dx(u^k)\|}$ *with* $0 < \varepsilon_k < 2$, *then* $\lim_{k\to\infty} z(u^k) = w_{LD}$, *or the algorithm finds* u^k *with* $\bar{w} \geq z(u^k) \geq w_{LD}$ *for some finite* k.

□

11.3 Lagrangean Heuristics and Variable Fixing

Suppose that we solve the Lagrangean dual by some algorithm, e.g. by the subgradient method described in the previous section. It makes sense to hope that once the multipliers u^k approach the optimal solution (of the dual), we may possibly extract some information about the original IP (11.1). In particular, we would hope that $x(u^k)$, the optimal solution of the problem 11.3 is «close » to an optimal solution of (11.1).

In this section we examine this idea for a particular example, the set-covering problem (see also Example 1.9 on page 8). In this problem we are given a finite ground set U and a collection $F \subseteq 2^N$ of subsets of N. There is a cost c_f associated with every set $f \in F$. We wish to find a subcollection of the sets in F of minimum cost such that each element in U is covered at least once:

$$\min\left\{\sum_{f\in F} c_f x_f : \sum_{f\in F} a_{if}x_f \geq 1 \text{ for } i \in N, x \in \mathbb{B}^F\right\}. \tag{11.7}$$

Here, as in Example 1.9 on page 8)

$$a_{if} := \begin{cases} 1 & \text{if element } i \text{ is contained in set } f \\ 0 & \text{otherwise.}\end{cases}$$

Let us consider the Lagrangean relaxation of (11.7) where we dualize all covering constraints. For $u \geq 0$ we have:

$$\begin{aligned} z(u) &= \min\left\{\sum_{f\in F} c_f x_f + \sum_{i\in N} u_i\left(1 - \sum_{f\in F} a_{if}x_f\right), x \in \mathbb{B}^F\right\} \\ &= \min\left\{\sum_{i\in N} u_i + \sum_{f\in F}\left(c_f - \sum_{i\in N} u_i a_{if}\right) x_f : x \in \mathbb{B}^F\right\} \end{aligned} \tag{11.8}$$

Observe that the problem (11.8) is trivial to solve. Let

$$\begin{aligned} F^+ &:= \left\{f \in F : c_f - \sum_{i\in N} u_i a_{if} > 0\right\} \\ F^- &:= \left\{f \in F : c_f - \sum_{i\in N} u_i a_{if} < 0\right\} \\ F^0 &:= \left\{f \in F : c_f - \sum_{i\in N} u_i a_{if} = 0\right\} \end{aligned}$$

We set $x_f = 1$ if $f \in F^-$ and $x_f := 0$ if $f \in F^+ \cup F^0$. otherwise. So, given $u \geq 0$, we can easily calculate $x(u)$ and the corresponding optimal value $z(u)$.

Given $x(u) \in \mathbb{B}^F$, we can use this vector to obtain a feasible solution x^H of (11.7): We drop all rows $i \in N$ where $\sum f \in F a_{if} x_f \geq 1$ and solve the remaining smaller set-covering problem by a greedy-heuristic (always choose a set such that the ratio of the cost over the number of elements newly covered is minimized). If y^* is the heuristic solution, then $x^H := x(u) + y^*$ is a feasible solution to (11.7).

Once we are given x^H we can use the dual information in the multipliers u for *variable fixing*. Suppose that $\bar{x}$ is a feasible solution for (11.7) with $c^T \bar{x} < c^T x^H$. Since we are dealing with a relaxation, we have

$$\sum_{i \in N} u_i + \sum_{f \in F} \left(c_f - \sum_{i \in N} u_i a_{if} \right) \bar{x}_f \leq c^T x < c^T x^H. \tag{11.9}$$

Let $f \in F^+$ and suppose that $\sum_{i \in N} u_i + \sum_{f \in F^-} \left(c_f - \sum_{i \in N} u_i a_{if} \right) \geq c^T x^H$. Then, (11.9) can only hold if $\bar{x}_f = 0$.

Similarly, let $f \in F^-$ and suppose that $\sum_{i \in N} u_i + \sum_{f \in F^- \setminus \{f\}} \left(c_f - \sum_{i \in N} u_i a_{if} \right) \geq c^T x^H$. Then, for (11.9) to hold we must have $\bar{x}_f = 1$.

A

Notation

This chapter is intended mainly as a reference for the notation used in these lecture notes and the foundations this work relies on. We assume that the reader is familiar with elementary graph theory, graph algorithmic concepts, and combinatorial optimization as well as with basic results from complexity theory. For detailed reviews we refer the reader to monographs and textbooks which are listed at the end of this chapter.

A.1 Basics

By $\mathbb{R}$ ($\mathbb{Q}$, $\mathbb{Z}$, $\mathbb{N}$) we denote the set of real (rational, integral, natural) numbers. The set $\mathbb{N}$ of natural numbers does not contain zero. $\mathbb{R}_0^+$ ($\mathbb{Q}_0^+$, $\mathbb{Z}_0^+$) denotes the nonnegative real (rational, integral) numbers.

The rounding of real numbers $x \in \mathbb{R}_+$ is denoted by the notation $\lfloor x \rfloor := \max\{\, n \in \mathbb{N} \cup \{0\} : n \leq x \,\}$ and $\lceil x \rceil := \min\{\, n \in \mathbb{N} : n \geq x \,\}$.

By 2^S we denote the *power set* of a set S, which is the set of all subsets of set S (including the empty set $\varnothing$ and S itself).

A.2 Sets and Multisets

A *multiset* Y over a ground set U, denoted by $Y \sqsubset U$, can be defined as a mapping $Y\colon U \to \mathbb{N}$, where for $u \in U$ the number $Y(u)$ denotes the multiplicity of u in Y. We write $u \in Y$ if $Y(u) \geq 1$. If $Y \sqsubset U$ then $X \sqsubset Y$ denotes a multiset over the ground set $\{\, u \in U : Y(u) > 0 \,\}$.

If $Y \sqsubset U$ and $Z \sqsubset U$ are multisets over the same ground set U, then we denote by $Y + Z$ their *multiset union*, by $Y - Z$ their *multiset difference* and by $Y \cap Z$ their *multiset intersection*, defined for $u \in U$ by

$$
\begin{aligned}
(Y + Z)(u) &= Y(u) + Z(u) \\
(Y - Z)(u) &= \max\{Y(u) - Z(u), 0\} \\
(Y \cap Z)(u) &= \min\{Y(u), Z(u)\}.
\end{aligned}
$$

The multiset $Y \sqsubset U$ is a subset of the multiset $Z \sqsubset U$, denoted by $Y \subseteq Z$, if $Y(u) \leq Z(u)$ for all $u \in U$. For a weight function $c\colon U \to \mathbb{R}$ the weight of a multiset $Y \sqsubset U$ is defined by $c(Y) := \sum_{u \in U} c(u)Y(u)$. We denote the cardinality of a multiset $Y \sqsubset U$ by $|Y| := \sum_{u \in U} Y(u)$.

Any (standard) set can be viewed as a multiset with elements of multiplicity 0 and 1. If X and Y are two standard sets with $X \subseteq Y$ and $X \neq Y$, then X is a *proper* *subset* of Y, denoted by $X \subset Y$. Two subsets $X_1 \subseteq Y$, $X_2 \subseteq Y$ of a standard set Y form a *partition* of Y, if $Y = X_1 \cup X_2$ and $X_1 \cap X_2 = \emptyset$.

A.3 Analysis and Linear Algebra

Reference books: [Rud76]

A *metric space* (X, d) consists of a nonempty set X and a *distance function* or *metric* $d\colon X \times X \to \mathbb{R}_+$ which satisfies the following three conditions:

(i) $d(x, y) > 0$ if $x \neq y$; $d(x, x) = 0$;

(ii) $d(x, y) = d(y, x)$;

(iii) $d(x, y) \leq d(x, z) + d(z, y)$ for all $z \in X$.

Inequality (iii) is called the *triangle inequality*. An example of a metric space is the set $\mathbb{R}^p$ endowed with the Euclidean metric which for vectors $x = (x_1, \dots, x_p) \in \mathbb{R}^p$ and $y = (y_1, \dots, y_p) \in \mathbb{R}^p$ is defined by

$$d(x, y) := \left(\sum_{i=1}^{p} (x_i - y_i)^2 \right)^{1/2} .$$

This metric space is usually referred to as the *Euclidean space*.

A *path* in a metric space (X, d) is a continuous function $\gamma\colon [0, 1] \to X$. The path γ is called *rectifiable*, if for all dissections $0 = t_0 < t_1 < \cdots < t_k = 1$ of the interval $[0, 1]$ the sum

$$\sum_{i=1}^{k} d(\gamma(t_i), \gamma(t_{i-1}))$$

is bounded from above. The supremum of the sums, taken over all dissections, is then referred to as the *length* of the path γ.

A.4 Growth of Functions

Reference books: [CLR90, AHU74]

Let g be a function from $\mathbb{N}$ to $\mathbb{N}$. The set $\mathcal{O}(g(n))$ contains all those functions $f\colon \mathbb{N} \to \mathbb{N}$ with the property that there exist constants $c > 0$ and $n_0 \in \mathbb{N}$ such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. A function f belongs to the set $\Omega(g(n))$, if and only if $g(n) \in \mathcal{O}(f(n))$. The notation $f(n) \in \Theta(g(n))$ means that $f(n) \in \mathcal{O}(g(n))$ and $f(n) \in \Omega(g(n))$. Finally, we write $f(n) \in o(g(n))$, if $\lim\limits_{n \to \infty} f(n)/g(n) = 0$.

A.5 Particular Functions

We use $\log_a$ to denote the *logarithm* function to the basis of a. We omit the basis in the case of $a = 2$ for the sake of convenience. By $\ln n$ we denote the *natural logarithm* of a positive number n, that is, $\ln n := \log_e n$.

A.6 Probability Theory

Reference books: [Fel68, Fel71, MR95]

A *probability space* $(\Omega, \mathbb{F}, \Pr)$ consists of a σ-field $(\Omega, \mathbb{F})$ with a probability measure Pr defined on it. When specifying a probability space, $\mathbb{F}$ may be omitted which means that the σ-field referred to is $(\Omega, 2^{\Omega})$.

In this thesis we are mainly concerned with the case that Ω is either the the set of real numbers $\mathbb{R}$ or an interval contained in $\mathbb{R}$. In this context a *density function* is a non-negative function $p\colon \mathbb{R} \to \mathbb{R}_+$ whose integral, extended over the real numbers, is unity, that is $\int_{-\infty}^{+\infty} p(x)\,dx = 1$. The density corresponds to the probability measure Pr, which satisfies

$$\Pr[\, x \in (-\infty, t] \,] = \int_{-\infty}^{t} p(x)\,dx.$$

A.7 Graph Theory

Reference books: [Har72, AMO93]

A *mixed graph* $G = (V, E, R)$ consists of a set V of *vertices* (or *nodes*), a set E of *undirected edges*, and a multiset R of *directed arcs*. We usually denote by $n := |V|$, $m_E := |E|$ and $m_R := |R|$ the number of vertices, edges and arcs, in G respectively. Throughout the thesis we assume that V, E, and R are all finite. If $R = \varnothing$, we briefly write $G = (V, E)$ and call G an *undirected graph* (or simply *graph*) with vertex set V and edge set E. If $E = \varnothing$, we refer to $G = (V, R)$ as a *directed graph* with vertex set V and arc (multi-) set R.

Each undirected edge is an unordered pair $[u, v]$ of distinct vertices $u \neq v$. The edge $[u, v]$ is said to be *incident* to the vertices u and v. Each arc is an ordered pair (u, v) of vertices which is incident to both u and v. We refer to vertex u as the *source* of arc (u, v) and to vertex v as its *target*. The arc (u, v) *emanates* from vertex u and *terminates* at vertex v. An arc (u, v) is *incident* to both vertices u and v. The arc (u, v) is an *outgoing arc* of node u and an *incoming arc* of vertex v. We call two vertices *adjacent*, if there is an edge or an arc which is incident with both of them.

Two arcs are called *parallel arcs* if they refer to copies of the same element (u, v) in the multiset R. Arcs (u, v) and (v, u) are termed *anti-parallel* or *inverse*. We write $(u, v)^{-1} := (v, u)$ to denote an inverse arc to (u, v). For a set R of arcs we denote by R^{-1} the set $R^{-1} := \{ r^{-1} : r \in R \}$.

Let $G = (V, E, R)$ be a mixed graph. A graph $H = (V_H, E_H, R_H)$ is a *subgraph* of G if $V_H \subseteq V$, $E_H \subseteq E$ and $R_H \subseteq R$. For a multiset $X \sqsubset E + R$ we denote by $G[X]$ the *subgraph of* G *induced by* X, that is, the subgraph of G consisting of the arcs and edges in X together with their incident vertices. A subgraph of G *induced by vertex set* $X \subseteq V$ is a subgraph with node set X and containing all those edges and arcs from G which have both endpoints in X.

For $v \in V$ we let R_v be the set of arcs in R emanating from v. The *outdegree* of a vertex v in G, denoted by $\deg^+_G(v)$, equals the number of arcs in G leaving v. Similarly, the *indegree* $\deg^-_G(v)$ is defined to be the number of arcs entering v. If $X \sqsubset R$, we briefly write $\deg^+_X(v)$ and $\deg^-_X(v)$ instead of $\deg^+_{G[X]}(v)$ and $\deg^-_{G[X]}(v)$. The *degree* of a vertex v in an undirected graph $G = (V, E)$ is defined to be the number of edges incident with v.

A subset C of the vertices of an undirected graph $G = (V, E)$ such that every pair of vertices is adjacent is called a *clique* of size $|C|$ in the graph G. A graph G whose vertex set forms a clique is said to be a *complete graph*.

A *path* P in an undirected graph $G = (V, E)$ is defined to be an alternating sequence $p = (v_1, e_1, v_2, \ldots, e_k, v_{k+1})$ of nodes $v_i \in V$ and edges $e_i \in E$, where for each triple (v_i, e_i, v_{i+1}) we have $e_i = (v_i, v_{i+1})$. We use equivalently the alternative notation $P = (v_1, v_2, \ldots, v_{k+1})$ and $P = (e_1, e_2, \ldots, e_k)$ when the meaning is clear. For directed graphs $G = (V, R)$, edges are replaced by arcs, and we require $r_i = (v_i, v_{i+1})$ and $r_i \in R \cup R^{-1}$ for each triple. If the stronger condition $r_i \in R$ holds, the path is called *directed*.

For mixed graphs, we define a *walk* which traverses arbitrarily edges and directed arcs. An *oriented walk* is a “directed version” of a walk in the sense that for any two consecutive vertices v_i and v_{i+1} we require that either x_i is an undirected edge $[v_i, v_{i+1}]$ between v_i and v_{i+1} or a directed arc (v_i, v_{i+1}) from v_i to v_{i+1}.

If all nodes of the path or walk are pairwise different (without considering the pair v_1, v_{k+1}), the path or walk is called *simple*. A path or walk with coincident start and endpoint is *closed*. A closed and simple path or walk is a *cycle*. An *Eulerian cycle* in a directed graph $G = (V, R)$ is a directed cycle which contains (traverses) every arc from R exactly once. The directed graph G is called *Eulerian* if it contains an Eulerian cycle. A *Hamiltonian path Hamiltonian cycle*) is a simple path (cycle) which touches every vertex in a directed (or undirected) graph.

A mixed graph $G = (V, E, R)$ is *connected* (*strongly connected*), if for every pair of vertices $u, v \in V$ with $u \neq v$ there is an walk (oriented walk) from u to v in G. A (strongly) connected subgraph of G which is maximal with respect to set inclusion is called *(strongly) connected component* of G.

A *tree* is a connected graph that contains no cycle. A node in a tree is called a *leaf* if its degree equals 1, and an *inner node* otherwise. A *spanning tree* of a graph G is a tree which has the same vertex set as G.

A *Steiner tree* with respect to a subset K of the vertices of an undirected graph G, is a tree which is a subgraph of G and whose vertex set includes K. The vertices in K are called *terminals*.

A *directed in-tree rooted at* $o \in V$ is a subgraph of a directed graph $H = (V, A)$ which is a tree and which has the property that for each $v \in V$ it contains a directed path from v to o.

Additional definitions to the basic ones presented above will be given in the respective contexts.

A.8 Theory of Computation

Reference books: [GJ79, Pap94, GLS88, CLR90]

Model of Computation

The *Turing machine* [GJ79] is the classical model of computation that was used to define the *computational complexity* of algorithms. However, for practical purposes it is fairly more convenient to use a different model. In the *random access machine* or *RAM model* [Pap94, MR95] we have a machine which consists of an infinite array of registers, each capable of containing an arbitrarily large integer, possibly negative. The machine is capable of performing the following types of operations involving registers and main memory: input-output operations, memory-register transfers, indirect addressing, arithmetic operations and branching. The arithmetic operations permitted are addition, subtraction, multiplication and division of numbers. Moreover, the RAM can compare two numbers and evaluate the square root of a positive number.

There are two types of RAM models used in literature. In the *log-cost RAM* the execution time of each instruction takes time proportional to the encoding length, i.e. proportional to the logarithm of the size of its operands, whereas in the *unit-cost RAM* each instruction can be accomplished in one time step. A log-cost RAM is equivalent to the Turing machine under a polynomial time simulation [Pap94]. In contrast, in general there is no polynomial simulation for a unit-cost RAM, since in this model we can compute large integers too quickly by using multiplication. However, if the encoding lengths of the operands occurring during the run of an algorithm on a unit-cost RAM are bounded by a polynomial in the encoding length of the input, a polynomial time algorithm on the unit-cost RAM will transform into a polynomial time algorithm on a Turing machine [GLS88, Pap94]. This argument remains valid in the case of nondeterministic programs.

For convenience, we will use the general unit-cost RAM to analyze the running time of our algorithms. This does not change the essence of our results, because the algorithms in which we are interested involve only operations on numbers that are not significantly larger than those in the input.

Computational Complexity

Classical complexity theory expresses the running time of an algorithm in terms of the "size" of the input, which is intended to measure the amount of data necessary to describe an instance of a problem. The running time of an algorithm on a specific input is defined to be the sum of times taken by each instruction executed. The *worst case time complexity* or simply *time complexity* of an algorithm is the function $T(n)$ which is the maximum running time taken over all inputs of size n (cf. [AHU74, GJ79, GLS88]).

An *alphabet* Σ is a nonempty set of characters. By Σ^* we denote the set of all strings over Σ including the empty word. We will assume that every problem Π has an (encoding independent) associated function *length* : $D_\Pi \to \mathbb{N}$, which is polynomially related to the input lengths that would result from a "reasonable encoding scheme". Here, $D_\Pi \subseteq \Sigma^*$ is the set of *instance*s of the problem Π, expressed as words over the alphabet Σ. For a more formal treatment of the input length and also of the notion of a "reasonable encoding scheme" we refer to [GJ79].

A *decision problem* is a problem where each instance has only one of two outcomes from the set $\{\text{yes}, \text{no}\}$. For a nondecreasing function $f\colon \mathbb{N} \to \mathbb{N}$ the *deterministic time complexity class* $\mathsf{DTIME}(f(n))$ consists of the decision problems for which there exists a deterministic Turing machine deciding the problem in $\mathcal{O}(f(n))$ time. Its nondeterministic counterpart $\mathsf{NTIME}(f(n))$ is defined analogously. The most important complexity classes with respect to this thesis are

$$\mathsf{P} := \bigcup_{k=1}^{\infty} \mathsf{DTIME}(n^k) \qquad \text{and} \qquad \mathsf{NP} := \bigcup_{k=1}^{\infty} \mathsf{NTIME}(n^k)\,.$$

Suppose we are given two decision problems Π and Π'. A *polynomial time transformation* is an algorithm t which, given an encoded instance I of Π, produces in polynomial time an encoded instance $t(I)$ of Π' such that the following holds: For every instance I of Π, the answer to Π is "yes" if and only if the answer to the transformation $t(I)$ (as an instance of Π') is "yes". A decision problem Π is called NP-*complete* if $\Pi \in \mathsf{NP}$ and every other decision problem in NP can be transformed to Π in polynomial time.

To tackle also optimization problems rather than just decision problems it is useful to extend the notion of a transformation between problems. Informally, a *polynomial time Turing reduction* (or just *Turing reduction*) from a problem Π to a problem Π' is an algorithm ALG, which solves Π by using a hypothetical subroutine ALG$'$ for solving Π' such that if ALG$'$ were a polynomial time algorithm for Π', then ALG would be a polynomial time algorithm for Π. More precisely, a polynomial time Turing reduction from Π to Π' is a deterministic polynomial time oracle Turing machine (with oracle Π') solving Π.

An optimization problem Π is called NP-*hard* ("at least as difficult as any problem in NP"), if there is an NP-complete decision problem Π' such that Π' can be Turing reduced to Π. Results from complexity theory (see e.g. [GJ79]) show that such an NP-hard optimization problem can not be solved in polynomial time unless $\mathsf{P} = \mathsf{NP}$.

B

Symbols

$\varnothing$	the empty set
$\mathbb{Z}$	the set of integers, that is, $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
$\mathbb{Z}_+$	the set of nonnegative integers $\mathbb{Z}_+ = \{0, 1, 2, \dots\}$
$\mathbb{N}$	the set of natural numbers, $\mathbb{N} = \mathbb{Z}_+$
$\mathbb{Q}$	the set of rational numbers
$\mathbb{Q}_+$	the set of nonnegative rational numbers
$\mathbb{R}$	the set of real numbers
$\mathbb{R}_+$	the set of nonnegative real numbers
2^A	the set of subsets of the set A
$\lvert A \rvert$	the cardinality of the (multi-) set A
$A \subseteq B$	A is a (not necessarily proper) subset of B
$A \subset B$	A is a proper (multi-) subset of B
$Y \sqsubset U$	Y is a multiset over the ground set U
$f(n) \in \mathcal{O}(g(n))$	f grows at most as fast as g (see page 166)
$f(n) \in \Omega(g(n))$	f grows at least as fast as g (see page 166)
$f(n) \in \Theta(g(n))$	f grows exactly as fast as g (see page 166)
$f(n) \in o(g(n))$	f grows more slowly than g (see page 166)
$\log_a$	logarithm to the basis of a
$\log$	logarithm to the basis of 2
$\ln$	natural logarithm (basis e)
$G = (V, E)$	undirected graph with vertex set V and edge set E
$G = (V, A)$	directed graph with vertex set V and edge set E
$\delta(S)$	set of edges that have exactly one endpoint in S
$x(S)$	$x(S) = \sum_{s \in S} x_s$

Bibliography

[Ada96] S. Adams, *The Dilbert principle*, HarperCollins, New York, 1996.

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1974.

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Networks flows*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[BDG88] J. L. Balcázar, J. Díaz, and J. Gabarró, *Structural complexity I*, EATCS monographs on theoretical computer science, Springer, 1988.

[BDG90] J. L. Balcázar, J. Díaz, and J. Gabarró, *Structural complexity II*, EATCS monographs on theoretical computer science, Springer, 1990.

[CC$^+$98] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver, *Combinatorial optimization*, Wiley Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, 1998.

[Chv83] V. Chvátal, *Linear programming*, W. H. Freeman and Company, 1983.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, MIT Press, 1990.

[Coo71] S. A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing, 1971, pp. 151–158.

[Fel68] W. Feller, *An introduction to probability theory and its applications*, 3 ed., vol. 1, John Wiley & Sons, Inc., 1968.

[Fel71] W. Feller, *An introduction to probability theory and its applications*, 2 ed., vol. 2, John Wiley & Sons, Inc., 1971.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and intractability (a guide to the theory of* NP*-completeness)*, W.H. Freeman and Company, New York, 1979.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer-Verlag, Berlin Heidelberg, 1988.

[Har72] F. Harary, *Graph theory*, Addison-Wesley Publishing Company, Inc., 1972.

[Kar72] R. M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (R. E. Miller and J. W. Thatcher, eds.), Plenum Press, New York, 1972, pp. 85–103.

[KN05] S. O. Krumke and H. Noltemeier, *Graphentheorische Konzepte und Algorithmen*, B.G. Teubner, 2005 (ngerman).

[Kru04] S. O. Krumke, *Integer programming*, Lecture notes, `http://www.mathematik.uni-kl.de/~krumke`, 2004.

[Lue84] D. G. Luenberger, *Linear and nonlinear programming*, 2 ed., Addison-Wesley, 1984.

[MR95] R. Motwani and P. Raghavan, *Randomized algorithms*, Cambridge University Press, 1995.

[NW99] G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*, Wiley-Interscience series in discrete mathematics and optimization, John Wiley & Sons, 1999.

[Pap94] C. M. Papadimitriou, *Computational complexity*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1994.

[Rud76] W. Rudin, *Principles of mathematical analysis*, 3 ed., McGraw Hill, 1976.

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

[Sch03] A. Schrijver, *Combinatorial optimization: Polyhedra and efficiency*, Springer, 2003.

[Wol98] L. A. Wolsey, *Integer programming*, Wiley-Interscience series in discrete mathematics and optimization, John Wiley & Sons, 1998.