UNIVERSITY OF OSLO
Department of Informatics

An introduction to convexity, polyhedral theory and combinatorial optimization

Geir Dahl

Kompendium 67
IN 330

September 19, 1997

An introduction to convexity, polyhedral theory and combinatorial optimization

Geir Dahl [1]

September 19, 1997

[1]University of Oslo, Institute of Informatics, P.O.Box 1080, Blindern, 0316 Oslo, Norway (Email:geird@ifi.uio.no)

Contents

Preface

This report consists of lecture notes for a graduate course in combinatorial optimization at the University of Oslo. A better, although longer, title of this course would be the report title "Convexity, polyhedral theory and combinatorial optimization".

The purpose of this report is to introduce the reader to convexity, polyhedral theory and the application of these areas to combinatorial optimization; the latter is known as polyhedral combinatorics. Compared to some other texts in polyhedral combinatorics we concentrate on the foundation in convex analysis. This is motivated by the fact that convexity leads to a good understanding of, e.g., duality and, in addition, convex analysis is very important for applied mathematics in general.

We assume that the student has a good general mathematical background and that she or he is familiar with mathematical proofs. More specifically, a good understanding of linear algebra is important. A secondary course in mathematical analysis is useful (topology, convergence etc.) as well as an introductory course in optimization. Some knowledge of algorithms and complexity is also required.

However, optimization is a part of applied mathematics, so we do want to model and solve problems. Therefore, discussions on methods, both general and more specific ones, follow the theoretical developments. Thus a main goal is that a (hard working) student after finishing this course shall (i) have a good theoretical foundation in polyhedral combinatorics, and (ii) have the ability (and interest) to study, model, analyze and "solve" difficult combinatorial problems arising in applications.

The present report should be viewed as a draft where several modifications are planned. For instance, some more material on matching and other combinatorial problems would be suitable. Still, judging from experience, it is hard (and not even desirable) to cover all the material in this report given the estimated work load of the mentioned course. This opens up for some flexibility in selecting material for lectures. It is also planned to find more exercises for a future version of the report.

It is suggested to read supplementary texts along with this report. Two highly recommended books are G.L. Nemhauser and L. Wolsey "Integer and combinatorial optimization" [29] and A. Schrijver "Theory of linear and integer programming" [35]. Another very good reference is the paper W.R. Pulleyblank "Polyhedral combinatorics" [32]. Although it has a different perspective, we recommend the book R. K. Ahuja, T. L. Magnanti and J. B. Orlin "Network flows: theory, algorithms, and applications" [33] where, in particular, many interesting applications are described.

Have fun!

Chapter 0

Introduction

0.1 Background and motivation

Mathematical optimization, or **mathematical programming**, is a fast growing branch of mathematics with a surprisingly short history. Most of its development has occured during the second half of this century. Basically one deals with the maximization (or minimization) of some function subject to one or more constraints.

As we know, mathematicians, for centuries (maybe even thousands of years) have studied linear algebraic *equations* and linear diophantine equations in a number of contexts. Problems came from e.g. mechanics, astronomy, geometry, economics and so on. Many of those problems had unique solutions, so there was really no "freedom" involved. However, more recently, problems have appeared from a number of applications, mathematical and non-mathematical, where there is typically several possible, or *feasible*, solutions to a problem. This naturally leads to the question of find a "best possible solution" among all those that are feasible in the specified sense. We shall illustrate this by some examples. First, let us mention that famous mathematicians like Bernoulli (1717), Lagrange (1788) and Fourier (1788) studied particle movements in mechanics where the particle was moving within some specified region R in space. Lagrange studied the case when R was described by one or more equations, while Fourier went further and allowed R to be described in terms of inequalities. Gauss also studied some of these mechanics problems, as well as approximation problems. He introduced the *least squares method* which reduces the problem of approximating a vector using a quadratic loss function (Euclidean norm) to the problem of solving a certain linear system. Fourier also studied these approximation problems, but with another loss function and in a restricted sense. He actually then discovered a simplified *simplex method* which, today, is a main method in mathematical optimization.

Today, optimization problems arise in all sorts of areas; this is the *age of optimization* as a scientist stated it in a recent talk. Modern society, with advanced technology and competive businesses typically needs to make best possible decisions which e.g. involve the best possible use of resources, maximizing some revenue, minimize production or design costs etc. In mathematical areas one may meet approximation problems like solving some equations "within some tolerance" but without using too many variables (resources). In computer science the VLSI area give rise to many optimization problems: plysical layout of microchips, routing, via minimization and so on. In telecommunications the physical design of networks leads to many different optimization problems, e.g. that of minimizing network design (or expansion) costs subject to constraints reflecting that the network can support the desired traffic. In fact, in many other areas, problems involving communication networks can be viewed as optimization problems. Also in economics (econometry) optimization models are used for e.g. describing money transfer between sectors in society or describing the efficiency of production units.

The large amount of applications, combined with the development of fast computers, has led to massive innovation in optimization. In fact, today optimization may be divided onto several fields, e.g. *linear programming, nonlinear programming, discrete optimization* and *stochastic optimization*. In this course we are concerned with discrete optimization and linear programming. In discrete optimization one optimizes some function over a discrete set, i.e., a set which countable or even finite. We shall mainly use the slightly more restricted term *combinatorial optimization* for the problems of interest here. Typically these are problems of choosing some "optimal subset" among a class of subsets of a given finite ground set. Many of the problems come from the network area, where finding a shortest path between a pair of points in a network is the simplest example.

The reader may now (for good reasons) ask: where does the title of these lecture notes enter the picture? Well, *polyhedral combinatorics* is an area where one studies combinatorial optimization problems using theory and methods from linear programming and *polyhedral theory*. All these terms will be discussed in detail later, but let us at this point just mention that linear programming is to maximize a linear function subject to (a finite number) of linear (algebraic) inequalities. Polyhedral theory deals with the feasible sets of linear programming problems, which are called *polyhedra*. Now, polyhedral theory may be viewed as a part of *convex analysis* which is the branch of mathematics where one studies *convex sets*, i.e., sets that contain the line segments between each pair of its points. A large part of this report is therefore devoted to convex analysis and polyhedral theory. Some people will probably say that it is too much focus on these areas, and they may be right. However, a second purpose of our approach is to give the reader a background in convexity which is useful for all areas in optimization, as well in other areas like approximation theory and statistics.

0.2 Optimization problems and terminology

We now present several optimization problems to be studied throughout the text. Also important terminology is introduced. First, however, we introduce some general notation.

Usually, we denote the (column) vector consisting of the i'th row of a matrix $A \in \mathbb{R}^{m,n}$ by a_i, so we have $a_i^T = (a_{i,1}, \ldots, a_{i,n})$. All vectors are considered as column vectors. When we write $x \leq y$ for vectors $x, y \in \mathbb{R}^n$, we mean that the inequality holds for all components, i.e., $x_i \leq y_i$ for $i = 1, \ldots, n$. Similarly, $x < y$ means that $x_i < y_i$ for $i = 1, \ldots, n$.

An **optimization problem** (or **mathematical programming problem**) is a problem (P):

$$\text{maximize } \{f(x) : x \in A\} \tag{1}$$

where $f : A \to \mathbb{R}$ is a given function defined on some specified set A (typically A is a subset of $\mathbb{R}^n$). A minimization problem is defined similarly. Each point in A is called a **feasible point**, or a **feasible solution**, and A is the **feasible region** (or **feasible set**. The function f is called the **objective function**. An optimization problem is called **feasible** if it has some feasible solution. A point x^* is an **optimal solution** of the problem (P) if $f(x^*) \geq f(x)$ for all $x \in A$. Thus an optimal solution maximizes the objective function among all feasible solutions. The **optimal value** of (P), denoted $v(P)$ is defined as $v(P) = \sup\{f(x) : x \in A\}$. (Recall that the supremum of a set of real numbers is the least upper bound of these numbers). For most problems of interest in optimization, this supremum is either attained, and then we may replace "sup" by "max", or the problem is unbounded as defined below. Thus, if x^* is an optimal solution, then $f(x^*) = v(P)$. Note that there may be several optimal solutions. We say that (P) is **unbounded** if, for any $M \in \mathbb{R}$, there is a feasible soluton x_M with $f(x_M) \geq M$, and we then write $v(P) = \infty$. Similarly, an unbounded minimization problem is such that for each $M \in \mathbb{R}$, there is a feasible soluton x_M with $f(x_M) \leq M$; we then write $v(P) = -\infty$. If the maximization problem (P) in (1) is infeasible (has no feasible solution), we write $v(P) = -\infty$. For infeasible minimization problem we define $v(P) = \infty$. Sometimes we may say, for two feasible solutions x_1 and

x_2 in (1), that x_1 is **better** than x_2 if $f(x_1) > f(x_2)$, and x_1 is **at least as good** as x_2 if $f(x_1) \geq f(x_2)$. Similar notions may be used for minimization problems.

Most optimization problems have additional structure compared to the problem in (1) and we define some important classes of problems next.

Consider matrices and vectors $A_1 \in \mathbb{R}^{m_1,n}$, $A_2 \in \mathbb{R}^{m_2,n}$, $b_1 \in \mathbb{R}^{m_1}$, $b_2 \in \mathbb{R}^{m_2}$ and $c \in \mathbb{R}^n$. The optimization problem (LP)

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & \\ \text{(i)} & A_1 x = b_1; \\ \text{(ii)} & A_2 x \leq b_2, \end{array} \tag{2}$$

is called a **linear programming problem** or **LP problem** for short. Thus in this problem the objective function is linear and the feasible set, let us call it P, is the solution set of a finite number of linear inequalities and equations.

We shall also be interested in another linear problem, the **integer linear programming problem**, (ILP) for short:

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & \\ \text{(i)} & A_1 x = b_1; \\ \text{(ii)} & A_2 x \leq b_2; \\ \text{(iii)} & x \text{ is integral.} \end{array} \tag{3}$$

Thus, in this problem the feasible set consists of all the integral points inside the feasible region of a linear programming problem. It seems natural that there should be some useful relations between the problems (LP) and (ILP), and, in fact, the study of such relations is one of the main topics in polyhedral combinatorics.

It is important to be aware of a main difference between the problems (ILP) and (LP). In terms of theoretical computational complexity, the (ILP) is NP-hard, while (LP) is polynomially solvable. Loosely speaking, this means that (LP) may be solved efficiently on computers using an algorithm with running time polynomially bounded by the "size" of the input problems, while for (ILP) no such efficient algorithm is known (or likely to exist). Typically, algorithms for solving the general (ILP) problem have exponential running time, so only small problems can be solved on a computer.

A **discrete optimization problem** is simply a problem of the form (1) where the feasible set A is a discrete set. A more restricted class of optimization problems may be defined as follows. Let E be a finite set (e.g., consisting of certain vectors or matrices), and let $\mathcal{F}$ be a family (class) of subsets of E, called the **feasible sets**. Also let $w : E \to \mathbb{R}_+$ be a nonnegative function defined on E; we call it a **weight function**. Define $w(F) := \sum_{e \in F} w_e$ for each $F \in \mathcal{F}$, so this is the total weight of the elements in F. We then call the optimization problem (CO)

$$\text{maximize}\{w(F) : F \in \mathcal{F}\} \tag{4}$$

a **combinatorial optimization problem**.

0.3 Examples of discrete models

This section simply contains several examples of the general optimization problems introduced in Section 0.2. The purpose is not only to make the reader acquainted with important problems, but also to illustrate the fantastic potential of discrete mathematical models in representing various phenomena in totally different fields of both science and society. It is important to stress that *modeling* itself is not the only task. We also want to *analyze* and *solve* these models. Combinatorial optimization and integer linear programming provides a rich theory and powerful methods for performing these tasks.

Example 0.1 *The famous* **Traveling Salesman Problem** *(TSP) may be described as follows: for a given set of cities with known distances between pairs of cities, find a shortest possible tour visiting each city exactly once. We introduce a set E consisting of every pair $\{u, v\}$ of cities u and v, and $w(\{u, v\})$ is defined as the distance between cities u and v. A feasible subset of E is the set of pairs of consecutive cities in a tour, and then the weight of this tour coincides with its length, as desired. The (CO) problem (4) then represents the TSP. We remark that the TSP has enjoyed an overwhelming interest since it was "rediscovered" around 1930; each year it is published about 100 papers on the subject (new algorithms, special cases, applications etc.). We shall discuss the TSP in Section 6.7.*

Discrete choices are often restricted to be binary (i.e., two alternatives), as in the TSP above. Either we visit cities i and j consecutively, or we do not. In such situations one can introduce a binary variable $x_j \in \{0, 1\}$ whose value indicates the choice that occurs.

Example 0.2 *The* **knapsack problem** *is a combinatorial optimization problem that may be presented as follows. A Norwegian mountain lover wants to pack her knapsack for today's walk. What should she bring? Available are* *items 1 to n, where the weight of item j is $a_j \geq 0$ and its (subjective) "value" is $w_j \geq 0$. Unfortunately, she cannot bring all the items, as the knapsack (or she) can not carry a weight larger than b (and $b < \sum_{j=1}^{n} a_j$). The problem is to decide which items to bring such that the weight condition is satisfied and such that the total value of the items brought on the trip is largest possible. To model this we introduce a binary variable x_j for each item j, and one can check that the following (ILP) problem represents the problem*

$$\max \{\sum_{j=1}^{n} w_j x_j : \sum_{j=1}^{n} a_j x_j \leq b, \ x \in \{0,1\}^n\}.$$

Loosely speaking, combinatorics deals with properties of finite sets. Some combinatorial optimization problems that involve packing and covering in finite sets are descibed next.

Example 0.3 *Let M be a finite set and $\mathcal{M} = \{M_j : j \in N\}$ a (finite) class of nonempty subsets of M, so N is the index set of these subsets. We say that a subset F of the index set N is a* **cover** *of M if $\cup_{j \in F} M_j = M$, i.e., each element in M lies in at least one of the selected subsets. We say that $F \subseteq N$ is a* **packing** *in M if the subsets M_j, $j \in F$ are pairwise disjoint, thus each element in M lies in at most one of the selected subsets. Finally, if $F \subseteq N$ is both a packing and a covering, it is called a* **partition** *of M. Let now w be a weight function defined on N, so $w_j \geq 0$ is the weight of $j \in N$. The* **minimum weight covering problem** *is to find a cover F with weight $w(F) := \sum_{j \in F} w_j$ as low as possible. The* **maximum weight packing problem** *is to find a packing F with weight $w(F)$ as large as possible.*

There are many combinatorial problems in graphs and networks, as e.g. the TSP. In fact, in the last half of this report we shall study many such problems in detail. A reader who wants to get an idea of what kind of problems these are, can have a look in Chapter 5. In order to avoid introducing graph terminology at this point, we just give one such example here.

Example 0.4 *Assume that m jobs are supposed to be performed by n persons (computers). Each job must be done by exactly one person, and each person can do at most one job. The cost of assigning job i to person j is assumed to be c_{ij}. The* **assignment problem** *is to assign the jobs to persons so as to minimize the total assignment cost. This problem is also called the* **minimum weight bipartite matching problem**. *It can be modeled by* *the following (ILP):*

$$
\begin{array}{llll}
\min & \sum_{i,j} c_{ij} x_{ij} & & \\
\text{subject to} & & & \\
\text{(i)} & \sum_{j=1}^{n} x_{ij} & = 1 & \text{for all } i; \\
\text{(ii)} & \sum_{i=1}^{m} x_{ij} & \leq 1 & \text{for all } j; \\
\text{(iii)} & 0 \leq x_{ij} \leq 1 & & \text{for all } i,j; \\
\text{(iv)} & x_{ij} \text{ is integral} & & \text{for all } i,j.
\end{array}
\tag{5}
$$

We see that the last two constraints assure that the vector x is binary. The variable x_{ij} is 1 if job i is assigned to person j, and 0 otherwise. The constraints (5)(i),(ii) assure that the mentioned restrictions on feasible assignments hold.

The model for the assignment problem in (5) has an interesting property. Consider the so-called linear programming relaxation of this model, which is obtained by removing the integrality constraint on the variables; thus feasible solutions may also be fractional. It turns out that among the optimal solutions of this LP problem there is always one which is integral! Thus, for this model, the optimal value of the (ILP) and that of the LP relaxation coincide, for every objective function c. This is an exceptional situation, but still very important from both a theoretical and a practical point of view. We shall study this in detail throughout this text.

0.4 An overview

In particular, in Chapter 1, we introduce some convex analysis in finite dimensional spaces. This chapter is partly motivated by the fact that most of the concepts and results discussed there are frequently used in polyhedral combinatorics. Secondly, it gives some foundation in convexity for studies in e.g. other branches of optimization. Then, in Chapter 2, we continue the study of convexity, but now in connection with linear inequalities. A powerful theory of polyhedra (the solution set of linear inequalities) is presented.

Chapter 3 is devoted to linear programming, and, in particular, to the main method for solving linear programs: the simplex method. Finally, for those who might believe that the author had forgotten about combinatorial problems, we come back to these problems in Chapter 4. It treats basic optimization problems in graphs and networks, e.g., the shortest path problem, the minimum spanning tree problem and the maximum flow problem. We also discuss algorithms for solving these problems. Chapter 5 on polyhedral combinatorics brings together the previous chapters: general theory and methods are given for attacking combinatorial problems via linear programming and polyhedral theory. Finally, the last chapter contains an overview of some methods for solving integer linear programming problems. The main focus is on linear programming based methods, but other general techniques are also discussed.

0.5 Exercises

Problem 0.1 *Here is a classical LP problem. A student at the University of Oslo wants to decide what kind of food to eat in order to minimize food expenses, but still get healthy food. Available are 5 different kinds of food (so it is a typical student cafeteria we have in mind), each containing certain amounts of energy (kcal), protein (g) and calcium (mg). The foods are* $F_1, \ldots, F_5$*. Each food is served in a given "size" (e.g., chicken with rice might be 220 g). The student requires that today's diet (which is to be decided) must contain a minimum amount of energy (e.g. no less than 2,500 kcal), a minimum amount of protein and a minimum amount of calcium. In addition the student has an upper bound on the number of servings of each food. The cost for each meal (NOK/serving) is assumed given. Formulate the student's diet problem as an LP problem. Discuss the role of integrality in this problem.*

Problem 0.2 *Let* $k \leq m$*, and consider a set of* m *linear inequalities* $a_i^T x \leq b_i$*,* $i = 1, \ldots, m$ *(where* $a_i \in \mathbb{R}^n$*). Formulate a model which represents that a point shall satisfy at least* k *of these constraints and in addition satisfy* $0 \leq x_j \leq M$ *for each* $j \leq n$*.*

Problem 0.3 *Approximation problems is also an area for linear programming, in particular when the* l_1 *or* l_∞ *norms are used. (Recall that* $\|z\|_\infty = \max_j |z_j|$*.) Let* $A \in \mathbb{R}^{m,n}$ *and* $b \in \mathbb{R}^m$*, and formulate the approximation problem* $\min\{\|Ax - b\|_\infty : x \in \mathbb{R}^n\}$ *as an LP problem.*

Problem 0.4 *Integer programs can be used to represent logical relations, as indicated in this exercise. Let* $P_1, \ldots, P_n$ *be logical statements, each being either true or false. Introduce binary variables, and represent the following relations via linear constraints.*

1. *Statement* P_1 *is true.*
2. *All statements are true.*
3. *At least (at most, exactly)* k *statements are true.*

4. *If P_1 is true, then P_2 is also true.*
5. *P_1 and P_2 are equivalent.*
6. *If either P_1 or P_2 is true, then at most two of the statements $P_3, \ldots, P_n$ are true.*

Problem 0.5 *Consider the following problem at our institute; one faces this problem each semester. A (repeating) weakly plan which assigns classes to rooms is to be constructed. Assume that the size of each class is known (it could be an estimate), and that the size (number of seats) of each room is also known. Formulate the problem of finding a feasible assignment of classes to rooms as an (ILP) problem. Then, construct some additional constraints that may be reasonable to impose on the schedule (be creative!), and formulate these in your model.*

Problem 0.6 *In this exercise you must really be creative! Figure out an optimization problem of interest to you (whatever it might be!) and try to formulate it as an (integer) linear programming problem!*

Chapter 1

Convexity in finite dimensions

In this chaper we give an introduction to convex analysis. Convexity is important in several applied mathematical areas like optimization, approximation theory, game theory and probability theory. A classic book in convex analysis is [34]. A modern text which treats convex analysis in combination with optimization is [19]. A comprehensive treatment of convex analysis is [38]. For a general treatment of convexity with application to theoretical statistics, see [37]. The book [39] also treats convexity in connection with a combinatorial study of polytopes.

We shall find it useful with some notation from set algebra. The (algebraic) sum, or Minkowski sum, $A+B$ of two subsets A and B of $\mathbb{R}^n$ is defined by $A+B := \{a+b : a \in A,\ b \in B\}$. Furthermore, for $\lambda \in \mathbb{R}$ we let λA denote the set $\{\lambda a : a \in A\}$. We write $A+x$ instead of $A+\{x\}$, and this set is called the **translate** of A by the vector x. It is usefule to realize that $A+B$ is the same as the union of all the sets $A+b$ where $b \in B$. Subtraction of sets is defined by $A-B := \{a-b : a \in A,\ b \in B\}$.

We leave as an exercise to verify the following set algebra identities from sets $A_1, A_2, A_3 \subseteq \mathbb{R}^n$ and real scalars λ_1, λ_2:

$$\begin{array}{ll}
\text{(i)} & A_1 + A_2 = A_2 + A_1; \\
\text{(ii)} & (A_1 + A_2) + A_3 = A_1 + (A_2 + A_3); \\
\text{(iii)} & \lambda_1(\lambda_2 A_1) = (\lambda_1\lambda_2)A_1; \\
\text{(iv)} & \lambda_1(A_1 + A_2) = \lambda_1 A_1 + \lambda_1 A_2.
\end{array} \tag{1.1}$$

Note that, although the properties above are familiar for real numbers, there are some properties that are not transfered to set algebra. For instance, we do not have $(\lambda_1+\lambda_2)A = \lambda_1 A + \lambda_2 A$ in general (why?).

We let $\|x\| = (x^T x)^{1/2}$ denote the Euclidean norm of a vector x. As usual the n-dimensional real vector space with inner product $x^T y = \sum_{j=1}^n x_j y_j$ is denoted by $\mathbb{R}^n$, and $\mathbb{R}_+$ denotes the set of nonnegative reals. We distinguish between the symbols $\subset$ and $\subseteq$, the first denotes *strict* containment for sets while the second allows equality of the sets involved. For vectors $x, y \in \mathbb{R}^n$ we write $x \leq y$ whenever $x_i \leq y_i$ for $i = 1, \ldots, n$. Similarly, $x < y$ means that $x_i < y_i$ for $i = 1, \ldots, n$. The **Cauchy-Schwarz inequality** says that $|x^T y| \leq \|x\| \|y\|$ for each $x, y \in \mathbb{R}^n$.

1.1 Some concepts from point set topology

First, we recall some concepts from point set topology in $\mathbb{R}^n$. A **closed ball** is a set of the form $\bar{B}(a, r) = \{x \in \mathbb{R}^n : \|x - a\| \leq r\}$ where $a \in \mathbb{R}^n$ and $r \in \mathbb{R}_+$, i.e. this set consist of all points with distance not larger than r from a. The corresponding **open ball**, defined whenever $r > 0$, is $B(a, r) = \{x \in \mathbb{R}^n : \|x - a\| < r\}$. A set $A \in \mathbb{R}^n$ is called **open** if it contains an open ball around each of its points, that is, for each $x \in A$ there is an $\epsilon > 0$ such that $B(x, \epsilon) \subseteq A$. For instance, in $\mathbb{R}$ each open interval $\{x \in \mathbb{R} : a < x < b\}$ is indeed an open set. A set F is called **closed** if its (set) complement $\bar{F} = \{x \in \mathbb{R}^n : x \notin F\}$ is open. A closed interval $\{x \in \mathbb{R} : a \leq x \leq b\}$ is a closed set. A sequence $\{x^{(k)}\}_{k=1}^{\infty} \subset \mathbb{R}^n$ **converges** to x if for each ball B around x there is an integer $K(B)$ such that $x^{(k)} \in B$ for all $k \geq K(B)$, and in that case we call x the **limit point** of the sequence and write $x^{(k)} \to x$.

The open sets in $\mathbb{R}^n$, defined via the Euclidean norm as above makes $\mathbb{R}^n$ into a **topological space**, i.e. the class τ of open sets have the following properties:

(i) $\emptyset, \mathbb{R}^n \in \tau$;

(ii) if $A, B \in \tau$, then $A \cap B \in \tau$;

(iii) if $A_i, i \in I$ is a family of open sets, then the union $\cup_{i \in I} A_i \in \tau$.

Thus, since closed sets are the complements of open sets, we get that the union of any *finite* family of closed sets is again closed, and that the intersection of *any* family of closed sets is closed. The **interior** $\text{int}(A)$ of a set A is defined as the largest open set contained in A; this coincides with the union of all open sets in A (in fact, one may take the union of all open balls contained in A and with center in A). For instance, we have $\text{int}(\bar{B}(a, r) = B(a, r))$. The **closure** $\text{cl}(A)$ of a set A is the smallest closed set containing A. We always have $\text{int}(A) \subseteq A \subseteq \text{cl}(A)$. Note that a set A is open iff $\text{int}(A) = A$, and that A is closed iff $\text{cl}(A) = A$. Finally, we define the **boundary** $\text{bd}(A)$ of A by $\text{bd}(A) = \text{cl}(A) \setminus \text{int}(A)$. An useful characterization of the closed sets is that F is closed if and only if it contains the limit point of each sequence of points in F that converges.

In convex analysis, we also need the concept of **relative topology**. Whenever A is a nonempty set in $\mathbb{R}^n$ we say that a set B is **open relative to** A if $B = B' \cap A$ for some open set B' in $\mathbb{R}^n$ (with usual topology). The family of sets that are open relative to A constitutes a topology on A (i.e. the properties (i)–(iii) above all hold). For a set C one can therefore introduce the **relative interior** $\mathrm{rint}(C)$ (relative to A) of a set C in the natural way: $\mathrm{rint}(C)$ is the union of all sets in C that are open relative to A. One can also introduce relative closure, but this is not of interest for our purposes. This is so because we shall always consider topologies relative to sets A that are closed in the usual topology, and then relative closure and usual closure coincides. However, the **relative boundary** defined by $\mathrm{rbd}(C) = \mathrm{cl}(C) \setminus \mathrm{rint}(C)$ is of interest later.

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is **continuous** if for each convergent sequence $\{x^{(k)}\}_{k=1}^{\infty} \subset \mathbb{R}^n$ with $x^{(k)} \to x$ we also have $f(x^{(k)}) \to f(x)$. If a set F is contained in some ball, it is called **bounded**. A set which is both closed and bounded is called **compact**. **Weierstrass' theorem** says that a continuous, real-valued function f on a compact set K achieves its supremum and infimum over that set, so there are points $x_1 \in K$ and $x_2 \in K$ such that $f(x_1) \leq f(x) \leq f(x_2)$ for all $x \in K$.

Whenever A and B are two nonempty sets in $\mathbb{R}^n$, we define the **distance** between A and B by $\mathrm{dist}(A, B) = \inf\{\|a - b\| : a \in A, b \in B\}$. For one point sets we may write $\mathrm{dist}(A, b)$ instead of $\mathrm{dist}(A, \{b\})$. Note that $\mathrm{dist}(a, b) = \|a-b\|$. From the definition of $\mathrm{dist}(A, B)$ it follows that we can find sequences of points $\{a^n\}_{n=1}^{\infty}$ in A and $\{b^n\}_{n=1}^{\infty}$ in B such that $\|a^n - b^n\| \to \mathrm{dist}(A, B)$. Sometimes, but not in general, one can find points $a \in A$ and $b \in B$ with $\|a - b\| = \mathrm{dist}(A, B)$. For instance this is possible if one of the two sets is compact. We discuss this in more detail in Section 1.5.

1.2 Affine sets

We here give an introduction to **affine algebra** which, loosely speaking, is similar to linear algebra, but where we remove the importance of the origin in the different concepts. The basic objects are affine sets which turn out to be linear subspaces translated by some fixed vector. It is useful to relate the concepts below to the corresponding ones in linear algebra.

An **affine set** $A \subseteq \mathbb{R}^n$ is a set which contains the **affine combination** $\lambda_1 x_1 + \lambda_2 x_2$ for any $x_1, x_2 \in A$ when the real numbers (weights) satisfy $\lambda_1 + \lambda_2 = 1$. Geometrically, this means that A contains the line through any pair of its points. A **line** $L = \{x_0 + \lambda r : \lambda \in \mathbb{R}\}$ through the point x_0 and with direction vector $r \neq 0$ is a basic example of an affine set. Another example is the set $P = \{x_0 + \lambda_1 r_1 + \lambda_2 r_2 : \lambda_1, \lambda_2 \in \mathbb{R}\}$ which is a two-dimensional "plane" going through x_0 and spanned by the nonzero vectors r_1 and r_2.

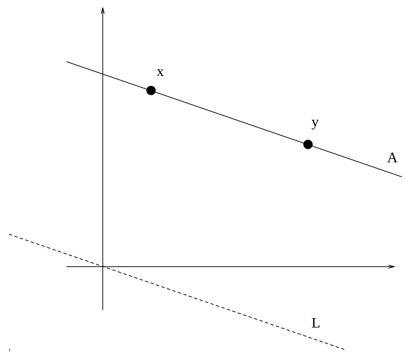

Figure 1.1: The affine hull of $\{x, y\}$

In Figure 1.1 we see a simple affine set, nemaely a line.

Recall that $L \subseteq \mathbb{R}^n$ is a **linear subspace** if $\lambda_1 x_1 + \lambda_2 x_2 \in L$ for any $x_1, x_2 \in L$, i.e., L is closed under the operation of taking linear combinations (of two, and therfore any finite number) of vectors. It follows that a linear subspace is also an affine set. The converse does not hold, but we have the following basic relation. We say that an affine set A is **parallel** to another affine set B if $A = B + x_0$ for some $x_0 \in \mathbb{R}^n$, i.e. A is a translate of B.

Proposition 1.1 *Let A be a nonempty subset of $\mathbb{R}^n$. Then A is an affine set if and only if A is parallel to a unique linear subspace L, i.e., $A = L + x_0$ for some $x_0 \in \mathbb{R}^n$.*

Proof. Assume that $A = L + x_0$ for some $x_0 \in \mathbb{R}^n$ and a linear subspace L of $\mathbb{R}^n$. Let $x_1, x_2 \in A$ and $\lambda_1, \lambda_2 \in \mathbb{R}$ satisfy $\lambda_1 + \lambda_2 = 1$. By assumption there are $y_1, y_2 \in L$ such that $x_1 = y_1 + x_0$ and $x_1 = y_2 + x_0$. This gives $\lambda_1 x_1 + \lambda_2 x_2 = \lambda_1(y_1 + x_0) + \lambda_2(y_2 + x_0) = \lambda_1 y_1 + \lambda_2 y_2 + (\lambda_1 + \lambda_2)x_0 = \lambda_1 y_1 + \lambda_2 y_2 + x_0 \in L + x_0 = A$ as L is a linear subspace, and therefore A is affine.

Conversely, assume that A is affine. Choose $x_0 \in A$ and define $L = A - x_0$. We claim that L is a linear subspace. To prove this, let $\mu \in \mathbb{R}$ and $x \in L$, so $x = a - x_0$ for some $a \in A$. Then $\mu x = \mu(a - x_0) = \mu a + (1 - \mu)x_0 - x_0 \in L$ because $\mu a + (1 - \mu)x_0 \in A$ is an affine combination of two elements in A. L is therefore closed under the operation of multiplying vectors by scalars. Next, for any $x_1, x_2 \in L$, there are suitable $a_1, a_2 \in A$ with $x_1 = a_1 - x_0$, $x_2 = a_2 - x_0$ and therefore $x_1 + x_2 = 2((1/2)a_1 + (1/2)a_2 - x_0) \in L$ since $(1/2)a_1 + (1/2)a_2 \in A$ (affine combination of elements in A) and since we have already shown that $2z \in L$ whenever $z \in L$. This proves that L is a linear subspace.

It remains to prove that A cannot be parallel to two distinct linear subspaces. So assume that $A = L_1 + x_1 = L_2 + x_2$ for linear subspaces L_1, L_2 and vectors x_1, x_2. This implies that $L_1 = L_2 + z$ where $z = x_2 - x_1$. But since L_1 is a linear subspace, $0 \in L_1$ and therefore L_2 must contain $-z$ and also z (as also L_2 is linear subspace). This gives $L_1 = L_2 + z = L_2$ as desired. $\square$

In the example of Figure 1.1 the unique linear subspace L parallel to A is shown.

We define the **dimension** $\dim(A)$ of an affine set A as the dimension of the unique linear subspace parallel to A. The maximal affine sets not equal to the whole space are of particular importance, these are the hyperplanes. More precisely, we define a **hyperplane** in $\mathbb{R}^n$ as an affine set of dimension $n-1$. For instance, the set A in Figure 1.1 is a hyperplane in $\mathbb{R}^2$ and in Figure 1.3 it is shown a hyperplane in $\mathbb{R}^3$. Each hyperplane gives rise of a decomposition of the space nearly as for linear subspaces. Recall that if $L \subset \mathbb{R}^n$ then L and its **orthogonal complement** $L^\perp = \{y \in \mathbb{R}^n : y^T x = 0 \text{ for all } x \in L\}$ have the property that each vector $x \in \mathbb{R}^n$ may be decomposed *uniquely* as $x = x_1 + x_2$ where $x_1 \in L$ and $x_2 \in L^\perp$, see Figure 1.2.

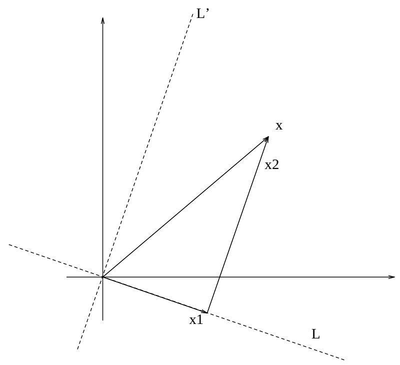

Figure 1.2: Orthogonal decomposition $x = x_1 + x_2$

Furthermore, we know that $\dim(L) + \dim(L^\perp) = n$, and, in particular, if L has dimension $n-1$, then $L^\perp$ has dimension 1, i.e., it is a line. Similarly, a hyperplane partitions the space into two parts, called halfspaces, such that the normal vector of the hyperplane generates a line (affine set of dimension 1). In fact, we have the following characterization of the hyperplanes.

Proposition 1.2 *Any hyperplane $H \subset \mathbb{R}^n$ may be represented by $H = \{x \in \mathbb{R}^n : a^T x = b\}$ for some nonzero $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, i.e. H is the solution*

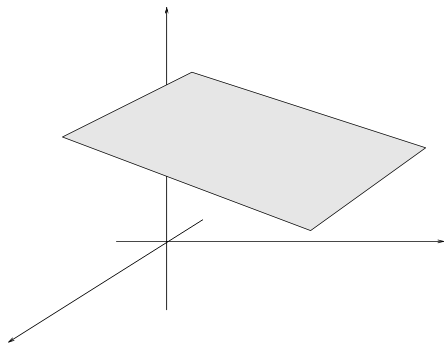

Figure 1.3: A hyperplane in $\mathbb{R}^3$

set of a nontrivial linear equation. Furthermore, any set of this from is a hyperplane. Finally, the equation in this representation is unique up to a scalar multiple.

Proof. Recall from linear algebra that we have $L = (L^\perp)^\perp$ for any linear subspace L. But according to Proposition 1.1, we have $H = L + x^0$ for some $x^0 \in \mathbb{R}^n$ and a linear subspace L of dimension $n-1$. As remarked above $L^\perp = \{\lambda a : \lambda \in \mathbb{R}\}$ for some nonzero $a \in \mathbb{R}^n$. We then get $x \in H \Leftrightarrow x - x^0 \in L \Leftrightarrow x - x^0 \in (L^\perp)^\perp \Leftrightarrow a^T(x - x^0) = 0 \Leftrightarrow a^T x = a^T x^0$ which shows that $H = \{x \in \mathbb{R}^n : a^T x = b\}$ for $b = a^T x^0$. The uniqueness (up to scalar multiple of the equation) in this representation follows from the uniqueness of L (see again Proposition 1.1) and the equivalences above. Similarly, one sees that each set being the solution set of a nontrivial linear equation is in fact a hyperplane.

$\square$

We can proceed along the same lines and show that affine sets are closely linked to systems of linear equations.

Proposition 1.3 *When $A \in \mathbb{R}^{m,n}$ and $b \in \mathbb{R}^m$ the solution set $\{x \in \mathbb{R}^n : Ax = b\}$ of the linear equations $Ax = b$ is an affine set. Furthermore, any affine set may be represented in this way. Therefore, the affine sets are precisely the sets obtained as intersections of hyperplanes.*

Proof. This can be proved by the same methods as in the proof of Proposition 1.2. The only difference is that the orthogonal complement $L^\perp$ may have higher dimension, say k, and we then choose a basis $b^1, \ldots, b^k$ for $L^\perp$ and let the matrix B have these basis vectors as rows.

$\square$

We say that $\sum_{j=1}^m \lambda_j x_j$ is an **affine combination** whenever $x_1, \ldots, x_m \in \mathbb{R}^n$ and the weights satisfy $\sum_{j=1}^m \lambda_j = 1$ (so this is a special linear combination of the vectors). Affine combinations play the same role for affine sets as linear combinations do for linear subspaces, as we shall see below. Let $S \subseteq \mathbb{R}^n$, and let $\text{aff}(S)$ be the intersection of all affine sets containing S. We call $\text{aff}(S)$ the **affine hull** of S. Note that this is an affine set as the intersection of any family of affine sets is again an affine set.

Proposition 1.4 *A set $A \subseteq \mathbb{R}^n$ is affine iff it contains all affine combinations of its points. The affine hull* $\text{aff}(S)$ *of a subset S of $\mathbb{R}^n$ consists of all affine combinations of points in S.*

Proof. We first note that if A contains all affine combinations of its points, it also contains affine combinations of two points which by definition means that A is affine. Next, assume that A is affine and let $x_1, \ldots, x_m \in A$ and $\lambda_1, \ldots, \lambda_m$ be such that $\sum_{j=1}^m \lambda_j = 1$. We must show that $x = \sum_{j=1}^m \lambda_j x_j \in A$. Since the λ_j's sum to 1, one of them must be nonzero, say $\lambda_1 \neq 0$. Then

$$x = \lambda_1 x_1 + \sum_{j=2}^m \lambda_j x_j = \lambda_1 x_1 + (1 - \lambda_1) \sum_{j=2}^m (\lambda_j/(1 - \lambda_1)) x_j. \tag{1.2}$$

Note here that $\sum_{j=2}^m \lambda_j/(1 - \lambda_1) = 1$, so the sum on the right-hand-side of (1.2) is an affine combination y of $m-1$ elements in A, and furthermore x is an affine combination of x_1 and y. Thus, by induction, we get the first part of the proposition.

Next, let W be the set of all affine combinations of points in S. Then $S \subseteq W$ (affine combination of one point!) and it is easy to check that W is affine. Therefore we must have $\text{aff}(S) \subseteq W$. By the first part of the proposition, as $\text{aff}(S)$ is affine, it contains all affine combinations of its points, and, in particular, such combinations of points in S. Thus $W \subseteq \text{aff}(S)$ and the proof is complete.

□

A set of vectors $X = \{x_0, x_1, \ldots, x_m\}$ are called **affinely independent** if $\dim(\text{aff}(\{x_0, x_1, \ldots, x_m\}) = m$. Note the resemblence to linear independence here. Affinely dependent vectors are vectors that are not affinely independent. For a set $X \subseteq \mathbb{R}^n$ the **affine rank** $r_a(X)$ of X is the maximum number of affinely independent vectors in X. We call the "usual" rank of X, namely the maximum number of linearly independent vectors in X, for the **linear rank** of X and denote this number by $r_l(X)$. Relations between linear and affine independence are given in the next proposition.

Proposition 1.5 *Let $X = \{x_0, x_1, \ldots, x_m\}$ be a set of $m+1$ vectors in $\mathbb{R}^n$. Then the following six statements are equivalent:*

(i) $\quad$ *X is affinely independent.*
(ii) $\quad$ *For each $w \in \mathbb{R}^n$ the set $X - w$ is affinely independent.*
(iii) $\quad$ *No vector in X is an affine combination of the others.*
(iv) $\quad$ *If $\sum_{j=0}^{m} \lambda_j x_j = 0$ and $\sum_{j=0}^{m} \lambda_j = 0$, then $\lambda_j = 0$ for all j.*
(v) $\quad$ *The m vectors $x_1 - x_0, \ldots, x_m - x_0$ are linearly independent.*
(vi) $\quad$ *The vectors $(x_0, 1), \ldots, (x_m, 1) \in \mathbb{R}^{n+1}$ are linearly independent.*

Furthermore, we have that
(vii) $\quad$ *if $0 \in aff(X)$, then $r_a(X) = r_l(X) + 1$;*
(viii) $\quad$ *if $0 \notin aff(X)$, then $r_a(X) = r_l(X)$.*

Finally, $dim(S) = r_a(S) - 1$ for each set S in $\mathbb{R}^n$.

We leave the proof as a useful exercise!

Affine independence is illustrated in Figure 1.4. The vectors x_0, x_1, x_2, x_3 are affinely independent, but linearly dependent while $x_1 - x_0, x_2 - x_0, x_3 - x_0$ are linearly independent, . The vectors x_1, x_3, x_4 are affinely dependent.

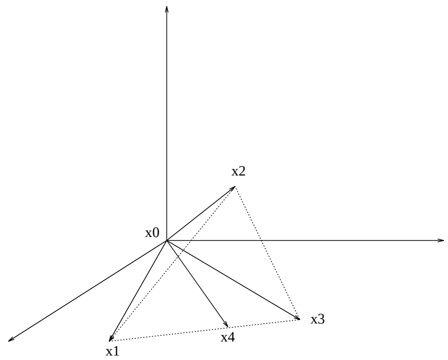

Figure 1.4: Affine independence

1.3 Convex sets and convex combinations

A set $C \subseteq \mathbb{R}^n$ is called **convex** if it contains line segments between each pair of its point, that is, if $\lambda_1 x_1 + \lambda_2 x_2 \in C$ whenever $x_1, x_2 \in C$ and $\lambda_1, \lambda_2 \geq 0$ satisfy $\lambda_1 + \lambda_2 = 1$. Equivalently, C is convex if and only if $(1-\lambda)C + \lambda C \subseteq C$ for every $\lambda \in [0, 1]$. Some examples of convex sets C_i and non-convex sets N_i in $\mathbb{R}^2$ are shown in Figure 1.5.

All affine sets are convex, but the converse does not hold. More generally, the solution set of a family (finite or infinite) of linear inequalities $a_i^T x \leq b_i$, $i \in I$ is a convex set. Furthermore, when $a \in \mathbb{R}^n$ and $r \in \mathbb{R}_+$ the ball $B(a, r)$ is convex, and this also holds for every norm on $\mathbb{R}^n$. We leave the proof of these facts as exercises.

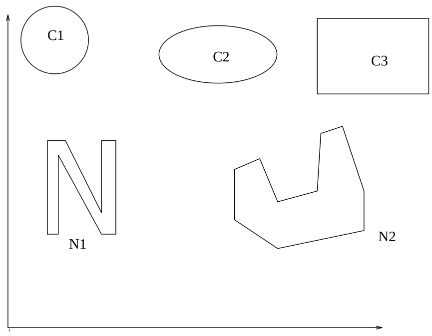

Figure 1.5: Convex and non-convex sets

The **dimension** of a convex set is defined as the dimension of its affine hull. More generally, one may define the dimension of *any* set as the dimension of its affine hull.

The set $K_n = \{z \in \mathbb{R}^n : z \geq 0,\ \sum_{j=1}^n z_j = 1\}$ is a convex set called the **standard simplex** in $\mathbb{R}^n$, see Figure 1.6.

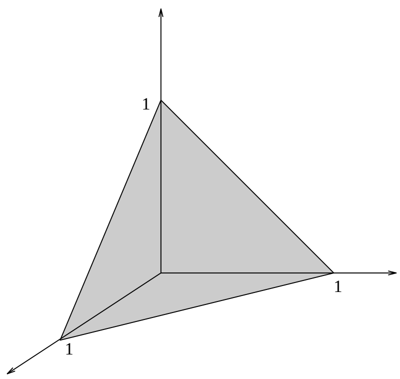

Figure 1.6: The standard simplex in $\mathbb{R}^3$

A **cone** $C \subseteq \mathbb{R}^n$ is a set which is closed under the operation of taking rays through its points, i.e. $\lambda x \in C$ whenever $\lambda \geq 0$ and $x \in C$. A **convex cone** is, of course, a cone which is convex. We see that a set C is a convex cone iff $\lambda_1 x_1 + \lambda_2 x_2 \in C$ whenever $\lambda_1, \lambda_2 \geq 0$ and $x_1, x_2 \in C$. Note that each linear subspace is a convex cone. A cone need not be convex, for instance, a set consisting of two distinct rays is a nonconvex cone. However, we shall only consider convex cones in the following, so we may sometimes use the term "cone" as a short term for "convex cone".

In convex analysis one is interested in certain special linear combinations of vectors that represent "mixtures" of points. When $x_1, \ldots, x_m \in \mathbb{R}^n$ and numbers ("weights") $\lambda_1, \ldots, \lambda_m \geq 0$ satisfy $\sum_{j=1}^m \lambda_j = 1$ (i.e., $\lambda =$

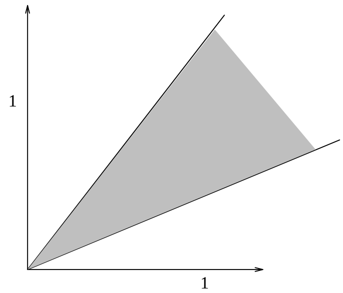

Figure 1.7: A convex cone in $\mathbb{R}^2$

$(\lambda_1, \ldots, \lambda_m) \in K_m$) we say that $x = \sum_{j=1}^m \lambda_j x_j$ is a **convex combination** of the points $x_1, \ldots, x_m$. Thus a set is convex iff it contains each convex combination of any two of its points. A statistician may prefer to view convex combinations as *expectations*; the weights define a probability distribution, and if X is a stochastic vector which attains the value x_k with probability λ_k, the expectation EX equals $\sum_{j=1}^m \lambda_j x$. With this interpretation, a set is convex iff it contains the expectation of all random variables with sample space $\{x_1, \ldots, x_m\}$. A **conical combination** of vectors $x_1, \ldots, x_m$ is a vector $\sum_{j=1}^m \lambda_j x_j$ for some weights $\lambda_j \geq 0$, $j = 1, \ldots, m$.

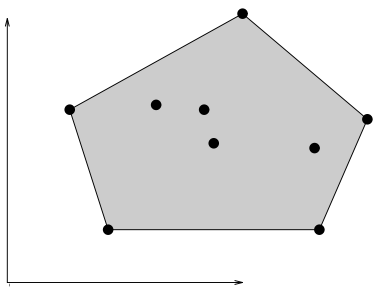

Figure 1.8: The convex hull of some points in $\mathbb{R}^2$

Proposition 1.6 *A set $C \subseteq \mathbb{R}^n$ is convex iff it contains all convex combinations of its points. A set $C \subseteq \mathbb{R}^n$ is a convex cone iff it contains all conical combinations of its points.*

Proof. Clearly, by our definition of convexity, it suffices to prove that if C is convex and $x^1, \ldots, x^m \in C$ and $\lambda \in K_m$, then $x = \sum_{j=1}^m \lambda_j x^j \in C$. We can show this with a similar method to the one used in the proof of Proposition 1.4. We note that some λ_j must be positive (otherwise $\lambda \notin K_m$), say $\lambda_1 > 0$

(and, of course, $\lambda \leq 1$), and we then have

$$x = \lambda_1 x^1 + (1 - \lambda_1) \sum_{j=2}^{m} (\lambda_j / (1 - \lambda_1)) x^j. \tag{1.3}$$

Here $\sum_{j=2}^{m} \lambda_j / (1 - \lambda_1) = 1$, so the sum on the right-hand-side of (1.3) is a convex combination y of $m-1$ elements in C, and x is a convex combination of x^1 and y. The desired result follows by induction, and similar arguments give the result for cones.

□

We define the **convex hull** $\text{conv}(S)$ of a set $S \subseteq \mathbb{R}^n$ as the intersection of all convex sets C containing S. Similarly, the **conical hull** $\text{cone}(S)$ of a subset $S \subseteq \mathbb{R}^n$ is the intersection of all convex cones containing S. Note that $\text{conv}(S)$ is a convex set and that $\text{cone}(S)$ is a convex cone; this follows from the fact that the intersection of an arbitrary family of convex sets (resp. convex cones) is again a convex set (resp. convex cone). It also follows that if C is convex, then $C = \text{conv}(C)$, and if C is a convex cone, then $C = \text{cone}(C)$.

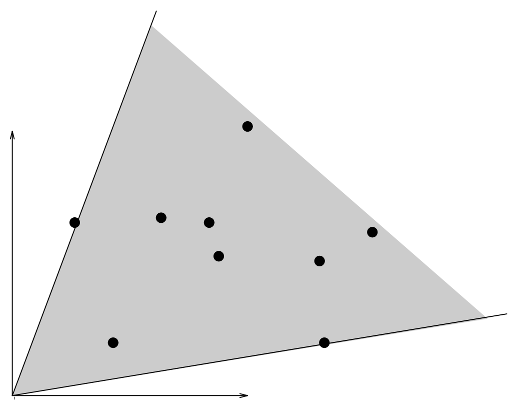

Figure 1.9: The conical hull of some points in $\mathbb{R}^2$

Recall from linear algebra that the linear subspace generated by a set of vectors is the smallest linear subspace containing those vectors. We have a similar relation for convex hulls (playing the role of "subspace generated by").

Proposition 1.7 *Let $S \subseteq \mathbb{R}^n$. Then* $\text{conv}(S)$ *consists of all convex combinations of points in S and* $\text{cone}(S)$ *consists of all conical combinations of points in S.*

Proof. Let W be the set of all convex combinations of points in S. We see that $S \subseteq W$ (convex combination of one point!) and that W is convex (see

Problem 1.8). Thus, by definition, we must have $\text{conv}(S) \subseteq W$. However, by Proposition 1.6, since $\text{conv}(S)$ is convex it contains all convex combinations of its points, and, in particular, such combinations of points in S which implies that $W \subseteq \text{conv}(S)$ and the proof is complete. The result for $\text{cone}(S)$ is proved similarly.

□

Convexity is preserved under a number of operations, and the next result describes a few of these.

Proposition 1.8 *(i) Let C_1, C_2 be convex sets in $\mathbb{R}^n$ and let λ_1, λ_2 be real numbers. Then $\lambda_1 C_1 + \lambda_2 C_2$ is convex.*

(ii) The closure of a convex set is again convex. The closure of a cone is a cone.

(iii) The intersection of any (even infinite) family of convex sets is a convex set.

(iv) Let $T : \mathbb{R}^n \to \mathbb{R}^m$ be an affine transformation, i.e., a function of the form $T(x) = Ax + b$, for some $A \in R^{m,n}$ and $b \in R^m$. Then T maps convex sets to convex sets.

Proof. See Problem 1.9!

□

There is a general technique for transforming results for convex sets into similar results for cones in a space of dimension one higher than the original space. Let $C \subseteq \mathbb{R}^n$ be a convex set and define

$$K_C = \text{cone}(\{(x, 1) \in \mathbb{R}^{n+1} : x \in C\}). \tag{1.4}$$

This cone is called the **homogenization** of C, see Figure 1.10.

Lemma 1.9 *Let C and K_C be as above and define $\bar{C} = \{(x, 1) \in \mathbb{R}^{n+1} : x \in C\}$. Then we have*

(i) $K_C = \{\lambda(x, 1) : x \in C, \lambda \geq 0\}$.

(ii) $\bar{C} = \{y \in K_C : y_{n+1} = 1\}$.

(iii) x is a convex combination of (affinely independent) vectors in C if and only if $(x, 1)$ is a conical combination of (linearly independent) vectors in $\bar{C}$.

Proof. (i) $\bar{C}$ is convex due to the convexity of C and because any convex combination Y of points in $\bar{C}$ must have $y_{n+1} = 1$. It is a general result that the conical hull of a convex set C coincides with the set of rays through points of C (see Problem 1.10). We then see that property (i) follows directly from this result.

Property (ii) is a simple consequence of property (i).

(iii) Let $a_1, \ldots, a_m \in C$ and $\lambda_1, \ldots, \lambda_m \geq 0$. Then $(x, 1) = \sum_{j=1}^m \lambda_j (a_j, 1)$ if and only if $x = \sum_{j=1}^m \lambda_j a_j$ and $\sum_{j=1}^m \lambda_j = 1$. Therefore conical combinations of elements in $\bar{C}$ corresponds to convex combinations of elements in C. Furthermore, by Proposition 1.5 (vi) we have that the vectors $a_1, \ldots, a_m$ are affinely independent if and only if $(a_1, 1), \ldots, (a_m, 1)$ are linearly independent and the property (iii) follows.

□

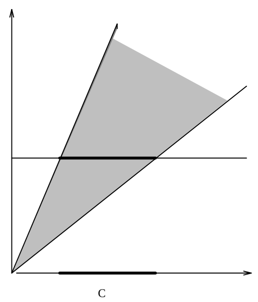

Figure 1.10: Homogenization

Sets being the convex or conical hull of a *finite* set are of special interest in the following.

A set $P \subset \mathbb{R}^n$ which is the convex hull of a finite number of points is called a **polytope**. Thus P is a polytope iff $P = \text{conv}(\{x_1, \ldots, x_m\}) = \{\sum_{j=1}^m \lambda_j x_j : \lambda \in K_m\}$ for certain $x_1, \ldots, x_m \in \mathbb{R}^n$. Examples of polytopes are shown in Figure 1.8, Figure 1.5 (only C3), and Figure 1.11.

A set K which is the conical hull of a finite number of points is called a **finitely generated cone**. So any finitely generated cone is of the form $K = \text{cone}(\{x_1, \ldots, x_m\}) = \{\sum_{j=1}^m \lambda_j x_j : \lambda_j \geq 0 \text{ for } j = 1, \ldots, m\}$ for suitable vectors $x^1, \ldots, x^m$. Some examples of finitely generated cones are found in Figure 1.7 and Figure 1.9.

In a sense cones may be viewed as objects "intermediate" of linear subspaces and general convex sets, as we discuss next. For a convex cone K in $\mathbb{R}^n$, we define its **polar cone** $K^\circ \subseteq \mathbb{R}^n$ by

$$K^\circ = \{y \in \mathbb{R}^n : y^T x \leq 0 \text{ for all } x \in K\}. \tag{1.5}$$

Since $y^T x = x^T y$ we see that each $x \in K$ represents a **valid inequality** $x^T y \leq 0$ which holds for all $y \in K^\circ$. Thus, the polar cone K° consists of the solution set of the infinite number of linear inqualities $x^T y \leq 0$ for each

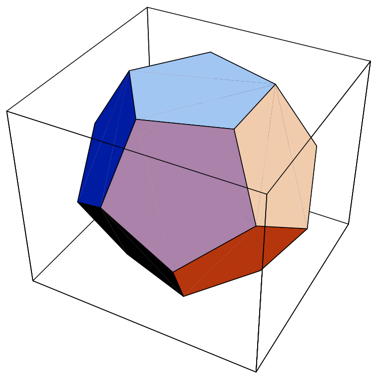

Figure 1.11: A polytope, the *dodecahedron*

element x in K, and, therefore, K° is a convex set. Some properties of the polar cone are given next.

Proposition 1.10 *K° is a closed convex cone. If K is a linear subspace, then K° equals the orthogonal complement $K^\perp$ of K.*

Proof. The fact that K° is a convex cone follows directly from the definition using the linearity of the scalar product. The continuity of the scalar product gives the closedness as follows. Let $\{y^k\}_{k=1}^\infty$ be a sequence of points in K° that converges to some point y. Then $(y^k)^T x \le 0$ for each $x \in K$, so the mentioned continuity of the function $z \to z^T x$ implies that $y^T x \le 0$, so $y \in K^\circ$ and K° is closed. Finally, assume that K is a linear subspace, and then $-x \in K$ whenever $x \in K$. Therefore, $K^\circ = \{y \in \mathbb{R}^n : y^T x = 0 \text{ for all } x \in K\}$ (for if $y^T x < 0$, then $y^T(-x) > 0$, so we have a violation). But this proves that $K^\circ = K^\perp$.

$\square$

We may therefore view polarity as a concept generalizing orthogonality. The polar cone of finitely generated cones will be of special interest to us in connection with linear programming.

Proposition 1.11 *If K is the finitely generated cone $K = \mathrm{cone}(\{c_1, \ldots, c_m\})$, then its polar cone is the solution set of a finite set of linear inequalities, $K^\circ = \{y \in \mathbb{R}^n : c_j^T y \le 0 \text{ for } j = 1, \ldots, m\}$.*

Proof. Since $c_1, \ldots, c_m \in K$, it follows that $K^\circ \subseteq \{y \in \mathbb{R}^n : c_j^T y \le 0 \text{ for } j = 1, \ldots, m\}$. Conversely, if y satisfies $c_j^T y \le 0$ for $j = 1, \ldots, m$, then we also have $(\sum_{j=1}^m \lambda_j c_j)^T y \le 0$ whenever $\lambda_j \ge 0$ for each j. But since any $y \in K$ is of the form $y = \sum_{j=1}^m \lambda_j c_j$ for suitable $\lambda_j \ge 0$, $j = 1, \ldots, m$, this shows that $y \in K^\circ$ as desired.

$\square$

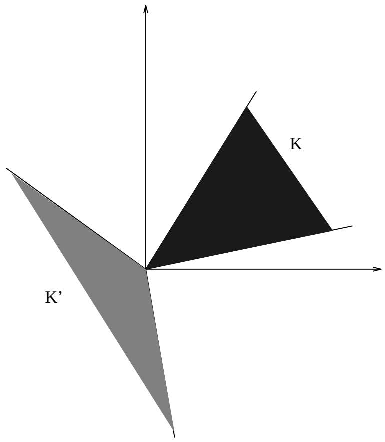

Figure 1.12: Polar cones

The remaining part of this section is devoted to some topological considerations for convex sets.

From the perspective of affine and convex sets the usual topology on $\mathbb{R}^n$ has limited value. The problem is that many interesting convex sets are not fulldimensional, i.e., they lie in affine sets of dimension strictly less than n. Such sets have empty interior. For instance, a line segment in $\mathbb{R}^2$ has nonempty interior, although the points except the two end points are "interior relative to the line". This deficiency of the usual topology is overcome by passing to relative topologies, see Section 1.1.

Let $C \subseteq \mathbb{R}^n$ be a convex set, and let $A = \text{aff}(C)$. Consider the topology relative to A (defined on A). Recall that the relative interior of C, denoted $\text{rint}(C)$ is the union of all sets that are open relative to A and also contained in C. Equivalently, $\text{rint}(C)$ is the largest relative open set contained in C. For instance, consider a convex set being a line segment in $\mathbb{R}^2$ and given by $C = \{(x, y) \in \mathbb{R}^2 : 0 \leq x < 1,\ y = 1\}$. Then we get $\text{rint}(C) = \{(x, y) \in \mathbb{R}^2 : 0 < x < 1,\ y = 1\}$ and the relative boundary is $\text{rbd}(C) = \{(0, 1), (1, 1)\}$.

A remarkable topological property of convex sets is that they are "very close" in a sense to their relative interior as the following theorem says.

Theorem 1.12 *If $C \subseteq \mathbb{R}^n$ is a nonempty convex set, then* $\text{rint}(C)$ *is also nonempty. Furthermore, the sets* $\text{rint}(C)$, C *and* $\text{cl}(C)$ *all have the same relative interior, relative boundary and closure.*

We omit the proof of this result, but remark that they are related to the following interesting property of convex sets: if x lies on the boundary of a convex set C and y lies in the relative interior of C, then every point on the line segment between x and y, except x, lies in the relative interior of C.

1.4 Carathéodory's theorem

The concept of a *basis* is very important in linear algebra (and in its generalization, matroid theory). For a linear subspace L of $\mathbb{R}^n$ each maximal set B of linearly independent vectors in L contain the *same number of vectors*. This number is called the **dimension** of L (comment: maximal means here that by extending the set by one more vector we get a linearly dependent set), and B is a **basis**. Any vector $x \in L$ may be expressed *uniquely* as a linear combination of elements in this basis. Therefore any linear combination of "lots of" vectors in L may be rewritten as a linear combination of the basis vectors, so only $\dim(L)$ vectors are used in this combination. A natural question is now: can we do similar reductions for convex combinations? Yes, the following theorem, called **Carathéodory's theorem**, shows that this is the case. Note that the proof actually gives an algorithm for performing this reduction, and it is related to linear programming as we shall see later. We prefer to consider the result for conical combinations first, and then turn to convex combinations.

Theorem 1.13 *Let $C = \text{cone}(G)$ for some $G \subseteq \mathbb{R}^n$ and assume that $x \in C$. Then x can be written as a conical combination of $m \leq n$ linearly independent vectors in G.*

Proof. Since C is a convex cone, it follows from Proposition 1.7 that there are $a_1, \ldots, a_m \in G$ and $\lambda \geq 0$ such that $x = \sum_{j=1}^m \lambda_j a_j$, in fact, we may assume that $\lambda_j > 0$ for each j. If $a_1, \ldots, a_m$ are linearly independent, then we must have $m \leq n$ and we are done.

Otherwise, $a_1, \ldots, a_m$ are linearly dependent, and we then claim that it is possible to modify the weights λ_j into λ'_j such that at least one λ'_j is zero and $x = \sum_{j=1}^m \lambda'_j a_j$. Thus we can reduce the number of elements in the conical combination by 1. To prove this, we first note that since $a_1, \ldots, a_m$ are linearly dependent, there are numbers $\mu_1, \ldots, \mu_m$ not all zero such that $\sum_{j=1}^m \mu_j a_j = 0$. We may assume that $\mu_1 > 0$ (some μ_j is nonzero, and if it is negative, we may multiply all weights by (-1); for simplicity we assume that $j = 1$). Define Δ^* by

$$\Delta^* = \max\{\Delta \geq 0 : \lambda_j - \Delta\mu_j \geq 0 \text{ for all } j \leq m\} = \min\{\lambda_j/\mu_j : \mu_j > 0\}. \tag{1.6}$$

As each $\lambda_j > 0$ and $\mu_1 > 0$, we get $0 < \Delta^* < \infty$. Let the modified weights λ'_j be given by $\lambda'_j = \lambda_j - \Delta^*\mu_j$ for $j \leq m$. It follows from (1.6) that $\lambda'_j \geq 0$ and that at least one λ'_j must be 0. Furthermore, $\sum_{j \leq m} \lambda'_j a_j = \sum_{j \leq m}(\lambda_j - \Delta^*\mu_j)a_j = \sum_{j \leq m} \lambda_j a_j - \Delta^* \sum_{j \leq m} \mu_j a_j = x - \Delta^* 0 = x$, so we have proved our claim.

By repeatedly applying this reduction we can reduce the number of elements in the conical combination representing x until we end up with positive coefficient only for linearly independent vectors and there can be at most n of these.

$\square$

Next, we give the version of Carathéodory's theorem for convex sets.

Corollary 1.14 *Let $C = \text{conv}(G)$ for some $G \subseteq \mathbb{R}^n$ and assume that $x \in C$. Then x can be written as a convex combination of $m \leq n+1$ affinely independent vectors in G.*

Proof. We use homogenization. Let $x \in C = \text{conv}(G)$, so x is a convex combination of elements in G (by Proposition 1.7). Then $(x, 1)$ is a conical combination of elements of the form $(g, 1)$ for $g \in G$ (see Lemma 1.9(iii)), and therefore, according to Theorem 1.13, $(x, 1)$ is also a conical combination of linearly independent vectors of the form $(g, 1)$ for $g \in G$. Using Lemma 1.9(iii) once again, it follows that x is a convex combination of affinely independent vectors in G.

$\square$

Carathéodory's theorem says that, for a *given* point $x \in \mathbb{R}^n$ in the convex hull of a set S of points, we can write x as a convex combination of at most $n+1$ affinely independent points from S. This, however, does not mean, in general, that there is a "convex basis" in the sense that the *same set* of $n+1$ points may be used to generate any point x. Thus, the "generators" has to be chosen specificly to each x. This is in contrast to the existence of a basis for linear subspaces. It should be noted that a certain class of convex sets, simplices, discussed below, has a "convex basis"; this is seen directly from the definitions below.

Carathéodory's theorem has some interesting consequences concerning decomposition of certain convex sets and convex cones.

Some special polytopes and finitely generated cones are of particular interest. A **simplex** S in $\mathbb{R}^n$ is the convex hull of a set X of affinely independent vectors in $\mathbb{R}^n$, see Figure 1.6. In particular, each simplex is a polytope and $\dim(S) = |X| - 1$. A **generalized simplex** is a cone K generated by (= conical hull of) n linearly independent vectors in $\mathbb{R}^n$. So K is a finitely generated cone and it is fulldimensional.

We next give a *simplicial decomposition theorem* for polytopes and finitely generated cones.

Theorem 1.15 *Each polytope in $\mathbb{R}^n$ can be written as the union of a finite number of simplices. Each finitely generated cone can be written as the union of a finite number of generalized simplices.*

Proof. Consider a polytope $P = \text{conv}(\{a_1, \ldots, a_m\}) \subset \mathbb{R}^n$. We define $M = \{1, \ldots, m\}$ and let $\mathcal{F}$ be the family of all subsets J of M for which a_j, $j \in J$ are affinely independent. Clearly, $\mathcal{F}$ is finite. For each $J \in \mathcal{F}$ let $P^J = \text{conv}(\{a_j : j \in J\})$, and note that P^J is a simplex and $P^J \subseteq P$.

We claim that $P = \cup_{J \in \mathcal{F}} P^J$. The inclusion $\supseteq$ is trivial, so we only need to prove that each $x \in P$ lies in some P^J. Let $x \in P$ and then, by Carathéodory's theorem (Corollary 1.14), x may be written as a convex combination of affinely independent vectors from $a_1, \ldots, a_m$. But this means that x lies in some P^J and the claim has been proved.

The corresponding result for convex cones may be proved similarly when we define the family $\mathcal{F}$ to consist of all subsets J of M for which a_j, $j \in J$ are linearly independent. We then apply the cone version of Carathéodory's theorem and the desired result follows.

$\square$

The decomposition result has an important consequence for, in particular, finitely generated cones as discussed next.

Proposition 1.16 *Each finitely generated cone in $\mathbb{R}^n$ is closed. Each polytope in $\mathbb{R}^n$ is compact.*

Proof. Let K' be a finitely generated cone in $\mathbb{R}^n$. By Theorem 1.15, K' is the union of a finite number of generalized simplices. Thus, it suffices to prove that every generalized simplex is closed (as the union of a finite set of closed sets is closed, see Section 1.1). Consider a generalized simplex $K = \text{cone}(\{a_1, \ldots, a_n\})$ where $a_1, \ldots, a_n$ are linearly independent. Let $A \in \mathbb{R}^{n,n}$ be the matrix with j'th column being a_j. We then see that $K = \{A\lambda : \lambda \in \mathbb{R}^n_+\}$ and that A is nonsingular. Consider a sequence of points $\{x^k\}_{k=1}^{\infty} \subseteq K$ which converges to some point z. If we can prove that $z \in K$, then K is closed (confer again Section 1.1 on topology). Since $x^k \in K$, there is some $\lambda^k \in \mathbb{R}^n_+$ with $x^k = A\lambda^k$ for each k, and therefore , as A is nonsingular, we have $A^{-1}x^k = \lambda^k$ for each k. Since $x^k \to z$ and any linear transformation is continuous (by the Cauchy-Schwarz inequality), we get $A^{-1}x^k \to A^{-1}z$, i.e., $\lambda^k \to A^{-1}z$. But the limit point of any convergent sequence in $\mathbb{R}^n_+$ must lie in $\mathbb{R}^n_+$ (as this set is closed!), so we get $A^{-1}z \in \mathbb{R}^n_+$ and therefore also $z = AA^{-1}z \in K$ as desired. This proves that any generalized simplex is closed, and we are done.

Finally, we prove the similar result for polytopes. This is easier. Assume that the simplex C is the convex hull of affinely independent points $a^1, \ldots, a^{n+1}$. Then we may write $C = \{A\lambda : \lambda \in K_{n+1}\}$ where $A \in \mathbb{R}^{n,n}$ is the matrix with j'th column being a_i and K_{n+1} is the standard simplex in $\mathbb{R}^{n+1}$. Now, K_{n+1} is closed and bounded, i.e., compact, and the linear transformation given by A is continuous (see Problem 1.11). Thus we can apply Weierstrass' theorem and conclude that C is compact.

$\square$

1.5 Separation of convex sets

Convex sets have many interesting properties. One of these is that disjoint convex sets can be separated by hyperplanes. In $\mathbb{R}^2$ this means that for any two disjoint convex sets we can find a line such that the two sets lie on opposite sides of this line (possibly intersecting the line). This is not true for non-convex sets. It turns out that this property in $\mathbb{R}^n$ may be viewed as a "theoretical core" of the linear programming duality theory. Separation of convex sets is also important in nonlinear optimization and other areas (e.g., game theory and statistical decision theory).

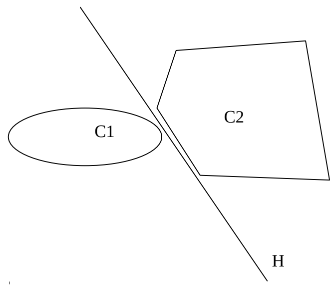

Figure 1.13: Separation of convex sets

Separation involves the concepts of a hyperplane and a halfspace as introduced next. For each $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$ we define the **halfspaces** $H_{\leq}(a,b) = \{x \in \mathbb{R}^n : a^T x \leq b\}$ and $H_{\geq}(a,b) = \{x \in \mathbb{R}^n : a^T x \geq b\}$, and also the **hyperplane** $H_{=}(a,b) = \{x \in \mathbb{R}^n : a^T x = b\}$. These two halfspaces are the closed halfspaces associated with the hyperplane $H_{=}(a,b)$. We call $H_{<}(a,b) := \mathrm{int}(H_{\leq}(a,b))$ and $H_{>}(a,b) := \mathrm{int}(H_{\geq}(a,b))$ the open halfspaces associated with $H_{=}(a,b)$. Geometrically, a hyperplane divides the space into two parts given by the associated closed halfspaces.

Loosely speaking, separation of disjoint convex sets means that one can find a hyperplane such that the two sets are contained in the opposite closed halfspaces defined by the hyperplane.

First, we study separation of a point from a convex set. Our proof will be based on geometry and topology, so we recommend to look through Section 1.1 before proceeding.

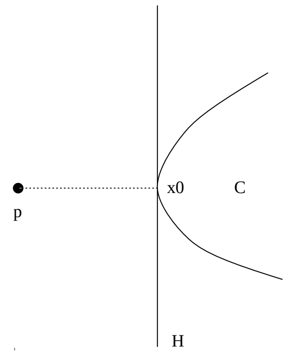

Figure 1.14: The proof idea

Theorem 1.17 *Let C be a nonempty closed convex set in $\mathbb{R}^n$ and let $p \notin C$. Then there exists a nonzero $a \in \mathbb{R}^n$ and an $\epsilon > 0$ such that*

$$a^T x \leq a^T p - \epsilon \ \textit{ for all } x \in C. \tag{1.7}$$

Furthermore, there is a unique point $x_0 \in C$ closest to p in C, i.e. $\|p - x_0\| = \mathrm{dist}(C, p)$, and we may choose $a = p - x_0$ in (1.7).

Proof. We first prove that $\inf_{x \in C} \|x - p\|$ is attained. Let $c \in C$ (which is possible as C is nonempty). Then $\inf_{x \in C} \|x - p\| = \inf_{x \in C'} \|x - p\|$ where $C' = \{x \in C : \|x - p\| \leq \|c - p\|\}$. Note that C' is bounded since it lies inside a ball of radius $\|c - p\|$ with center p. In addition C' is closed (as the intersection of the mentioned ball and C), and therefore C' is compact. Since the norm is a continuous function, it follows from Weierstrass' theorem that $\inf_{x \in C} \|x - p\| = \|x_0 - p\|$ for some $x_0 \in C$ as claimed. (Note: in these arguments we did not use convexity, only that C is nonempty and closed, so we have shown that $\mathrm{dist}(p, C)$ is attained under these assumptions)

Let $x \in C$ and $0 < t < 1$. Since C is convex, $(1 - t)x_0 + tx \in C$ and therefore $\|(1 - t)x_0 + tx - p\| \geq \|x_0 - p\|$. By squaring both sides and calculating the inner products we obtain $\|x_0 - p\|^2 + 2t(x_0 - p)^T (x - x_0) + t^2 \|x - x_0\|^2 \geq \|x_0 - p\|^2$. We now subtract $\|x_0 - p\|^2$ on both sides, divide by t and let $t \to 0^+$, and get $(x_0 - p)^T (x - x_0) \geq 0$ or equivalently $(x_0 - p)^T x \geq (x_0 - p)^T p + \|x_0 - p\|^2$. Define $a = p - x_0$, so $a \neq 0$ (as $p \notin C$) and the inequality becomes $a^T x \leq a^T p - \|p - x_0\|^2$ which holds for each $x \in C$. Thus we have shown (1.7). It only remains to prove that x_0 is the *unique* point in C closest to p. Let $r = \|p - x_0\| = \mathrm{dist}(p, C)$. Then the set of points within distance r from p is the ball $\bar{B}(p, r)$. Let $b = a^T x_0$, and from (1.7) we see that C is contained in the halfspace $H_{\leq}(a, b)$ and $\bar{B}(p, r)$ is contained in the opposite halfspace $H_{\geq}(a, b)$ and x_0 is the unique point in

$\bar{B}(p, r)$ which also lies in the hyperplane $H_=(a, b)$. The desired uniqueness result follows directly from these relations.

□

A more specialized version of this separation theorem is obtained if C is a convex cone.

Corollary 1.18 *Let C be a non-empty closed convex cone in $\mathbb{R}^n$ and let $p \notin C$. Then there exists an $a \in \mathbb{R}^n \setminus \{0\}$ such that*

$$a^T x \leq 0 < a^T p \;\; \textit{for all } x \in C. \tag{1.8}$$

Proof. We apply Theorem 1.17 and let a be as in (1.7). We claim that $a^T x \leq 0$ for all $x \in C$. In fact, if $a^T x > 0$ for some $x \in C$, then also $\lambda x \in C$ for each $\lambda \geq 0$ (as C is a cone), and therefore $a^T(\lambda x) = \lambda a^T x \to \infty$ as $\lambda \to \infty$. But this contradicts that $a^T(\lambda x) \leq a^T p - \epsilon$, and the claim follows. Furthermore, we have that $a^T p > 0$; this follows from (1.7) as $0 \in C$ and $a^T 0 = 0$.

□

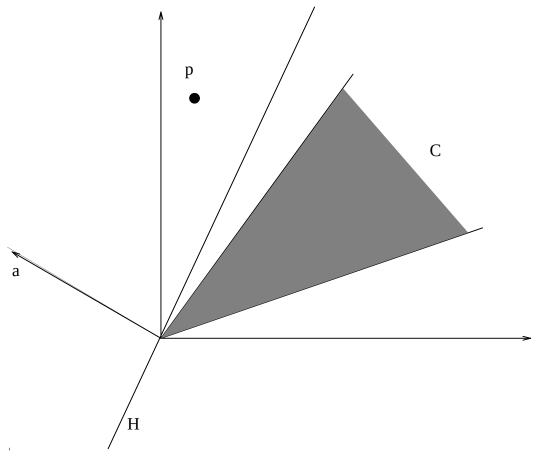

Figure 1.15: Separation for convex cones

From the perspective of linear programming duality, it is the previous "point-set" separation theorem that is needed. However, for other applications, it may be of interest to give more general separation theorems as we do next.

We first introduce different notions for separating sets. Let C_1 and C_2 be nonempty sets, and let $a \in \mathbb{R}^n$, $b \in \mathbb{R}$. We say that the hyperplane $H_=(a, b)$ **weakly separates** C_1 and C_2 if C_1 and C_2 are contained in *different* sets among the two closed halfspaces $H_\leq(a, b)$ and $H_\geq(a, b)$. If, in addition, we have that $\text{dist}(C_i, H_=(a,b)) > 0$ for $i = 1,2$, the hyperplane $H_=(a,b)$ is said to **strongly separate** C_1 and C_2.

It should be noted that weak separation is indeed a weak concept because it allows the following improper separation. Let A be the affine hull of $C_1 \cup C_2$ and assume that A is not fulldimensional. Then we can find a nonzero vector $a \in L^\perp$ where L is the unique linear subspace parallel to A. Then $a^T x$ is constant, say equal to b, for $x \in A$, and therefore the hyperplane $H_=(a,b)$ weakly separates C_1 and C_2. Here both the sets C_1 and C_2 are contained in this hyperplane, so this kind of separation is not of much interest. Therefore, we introduce a stronger requirement next. We say that a hyperplane $H_=(a,b)$ **properly separates** C_1 and C_2 if it weakly separates the two sets and, in addition, either C_1 or C_2 intersects one of the two *open* halfspaces associated with $H_=(a,b)$.

We are now prepared to give results concerning both strict and proper separation of convex sets.

Theorem 1.19 *Let C_1 and C_2 be nonempty convex sets. Then the following statements are equivalent.*

(i) *C_1 and C_2 are strongly separated by some hyperplane.*

(ii) *There exists an nonzero $a \in \mathbb{R}^n$ with $\sup_{x \in C_1} a^T x < \inf_{y \in C_2} a^T y$.*

(iii) $\text{dist}(C_1, C_2) > 0$.

Proof. (i) $\Rightarrow$ (ii): Assume that (i) holds, so there exists a nonzero $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$ with $C_1 \subseteq H_\leq(a,b)$, $C_2 \subseteq H_\geq(a,b)$, and $\text{dist}(C_1, H_=(a,b)) = \epsilon_1 > 0$, $\text{dist}(C_2, H_=(a,b) = \epsilon_2 > 0$. We may assume that $\|a\| = 1$ as otherwise we could scale a and b suitably. Define $\epsilon = \min\{\epsilon_1, \epsilon_2\}$, so $\epsilon > 0$. Let $L = \text{span}\{a\} = \{\lambda a : \lambda \in \mathbb{R}\}$ (a subspace of dimension 1 being a line through the origin with direction vector a) and let $L^\perp$ be its orthogonal complement, i.e. $L^\perp = \{x \in \mathbb{R}^n : a^T x = 0\}$. Let $c \in C_1$. We may then decompose c as $c = c_1 + c_2$ where $c_1 \in L$ and $c_2 \in L^\perp$. Here $c_1 = \lambda a$ for suitable $\lambda \in \mathbb{R}$. Now, $\epsilon \leq \epsilon_1 \leq \text{dist}(C_1, H_=(a,b)) \leq \text{dist}(c, H_=(a,b)) = \text{dist}(c_1, H_=(a,b)) = \|\lambda a - b\, a\| = |\lambda - b|\|a\| = |\lambda - b|$. We used the fact that the projection of λa onto the hyperplane $H_=(a,b)$ is the point $b\, a$. We must have $\lambda \leq b$ as $C_1 \subseteq H_\leq(a,b)$, so it follows that $\lambda \leq b - \epsilon$ which again gives $a^T c = a^T(c_1 + c_2) = a^T c_1 + a^T c_2 = a^T c_1 = a^T \lambda a = \lambda \leq b - \epsilon$. This inequality holds for all $c \in C_1$, and therefore $\sup_{x \in C_1} a^T x \leq b - \epsilon$. By similar arguments we obtain that $\inf_{y \in C_2} a^T y \geq b + \epsilon$, which shows that statement (ii) holds.

(ii) $\Rightarrow$ (iii): Assume that (ii) holds, and that $\text{dist}(C_1, C_2) = 0$; we shall deduce a contradiction and thereby obtain the desired implication. Choose sequences $\{x^n\}_{n=1}^\infty \subseteq C_1$ and $\{y^n\}_{n=1}^\infty \subseteq C_2$ such that $\|x^n - y^n\| \to \text{dist}(C_1, C_2) = 0$. This implies that $a^T(x^n - y^n) \to 0$ since each linear function is continuous (confer the Cauchy-Schwarz inequality). But this contradicts the strict inequality in (ii) and the implication follows.

(iii) $\Rightarrow$ (i): Assume that (iii) holds. Then $0 \not\in \mathrm{cl}(C_1 - C_2)$. From (1.1) we see that $C_1 - C_2$ is convex, and this implies that its closure $\mathrm{cl}(C_1 - C_2)$ is convex. We now apply Theorem 1.17 with $C = C_1 \bar{-} C_2$ and $p = 0$ and obtain a nonzero vector a such that $a^T z \leq a^T 0 - \epsilon = -\epsilon$ for each $z \in \mathrm{cl}(C_1 - C_2)$. In particular, we get (for $z = c_1 - c_2$) $a^T c_1 \leq a^T c_2 - \epsilon$ for all $c_1 \in C_1$, $c_2 \in C_2$. Let $b = \sup_{x \in C_1} a^T x$, and we have shown that $a^T x \leq b$ and $a^T y \geq b + \epsilon$ for all $x \in C_1$, $y \in C_2$. Thus $C_1 \subseteq H_{\leq}(a, b)$ and $C_2 \subseteq H_{\geq}(a, b + \epsilon)$, so the hyperplane $H_{=}(a, b + \epsilon/2)$ separates C_1 and C_2 strongly.

□

As remarked before, it is not generally true that $\mathrm{dist}(A, B) = \|a - b\|$ for suitable $a \in A$ and $b \in B$. However, under a compactness assumption this holds as described next.

Lemma 1.20 *Let A and B be two nonempty closed sets in $\mathbb{R}^n$ where at least one of these sets is bounded as well (and therefore compact). Then there exists $a \in A$ and $b \in B$ with $\|a - b\| = \mathrm{dist}(A, B)$. If, in addition, the sets are disjoint, we have $\mathrm{dist}(A, B) > 0$.*

Proof. Say that A is compact, i.e. closed and bounded. Pick an element $\tilde{b} \in B$ and consider the set $\tilde{B} = \{b \in B : \mathrm{dist}(A, b) \leq \mathrm{dist}(A, \tilde{b})\}$. This set $\tilde{B}$ is nonempty (it contains $\tilde{b}$) and compact. Furthermore, we have

$$\inf\{\|a - b\| : a \in A,\ b \in B\} = \inf\{\|a - b\| : a \in A,\ b \in \tilde{B}\}. \tag{1.9}$$

Note here that $A \times \tilde{B}$ is compact (this follows from Tychonoff's theorem, but can also be shown directly) and the function $\|a - b\|$ is continuous on this set. By Weierstass' theorem each continuous function on a compact set achieves its infimum, so it follows that that the last infimum in (1.9) is achieved by some $a \in A$ and $b \in \tilde{B} \subseteq B$. Assume now that A and B are disjoint, and let $a \in A$, $b \in B$ be such that $\|a - b\| = \mathrm{dist}(A, B)$. The disjointness implies that $a \neq b$, so $0 < \|a - b\| = \mathrm{dist}(A, B)$ which completes the proof.

□

Corollary 1.21 *Let C_1 and C_2 be disjoint convex sets where at least one of these is compact. Then C_1 and C_2 are strongly separated by some hyperplane.*

Proof. This is an immeditate consequence of Lemma 1.20 and Theorem 1.19.

□

Our last separation theorem involves proper separation, but we leave out the proof.

Theorem 1.22 *Let C_1 and C_2 be nonempty convex sets with disjoint relative interior ($\mathrm{rint}(C_1) \cap \mathrm{rint}(C_1) = \emptyset$). Then C_1 and C_2 can be properly separated by a hyperplane.*

In particular, if C_1 is a nonempty convex set and $p \in \mathrm{rbd}(C_1)$, then $\{p\}$ and C_1 can be properly separated by a hyperplane.

This result is important in connection with a theory of generalized gradients which again is central in e.g. nondifferentiable optimization.

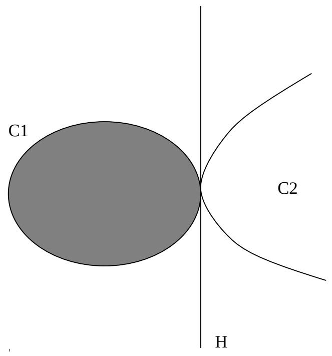

Figure 1.16: Proper separation

We shall give one important application of the separation theory to polarity. Recall from (1.5) that the polar of a convex cone $K \subseteq \mathbb{R}^n$ is the set $K^\circ = \{y \in \mathbb{R}^n : y^T x \leq 0 \text{ for all } x \in K\}$. What happens if we apply the polarity operation twice to the cone K? In the next result we give the form of the **bipolar** $K^{\circ\circ} := (K^\circ)^\circ$.

Proposition 1.23 *Let K be a convex cone. Then we have that $K^{\circ\circ} = \mathrm{cl}(K)$.*

Proof. For each $x \in K$ and each $y \in K^\circ$ we have that $y^T x \leq 0$ (by the definition of K°). But this immediately gives that $x \in K^{\circ\circ}$, so we have $K \subseteq K^{\circ\circ}$. By taking the closure on both sides of this inequality we obtain $\mathrm{cl}(K) \subseteq K^{\circ\circ}$ since we know from Proposition 1.10 that $K^{\circ\circ}$ is a closed cone.

In order to prove the opposite inclusion, we shall use separation. Note that, by Proposition 1.8, the set $\mathrm{cl}(K)$ is convex, in fact it is easy to see that it is a closed convex cone. Assume that $\mathrm{cl}(K) \subset K^{\circ\circ}$ holds with strict inclusion, so there is some $p \in K^{\circ\circ} \setminus \mathrm{cl}(K)$. From Corollary 1.18 there is some nonzero vector $a \in \mathbb{R}^n$ such that $a^T x \leq 0 < a^T p$ for all $x \in K$. From the last inequality we get that $a \in K^\circ$ and therefore, for any $z \in K^{\circ\circ}$ we must have (by definition of the bipolar) that $a^T z \leq 0$. In particular, this must hold for $z = p \in K^{\circ\circ}$, so $a^T p \leq 0$. But this contradicts the separation inequality $a^T p > 0$, and it follows that $\mathrm{cl}(K) = K^{\circ\circ}$ as desired.

$\square$

For an illustration of the previous theorem, see Figure 1.12.

It follows from the last result that when K is a closed convex cone we have that $K = K^{\circ\circ}$. This relation generalizes the well-known fact that $L^{\perp\perp} = L$ for a linear subspace L.

1.6 Exercises

Problem 1.1 *Prove the relations in (1.1).*

Problem 1.2 *Show that the intersection of any family of affine sets is an affine set.*

Problem 1.3 *Prove Proposition 1.5.*

Problem 1.4 *Consider a family (possibly infinite) of linear inequalities $a_i^T x \leq b_i$, $i \in I$, and C be its solution set, i.e., C is the set of points satisfying all the inequalities. Prove that C is a convex set.*

Problem 1.5 *Consider the unit disc $S = \{(x_1, x_2) \in \mathbb{R}^2 : x_1^2 + x_2^2 \leq 1\}$ in $\mathbb{R}^2$. Find a family of linear inequalities as in the previous problem with solution set S.*

Problem 1.6 *Show that the ball $B(a, r)$ is convex (for $a \in \mathbb{R}^n$ and $r \in \mathbb{R}_+$).*

Problem 1.7 *Show that (i) the intersection of any family of convex sets is a convex set, and that (ii) the intersection of any family of convex cones is a convex cone.*

Problem 1.8 *Let $S \subseteq \mathbb{R}^n$ and let W be the set of all convex combinations of points in S. Prove that W is convex.*

Problem 1.9 *Prove Proposition 1.8.*

Problem 1.10 *Prove that the conical hull of a convex set $S \subseteq \mathbb{R}^n$ coincides with the set of rays through points of S, i.e., $\mathrm{cone}(S) = \{\lambda x : x \in S,\ \lambda \geq 0\}$.*

Problem 1.11 *Prove that (i) the linear transformation $x \to a^T x$, for given $a \in \mathbb{R}^n$ is continuous, and that (ii) the standard simplex K_m is compact.*

Problem 1.12 *Let C be a simplex being the generated by a set $S$of (affinely independent) points. Show that each point in C can be written* uniquely *as a convex combination of the points in S. What about the points outside C: can they be written as convex, affine or linear combination of points in S? Uniquely?*

Chapter 2

Theory of polyhedra, linear inequalities and linear programming

From chapter 1 we have now available a theory of convex sets. The purpose of this chapter is to apply and extend this theory to a very important class of convex sets called polyhedra. The interest in polyhedra is motivated by the fact that these are the feasible sets in linear programming and that they arise in integer linear programming. Moreover, the "mathematics of polyhedra" is a fascinating subject in itself, although we shall not go to far into this theory.

A main influence of the presentation given here is the classic book [35] on linear and integer linear programming, as well as the paper [32]. We also recommend [39] where the combinatorics of polytopes is the main theme. Both these books contain a large number of further references.

2.1 Polyhedra and linear systems

We introduce the basic objects of this chapter, polyhedra and linear inequalities.

A **linear inequality** is an inequality of the form $a^T x \leq \beta$ where $a \in \mathbb{R}^n$ is non-zero and $\beta \in \mathbb{R}$. Note that a **linear equality (equation)** $a^T x = \beta$ may be written as the two linear inequalities $a^T x \leq \beta$, $a^T x \geq \beta$. A **linear system**, or **system** for short, is a finite set of linear inequalities, so it may be written in matrix form as $Ax \leq b$ where $A \in \mathbb{R}^{m,n}$ and $b \in \mathbb{R}^m$. The i'th inequality of the linear system $Ax \leq b$ is the linear inequality $a_i^T x \leq b_i$ (so $a_i^T = (a_{i,1}, \ldots, a_{i,n})$). A linear system is **consistent** if it has at least one solution, i.e., there is an x_0 satisfying $Ax_0 \leq b$. An inequality $a_i^T x \leq b_i$

from the system $Ax \le b$ is **active** in a point x_0 (with $Ax_0 \le b$) if it satisfies $a_i^T x_0 = b_i$, i.e., this inequality holds with equality in x_0.

We are (very!) interested in the set of points that satisfy linear systems. A **polyhedron** $P \subseteq \mathbb{R}^n$ is the solution set of a linear system, i.e., $P = \{x \in \mathbb{R}^n : Ax \le b\}$ for some linear system $Ax \le b$. An example of an (unbounded) polyhedron is shown in Figure 2.1; it is the solution set of three linear inequalities. A **halfspace** is the solution set of a single linear inequality, see Figure 2.2.

Proposition 2.1 *Any polyhedron in $\mathbb{R}^n$ is a closed convex set, and it is the intersection between a finite number of halfspaces.*

Proof. Let $P = \{x \in \mathbb{R}^n : Ax \le b\}$, so we have $P = \cap_{i=1}^m \{x \in \mathbb{R}^n : a_i^T x \le b_i\}$ which shows that P is the intersection of a finite number of halfspaces. Thus, the result will follow if we show that each halfspace is both closed and convex. This, however, is easy to verify from the definitions (or like this: a halfspace is closed as the inverse image of the closed set $\{y \in \mathbb{R} : y \le b\}$ under the continuous function $x \to y = a^T x$).

$\square$

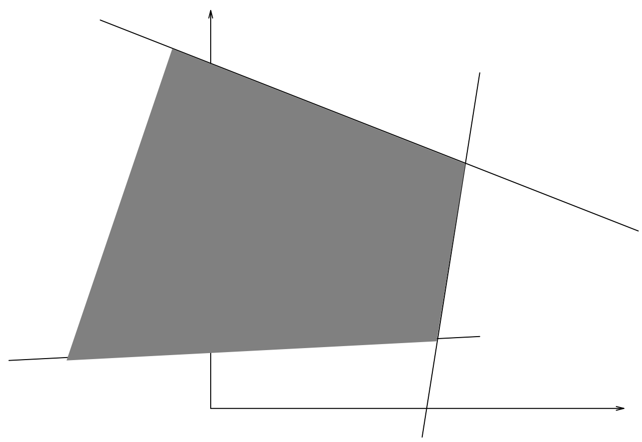

Figure 2.1: A polyhedron

We say that two linear systems are **equivalent** if they have the same solution set, i.e., if the associated polyhedra coincide. A linear system $Ax \le b$ is called **real** (resp. **rational**) if all the elements in A and b are real (resp. rational). A polyhedron is **real** (resp. **rational**) if it is the solution set of a real (resp. rational) linear system. Note that a rational linear system is equivalent to a linear system with all coefficients being integers; we just multiply each inequality by a suitably large integer. (Remark: an *integral* polyhedron is *not* defined through an integral linear system, but we return to this later).

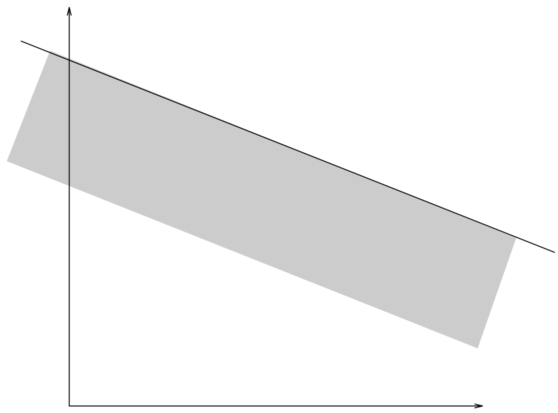

Figure 2.2: A halfspace

2.2 Farkas' lemma and linear programming duality

One of the main topics in this section is the consistency of linear systems, i.e., the question of whether a given linear system has a solution. This problem is connected to linear programming. In fact, consider an LP problem (P) max $\{c^T x : Ax \leq b\}$ having optimal value $v(P)$. We then see that $v(P) \geq \alpha$ if and only if the linear system $Ax \leq b$, $c^T x \geq \alpha$ has a solution x. Thus after a study of consistency, we pass on to applications of these results to linear programming.

Consider first a linear system of equations $Ax = b$. From linear algebra we recall a simple characterization of whether this system has a solution; this is the so-called *Fredholm's alternative*.

Theorem 2.2 *Let $A \in \mathbb{R}^{m,n}$ have columns $a_1, \ldots, a_n$ and let $b \in \mathbb{R}^m$. The system $Ax = b$ has a solution if and only if $y^T b = 0$ for all $y \in \mathbb{R}^m$ with $y^T a_j = 0$, $j = 1, \ldots, n$.*

This (algebraic) result may be interpreted in different ways. The *geometric content* is obtained by observing that $Ax = b$ has a solution x if and only if $b \in L = \text{span}(\{a_1, \ldots, a_1\})$ (recall from block multiplication of matrices that $Ax = \sum_{j=1}^{n} x_j a_j$). But L is a linear subspace and by the orthogonal decomposition theory we have $L = (L^\perp)^\perp$, so we obtain that $Ax = b$ is consistent iff $b \perp y$ for each $y \in L^\perp$, which is precisely the statement in Fredholm's result. Finally, we may give an *inference* interpretation as follows. Let us say that an equation which is a linear combination of the equations in $Ax = b$ is *implied* by the system. Thus an implied equation is of the form $(y^T A)x = y^T b$ for some vector $y \in \mathbb{R}^m$. An obvious fact is that a consistent system cannot have an implied equality of the form $0^T x = \beta$ for some $\beta \neq 0$; such an equation is clearly inconsistent. Fredholm's alternative says that $Ax = b$ is consistent *if and only if* there is no implied equality which is inconsistent.

The next result, called **Farkas' lemma** generalizes Fredholm's alternative to general linear systems.

Theorem 2.3 *Let $A \in \mathbb{R}^{m,n}$ and $b \in \mathbb{R}^m$. Then the linear system $Ax = b$, $x \geq 0$ has a solution if and only if $y^T b \geq 0$ for each $y \in \mathbb{R}^m$ with $y^T A \geq 0$.*

Proof. The system $Ax = b$, $x \geq 0$ has a solution if and only if $b \in K :=$ cone$(\{a^1, \ldots, a^n\})$, where these vectors are the column vectors of A. But from Proposition 1.16 K is closed and combining this with the form of the bipolar in Proposition 1.23 we get $K = K^{\circ\circ}$. Thus we get the following equivalences:

$$\begin{aligned}
&b \in K \Leftrightarrow b \in K^{\circ\circ} \Leftrightarrow \\
&y^T b \leq 0 \ \text{ for each } y \in K^\circ \Leftrightarrow \\
&y^T b \leq 0 \ \text{ for each } y \in \mathbb{R}^m \text{ with } y^T A \leq 0 \Leftrightarrow \\
&y^T b \geq 0 \ \text{ for each } y \in \mathbb{R}^m \text{ with } y^T A \geq 0.
\end{aligned}$$

and the proof is complete.

□

Farkas' lemma may therefore be viewed as a polarity result which, in our development, is based on two results: (i) the closedness of finitely generated cones (due to Carathéodory's theorem) and (ii) a separation theorem for convex cones. The geometric interpretation of Farkas' lemma is illustrated in Figure 2.3. Here the matrix A has the two columns a^1 and a^2 and K is the cone generated by these two vectors. We see that $Ax = b_1$ has nonnegative solution as $b_1 \in K$, while $Ax = b_2$ has no nonnegative solution as there is a hyperplane defined by y which separates b_2 from K.

Variants of Farkas' lemma are obtained by studying the consistency of other linear systems. One can transform one linear system into another by e.g. introducing additional variables. For instance, the system (i) $Ax \leq b$ is "equivalent" to the system (ii) $Ax + y = b$, $y \geq 0$ in the sense that we can transform a solution of one of the systems into a solution of the other. In fact, if x is a solution of (i), then (x, y) where $y := b - Ax$ is a solution of (ii) (because $Ax \leq b$ gives $y \geq 0$ and $Ax + y = b$). Conversely, if (x, y) is a solution of (ii), then $Ax = b - y \leq b$ so x is a solution of (i). The variable y above is called a **slack variable** as its components represent the slack (difference) in each of the original inequalities. There are two other transformation tricks to

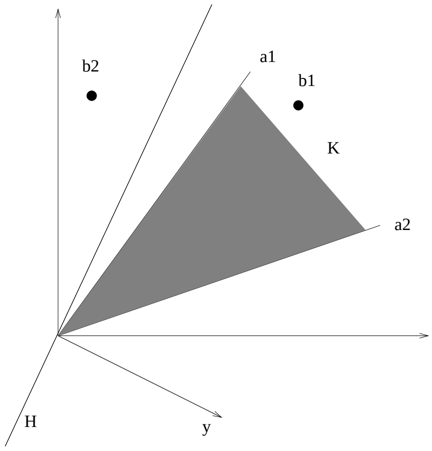

Figure 2.3: Farkas' lemma

be aware of. First, we can clearly convert each equation into two inequalities, i.e., $a^T x = \beta$ is equivalent to $a^T x \le \beta$, $-a^T x \le -\beta$. Secondly, simple variable substitutions are useful, for instance, a **free** variable $x \in \mathbb{R}^n$ which can be any real number (e.g., it is not restricted to be nonnegative) may be replaced by the two *nonnegative* variables $x_1, x_2 \in \mathbb{R}^n$ through $x = x_1 - x_2$. By using one or more of these transformations one may get alternative (equivalent) forms of Farkas' lemma, formulated for different types of linear system. One such alternative formulation is as follows.

Corollary 2.4 *Let $A \in \mathbb{R}^{m,n}$ and $b \in \mathbb{R}^m$. Then the linear system $Ax \le b$ has a solution if and only if $y^T b \ge 0$ for each $y \in \mathbb{R}^m$ with $y \ge 0$ and $y^T A = 0$.*

Proof. We introduce a nonnegative slack variable u and see that $Ax \le b$ has a solution iff $Ax + u = b$, $u \ge 0$ has a solution. Next, we introduce nonnegative variables x_1, x_2 by $x = x_1 - x_2$, and we obtain the new system $A(x_1 - x_2) + u = b$, $x_1, x_2 \ge 0$, $u \ge 0$ or in matrix form

$$\begin{gathered} \begin{bmatrix} A & -A & I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ u \end{bmatrix} = b, \\ x_1 \ge 0,\ x_2 \ge 0,\ u \ge 0. \end{gathered}$$

and this system is equivalent to the original system $Ax \le b$. By Farkas' lemma the new system has a solution iff $y^T b \ge 0$ for each y satisfying $y^T A \ge 0$, $y^T(-A) \ge 0$ and $y^T I \ge 0$, i.e. $y^T b \ge 0$ for each $y \ge 0$ satisfying $y^T A = 0$. $\square$

Note that from Corollary 2.4 we obtain Fredholm's alternative as follows. A linear system of equations $Ax = b$ has a solution iff $Ax \leq b, -Ax \leq -b$ has a solution which again is equivalent to $y_1^T b + y_2^T(-b) \geq 0$ for each $y_1, y_2 \geq 0$ with $y_1^T A + y_2^T(-A) = 0$. By replacing $y_1 - y_2$ by y (without any simple bounds), we get the equivalent condition $y^T b \geq 0$ for each y with $y^T A = 0$, which clearly is equivalent to $y^T b = 0$ for each y with $y^T A = 0$ (as $y^T A = 0$ implies that $(-y)^T A = 0$).

We now turn to linear programming theory. A central concept in this area is **duality** which represents certain close connections between pairs of LP problems. Recall from Chapter 0 that the optimal value of an optimization problem (Q) is denoted by $v(Q)$ (and if the problem is unbounded or infeasible $v(Q)$ is suitably defined as either $-\infty$ or ∞).

Consider a linear programming problem

$$\text{(P)} \quad \max\{c^T x : Ax \leq b\}. \tag{2.1}$$

We shall call (P) the **primal problem**. We associate another LP problem with (P), which we call the **dual problem**:

$$\text{(D)} \min\{y^T b : y^T A = c^T, y \geq 0\}. \tag{2.2}$$

We may motivate the study of this dual problem in the following way. Assume that one wants to find an upper bound on the optimal value of problem (P). This may be derived from the system $Ax \leq b$ by taking nonnegative linear combinations. Thus, let $y \in \mathbb{R}^m$ be nonnegative and assume that y is chosen such that $y^T A = c^T$ which means that the objective c is written as a conical combination of the rows in A. Then each feasible solution of (P) also satisfies the new inequality $(y^T A)x \leq y^T b$ (because y is nonnegative), or, equivialently, $c^T x \leq y^T b$. Therefore $y^T b$ is an upper bound on the optimal value $v(P)$ of (P), and it follows that the dual problem (D) is to find the *best upper bound* on $v(P)$ in terms of such conical combinations.

The following theorem, called the duality theorem of linear programming, represents the heart of the duality theory. It says that *the optimal values of the two problems are the same.*

Theorem 2.5 *Let $A \in \mathbb{R}^{m,n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. Assume that each of the two systems $Ax \leq b$ and $y^T A = c^T$, $y \geq 0$ is consistent. Then we have*

$$\max\{c^T x : Ax \leq b\} = \min\{y^T b : y^T A = c^T, \ y \geq 0\}. \tag{2.3}$$

Proof. Let x satisfy $Ax \leq b$ and y satisfy $y^T A = c^T, y \geq 0$ (such vectors exist by the consistency assumption). Then we get $c^T x = (y^T A)x = y^T(Ax) \leq y^T b$, and it follows that

$$\max\{c^T x : Ax \leq b\} \leq \min\{y^T b : y^T A = c^T, y \geq 0\}. \tag{2.4}$$

This inequality, called **weak duality**, is therefore easy to prove (but still useful, as we shall see later). The opposite inequality will follow if we show that the linear system $Ax \leq b,\ y^T A = c^T,\ y \geq 0,\ c^T x \geq y^T b$ has a solution. We write this system in matrix form

$$\begin{bmatrix} A & 0 \\ -c^T & b^T \\ 0 & A^T \\ 0 & -A^T \\ 0 & -I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} b \\ 0 \\ c \\ -c \\ 0 \end{bmatrix}$$

By Farkas' lemma (Corollary 2.4) this system has a solution if and only if for each $u \in \mathbb{R}^m$, $v \in \mathbb{R}$, $w_1, w_2 \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$ satisfying

$$\begin{array}{ll} \text{(i)} & u \geq 0,\ v \geq 0,\ w_1,\ w_2 \geq 0,\ z \geq 0; \\ \text{(ii)} & u^T A - vc^T = 0; \\ \text{(iii)} & vb^T + (w_1 - w_2)^T A^T - z^T = 0 \end{array} \tag{2.5}$$

we also have

$$u^T b + (w_1 - w_2)^T c \geq 0. \tag{2.6}$$

Let (u, v, w_1, w_2, z) satisfy (2.5). Consider first the case when $v = 0$. Then $u^T A = 0$ and $(w_1 - w_2)^T A^T = z^T$. Let x satisfy $Ax \leq b$ and y satisfy $y^T A = c^T,\ y \geq 0$. Combining these inequalities we get $u^T b + (w_1 - w_2)^T c \geq u^T(Ax) + (w_1 - w_2)^T (A^T y) = (u^T A)x + y^T A(w_1 - w_2) = 0 + y^T z \geq 0$ so (2.6) holds. Next, consider the case when $v > 0$. Then $u^T(vb) = u^T(z - A(w_1 - w_2)) = u^T z - (u^T A)(w_1 - w_2) = u^T z - vc^T(w_1 - w_2) \geq -vc^T(w_1 - w_2)$. We divide this inequality by (the positive number) v, and get $u^T b + c^T(w_1 - w_2) \geq 0$, so again (2.6) holds. Thus we have verified the consistency condition in Farkas' lemma, and the equation (2.3) follows.

□

There is a nice geometrical interpretation of the duality theorem, which is illustrated for $n = 2$ in Figure 2.4. Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be the feasible set in the primal LP (P), and assume that x_0 is an optimal solution. Let $a_i^T x \leq b_i$ for $i \in I'$ be the set of active inequalities (among $Ax \leq b$) in x_0, in the figure these are the inequalities $a_1^T x \leq b_1$ and $a_2^T x \leq b_2$. We see, intuitively, that the optimality of x_0 is equivalent to that c lies in the

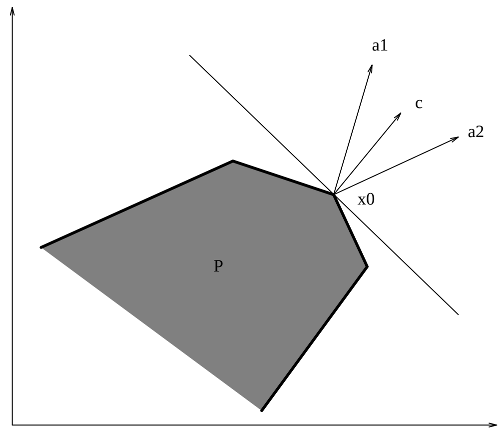

Figure 2.4: LP duality

cone generated by a_i, $i \in I'$, say $c^T = \sum_{i \in I'} y_i a_i$ for nonnegative numbers y_i, $i \in I'$. By defining a vector $y \in \mathbb{R}^m$ by adding zero components for indices $i \in I \setminus I'$, we see that the mentioned condition means that y is feasible in the dual problem, i.e. $y^T A = c^T$ and $y \geq 0$. But now we have $v(P) = c^T x_0 = (y^T A) x_0 = y^T (A x_0) = \sum_{i \in I'} y_i (A x_0)_i = \sum_{i \in I'} y_i b_i = y^T b \geq v(D)$ so $v(P) \geq v(D)$. Combined with weak duality $v(P) \leq v(D)$ (see (2.4)), we therefore get $v(P) = v(D)$. Note that this is not a complete proof; we did not actually prove that x_0 is optimal in (P) iff b lies in the mentioned cone.

We can obtain a more general version of the duality theorem in which one only requires that *at least* one of the two LP problems is feasible. Again, Farkas' lemma is the main tool in the proof.

Theorem 2.6 *Let $A \in \mathbb{R}^{m,n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. Consider the two dual LP problems* (P) $\max\{c^T x : Ax \leq b\}$ *and* (D) $\min\{y^T b : y^T A = c^T, y \geq 0\}$ *and assume that at least one of the two problems is feasible. Then we have that*

$$\sup \{c^T x : Ax \leq b\} = \inf \{y^T b : y^T A = c^T,\ y \geq 0\}. \tag{2.7}$$

Proof. If both problems are feasible, then (2.7) holds, even with the supremum and infimum replaced by maximum and minimum, respectively; this is Theorem 2.5.

Next, assume that (P) is feasible, but that (D) is not. Then the linear system $A^T y = c$, $y \geq 0$ is inconsistent, and by Farkas' lemma there exists an $z \in \mathbb{R}^n$ such that $z^T A^T \leq 0$, $z^T c > 0$, i.e., $Az \leq 0$, $c^T z > 0$. Let x_0 be a feasible solution of (P) (this problem is feasible), and consider, for $\lambda \geq 0$, the point $x(\lambda) = x_0 + \lambda z$. Since $Ax(\lambda) = Ax_0 + \lambda Az \leq Ax_0 \leq b$, we have $x(\lambda) \in P$. But $c^T x(\lambda) = c^T x_0 + \lambda c^T z \to \infty$ as $\lambda \to \infty$, so (P) must be unbounded. Thus we have $v(P) = \infty = v(D)$ as (D) is infeasible, and (2.7) holds.

The case when (P) is infeasible, but (D) feasible, can be treated similarly, and one obtains that both values are $-\infty$.

$\square$

For a pair of dual linear programs there are only four different situations that can arise concerning the values of these optimization problems.

Corollary 2.7 *Consider a pair of linear programs (P) and (D) as above. Then exactly one of the following four situations occurs.*

(i) *Both $v(P)$ and $v(D)$ are finite; and then these values are equal and attained.*

(ii) *(P) is feasible and unbounded while (D) is infeasible.*

(iii) *(D) is feasible and unbounded while (P) is infeasible.*

(iv) *Both (P) and (D) are infeasible.*

Proof. Consider first when (P) is feasible. From Theorem 2.6 we see that we are either in case (i) or (ii) depending on whether (P) is bounded or not. Next, assume that (P) is infeasible. Then, if (D) is feasible, we must have $v(D) = v(P) = -\infty$ so case (iii) occurs. Alternatively, (D) is infeasible, and then we are in case (iv).

$\square$

2.3 Implicit equations and dimension of polyhedra

We shall study the dimension of polyhedra and obtain a dimension formula which generalizes the dimension formula for linear transformations.

Throughout this section we shall write all linear systems either as $Ax \leq b$ or $Ax \leq b$, $Cx = d$. Thus we assume that all "$\geq$" inequalities have been rewritten to the form given. Let $I = \{1, \ldots, m\}$ be the index set of the inequalities, so $Ax \leq b$ can be written as $a_i^T x \leq b_i$ for $i \in I$.

An inequality $a_i^T x \leq b_i$ in a linear system $Ax \leq b$ is called an **implicit equality** if each solution x of $Ax \leq b$ also satisfies $a_i^T x = b_i$. For instance, consider the system $-x_1 \leq 0$, $-x_2 \leq 0$, $x_1 + x_2 \leq 0$. The solution set consists of 0 alone, and each of the inequalities is an implicit equality. We also define implicit equalities in a similar manner for linear systems with both inequalities and equalities, and then clearly all the equalities are implicit equalities.

For a linear system $Ax \leq b$ we let $A^= x = b^=$ denote the subsystem consisting of the implicit equalities, and $A^+ x \leq b^+$ denotes the remaining inequalities. Therefore, the three systems (i) $Ax \leq b$, (ii) $A^+x \leq b^+$, $A^=x = b^=$ and (iii) $A^+x \leq b^+$, $A^=x \leq b^=$ are all equivalent. Let $I^=$ and I^+ be the index set of the implicit and the remaining inequalities, respectively, and these sets therefore represent a partition of the index set I.

An **inner point** x_0 in the polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is a a point in P which satisfies $A^=x_0 = b^=$, $A^+x_0 < b^+$ (recall that here $<$ means that all of the inequalities are strict).

Lemma 2.8 *Each nonempty polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ has an inner point.*

Proof. From the definition of implicit equalities (since P is nonempty), it follows that, for each $i \in I^+$, there is an $x^i \in P$ with $a_i^T x^i < b_i$. Let $x_0 = (\sum_{i \in I^+} x^i)/|I^+|$, so this is a convex combination of the points $x^i, i \in I^+$ and therefore $x_0 \in P$ and $A^=x_0 = b^=$. Furthermore, we see that $A^+x_0 < b^+$, as desired.

□

Recall that **dimension** of a set of points in $\mathbb{R}^n$, e.g. a polyhedron, is the dimension of its affine hull. A polyhedron in $\mathbb{R}^n$ is **fulldimensional** if $\dim(P) = n$. If a polyhedron P has an implicit equality, then P lies in the hyperplane defined by this implicit equality, and consequently P can not be fulldimensional. More generally, we can describe the affine hull and the dimension of a polyhedron in terms of the implicit equalities.

Proposition 2.9 *Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be a nonempty polyhedron. Then*

$$aff(P) = \{x \in \mathbb{R}^n : A^=x = b^=\} = \{x \in \mathbb{R}^n : A^=x \leq b^=\} \tag{2.8}$$

and

$$\dim(P) = n - \operatorname{rank}(A^=). \tag{2.9}$$

Proof. Let $A_1 = \{x \in \mathbb{R}^n : A^=x = b^=\}$ and $A_2 = \{x \in \mathbb{R}^n : A^=x \leq b^=\}$. Since $P \subseteq A_1$ and A_1 is an affine set, the affine hull of P must be contained in A_1. Since $A_1 \subseteq A_2$, it only remains to prove that $A_2 \subseteq \operatorname{aff}(P)$. Let x_0 be an inner point of P (which exists by Lemma 2.8 as P is nonempty) and let $x \in A_2$. If $x = x_0$, then $x \in P \subseteq \operatorname{aff}(P)$. If $x \neq x_0$, define $z = (1-\lambda)x_0 + \lambda x$ for $\lambda > 0$ "small". Since both x_0 and x lie in A_2, we have $A^=z \leq b^=$. Furthermore, by choosing λ small enough, z will be near enough to x_0 to assure that all the inequalities that are not implicit equalities hold with strict inequality in z. Thus $z \in P$, and therefore the line through x_0 and x must be contained in $\operatorname{aff}(P)$, and in particular, $x \in \operatorname{aff}(P)$. Thus $A_1 = A_2 = \operatorname{aff}(P)$ and (2.8) has been shown.

The dimension of P is (by definition) the dimension of aff(P). But the dimension of an affine set is the dimension of the unique linear subspace parallel to it. Thus, by combining these remarks with the first part of the proof, we get $\dim(P) = \dim(\{x \in \mathbb{R}^n : A^= x = 0\}) = n - \text{rank}(A^=)$ where the last equality is due to the dimension formula for linear transformations.

□

Recall from Chapter 1 the concept of *relative interior* of point sets. In connection with a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ the relative topology we consider is the topology relative to the affine hull of P, i.e., aff(P) $=$ $\{x \in \mathbb{R}^n : A^= x = b^=\}$. For a polyhedron P we therefore see that its relative interior is given by

$$\text{rint}(P) = \{x \in \mathbb{R}^n : A^= x = b^=,\ A^+ x < b^+\}. \tag{2.10}$$

This shows that a point in P is a relative interior point if and only if it is an inner point.

We next consider interior points of a polyhedron for the usual topology on $\mathbb{R}^n$, see Section 1.1. An **interior point** x of a polyhedron P is a point in P with the property that $B(x, \epsilon) \subseteq P$ for some $\epsilon > 0$, i.e., P contains a ball around x. Note that each interior point is also an inner point, but the converse does not hold. We can characterize the fulldimensional polyhedra as follows.

Corollary 2.10 *The following conditions are equivalent for a polyhedron* $P = \{x \in \mathbb{R}^n : Ax \leq b\}$.

(i) P *is fulldimensional.*

(ii) P *has an interior point.*

(iii) $Ax \leq b$ *has no implicit equalities.*

Proof. From Proposition 2.9 we see that P is fulldimensional iff $\text{rank}(A^=) =$ 0, i.e. there are no implicit equalities, so the equivalence of (i) and (iii) follows. Now, if there are no implicit equalities ((iii) holds), choose an inner point x_0 (which exixts by Lemma 2.8) so $Ax_0 < b$. But then it is easy to see that x_0 is also an interior point, i.e. (ii) holds. Conversely, if (ii) holds, P must contain some ball. But this ball is fulldimensional, and therefore P is fulldimensional as well and (i) holds.

□

These concepts are illustrated in Figure 2.5, where P is the polyhedron $P = \{x \in \mathbb{R}^3 : 0 \leq x_i \leq 1 \text{ for } i = 1, 2, 3\}$, i.e., the unit cube, and where F is the polyhedron being the intersection between P and the hyperplane $\{x \in \mathbb{R}^3 : x_2 = 1\}$. Then P is fulldimensional, while F has dimension 2 with implicit equality $x_2 = 1$. The point x is an interior point of P, y is an inner point of F, but z is not an inner point of F.

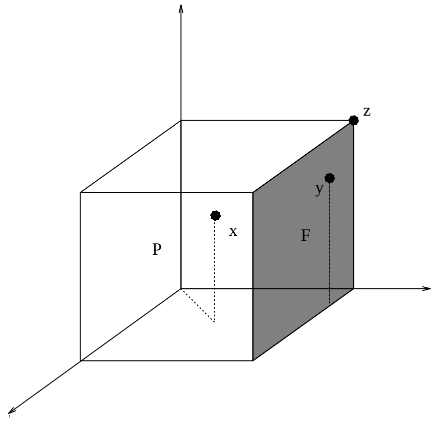

Figure 2.5: Dimension, interior and inner points

2.4 Interior representation of polyhedra

A linear programming problem max $\{c^T x : x \in P\}$ over a nonempty polyhedron P may have finite or infinite optimal value. The main goal in this section is to show that, in both cases, there are always optimal points or directions of a very special kind, and, furthermore, that all these points and directions span the polyhedron P through convex and conical combinations. We shall find it convenient to develop the results for so-called pointed polyhedra, and then extend these to the general situation afterwards.

The **lineality space** of a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is the set $\text{lin.space}(P) = \{x \in \mathbb{R}^n : Ax = 0\}$. Note that $y \in \text{lin.space}(P)$ iff for each $x \in P$ we have $L_x \subseteq P$ where $L_x = \{x + \lambda y : \lambda \in \mathbb{R}\}$ is the line going through x with direction vector y. Thus the lineality space is the nullspace (kernel) of the matrix A and it is therefore a linear subspace of $\mathbb{R}^n$. P is called **pointed** if $\text{lin.space}(P) = \{0\}$, i.e., P contains no lines, or equivalently, $\text{rank}(A) = n$. Note that if L is a linear subspace (and therefore a polyhedron), then we have $L = \text{lin.space}(L)$. In Figure 2.6 the polyhedron P is pointed, while Q is not. The lineality space of Q is $\text{span}(\{e_1\})$ where $e_1 = (1, 0, \ldots, 0)$.

We now define the basic objects that, as we shall show, span polyhedra. Consider a pointed nonempty polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. A point $x_0 \in P$ is called a **vertex** of P if x_0 is the (unique) solution of n linearly independent equations from the system $Ax = b$. (Remark: when we say that equations $Cx = d$ are linearly independent we actually mean that the row vectors of C are linearly independent). We shall see below that P must have at least one vertex (as it is pointed and nonempty). We say that $x_0 \in P$ is an **extreme point** of P if there are no points $x_1, x_2 \in P$ both distict from x_0 with $x = (1/2)x_1 + (1/2)x_2$. Let $r \in \mathbb{R}^n$ be such that $r \neq 0$ and $Ar \leq 0$,

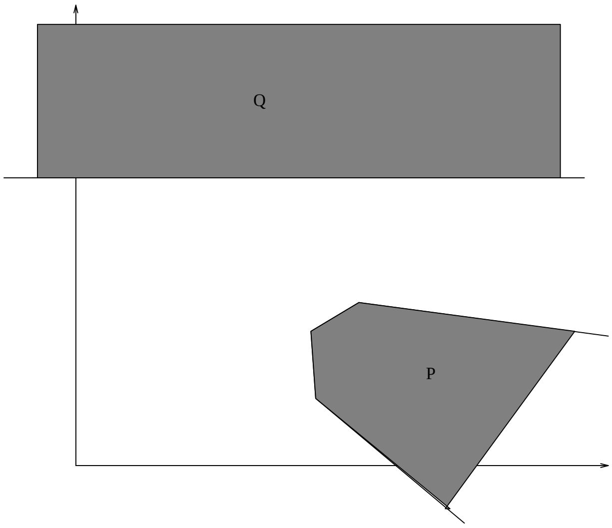

Figure 2.6: Lineality space

and define the set $R(x;r) = \{x + \lambda r : \lambda \geq 0\} = x + \text{cone}(\{r\})$ for each $x \in P$. Then we have that $R(x;r) \subseteq P$ as $A(x + \lambda r) = Ax + \lambda Ar \leq b$. The set $R(x;r)$ is called a **ray** of P, in fact, we call it the ray in *direction r in the point x*. The direction of a ray is defined up to a positive scalar multiple, i.e., $R(x;r_1) = R(x;r_2)$ iff $r_1 = \lambda r_2$ for some positive scalar λ. Note that a bounded polyhedron has no ray. Two rays are parallel if they have parallel direction vectors. A ray $R(x;r)$ is an **extreme ray** if there are no non-parallel rays $R(x;r_1)$ and $R(x;r_2)$ with $r = (1/2)r_1 + (1/2)r_2$ (Note that this definition is independent of $x \in P$). In Figure 2.7 the polyhedron (cone) K has extreme rays $R(0,r_1)$ and $R(0,r_2)$, while $R(0,r_3)$ is a ray, but not an extreme ray. K has one vertex, namely the origin. In the polyhedron P the points $x_1, \ldots, x_5$ are vertices, but x_7, x_8 and x_9 are not vertices. (We leave it as an exercise to find x_6!). P has no rays, as it is bounded.

Proposition 2.11 *(i) A point $x_0 \in P$ is a vertex of $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ if and only if it is an extreme point of P.*

(ii) A ray of P is an extreme ray if and only if it has a direction vector r such that $A'r = 0$ for some subsystem $A'x = 0$ consisting of $n-1$ linearly independent equations from $Ax = 0$.

Proof. (i) Assume first that x_0 is a vertex of P, so there is a linearly independent subsystem $A'x \leq b'$ of $Ax \leq b$ such that x_0 is the unique solution of $A'x = b'$. Assume that $x_0 = (1/2)x_1 + (1/2)x_2$ for some $x_1, x_2 \in P$. Then $A'x_i = b'$ for $i = 1, 2$ (otherwise, since $A'x \leq b'$ are valid inequalities, we

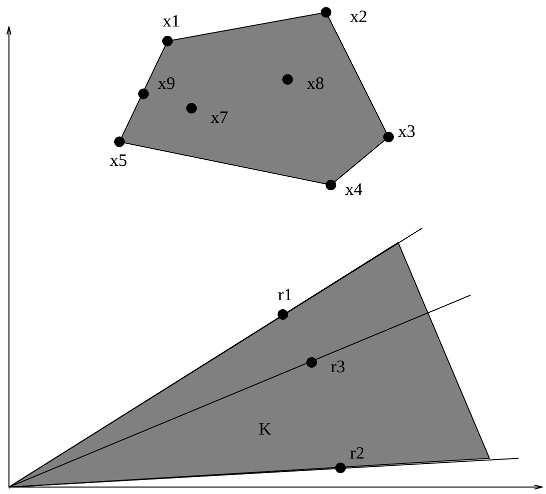

Figure 2.7: Vertices and extreme rays

would get the contradiction $a_i^T x_0 < b_i$ for some inequality in $A'x \leq b'$). But since the equalities in $A'x = b'$ are linearly independent, i.e., A' is nonsingular, we get $x_1 = x_2 = x_0$, which proves that x_0 is an extreme point.

Assume next that $x_0 \in P$ is *not* a vertex of P, and let $A'x \leq b'$ be the inequalities among $Ax \leq b$ that are satisfied by equality in x_0. From the assumption, we have that $\text{rank}(A') < n$. Thus there is a nonzero vector $z \in \mathbb{R}^n$ with $A'z = 0$. We may now find $\epsilon > 0$ "suitably small" such that the two points $x_1 = x_0 - \epsilon z$ and $x_2 = x_0 + \epsilon z$ both *strictly* satisfy each inequality in $Ax \leq b$, but not in $A'x \leq b'$. Furthermore, for each inequality $a_i^T x \leq b_i$ in $A'x \leq b'$ we have $a_i^T x_1 = a_i^T x_0 - \epsilon a_i^T z = a_i^T x_0 \leq b_i$ and similarly $a_i^T x_2 \leq b_i$. Thus, $x_1, x_2 \in P$ and $x_0 = (1/2)x_1 + (1/2)x_2$, and x_0 is not a vertex of P, which completes the proof of (i).

The proof of (ii) is similar, and is left for an exercise.

□

An important consequence of the previous proposition is that the number of vertices and extreme rays is finite.

Corollary 2.12 *Each pointed polyhedron has a finite number of vertices and extreme rays.*

Proof. According to Proposition 2.11 each vertex of a pointed polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is obtained by setting n linearly independent inequalities among the m inequalities in the defining system $Ax \leq b$ to equality. But there are only a finite number of such choices of subsystems (in fact, at most $m!/(n!(m-n)!)$), so the number of vertices is finite. For similar reasons the number of extreme rays is finite (at most $m!/((n-1)!(m-n+1)!)$). □

The next step in our project of proving the interior representation theorem for polyhedra is to show that we obtain optimal solutions that are vertices or extreme rays when solving a linear programming problem over a pointed polyhedron P.

Proposition 2.13 *Consider a nonempty pointed polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ and consider the LP problem*

$$\max\{c^T x : x \in P\}. \tag{2.11}$$

(i) If $v(P)$ is finite, there is a vertex x_0 which is an optimal solution of (2.11), i.e., $c^T x_0 = v(P)$.

(ii) If $v(P) = \infty$ (problem (2.11) is unbounded) , then there is an extreme ray $R(x; r) = x + \mathrm{cone}(\{r\})$ of P with $c^T r > 0$, and therefore $R(x; r) \subseteq P$ and $x + \lambda r \to \infty$ whenever $\lambda \to \infty$ for any $x \in P$.

Proof. (i) Let $v^* = v(P)$ be the finite optimal value, and define the set $F = \{x \in P : c^T x = v^*\}$, so F consists of all optimal solutions in (2.11), and F is nonempty. Consider the pair of dual LP's given in (2.1) (or (2.11)) and (2.2). It follows from the LP duality theory, see Theorem 2.6 and Corollary 2.7, that (D) is feasible and that $\max\{c^T x : Ax \leq b\} = \min\{y^T b : y^T A = c^T,\ y \geq 0\}$. Consider an optimal dual solution y^*, so $(y^*)^T A = c^T$, $y^* \geq 0$ and $(y^*)^T b = v(D) = v(P)$. Define $I' = \{i \in I : y_i > 0\}$. We claim that

$$F = \{x \in P : a_i^T x = b_i \text{ for all } i \in I'\}. \tag{2.12}$$

To see this, note that for each $x \in P$ we have $c^T x = ((y^*)^T A)x = (y^*)^T Ax = \sum_{i \in I} y_i^*(Ax)_i = \sum_{i \in I'} y_i^*(Ax)_i \leq \sum_{i \in I'} y_i^* b_i = v(P)$. Thus we have $c^T x = v(P)$ if and only if $a_i^T x = b_i$ for each $i \in I'$, and (2.12) follows.

For each $\bar{I} \subseteq I$ we define the polyhedron $P(\bar{I}) = \{x \in \mathbb{R}^n : a_i^T x = b_i \text{ for } i \in \bar{I},\ a_i^T x \leq b_i \text{ for } i \in I \setminus \bar{I}$. Thus, $F = P(I')$ and therefore F is a polyhedron contained in P (in fact, F is a *face* of P as we shall discuss later).

We claim that F contains a vertex of P and prove this by a dimension reduction argument. Let initially $\bar{I} = I'$. Let $m = \dim(F)$, so $0 \leq m \leq \dim(P)$. If $m = 0$, then F consists of one point x_0 only, and this point must be a vertex (since $\mathrm{rank}(\{a_i : i \in I'\}) = n$, there must be n linearly independent vectors among a_i, $i \in I'$ and x_0 is a vertex) and we are done. If $m > 0$, choose a point $x_0 \in F$, and a nonzero vector $z \in \mathrm{aff}(F)$ (which is possible since this set has dimension $m > 1$). Note that $a_i^T z = 0$ for each $i \in I'$ as all the vectors $a_i, i \in I'$ belongs to the equality system of F. Consider the line $L = \{x_0 + \lambda z : \lambda \in \mathbb{R}\}$ which then intersects F, i.e., $x_0 \in L \cap F$. On the other hand, as P is pointed, P, and therefore F, cannot contain the line L. Thus there is an $i \in I \setminus I'$ such that the halfspace defined by $a_i^T x \leq b_i$ does not contain L. Consider the unique point x_1 in the intersection between L and the hyperplane $\{x \in \mathbb{R}^n : a_i^T x = b_i\}$. Note that $x_1 \in F$. Update I' by setting $I' := I' \cup \{i\}$. Then $F(I')$ is a polyhedron of dimension at most $m - 1$ and $x_1 \in F(I') \subset F$. We can now repeat this procedure of adding inequalities set to equality, and eventually we will have a polyhedron $F(I') \subset F$ which consists of only one point; a vertex of P which proves (i).

(ii) Assume that (2.11) is unbounded, so the dual problem is infeasible (by Corollary 2.7). Then it follows from Farkas' lemma (exactly as in the proof of Corollary 2.6) that there is a vector $r_0 \in \mathbb{R}^n$ with $Ar_0 \leq 0$ and $c^T r_0 > 0$. Thus r_0 is the direction vector of some ray of P, and we now explain how to find an extreme ray. Consider the LP problem (Q) max $\{c^T z : Az \leq 0,\ c^T z \leq 1\}$. This problem is feasible, as λr_0 is a feasible solution for suitable $\lambda \geq 0$, and clearly (Q) is bounded. We can therefore apply part (i) of this proof, and obtain an optimal solution r of (Q) which is a vertex of the polyhedron defined by $Az \leq 0,\ c^T z \leq 1$. We have that $v(Q) = 1$ (an optimal solution is λr_0 for $\lambda = 1/(c^T r_0)$). Thus any optimal solution of (Q), and therefore also r, must satisfy $c^T z \leq 1$ with equality. Therefore we can find a subsystem $A'z \leq 0$ of $Az \leq 0$ consisting of $n - 1$ vectors such that these vectors augmented with c are linearly independent and such that the unique solution of $A'z = 0,\ c^T z = 1$ is r. It follows from Proposition 2.11 that r a direction vector of an extreme ray of P and, in addition, we have $c^T r > 0$, so (ii) has been proved.

□

Consider an LP problem $\max\{c^T x : x \in P\}$ where P is a bounded polyhedron, which, therefore, has no ray. Then the previous proposition shows that we can solve the LP problem by comparing the objective function for all the vertices, and there is a finite number of these. This is not a good algorithm in the general case, because the number of vertices may be huge, even with a rather low number of inequalities in the defining system $Ax \leq b$. However, the fact that LP's may be solved by searching through vertices only has been the starting point for another algorithm, the *simplex method*, which, in practice, is a very efficient method for solving linear programming problems.

We next combine the previous optimization result with Farkas' lemma and obtain the following **interior representation theorem for polyhedra**.

Theorem 2.14 *Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be a nonempty pointed polyhedron with vertex set V. Let R be a set consisting of exactly one direction vector for each extreme ray of P. Then P can be decomposed as follows*

$$P = \text{conv}(V) + \text{cone}(R). \tag{2.13}$$

Proof. Let $Q = \text{conv}(V) + \text{cone}(R)$, and also $V = \{v_1, \ldots, v_t\}$ and $R = \{r_1, \ldots, r_m\}$. For each $i \leq t$, the vertex v_i must satisfy $Av_i \leq b$, and for each $j \leq m$ the ray directon vector r_j satisfies $Ar_j \leq 0$. A point $x \in Q$ can be written as $x = \sum_i \lambda_i v_i + \sum_j \mu_j r_j$ for suitable $\lambda_i \geq 0$ for each $i \leq t$ such that $\sum_i \lambda_i = 1$ and $\mu_j \geq 0$ for $j \leq m$. Then we have $Ax = \sum_i \lambda_i Av_i + \sum_j \mu_j Ar_j \leq \sum_i \lambda_i b + \sum_j \mu_j 0 = b \sum_i \lambda_i = b$, so $x \in P$. Therefore, $Q \subseteq P$.

To prove the converse inclusion, assume that $x_0 \notin Q$. Thus the linear system $\sum_{i \leq t} \lambda_i v_i + \sum_{j \leq m} \mu_j r_j = x_0, \;\; \sum_{i \leq t} \lambda_i = 1, \; \lambda \geq 0, \mu \geq 0$ has no solution (λ, μ). Therefore, by Farkas' lemma (Theorem 2.3) there is $c \in \mathbb{R}^n$ and $d \in \mathbb{R}$ such that $c^T v_i + d \leq 0$ for $i \leq t, \;\; c^T r_j \leq 0$ for $j \leq m$ and $c^T x_0 + d > 0$. Consider the LP problem (P) max $\{c^T x : x \in P\}$. It follows from Proposition 2.13 that (P) must be bounded (for, otherwise, there would be an direction vector r of an extreme ray with $c^T r > 0$ which is impossible as $c^T r_j \leq 0$ for all j). Thus, again by Proposition 2.13, there must an optimal solution of (P) which is a vertex. Since $c^T v_i + d \leq 0$, the optimal value of (P) can not be greater than $-d$. But $c^T x_0 > -d$, so x_0 can not be feasible in (P), i.e., $x_0 \notin P$. This proves that $P \subseteq Q$ and the proof is complete.

□

In order to generalize this result to hold for all polyhedra, not just pointed ones, we show how to decompose a polyhedron in terms of a linear subspace and a pointed polyhedron.

Lemma 2.15 *Let P be a nonempty polyhedron. Then P can be decomposed as $P = L + Q$, where $L = \text{lin.space}(P)$ and $Q = P \cap L^\perp$ is a pointed polyhedron.*

Proof. Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. Since $L = \{x \in \mathbb{R}^n : Ax = 0\}$ is a linear subspace, we may view $\mathbb{R}^n$ as the direct sum of L and $L^\perp$. Let $Q = P \cap L^\perp$. We claim that $P = L + Q$. To see this, decompose $x \in P$ by $x = x_1 + x_2$ where $x_1 \in L$ and $x_2 \in L^\perp$. Since $x_1 \in L$, we have $Ax_1 = 0$ and combined with $A(x_1 + x_2) \leq b$, we get that $Ax_2 \leq b$ which proves that $x_2 \in P$. Thus, $x = x_1 + x_2 \in L + Q$, and we have shown that $P \subseteq L + Q$. The opposite inclusion is direct: if $x = x_1 + x_2$ with $x_1 \in L$ and $x_2 \in Q$, then $Ax_1 = 0$ and $Ax_2 \leq b$, so $Ax = Ax_2 \leq b$ and $x \in P$. It remains to prove that Q is pointed. Assume not, and let $L' = \{x_0 + \lambda r : \lambda \in \mathbb{R}\}$, where $r \neq 0$, be a line in $Q = P \cap L^\perp$. But then $r \in \text{lin.space}(P) = L$ and $r \in \text{lin.space}(L^\perp) = L^\perp$

(by a previous remark). Thus, $r \in L \cap L^{\perp} = \{0\}$, and $r = 0$; a contradiction. Thus Q contains no line and is therefore pointed.

□

Proposition 2.16 *Any nonempty polyhedron P may be decomposed by*

$$P = \text{conv}(V) + \text{cone}(R) + L \tag{2.14}$$

where $L = \text{lin.space}(P)$, and where V is the set of vertices of the pointed polyhedron $P \cap L^{\perp}$, and R consists of one direction vector for each extreme ray of $P \cap L^{\perp}$.

The decomposition in (2.14) is minimal in the sense that if $P = \text{conv}(V') + \text{cone}(R') + L'$ where L' is a linear subspace, then $L' = L$, $V \subseteq V'$ and $R \subseteq R'$.

Proof. The existence of the decomposition in (2.14) is an immediate consequence of Theorem 2.14 and Lemma 2.15, and the proof of the minimality is left for an exercise.

□

A **polyhedral cone** is a cone of the form $\{x \in \mathbb{R}^n : Ax \leq 0\}$ with $A \in \mathbb{R}^{m,n}$, i.e., a polyhedron defined by a finite set of homogeneous linear inequalities. The next result, called the **Farkas-Minkowski-Weyl theorem**, establishes the equivalence between polyhedral cones and finitely generated cones. Note that the proof uses polarity to obtain one part of this equivalence.

Theorem 2.17 *A convex cone is polyhedral if and only if it is finitely generated.*

Proof. The result is trivial for empty cones, so let $K = \{x \in \mathbb{R}^n : Ax \leq 0\}$ be a nonempty polyhedral cone. First we note that the only vertex of a cone (if any) is 0. Therefore, in Proposition 2.16 we must have $V = \{0\}$ and $P = \text{cone}(R) + L$ for some finite set R and a linear subspace L. But, each linear subspace is also a finitely generated cone which is seen as follows. Select a basis $a_1, \ldots, a_s$ for L, so $L = \text{span}(\{a_1, \ldots, a_s\})$. Let $u = -\sum_{j \leq s} a_j$. Then it is easy to check that $L = \text{cone}(\{u, a_1, \ldots, a_s\})$. Therefore we see that $P = \text{cone}(R')$ for a finite set R', and P is finitely generated.

Conversely, let $K \subseteq \mathbb{R}^n$ be a finitely generated cone, so $K = \text{cone}(R)$ for a finite set $R = \{a_1, \ldots, a_s\}$. K is a closed set (see Proposition 1.16) and combining this with the polarity result of Proposition 1.23 we have $K = K^{\circ\circ}$. Next, using Proposition 1.11, we have $K^{\circ} = \{x \in \mathbb{R}^n : a_i^T x \leq 0 \text{ for } i = 1, \ldots, s\}$ so K° is a polyhedral cone. We can therefore apply the first part of the proof to K° and get $K^{\circ} = \text{cone}(\{b_1, \ldots, b_m\})$ for suitable vectors $b_1, \ldots, b_m$. Thus we have $K = K^{\circ\circ} = (\text{cone}(\{b_1, \ldots, b_m\})^{\circ} = \{x \in \mathbb{R}^n : b_i^T x \leq 0 \text{ for } i = 1, \ldots, m\}$ (again using Proposition 1.11). Thus K is a polyhedral cone, and the proof is complete.

□

Note the content of the Farkas-Minkowski-Weyl theorem: a convex cone is the solution set of a *finite* number of linear inequalities iff it is generated by a *finite* number of vectors. Thus, the theorem establishes an equivalence between a finite exterior description and a finite interior description. A similar result may now be shown for polyhedra and it is called the **the decomposition theorem for polyhedra** or **Motzkin's representation theorem**.

Theorem 2.18 *For each polyhedron $P \subseteq \mathbb{R}^n$ there are finite sets V and R such that $P = \text{conv}(V) + \text{cone}(R)$.*

Conversely, if V and R are finite sets, then $\text{conv}(V) + \text{cone}(R)$ *is a polyhedron.*

Proof. Assume that P is a nonempty polyhedron, and then we have from Proposition 2.16 that $P = \text{conv}(V) + \text{cone}(R') + L$ where L is a linear subspace and V and R' are finite sets. But as we saw in the proof of the Farkas-Minkowski-Weyl theorem, L may be written as a finitely generated cone, say $L = \text{cone}(R'')$. Let $R = R' \cup R''$ and note that $\text{cone}(R) = \text{cone}(R') + \text{cone}(R'')$. Therefore $P = \text{conv}(V) + \text{cone}(R)$ as desired.

Conversely, let V and R be finite sets and define $Q = \text{conv}(V) + \text{cone}(R)$. Note that $x \in Q$ if and only if $(x, 1) \in Q'$ where $Q' = \text{cone}(\{(v, 1) \in R^{n+1} : v \in V\} \cup \{(r, 0) \in R^{n+1} : r \in R\})$. The cone Q' may be viewed as the sum of the homogenization of Q (see section 1) and the cone generated by the $(r, 0)$, $r \in R$. We apply Theorem 2.17 so there is a $Ax + cx_{n+1} \leq 0$ with solution set Q'. Thus $x \in Q$ iff $(x, 1)$ satisfies this linear system, i.e., $Ax \leq -c$. Thus, P is a polyhedron as desired.

□

In Figure 2.8 we show a polyhedron P given by $P = \text{conv}(V) + \text{cone}(R)$.

An important corollary of the decomposition theorem for polyhedra concerns polytopes. Recall that a polytope is the convex hull of a finite number of points. The next result is called the **finite basis theorem for polytopes**.

Theorem 2.19 *A set P is a polytope if and only if it is a bounded polyhedron.*

Proof. Let P be a polytope, say $\text{conv}(V)$ for a finite set $V \subset \mathbb{R}^n$. It then follows from the second part of Theorem 2.18 that P is a polyhedron. Furthermore, P is bounded (it is contained in the ball with center in the origin

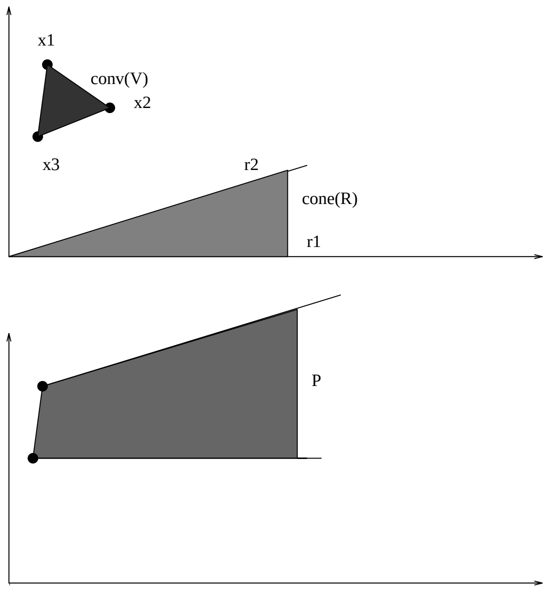

Figure 2.8: Decomposition theorem for polyhedra

and radius $\max_{v \in V} \|v\|$). Conversely, let P be a bounded polyhedron, and let $P = \text{conv}(V) + \text{cone}(R)$ be the decomposition of Theorem 2.18. Since P is bounded, we must have $R = \emptyset$, so P is a polytope as desired.

□

2.5 Faces of polyhedra and exterior representation of polyhedra

In the previous section we saw that for a pointed polyhedron P there is always an optimal vertex solution of an LP problem $\max \{c^T x : x \in P\}$. In this section we study the set of optimal solutions of LP problems for general polyhedra. We also study minimal exterior representation of polyhedra.

Let $c \in \mathbb{R}^n$ be a nonzero vector. Associated with c we have the hyperplanes $H_=(c, \alpha) = \{x \in \mathbb{R}^n : c^T x = \alpha\}$ for each $\alpha \in \mathbb{R}$. Now, view c as an objective function and consider an optimization problem (P) max $\{c^T x : x \in S\}$ over some set $S \subseteq \mathbb{R}^n$. Note that the hyperplane $H_=(c, \alpha)$ contains all the points (in $\mathbb{R}^n$) with equal value on the objective function, and similarly, $H_=(c, \alpha) \cap S$ consist of all the feasible points in S with equal objective function value. The optimal value of (P) is the largest possible value α, say $\alpha = \alpha_0$, with the property that the set $H_=(c, \alpha) \cap S$ is nonempty. Then $H_=(c, \alpha_0) \cap S$ is the set of optimal solutions. For an arbitrary set S

of feasible solutions the optimal set $H_=(c, \alpha_0) \cap S$ may be very "complex", for instance, it may be nonconvex. However, in the case of interest to us, when S is a polyhedron, this set $H_=(c, \alpha_0) \cap S$ turns out to be very nice, it is another polyhedron. This is discussed in detail below.

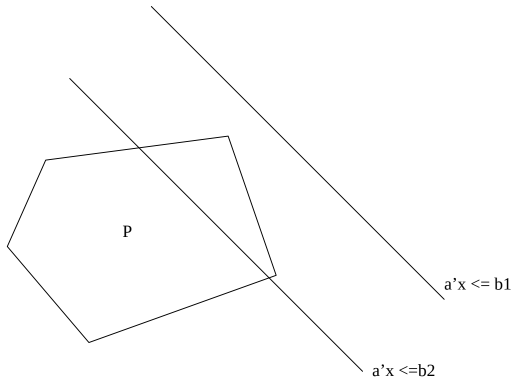

Figure 2.9: Valid inequality

Consider a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. A **valid inequality** for P is a linear inequality $c^T x \leq \alpha$ satisfied by all points in P, i.e., $P \subseteq H_\leq(c, \alpha) = \{x \in \mathbb{R}^n : c^T x \leq \alpha\}$. Note that $c^T x \leq \alpha$ is a valid inequality if and only if max $\{c^T x : x \in P\} \leq \alpha$. A valid inequality $c^T x \leq \alpha$ is **supporting** if there is some point $x_0 \in P$ with $c^T x_0 = \alpha$, i.e., the hyperplane $H_=(c, \alpha)$ intersects P. Thus, $a^T x \leq \alpha$ is a supporting inequality if and only if max $\{c^T x : x \in P\} = \alpha$. In Figure 2.9 the inequality $c^T x \leq b_1$ is valid for P, but it is not supporting. The inequality $c^T x \leq b_2$ is not valid for P. A **face** of P is a set F of the form

$$F = H_=(c, \alpha) \cap P = \{x \in P : a^T x = \alpha\} \tag{2.15}$$

for a valid inequality $c^T x \leq \alpha$; we say that F is the face **induced** or **defined** by $c^T x \leq \alpha$. We also call P itself a face of P (and we could say that it is induced by the valid inequality $\mathbf{0}^T x \leq 0$). The empty set and P are **trivial faces**, while all other faces are called **proper faces**. We see that the proper faces are obtained from supporting inequalities for which the associated hyperplane does not contain P. Thus the faces of a polyhedron are those point sets on the boundary of P that can be obtained as the set of optimal solutions of LP problems over P, or equivalently, intersection between P and its supporting hyperplanes. In Figure 2.10 we show a valid inequality for a polyhedron P and the induced face F.

There is another useful description of faces of polyhedra that connect these sets to the defining inequalities $Ax \leq b$ of the polyhedron P. In fact,

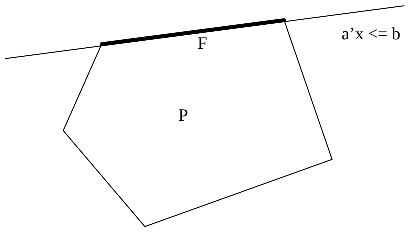

Figure 2.10: Face of a polyhedron

we have met this result before, but then it was hidden in the the proof of Proposition 2.13.

Theorem 2.20 *Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. A nonempty set F is a face of P if and only if*

$$F = \{x \in P : A'x = b'\} \tag{2.16}$$

for some subsystem $A'x \leq b'$ of $Ax \leq b$.

Proof. Let F be a nonempty face of P, so $F = H_=(c, \alpha) \cap P$ for a supporting inequality $c^T x \leq \alpha$. Then the optimal value $v(P)$ of the LP problem (P) max $\{c^T x : Ax \leq b\}$ satisfies $v(P) = \alpha < \infty$ and furthermore $F = \{x \in P : c^T x = v(P)\}$. It follows from the duality theorem, Theorem 2.6, that the LP dual of problem (P) is feasible and that $\max\{c^T x : Ax \leq b\} = \min\{y^T b : y^T A = c^T, y \geq 0\}$. Let y^* be an optimal dual solution, so $(y^*)^T A = c^T$, $y^* \geq 0$ and $(y^*)^T b = v(D) = v(P)$, and define $I' = \{i \in I : y_i > 0\}$. We claim that (2.16) holds with the subsystem $A'x \leq b'$ consisting of the inequalities $a_i^T x \leq b_i$ for $i \in I'$. To see this, note that for each $x \in P$ we have $c^T x = ((y^*)^T A)x = (y^*)^T Ax = \sum_{i \in I} y_i^* (Ax)_i = \sum_{i \in I'} y_i^* (Ax)_i \leq \sum_{i \in I'} y_i^* b_i = v(P)$. Thus we have $c^T x = v(P)$ if and only if $a_i^T x = b_i$ for each $i \in I'$, and (2.16) holds.

Conversely, assume that the set F satisfies (2.16) for some subsystem $A'x \leq b'$ consisting of inequalities $a_i^T x \leq b_i$, $i \in I'$ from $Ax \leq b$. Let $c = \sum_{i \in I'} a_i$ and $\alpha = \sum_{i \in I'} b_i$. Then $c^T x \leq \alpha$ is a valid inequality for P (it is a sum of other valid inequalities). Furthermore F is the face induced by $c^T x \leq \alpha$, i.e., $F = \{x \in P : c^T x = \alpha\}$. In fact, a point $x \in P$ satisfies $c^T x = \alpha$ if and only if $a_i^T x = b_i$ for each $i \in I'$.

□

We therefore see that the faces of $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ are precisely the sets of the form $F = \{x \in \mathbb{R}^n : A'x = b', A''x \leq b''\}$ for some partition $A'x \leq b'$, $A''x \leq b''$ of the linear system $Ax \leq b$.

Corollary 2.21 *Consider a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. Then* *(i) P has a finite number of faces.*

(ii) Each face of P is a polyhedron.

(iii) Let F be a face of P (so F is a polyhedron by (ii)). Then a subset G of F is a face of F if and only if it is a face of P.

Proof. All these results follow from Theorem 2.20. (i) Each face is obtained by setting a subsystem of inequalities from $Ax \leq b$ to equality, and there are a finite number, in fact 2^m, of such subsystems. (Remark: there are 2^m subsystems, but many of these may induce the same face. In fact, estimating the number of faces is a discipline on its own, within discrete geometry).

(ii) This is clear as each face is the solution set of a finite set of linear inequalities and equations.

(iii) Let $F = \{x \in \mathbb{R}^n : A'x = b', A''x \leq b''\}$. Each face G of F is obtained by setting some of the inequalities in $A''x \leq b''$ to equality and so this is also a face of P. Conversely, let G be a face of P with $G \subseteq F$, say $G = \{x \in P : A'x = b'\}$ and $F = \{x \in P : A''x = b''\}$. Since $G \subseteq F$, we must have $G = \{x \in P : A'x = b', A''x = b''\}$, and therefore G is a face of F. $\square$

Faces of polyhedra are interesting from an optimization viewpoint, and, in particular, one is concerned with minimal and maximal faces. A **minimal face** of P is a minimal nonempty face, i.e., a nonempty face of P not strictly containing any other nonempty face of P. Similarly, a **maximal face** of P is a face $F \neq P$ not strictly contained in any proper face of P. Each maximal face of P is also called a **facet** of P.

The minimal faces of P are related to the lineality space of P and the facets are related to a minimal exterior representation of P. We shall discuss these relations in detail in the following. First, we give a characterization of the minimal faces due to Hoffman and Kruskal (1956).

Theorem 2.22 *Let $\emptyset \neq F \subset P = \{x \in \mathbb{R}^n : Ax \leq b\}$. Then F is a minimal face of P if and only if F is an affine set of the form*

$$F = \{x \in \mathbb{R}^n : A'x = b'\} \tag{2.17}$$

for some subsystem $A'x \leq b'$ of $Ax \leq b$.

Proof. Assume that F is a minimal face of P, say $F = \{x \in \mathbb{R}^n : A'x = b', A''x \leq b''\}$, where we may assume that each inequality in $A''x \leq b''$ is nonredundant and not an implicit equality. Then, one of the inequalities in $A''x \leq b''$ has the property that if we set it to equality we obtain a nonempty face strictly contained in F, which contradicts the minimality of F. Thus, the system $A''x \le b''$ must be empty, and $F = \{x \in \mathbb{R}^n : A'x = b'\}$ which proves that F is an affine set of the form (2.17).

Conversely, assume that F is given by (2.17) which we rewrite by replacing each equality by two opposite inequalities. By Theorem 2.20, any face of F is obtained by setting some of these inequalities to equality, but clearly this does not change the solution set F (since all the inequalities are implicit equalities). Thus F is a minimal face.

□

Corollary 2.23 *Each minimal face F of $P = \{x \in \mathbb{R}^n : Ax \le b\}$ is a translate of* lin.space(P)*, and therefore* $\dim(F) = n - \text{rank}(A)$*. If P is pointed, the minimal faces are precisely the one-point sets consisting of the vertices of P.*

Proof. Let F be a minimal face of P, so by Theorem 2.22 we have that F is the affine set $F = \{x \in \mathbb{R}^n : A'x = b'\}$ for some subsystem $A'x \le b'$ of $Ax \le b$. Let $x_0 \in F$ (F is nonempty) and then $F = x_0 + L$ where L is the unique linear subspace parallel to F, i.e., $L = \{x \in \mathbb{R}^n : A'x = 0\}$. Since A' contains a subset of the rows of A, we have $\text{rank}(A') \le \text{rank}(A)$. In fact, we here have equality, for, if not, there would be an inequality $a_i^T x \le b_i$ in $Ax \le b$ such that a_i is not a linear combination of the rows of A', which then gives the contradiction $F \subseteq \{x : A'x = b',\ a_i^T x \le b_i\} \subset \{x : A'x = b'\} = F$ (where the last inclusion is strict). We therefore get that $L = \text{lin.space}(P)$.

□

Note the following important consequence of the previous corollary: a point x in a polyhedron P is a vertex of P if and only if it is the unique optimal solution of an LP problem over P. This fact is sometimes useful in finding all the vertices of a polyhedron.

An **edge** of a polyhedron is a bounded face of dimension one (so it is not a ray). Each edge is the line segment between two vertices and these two vertices are said to be **adjacent**.

There is a convex cone which is naturally associated with a polyhedron. For a polyhedron $P = \{x \in \mathbb{R}^n : Ax \le b\}$ we define its **characteristic cone** $\text{char.cone}(P) = \{y \in \mathbb{R}^n : Ay \le 0\}$. Each nonzero element in char.cone(P) is called an **infinite direction** of P , see also Section 2.4. (Some authors also use the term ray here, but we reserve the notation ray for the geometric object $R(x;r)$ defined previously). We leave it as an exercise to show that

$$\begin{array}{ll} \text{(i)} & \text{char.cone}(P) = \{y \in \mathbb{R}^n : x + y \in P \text{ for all } x \in P\}; \\ \text{(ii)} & \text{lin.space}(P) = \text{char.cone}(P) \cap -\text{char.cone}(P). \end{array} \tag{2.18}$$

As an application of the theory developed, let us consider again a pair of dual LP problems and relate their optimal solutions. Consider the linear programming problems

$$\text{(P)} \quad \max\{c^T x : x \in P\}, \tag{2.19}$$

and

$$\text{(D)} \quad \min\{u^T b : u \in Q\} \tag{2.20}$$

where we, for simplicity, assume that both polyhedra P and Q are pointed. Let these polyhedra be given by

$$\begin{aligned} P &= \{x \in \mathbb{R}^n : x \geq 0,\ Ax \leq b\} \\ &= \text{conv}(V_P) + \text{cone}(R_P); \\ Q &= \{u \in \mathbb{R}^m : u \geq 0,\ u^T A \geq c^T\} \\ &= \text{conv}(V_Q) + \text{cone}(R_Q), \end{aligned} \tag{2.21}$$

where V_P, resp. R_P, is the (finite) set of extreme points (vertices) and directions corresponding to extreme rays of P. V_Q resp. R_Q are similar objects defined for Q.

Theorem 2.24 *For the problems (P) and (D) the following hold.*
(i) (P) is feasible iff $b^T r \geq 0$ *for all* $r \in R_Q$*.*
(ii) When (P) is feasible the following statements are equivalent:
  a) $v(P) = \infty$*.*
  b) There is an $r \in R_P$ *with* $c^T r > 0$*.*
  c) $Q = \emptyset$*.*
(iii) If $v(P)$ *is finite, then* $v(P) = \max_{v \in V_P} c^T v = \min_{u \in V_Q} u^T b = v(D)$

Proof. (i) From Farkas' lemma we see that (P) is feasible (i.e., $P \neq \emptyset$) iff $\nu^T b \geq 0$ for all $\nu \geq 0$ with $\nu^T A \geq 0$. It follows from Theorem 2.17 that char.cone$(Q) = \{\nu \in R^m_+ : \nu^T A \geq 0\} = \text{cone}(R_Q)$. Thus $\nu^T b \geq 0$ for all $\nu \geq 0$ with $\nu^T A \geq 0$ iff $r^T b \geq 0$ for all $r \in R_Q$.

(ii) It follows from the interior representation of P that $v(P)$ is finite iff $c^T r \leq 0$ for all $r \in R_P$, and the equivalence for the dual problem follows similarly.

(iii) This follows by combining the LP duality theorem with the interior representations above.

□

We now turn to a study of minimal exterior descriptions of polyhedra. The main goal is to show that, in addition to the implicit equality system, these descriptions are given by inequalities that define facets of the polyhedron.

In order to discuss minimal representations of polyhedra we find it convenient to write the linear system with both inequalities and equalities. Consider a linear system

$$\begin{array}{ll} a_i^T x = b_i & \text{for } i \in I_1; \\ a_i^T x \le b_i & \text{for } i \in I_2, \end{array} \tag{2.22}$$

and let P be its solution set. We call a constraint (inequality or equality) in (2.22) **redundant** if the solution set is unchanged if this constraint is removed. We call the system (2.22) **minimal** if

(i) no inequality can be set to equality without reducing the solution set;

(ii) no inequality or equality is redundant.

A **minimal representation** of a polyhedron P is a minimal linear system of the form (2.22) with solution set P. It is clear that every polyhedron has a minimal representation (one simply removes redundant constraints as long as possible, see Problem 2.9). An interesting result is that one also has a kind of uniqueness of a minimal representation as discussed next.

Theorem 2.25 *Let P be the polyhedron defined by (2.22) and assume that P is nonempty. Then (2.22) is a minimal representation of P if and only if (i) the vectors a_i, $i \in I_1$ are linearly independent, and (ii) there is a bijection (one-to-one and onto mapping) between the facets of P and the inequalities $a_i^T x \le b_i$ for $i \in I_2$ given by $F = \{x \in P : a_i^T x = b_i\}$.*

Proof. Assume first that the system (2.22) is minimal. If an a_i for some $i \in I_1$ is a linear combination of the other vectors in I_1, then either the equation $a_i^T x = b_i$ is redundant, or the equation system is inconsistent (and $P = \emptyset$); in any case a contradiction. Thus a_i, $i \in I_1$ are linearly independent.

Next, let $i \in I_2$ and let P' be the solution system when the inequality $a_i^T x \le b_i$ is removed from (2.22). As $a_i^T x \le b_i$ is nonredundant, there is a point $x_1 \in P' \setminus P$. Thus, x_1 satisfies all the constraints in (2.22) except that $a_i^T x_1 > b_i$. As shown before (see Lemma 2.8), P has an inner point x_2, i.e., $a_i^T x_2 = b_i$ for $i \in I_1$, $a_i^T x < b_i$ for $i \in I_2$. Therefore, some convex combination $\bar{x}$ of x_1 and x_2 must satisfy $a_j^T \bar{x} = b_j$ for $j \in I_1$ and $a_j^T x < b_j$ for $j \in I_2 \setminus \{i\}$ and finally $a_i^T \bar{x} = b_i$. Let F be the face of P induced by the inequality $a_i^T x \le b_i$, and note that $\bar{x} \in F$. Therefore the equality set of F consists of the constraints with index set $I_1 \cup \{i\}$ (remark: $\bar{x}$ shows that the equality set can not contain other inequalities). F is therefore a maximal face because any face G strictly containing F must have equality system $a_i^T x = b_i$ for $i \in I_1$, and this is only the case for the trivial face $G = P$. Therefore, F is a maximal face, i.e., a facet of P induced by the inequality $a_i^T x \le b_i$.

Furthermore, no other inequality can induce the same facet, because the point $\bar{x}$ above satisfies $a_i^T x \leq b_i$ with equality, but it satisfies no other inequality in $a_j^T x \leq b_j$ for $j \in I_2$ with equality. Finally, we note that each facet is of the form $F = \{x \in P : a_i^T x = b_i\}$ for some $i \in I_2$ (as a facet is also a face, so it is induced by some inequality, and, furthermore, this inequality cannot be an implicit equality, for that would not give a proper face). This proves that condition (ii) holds.

Conversely, assume that conditions (i) and (ii) both hold. Consider an inequality $a_i^T x \leq b_i$ for an $i \in I_2$. If we set this inequality to equality, we obtain by (ii) a facet of P, and this is a proper subset of P. Therefore condition (i) in the definition of a minimal system is satisfied. Furthermore, this also shows that the equality system of (2.22) is $a_j^T x = b_j$ for $j \in I_1$. We next prove that each equality $a_j^T x = b_j$ for $j \in I_1$ is nonredundant. Let A be the affine hull of P, i.e., this is the solution set of the equality system $a_j^T x = b_j$ for $j \in I_1$. Since the vectors a_i, $i \in I_1$ are linearly independent, the removal of one of these equations results in a solution set being an affine set A' with $\dim(A') = \dim(A) + 1$. By the dimension formula for polyhedra it follows that we get $\dim(P') = \dim(P) + 1$ where P' is the solution set of the system obtained by removing $a_i^T x = b_i$ from (2.22). This proves that each equality in (2.22) is nonredundant. It only remains to prove that each inequality is nonredundant, so consider an inequality $a_i^T x \leq b_i$ for some $i \in I_2$. Let x_1 be an interior point in the facet $F = \{x \in P : a_i^T x = b_i\}$ and let x_2 be an interior point of P (note: such points exist as both polyhedra are nonempty). It is easy to see that the line through these two points contains a point which satisfies all the constraints in (2.22) except for $a_i^T x \leq b_i$. Thus this inequality is nonredundant and the proof is complete.

□

We therefore see that, except for implicit equalities, we only need the facet defining inequalities in an exterior representation of a polyhedron.

An interesting property of a proper face of a polyhedron is that it has lower dimension than the polyhedron.

Lemma 2.26 *Let F be a proper face of a polyhedron P (so F is nonempty and strictly contained in P). Then we have* $\dim(F) < \dim(P)$.

Proof. Since $F \subset P$, there is an $x_0 \in P \setminus F$. Furthermore, F is a face of P so $F = P \cap H$ for some hyperplane H. Then $x_0 \notin H$ (for $x_0 \in H$ implies that $x_0 \in P \cap H = F$; a contradiction). But H is an affine set, and therefore $\mathrm{aff}(F) \subseteq H$, and $x_0 \notin \mathrm{aff}(F)$. This gives $\dim(P) = \dim(\mathrm{aff}(P)) \geq \dim(\{x_0\} \cup \mathrm{aff}(F)) > \dim(\mathrm{aff}(F)) = \dim(F)$. (The strict inequality is due to the fact that $\mathrm{aff}(F)$ is strictly contained in $\mathrm{aff}(\{x_0\} \cup \mathrm{aff}(F)))$.

□

We conclude this section with a result which gives further characterization of the facet defining inequalities. This theorem is often used in polyhedral combinatorics.

Theorem 2.27 *Let P be the solution set of a linear system of the form (2.22) which is assumed to be minimal and consistent. Assume that F is a nontrivial face of P. Let $m = |I_2|$. Then the following statements are equivalent.*
(i) F is a facet of P.
(ii) $\dim(F) = \dim(P) - 1$.
(iii)If $a, \bar{a} \in \mathbb{R}^n$ and $\alpha, \bar{\alpha} \in \mathbb{R}$ satisfy
$F = \{x \in P : a^T x = \alpha\} = \{x \in P : \bar{a}^T x = \bar{\alpha}\}$ where both $a^T x \leq \alpha$ and $\bar{a}^T x \leq \bar{\alpha}$ are valid inequalities for P, then there exist $\lambda \in \mathbb{R}^m$ and a positive $\gamma \in \mathbb{R}$ such that $\bar{a} = \gamma a + \lambda A^=$, $\bar{\alpha} = \gamma\alpha + \lambda b^=$.

Proof. Let $A^= x = b^=$, $A^+ x \leq b^+$ be a minimal system for P, and let I_1 and I_2 be the index sets of these two subsystems, respectively. We prefer to show first the equivalence of (i) and (ii), and then the equivalence of (ii) and (iii).

(i) $\Leftrightarrow$ (ii): Assume that F is a facet of P. By Theorem 2.25, there is an $i \in I_2$ such that $F = \{x \in P : a_i^T x = b_i\}$, and (see the proof of that theorem) the equality system of F is $a_j^T x = b_j$ for $j \in I_1$, $a_i^T x = b_i$. The dimension formula gives that $\dim(F) \geq \dim(P) - 1$, and since F is a nontrivial face we also get $\dim(F) \leq \dim(P) - 1$ (see Lemma 2.26), and (ii) holds. Conversely, assume that condition (ii) holds. If F is not a facet (but still a face), there is a nontrivial face G such that $F \subset G \subset P$ (all inclusions strict). By applying Lemma 2.26 twice, we get the contradiction $\dim(F) \leq \dim(P) - 2$. This proves that condition (i) must hold, as desired.

(ii) $\Leftrightarrow$ (iii): Assume that (ii) holds and that $a^T x \leq \alpha$ and $\bar{a}^T x \leq \bar{\alpha}$ are valid inequalities for P that both induce the same facet F. Therefore F is the solution system of the original system (2.22) augmented with the two equations $a^T x = \alpha$ og $\bar{a}^T x = \bar{\alpha}$. Let $Y = \{a_i : i \in I_1\}$, so $|Y| = m$. Note that neither a nor $\bar{a}$ can be a linear combination of the vectors in Y (for this would violate that F is nonempty and has dimension *less* than that of P). On the other hand, the rank of $Y \cup \{a, \bar{a}\}$ can not be larger than $m+1$ because that would imply that $\dim(F) < \dim(P) - 1$. These two observations therefore give that $\text{rank}(Y \cup \{a, \bar{a}\}) = m + 1$, from which we easily get the desired relation between a and $\bar{a}$, and then also between α and $\bar{\alpha}$ as in (iii).

To prove the converse, we assume that F is *not* a facet of P. But then F is the face of P obtained by setting at least two of the inequalities $a_i^T x \leq b_i$ to equality, say for $i = r$ and $i = s$ (see Theorem 2.25). The minimality of (2.22) implies that we can find a point $\bar{x}$ which satisfies all the constraints in (2.22) except $a_k^T x \leq b_k$ (so $a_k^T \bar{x} > b_k$). Then (as we have done previously) a suitable convex combination $\tilde{x}$ of $\bar{x}$ and an inner point of P, satisfies $a_i^T\tilde{x} < b_i$ for $i \in I_2 \setminus \{k\}$, and $a_k^T\tilde{x} > b_k$. Let H consist of those $i \in I_2$ for which $a_i^T x \leq b_i$ is an implicit equality for F. Let $\lambda_i > 0$ for $i \in H$ and define $a = \sum_{i\in H} \lambda_i a_i$ and $\alpha = \sum_{i\in H} \lambda_i b_i$. We see that $a^T x \leq \alpha$ is then a valid inequality for P and that it induces F. Also note that $k \in H$. By suitable choices of λ_i, for $i \in H$ we can find two such valid inequalities $(a^1)^T x \leq \alpha^1$ (choose λ_k small enough) and $(a^2)^T x \leq \alpha^2$ (choose λ_k large enough) such that $(a^1)^T\tilde{x} \leq \alpha^1$ and $(a^2)^T\tilde{x} > \alpha^2$. This shows that $(a^2)^T x \leq \alpha^2$ is not a positive multiple of $(a^1)^T x \leq \alpha^1$ plus some linear combination of the equations $a_i^T x = b_i$ for $i \in I_1$, i.e., condition (iii) does not hold, and the proof is complete.

□

If the polyhedron P is fulldimensional, the previous Theorem shows that there is a *unique* minimal representation of P. Here the uniqueness is up to the multiplication of an inequality by a positive scalar, and the inequalities involved correspond to the facets of P.

2.6 Exercises

Problem 2.1 *Prove Fredholm's alternative directly from Theorem 2.3.*

Problem 2.2 *Show that* $y \in \text{lin.space}(P)$ *iff for each* $x \in P$ *we have* $L_x \subseteq P$ *where* $L_x = \{x + \lambda y : \lambda \in \mathbb{R}\}$.

Problem 2.3 *Prove that for a linear subspace* L *in* $\mathbb{R}^n$ *we have that* $L = \text{lin.space}(L)$.

Problem 2.4 *Prove (ii) in Proposition 2.11.*

Problem 2.5 *Prove the minimality of the decomposition in Corollary 2.16.*

Problem 2.6 *Let* $P \subset \mathbb{R}^2$ *be the polyhedron defined by the following linear inequalities:* $x_1 \geq 0,\ x_2 \geq 0,\ x_1 + x_2 \geq 0,\ x_1 \leq 3,\ x_2 \leq 2,\ x_1 + x_2 \leq 4,\ 2x_1 + 3x_2 \leq 15$. *Illustrate* P *and the inequalities in the plane. Find the face of* P *induced by each of the inequalities. What is the dimension of* P*, and of all the faces? Which inequalities are redundant? Find a minimal representation of* P*. Is this polyhedron a polytope? What is the lineality space and the characteristic cone? You should also give formal proofs for all these properties (not just use geometric intuition!).*

Problem 2.7 *Let* $Q \subset \mathbb{R}^2$ *be the polyhedron defined by the following linear inequalities:* $x_1 \leq 3,\ x_2 \leq 2,\ x_1 + x_2 \leq 4,\ 2x_1 + 3x_2 \leq 15$. *Answer now the same questions as in the Problem 2.6.*

Problem 2.8 *Show that each linear subspace is a finitely generated cone (see the comments in the proof of Theorem 2.17).*

Problem 2.9 *Show that a minimal representation of a polyhedron does indeed exist.*

Problem 2.10 *Let $P \subseteq \mathbb{R}^n$ be a polyhedron of dimension m and with the dimension of its lineality space being d. Show that there is a sequence of faces $F_d, F_{d+1}, \ldots, F_m$ of P such that $\dim(F_j) = j$ for $j = d, \ldots, m$. Also prove that this sequence may be chosen such that F_j is a facet of F_{j+1} for $j = d, \ldots, m-1$.*

Problem 2.11 *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and let $T : \mathbb{R}^n \to \mathbb{R}^m$ be an affine transformation, that is, $T(x) = Cx + d$ for some $m \times n$-matrix C and vector $d \in \mathbb{R}^m$. Consider an inner representation $P = \mathrm{conv}(V) + \mathrm{cone}(R)$ of P. Show that the image $T(P)$ of P under T (i.e., $T(P) = \{T(x) \in \mathbb{R}^m : x \in P\}$) satisfies*

$$T(P) = \mathrm{conv}(T(V)) + \mathrm{cone}(T(R)).$$

Thus, $T(P)$ is a polyhedron as well. Prove next that each face of $T(P)$ is the image under T of some face of P. As a concrete example, let P be the solution set of the (nonlinear) inequalities $|x_i| \leq 1$ for $i = 1, 2, 3$. Explain why P is a polyhedron and find inner and outer descriptions. Let the linear transformation T be given by $T(x_1, x_2, x_3) = (x_1, x_2)$ so this is projection into the xy-plane. Determine $T(P)$. We mention that the polytope P in the example is called a crosspolytope (simialr in $\mathbb{R}^n$). It may also be seen that the set of points with norm no greater than 1 in the l_∞-norm.

Chapter 3

The simplex method

In Chapter 2 we discussed linear programming theory. In particular we studied relations between the optimal solutions in a pair of dual LP problems. The purpose here is to describe the simplex method for solving linear programming problems numerically. Efficient implementations of this algorithm are among the fastest LP codes available today.

There is a vast literature on linear programming. The classical text is G. Dantzig's book [8] on linear programming. More recent literature includes [4], [24], [25] and [29]. The famous book by A. Schrijver [35] treats LP from a theoretical point of view and contains a lot of literature references on the subject. For a presentation on recent algorithmic developments (ellipsoid method, Karmarkar's method etc.), see the survey paper [13]. A comprehensive treatment of the ellipsoid method in connection with implicitly described polyhedra is the book [26].

3.1 Some basic ideas

We use a small example in order to motivate the algebraic manipulations of the simplex method. The underlying geometric ideas are also treated.

Consider the LP problem (P) $\max\{x_1+x_2 \mid x_1 \geq 0,\ x_2 \geq 0,\ x_1 \leq 3,\ x_2 \leq 2.5,\ x_1 + 2x_2 \leq 5\}$ illustated in Figure 3.1. Let P be the feasible set in (P) and note that P is a polytope.

Geometrically we can solve this problem as follows. Let $c = (1, 1)$ be the objective function and consider a hyperplane $H_=(c, b) = \{x \in \mathbb{R}^2 \mid c^T x = b\}$ for each $b \in \mathbb{R}$; these are all parallel hyperplanes (i.e., they are all parallel to the same linear subspace). Thus $H_=(c, b)$ consists of those solutions (points) with the same value b on the objective function. We therefore find the optimal value $v(P)$ as the maximum b such that $F := H_=(c, b) \cap P \neq \emptyset$, and then F

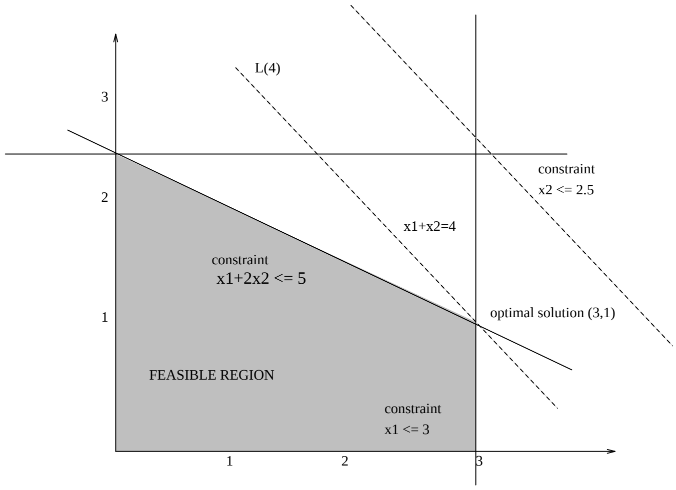

Figure 3.1: The example LP

is the set of optimal solutions. In the example, $v(P) = 4$ and $F = \{(3, 1)\}$. Although this geometric approach can only be used to solve problems with two (or possibly three) variables, it gives some (geometric) understanding of what happens in higher dimensional spaces as well.

Now, let us take a look at the algebraic side. First, recall from Chapter 2 that there is an optimal solution which is also a vertex because the polyhedron P is both pointed and bounded. Consider the vertex $x^0 = (0, 0)$ of P. This point is the unique solution obtained by setting the two defining inequalities $x_1 \geq 0$ and $x_1 \geq 0$ to equality. This solution is not optimal, which may be explained by the fact that (at least) one of the *active* constraints, $x_1 = 0$, hinders us in improving the objective function. In fact, if we remove the constraint $x_1 = 0$ (defining x^0), but maintain the constraint $x_2 = 0$, we make an improvement possible. More precisely, the set $\{x \in P \mid x_2 = 0\}$ defines an edge of P (a face of dimension 1) and we can move along this edge until we meet another vertex $x^1 = (3, 0)$, and this vertex is the solution of the original equation $x_1 = 0$ combined with the defining inequality $x_1 \leq 3$ which has now become active. This process of moving from a vertex to another (adjacent) vertex along an edge of the feasible polyhedron is precisely the geometric idea underlying the simplex algorithm. The algebraic operations are: (i) maintaining a set of active constraints defining the current vertex, (ii) testing if loosening one of these constraints may improve the objective function, and, if so, (iii) calculate the new solution obtained by moving as far as feasibility allows, and we then find a new active constraint (linearly independent of the others). Finally, we should mention that optimality of the last solution corresponds to finding a "matching" dual feasible solution so that weak duality serves as a stopping rule!

The method is given in detail in the next section. Actually the simplex method consists of two stages: first one finds a vertex (if any) and then, starting from this initial vertex, one finds an optimal vertex. In both stages the *simplex algorithm* is used: it solves an LP problem when an initial vertex solution is given. The method has its origins in some papers by the physisist Joseph Fourier (ca. 1820) on problems in mechanics, but it was fully developed into an efficient algorithm by George Dantzig around 1940. By the way, the LP duality theory was a joint effort of Dantzig and the famous mathematicial John von Neumann (known for his work in functional analysis, game theory and also in the initial developments of computers).

3.2 The simplex algorithm

We shall describe the (primal) simplex algorithm for LP problems of the following form

$$\max\{c^T x \mid Ax = b,\ x \geq 0\} \tag{3.1}$$

where $A \in \mathbb{R}^{m,n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. We assume that the coefficient matrix A has full row rank, i.e., the row vectors are linearly independent. This assumption is valid (both theoretically and computationally). Anyway, the assumption implies that $m \leq n$, and we note that the feasible set $P = \{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\}$ is a pointed polyhedron. It is pointed because P is contained in the nonnegative orthant which is pointed. Thus the minimal faces of P are vertices, and from Proposition 2.13 we know that, if P is nonempty, there is an optimal solution which is a vertex, or, if the problem is unbounded, there is an extreme ray along which the objective function increases without bounds.

Some notational remarks are in order. The subvector of a vector $z \in \mathbb{R}^J$ (resp. matrix $M \in \mathbb{R}^{I \times J}$) corresponding to (column indices) $J' \subseteq J$ is denoted by $z_{J'}$ (resp. $M_{J'}$). We shall also write $M = \begin{bmatrix} M_B & M_N \end{bmatrix}$ and $z = \begin{bmatrix} z_B & z_N \end{bmatrix}$. (Note that this notation could be misinterpreted; it does *not* mean that $B = \{1, \ldots, m\}$ and $N = \{m+1, \ldots, n\}$).

We introduce some useful concepts that will turn out to be algebraic counterparts to vertices. Let $J = \{1, \ldots, n\}$ be the index set of the variables, and let $B = \{B_1, \ldots, B_m\}$ be a subset of J consisting of m variable indices and define $N = J \setminus B$. Thus A_B is a square $m \times m$ matrix. If A_B is nonsingular, it is called a **basis** and we then call B a **basis index set**. Note that, since the rows of A are linearly independent, its row rank, and also its column rank, is m. Therefore, there is at least one basis index set. Note that when A_B is a basis, then its columns constitute a vector basis in $\mathbb{R}^m$.

Associated with a basis index set we have the **primal basic solution** $x^B = (x_B, x_N)$ where $x_N = 0$ and $x_B = A_B^{-1}\, b$. The variables in x_B (i.e., x_j with $j \in B$) are called **basic variables** and the variables in x_N are the **nonbasic variables**. We see that a basic solution x^B satisfies all the *equations* in (3.1). However, x^B may not be nonnegative, but if it is, x^B is called a **basic feasible solution**, as we then have that $x^B \in P$.

As we have suggested above, basic feasible solutions and vertices are the same, as stated next.

Lemma 3.1 *The set of vertices of P coincides with the set of basic feasible solutions in (3.1)*

Proof. Let x^B be a basic feasible solution in (3.1). To prove that x^B is a vertex, it suffices, by Corollary 2.23, to find an objective function $d \in \mathbb{R}^n$ such that x^B is the *unique* optimal solution of the LP problem (Q) min $\{d^T x \mid x \in P\}$. Let $d = \sum_{j \in N} e_j$ where e_j denotes the jth coordinate vector. Since each point in P is nonnegative, we must have that $v(Q) \geq 0$. But $d^T x^B = 0$, so x^B is an optimal solution and $v(Q) = 0$. Furthermore, each optimal solution y in (Q) must satisfy $y_N = 0$ and from $Ay = b$ we then get $A_B y_B + A_N y_N = b$ and $A_B y_B = b$. But A_B is nonsingular, so $y_B = A_B^{-1} b = x_B$, and we have $y = x^B$, proving the desired uniqueness.

Conversely, let x_0 be a vertex of P. Then x_0 is determined by n linearly independent inequalities from the linear system $Ax = b,\ x \geq 0$. Since A has rank m, this subsystem must contain $n - m$ inequalities from $x \geq 0$, say $x_j = 0$ for some $N \subset J$ with $|N| = n - m$. Thus the 2×2 block matrix C corresponding to the mentioned subsystem

$$\begin{bmatrix} A_B & A_N \\ 0 & I \end{bmatrix}$$

is nonsingular and therefore $0 \neq \det(C) = \det(A_B)\det(I) = \det(A_B)$, so A_B is nonsingular, and it follows that $x_0 = x^B$, i.e., a basic feasible solution.

□

The **simplex algorithm** solves an LP problem of the form (3.1) when an initial basic feasible solution (vertex) is given. Iteratively one moves from one vertex to the next as long as the objective function increases until no further improvement can be made. More precisely, the algorithm generates a sequence of bases, each corresponding to a vertex of P. In this process it is crucial to be able to determine (efficiently) if a given basic feasible solution x^B is optimal. This is where the LP duality theory is useful. From Theorem 2.5 it follows that x^B is optimal if and only if $c^T x^B$ equals the objective value for some dual feasible solution y^B. In fact, the simplex algorithm produces such a candidate y^B and the desired optimality test corresponds to checking if y^B is feasible in the dual LP problem of (3.1). The details are given next.

The LP dual of the primal problem (P) is the LP problem (D) given by

$$\min\{y^T b \mid y^T A \geq c^T\}. \tag{3.2}$$

Let B be a basis index set, A_B the associated basis and x^B the (primal) basic solution as above. We also define an associated **dual basic solution** $y^B \in \mathbb{R}^m$ by $(y^B)^T = c_B^T A_B^{-1}$. We then have that $(y^B)^T A = (y^B)^T \begin{bmatrix} A_B & A_N \end{bmatrix} = \begin{bmatrix} c_B^T & (y^B)^T A_N \end{bmatrix}$. Therefore, a dual basic solution is a **dual basic feasible solution** if and only if $(y^B)^T A_N \geq c_N$, which means that y^B is feasible in (D). The two basic solutions x^B and y^B are called **complementary** because they satisfy the **complementary slackness condition**

$$((y^B)^T A - c^T)x^B = 0 \tag{3.3}$$

We now have the following result.

Proposition 3.2 *Assume that B is a basis index set such that the associated basic solutions x^B and y^B are primal feasible and dual feasible, respectively. Then x^B is optimal in (P) and y^B is optimal in (D).*

Proof. We have that $c^T x^B = c_B^T x_B + c_N^T x_N = c_B^T x_B = (y^B)^T A_B A_B^{-1} b = (y^B)^T b$. On the other hand, using LP duality (Theorem 2.5) and the fact that x^B is feasible in (P) and y^B is feasible in (D) we get $c^T x^B \leq v(P) = v(D) \leq (y^B)^T b$. As shown, the first and last expression here are equal, and the desired result follows.

$\square$

Remark. Note that this proof only relied on weak duality ($v(P) \leq v(D)$)! Acually, one can prove the LP duality theorem (strong duality) by proving the correctness of the simplex algorithm. For that approach the finiteness of the simplex algorithm is a main task.

The simplex algorithm generates a sequence of basis index sets $B^{(1)}, \ldots, B^{(N)}$ with the properties that:

(i) each primal basic solution $x^{B^{(j)}}$, for $j = 1, \ldots, N$, is feasible in (P),

(ii) the dual basic solutions $y^{B^{(j)}}$ for $j = 1, \ldots, N-1$ are all infeasible in (D), while $y^{B^{(N)}}$ is feasible in (D),

(iii) consecutive basis index sets satisfy $|B^{(j)} \setminus B^{(j+1)}| = 1$ and $|B^{(j+1)} \setminus B^{(j)}| = 1$.

Thus, in the last iteration $t = N$ we get, for the first time, a dual basic solution, which is feasible. Therefore, by Proposition 3.2 $x^{B^{(N)}}$ is optimal in (P) and $y^{B^{(N)}}$ is optimal in (D).

We remark that a there is a related algorithm, called the **dual simplex algorithm**. It is in a sense complementary to the primal algorithm, and generates a sequence of basis index sets such that the associated dual solutions (i.e., y^B) are feasible in (D), but the associated primal solutions are infeasible in (P), except for at termination.

One of the reasons for the success of the simplex algorithm is that the transition from the pair of primal and dual basic solutions (x^B, y^B) to the next pair can be done with little computational effort. We describe how this can be done.

For a basis index set B, we assume that $B = \{B_1, \ldots, B_m\}$ is such that $B_1 < \ldots < B_m$. In the associated primal basic solution $x = x^B = (x_B, x_N)$ we call the variable x_{B_i} the i'th basic variable. Assume now that x^B is feasible in (P), i.e. $A_B^{-1} b \geq 0$. Geometrically, x^B is a vertex of the polyhedron P and it satisfies the n linearly independent equations given by $x_{B_i} = (A_B^{-1} b)_i$, $i = 1, \ldots, m$, and $x_j = 0$ for $j \in N$. We can reformulate the original LP problem (3.1) into an equivalent problem $P(B)$ relative to the current basis B

$$c_B \bar{b} + \max\{\bar{c}_N^T x_N \mid x_B + \bar{A}_N x_N = \bar{b};\ x_B \geq 0,\ x_N \geq 0\}. \tag{3.4}$$

where we define $\bar{A}_N = A_B^{-1} A_N$, $\bar{b} = A_B^{-1} b$ and $\bar{c}_N^T = c_N^T - c_B^T A_B^{-1} A_N$. In fact, the feasible region, as well as the objective functions in the two problems (P) and (P(B)) coincide; we have just introduced the partitioning and premultiplied the equation system with the inverse of the basis, i.e. A_B^{-1}. This problem has a convenient form for analyzing the effect of variation in the nonbasic variables x_N. First, we notice, as expected, that when we let $x_N = 0$, then we obtain the solution $x^B = (\bar{b}, 0)$ in (P(B)). Is this solution optimal? To answer this question, consider the objective function $\bar{c}_N^T = c_N^T - c_B^T A_B^{-1} A_N$ called the **reduced cost vector**. If $\bar{c} \leq 0$, then the associated dual basic solution y^B is also feasible (check this!), and then the two basic solutions are optimal in their respective problems (see Proposition 3.2). If, however, for some j, say $j = r$, we have that $\bar{c}_r > 0$, then an increase of the nonbasic variable x_r leads to a solution which is at least as good as the current one. Starting in $(\bar{b}, 0)$, we now want to increase the variable x_r as much as possible, while maintaining all the other nonbasic variables at 0. Let $x_r = \lambda_r \geq 0$, and we then obtain the solution $x(\lambda_r)$ given by

$$\begin{array}{l} x(\lambda_r)_B = \bar{b} - \bar{a}_r\lambda_r, \\ x(\lambda_r)_r = \lambda_r, \\ x(\lambda_r)_j = 0 \text{ for } j \in N \setminus \{r\}. \end{array} \tag{3.5}$$

How much can we increase λ_r? Since all the equations hold for any λ_r (by construction), it is only the nonnegativity of the basic variables that may cause trouble. The i'th basic variable, $x(\lambda_r)_{B_i}$ is given by $x(\lambda_r)_{B_i} = \bar{b}_i - \bar{a}_{i,r}\lambda_r$. Note that, because A_B is a primal feasible basis, we have $\bar{b} \geq 0$. We therefore see that, for each i

(i) if $\bar{a}_{i,r} \leq 0$, then $x(\lambda_r)_{B_i} \geq 0$ for all $\lambda_r \geq 0$, and

(ii) if $\bar{a}_{i,r} > 0$, then $x(\lambda_r)_{B_i} \geq 0$ if and only if $0 \leq \lambda_r \leq \bar{b}_i/\bar{a}_{i,r}$.

We therefore see that the maximum permittable λ_r for which $x(\lambda_r)$ is primal feasible is λ_r^* given by

$$\lambda_r^* = \min\,\{\bar{b}_i/\bar{a}_{i,r} \mid i \leq m,\ \bar{a}_{i,r} > 0\}. \tag{3.6}$$

and we define $\lambda_r^* = \infty$ if $\bar{a}_r \leq 0$. If λ_r^* is finite, we let

$$B^*(r) = \{B_i \mid i \leq m,\ \bar{a}_{i,r} > 0,\ \bar{b}_i/\bar{a}_{i,r} = \lambda_r^*\} \tag{3.7}$$

which is the index set of basic variables that will become zero in the solution $x(\lambda_r^*)$ (why!).

We say that the primal basic solution x^B is **degenerate** if $x_{B_i} = 0$ for some $i \leq m$. Otherwise, the solution is **nondegenerate**, i.e., all the basic variables are strictly positive. We observe that the new solution $x(\lambda_r^*)$ is primal feasible, and, furthermore, that $x(\lambda_r^*) \neq x(0) = x^B$ if and only if $\lambda_r^* > 0$. In particular, we see that if x^B is nondegenerate, then $\lambda_r^* > 0$. If x^B is degenerate, then we may, or may not, have $\lambda_r^* = 0$. Whenever λ_r^* is finite, the new solution $x(\lambda_r^*)$ is also a *basic* solution; this follows from the next proposition.

Proposition 3.3 *Consider a basis index B and an $r \in J \setminus B$. Define λ_r^* and $x(\lambda_r^*)$ as in (3.5) and (3.6). Let $F = \{x(\lambda_r) \mid 0 \leq \lambda_r \leq \lambda_r^*\}$. Then the following statements all hold.*

(i) If $\lambda_r^* = 0$, the vertex $x_0 = x^B$ is degenerate and each of the sets $(B \setminus \{s\}) \cup \{r\}$ for $s \in B^*(r)$ is a basis index set with associated basic feasible solution x_0.

(ii) If $0 < \lambda_r^* < \infty$, each of the sets $(B \setminus \{s\}) \cup \{r\}$ for $s \in B^*(r)$ is a basis index set with associated basic feasible solution $x(\lambda_r^*)$. Furthermore,

$F = \{x(\lambda_r) \mid 0 \leq \lambda \leq \lambda_r^*\} = \text{conv}(\{x^B, x(\lambda_r^*)\})$ *is an edge (one-dimensional face) of P which joins the two vertices x^B and $x(\lambda_r^*)$.*

(iii) If $\lambda_r^* = \infty$, then the set $F = \{x(\lambda) \mid \lambda \geq 0\}$ is an extreme ray of P (i.e., an unbounded one-dimensional face).

Proof. This result is essentially obtained by using the following basic result from linear algebra: an $m \times m$ matrix A_B is nonsingular if and only if for some $b \in \mathbb{R}^m$ the linear system $A_B x = b$ has a *unique* solution.

Consider first the situation with $\lambda_r^* < \infty$, and let $s \in B^*(r)$. Let $B' = (B \setminus \{s\}) \cup \{r\}$ and $N' = N \setminus B'$. Then , by definition of λ_r^*, the linear system $A_{B'} x = b$ has a unique solution, namely $x(\lambda_r^*)$, and from the mentioned linear algebra result it follows that $A_{B'}$ is nonsingular, and B' is a basis index set. All the results now follow since feasible basic solutions are vertices of P, and the line segment between two vertices is an edge of P. The statement in (iii) follows in a similar manner.

□

The previous discussion therefore describes how the simplex algorithm goes from one (primal) basic feasible solution to the next, and adjacent, basic feasible solution. The detailed algorithm is as follows.

The simplex algorithm.

Step 0. (Initialization) Let B be an initial basis index set such that the associated primal basic solution x^B is feasible. Calculate $x_B = A_B^{-1} b$ and $(y^B)^T = c_B^T A_B^{-1}$.
Step 1. (Optimality check) Calculate the reduced cost $\bar{c}_N^T = c_N^T - (y^B)^T A_N$. If $\bar{c}_N^T \leq 0$, then terminate; x^B is optimal in (P) and y^B is optimal in (D). Otherwise, choose an $r \notin B$ with $\bar{c}_r > 0$ and go to Step 2.
Step 2. (Pivoting) Determine λ_r^* and $B^*(r)$ from (3.6) and (3.7). If $B^*(r)$ is empty, then (P) is unbounded, $x(\lambda) \in P$ for all $\lambda \geq 0$ and $c^T x(\lambda) \to \infty$ as $\lambda \to \infty$. Otherwise, choose an $s \in B^*(r)$ and update the basis index set by $B := (B \setminus \{s\}) \cup \{r\}$. Determine the new primal and dual basic solutions x^B and y^B, and return to Step 1.

There are several questions that are natural to ask at this point. First, does the algorithm work, i.e., does it terminate? Note that it is not obvious that the algorithm works because we may have degenerate pivots, i.e., two consecutive bases may correspond to the same vertex. Therefore, in principle, it might be that the algorithm gets stuck in some vertex and never reaches an optimal one. However, the algorithm *does* work, provided that one uses specific principles for selecting new basic variables and also (in case of ties) outgoing basic variables. We show this in the next section.

Another important question concerns the problem of finding an initial vertex. Remember that the simplex algorithm takes a feasible basic solution as input, so we need a device for producing this initial solution, or prove that the problem is infeasible. It is remarkable that we can use the simplex algorithm for this task as well, see Section 3.4.

A final question is how we can obtain efficient implementations of the simplex algorithm (or simplex method). The straightforward implementation of the simplex algorithm as described above, would require that we (in addition to some matrix products) solve two $m \times m$ linear equation systems with the basis A_B, or its transpose, as the coefficient matrix. In fact, to find x_B and y^B we solve $A_B x_B = b$ and $(y^B)^T A_B = c_B^T$, respectively. The important observation is that these two systems are "nearly equal" from one iteration to the next as the basis index sets differ in one index only. In fact, one can find an LU-factorization of the basis in terms of so called *eta-matrices* (representing the pivot operation on the coefficient matrix). This factorization may then be updated in each iteration by adding suitable eta- matrices (and possibly permutation matrices). Two such schemes, the Bartels-Golub method and the Forrest-Tomlin method, are used in efficient simplex implemetations today, for further informations on this topic, see e.g. [4].

3.3 The correctness of the simplex algorithm

The purpose of this section is to prove that the simplex algorithm works, i.e., that it correctly solves the linear programming problem in a finite number of iterations whenever a primal basic feasible solution is given.

As pointed out in the previous section, there might be a problem that one gets stuck in a (non-optimal) vertex x_0 in the sense that , from a certain iteration on, all the primal basic feasible solutions are equal to x_0. We have seen that this can only happen if the basic feasible solution x^B is degenerate, i.e., at least one basic variable is zero. Since $\lambda_r^* > 0$ we then get a new solution. Let us first see that the simplex algorithm works if all the vertices are nondegenerate.

Theorem 3.4 *Consider the LP problem max* $\{c^T x \mid Ax = b,\ x \geq 0\}$ *where A is of full row rank, and where a basic feasible solution x_0 is known. Assume that all bases are nondegenerate. Then the simplex algorithm, with x_0 as the initial solution, will terminate in a finite number of steps.*

Proof. From the discussion of the simplex algorithm, and the nondegeneracy assumption, we see that the step length λ_r^* is strictly positive in each pivot operation. Therefore, the objective function must strictly increase in each iteration (as the reduced cost of the incoming variable x_r is positive). But then we cannot generate a cycle of bases, i.e., eventually end up with a basis that was considered before (as the objective function has increased in the meantime). Since the number of bases is finite (at most $n!/(m!(n-m)!)$), the simplex algorithm must terminate in a finite number of iterations.

$\square$

From this proof we see that the crucial step in the finiteness argument was to assure that we do not get "cycling", that is, a cycle of bases. Therefore, in the general situation with possibly degenerate bases, if we can find a rule preventing cycling, the simplex algorithm will work. It turns out that several such rules can be found. We shall study a very simple one, called **Bland's rule**.

We first observe that the simplex algortihm, as descibed in Section 3.2, leaves several choices open for different specifications. First, we have not said how to choose the new basic variable (entering variable) x_r. Secondly, there may be alternative choices for the leaving basic variable; this happens whenever the index set $B^*(r)$ has more than one element. Whenever principles for making these two decisions have been specified, we have a completely specified simplex algortihm. Blands's rule consists of the following simple principles: (i) as an entering variable we choose the nonbasic variable with lowest index having positive reduced cost, and (ii) as a leaving variable we choose the lowest index in $B^*(r)$. It is a remarkable fact that this simple rule handles the problem of cycling. By a **degenerate pivot** we mean a pivot, or simplex iteration, in which $\lambda = 0$, so we remain in the same vertex of the feasible polyhedron. The presentation is largely based on [25].

Theorem 3.5 *The simplex algorithm with Bland's rule terminates.*

Proof. A main goal in the proof is to show this result.

Claim: Consider a sequence of degenerate simplex iterations $t, t+1, \ldots, t'$. Let, in iteration t, the nonbasis index set be N^t and assume that the nonbasic variable x_q enters the basis. Let $N^t_> = \{j \in N^t \mid j > q\}$. If x_q leaves the basis in iteration t', then some $r \in N^t_>$ must have entered the basis in one of the intermediate iterations $t+1, \ldots, t'-1$.

Before we prove the claim, let us explain why it implies the termination of the simplex algorithm equipped with Bland's rule. It is clear that if the algorithm does *not* terminate, it must cycle, i.e., return to a previously generated basis and repeat this cycle infinitely. (Otherwise we would go through all bases and find an optimal one or an unbounded solution; this is due to the finite number of bases). Consider such a cycle of iterations. Then this cycle must consist of only degenerate pivots, otherwise we could not return to the same basis (for the objective function would have changed). So consider such a cycle of degenerate iterations and let q be *the largest* index among the nonbasic variables that enter in one of these degenerate iterations. But the claim then gives the desired contradiction due to the maximality of q. Therefore the algorithm can not cycle, and it terminates as there is only a finite number of bases.

Thus, it only remains to prove the claim. To do this, assume (as above) that the iterations $t, t+1, \ldots, t'$ are all degenerate, and that x_q enters the basis in iteration t and leaves in iteration t'. Let N^t and $N^t_>$ be as described in the claim and also let $N^t_< = \{j \in N^t \mid j < q\}$. Assume that there is no x_r with $r \in N^t_>$ that enters the basis in any of the iterations $t+1, \ldots, t'$. We shall deduce a contradiction from this.

Let x_p be the nonbasic variable that enters the basis in iteration t' (when x_q leaves). Let B be the basis index set in iteration t and B' the basis index set in iteration t'. Consider now the LP problem in iteration t, relative to the current basis B:

$$\max\{\bar{c}_N^T x_N \mid x_B + \bar{A}_N x_N = \bar{b};\ x_B \geq 0,\ x_N \geq 0\}. \tag{3.8}$$

Thus, $\bar{c}_q > 0$ as x_q enters the basis, and also $\bar{c}_j \leq 0$ for $j \in N^t_<$; this is due to the "entering principle" of Bland's rule. We may now view this problem as an "original LP problem" with coefficient matrix $[I, \bar{A}]$ and objective function $[0, \bar{c}_N^T]$. Since x_p enters in the t''th iteration, this variable must have positive reduced cost, i.e.,

$$\bar{c}_p - \bar{c}_{B'}^T B'^{-1} \bar{a}_p > 0. \tag{3.9}$$

Now, $\bar{c}_p \leq 0$. (For if $\bar{c}_p > 0$, then x_p was nonbasic in iteration t, so $p \in N^t$. But $p \notin N^t_>$ for, by assumption, no variable with index in $N^t_>$ enters in iterasjon t' as x_p does. Clearly, $p \neq q$, so we must have $p \in N^t_<$ and thus $\bar{c}_p \leq 0$, see above). Therefore, from (3.9), we get $\bar{c}_{B'}^T B'^{-1} \bar{a}_p < 0$, or $\bar{c}_{B'}^T y_p < 0$ where we define $y_p = B'^{-1} \bar{a}_p$. Let the pivot element be y_{qp} (recall: x_p enters, and x_q leaves the basis in this iteration). Then $y_{qp} > 0$ (otherwise x_q would not leave the basis) and from above we have $\bar{c}_q > 0$. Thus, in the negative inner product $\bar{c}_{B'}^T y_p$ the contribution from the q'th component is $\bar{c}_q y_{qp} > 0$, and we conclude that there must exist some other basic variable r (in B') with $\bar{c}_r y_{rp} < 0$. Note that $\bar{c}_r \leq 0$ for all basic variables except x_q in B' as all these variables correspond either to indices in $N^t_<$ or basic variables in iteration t (and then $\bar{c}_r = 0$). Therefore, we must have an $r \in N^t_<$ with $\bar{c}_r < 0$ and $y_{rp} > 0$. We have that $x_r = 0$ since the primal solution x have been unchanged during the intermediate iterations (they were degenerate) and it was nonbasic in iteration t (and therefore zero). However, this shows that the r'th basic variable was a candidate for leaving the basis in the ratio test, and since $r < q$, we have arrived at a contradiction.

This proves the claim above and also the theorem.

□

3.4 Finding an initial vertex

In this section we describe how to find an initial vertex of the polyhedron $P = \{x \in \mathbb{R}^n \mid Ax = b,\ x \geq 0\}$ being the feasible set of the LP problem (P) given in (3.1).

In (P) we may assume that $b \geq 0$; this is accomplished by multiplying suitable rows by -1. Consider the following LP problem (PI), called the **phase I problem**:

$$\min\{\mathbf{1}^T s \mid Ax + s = b,\ x \geq 0,\ s \geq 0\} \tag{3.10}$$

where $\mathbf{1} = (1, \ldots, 1)$. Compared to the original problem (P) the phase I LP problem contains artificial variables, one for each row, and the objective is to minimize the sum of these artificial variables. A cruical property is that $v(PI) = 0$ if and only if there is a feasible solution (x, s) of (PI) with $s = 0$. In this case x is a feasible solution of (P). Thus by solving (PI) one determines if the problem (P) is feasible, and, if it is, one gets a feasible solution. Furthermore, it is easy to find an initial basic feasible solution to (PI) (this is the whole point!); just let $x = 0$ and $s = b$. Thus all variables in s are basic, and the corresponding basis is the identity matrix.

Next we explain how to find, in the case of (P) feasible, a *basic* feasible solution through solving (PI). Let (x, s) be the optimal basic feasible solution of (PI), and assume that (P) is feasible, so $v(PI) = 0$. Thus we have that $s = 0$. If now all the variables in s are nonbasic, the current basis matrix is in fact a basis matrix in A (i.e. a $m \times m$ nonsingular submatrix). In this case we found an initial basic feasible solution of (P) using this basis.

It remains to discuss the case when at least one artificial variable is basic in the optimal basic feasible solution of (PI). For notational simplicity, say that $s_1, \ldots, s_k$ are basic variables in s and that $x_1, \ldots, x_t$ are nonbasic variables in x. The idea is to use a pivot operation to drive one of the variables $s_1, \ldots, s_k$ out of the basis such that it is replaced by one of the variables $x_1, \ldots, x_t$. Note that in this basis change the objective function is of no interest, but making pivot operations is still possible. In fact, it corresponds precisely to a pivot operation as in Gaussian elimination on a matrix. In this way one can gradually switch variables until either (i) all artificial variables are nonbasic, or (ii) no more switching may be done. In case (i) we have reached our goal of finding a primal feasible basis, and we may proceed to phase II of the simplex method. In case (ii) the switching got stuck because in the submatrix in which we wanted to choose a pivot element all elements were zero so no such pivots could be done. One can see that this reveals that certain rows in $Ax = b$ are linearly dependent on the others. One may then delete these redundant rows, and the preceeding pivoting has then also produced a basis in this reduced system. Thus, also in this case a primal feasible basis was found, and one may proceed to phase II of the simplex method.

Chapter 4

Graph optimization

Most problems studied in combinatorial optimization involve looking for certain structures in graphs. Furthermore, many applied problems in e.g. computer science, VLSI, telecommunications, scheduling etc. are fruitfully modeled by graphs, possibly through some kind of optimization problem in that graph. In this chapter we give a brief introduction to some basic concepts and results in graph theory. Then we study some classical optimization problems in graphs, e.g., the shortest path problem, with focus on the mathematical ideas underlying efficient algorithms for solving these problems.

For further treatment of graph optimization and combinatorial optimization, see [33], [25], [2], [14], [21], [23] and [29]. Some central books in graph theory are [1],[3] and [14]. It may also be useful to study parts of the book [15] (discrete mathematics) along with this chapter.

4.1 Graphs and digraphs

We give a very short introduction to graph theory. Books that can be recommended for further reading include [3], and [1].

A **graph** is an ordered pair $G = (V, E)$, where V is a finite set with elements called **nodes** or **points** and E is a finite set of unordered pairs from V; each such pair is called an **edge** or a **line**. Any graph may be visualized in $\mathbb{R}^2$ by drawing the nodes as distinct points and each edge as a curve joining the two nodes. Such a drawing of the graph is called an **embedding** of the graph. The number n of nodes is called the **order** of the graph, and the number m of edges is called the **size** of the graph. In Figure 4.1 a graph of order 8 is indicated.

Graphs are typically used to model finite sets where there is some kind of association between pairs of elements. For instance, in a computer network

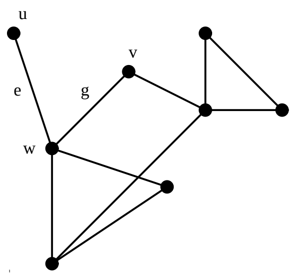

Figure 4.1: A graph

the nodes may correspond to computers and edges to direct communication lines as cables. In fact, graphs are well suited for representing communications systems in other areas as well (railway, roads, electricity). As another example, consider a matrix $A \in \mathbb{R}^{m,n}$ and associate a node $i \in I$ with the i'th row and a node $j \in J$ with the j'th column. Let the $G = (V, E)$ have node set $V = I \cup J$ and let E contain the edge iff $a_{ij} \neq 0$. This graph represents the sparsity structure of the graph and may be used in efficient specialized (e.g.) Gaussian elimination algorithms for solving linear equations $Ax = b$ with some sort of band structure.

For an edge $e = \{u, v\} \in E$ we normally write $e = [u, v]$ or $e = uv$. Note that since this is an unordered pair we have $uv = vu$. When $e = uv$ we say that e is **between**, or **joins**, u and v and that u and v are **incident** to e. Two nodes are **adjacent**, or **neighbors**, if there is some edge between them, and these nodes are called the **endnodes** of that edge. Two edges are **neighbors** if they share a node. A node v and an edge e are **incident** if v is an endnode of e. Sometimes a graph is allowed to contain **loops** of the form uu. In the graph of Figure 4.1 we see that e.g. nodes u and w are adjacent, but u and v are not. The edge e is between u and w, and the edges e and g are neighbors. Two edges are **parallel** if they have the same endnodes. Unless otherwise stated we consider only loopless graphs without parallels. Graphs with parallels are often called **multigraphs**. Note that graph terminology varies slightly from one text to another, so definitions should be studied carefully!

We have already seen that a graph may be used to represent a matrix in a certain sense. Conversely, there are certain matrices associated with a graph that are useful for discussing graph properties and graph optimization problems. Consider a graph $G = (V, E)$ and define the **node-edge incidence matrix** $A \in \mathbb{R}^{n,m}$ with elements being 0 or 1 as follows. A has one row for each node i and and a column for each edge e, and $a_{i,e} = 1$ if i is incident to e (i.e., i is an endnode of e) and $a_{i,e} = 0$ otherwise. All the ones in row i of A are in columns that correspond to the set $\delta(i)$ of edges being incident to node

i; this set is often called the **star** of node i. The row sum $d(i) = d_i = |\delta(i)|$ is the number of edges incident to i (= the number of neighbor nodes to i); we call this number the **degree** of node i. More generally, for $S \subset V$, we let $\delta(S)$ denote the set of edges with one endnode in S and the other endnode in $\bar{S} = V \setminus S$; such a set is called a **cut**.

An **isolated** node is a node of degree 0, i.e., without neighbors.

A **walk** (in a graph) is a node-edge sequence $W : v_0, e_0, v_1, e_1, \ldots, v_{n-1}, e_{n-1}, v_n, e_n, v_{n+1}$ where $e_i = [v_i, v_{i+1}] \in E$ for $i = 0, \ldots, n$. The **endnodes** of W are v_0 and v_n, and we also call W a v_0v_n-walk or a walk between v_0 and v_n. Note that in a walk we may have repeated nodes or edges. If, however, there are no repeated nodes, we call the walk a **path** between v_0 and v_n, or a v_0v_n-path. Usually, we use the symbol P for paths. All the nodes in a path, except the two endnodes, are called **internal nodes**. A walk for which the endnodes coincide, is called a **circuit**. A circuit for which all nodes are of degree two, is called a **cycle**. A graph without cycles is called **acyclic** or a **forest**. The **length** of a walk (path etc) is its number of edges. A walk (path etc) is called **odd** (resp. **even**) if it has odd (resp. even) length. A walk W, a path P and a cycle C is shown in Figure 4.2.

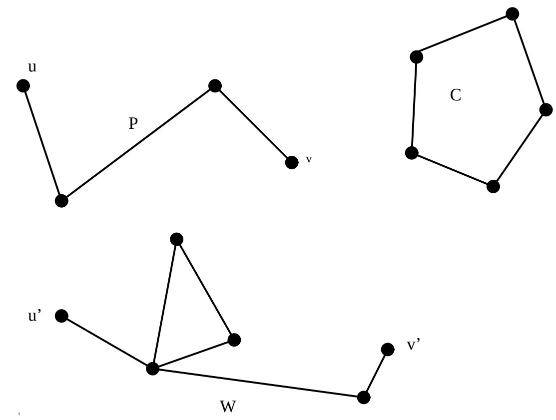

Figure 4.2: Path, walk and cycle

There is another matrix which may be used to represent the graph. Let the graph $G = (V, E)$ have node set $\{v_1, \ldots, v_n\}$. The **adjacency matrix** $A \in \mathbb{R}^{n,n}$ is a 0/1-matrix where the (i, j)'th element is 1 if $[v_i, v_j] \in E$ and 0 otherwise. Note that A is symmetric and that it has zeros on the diagonal. The powers of this matrix contains information about walks in the graph.

Proposition 4.1 *Let A be the adjacency matrix of a graph G and let $n \geq 1$. Then the (i, j)'th entry of A^n, the n'th power of A^n, equals the number of different v_iv_j-walks of length n in G.*

Proof. Let N^n be the $n \times n$ matrix where $N^n_{i,j}$ denotes the total number of v_iv_j-walks of length n in G. Clearly, we have $N^1 = A$. Every v_iv_j-walk of length n arises from a v_iv_k-walk of length $n-1$ by adding the edge v_kv_j. Therefore we must have $N^n_{i,j} = \sum_{v_k \in \delta(v_j)} N^{n-1}_{i,k} = \sum_{v \in V} N^{n-1}_{i,k} a_{k,j}$. But this proves that if $N^{n-1} = A^{n-1}$, we also have $N^n = A^n$, and the desired result follows by induction.

□

We remark that the previous result is similar to a property of the transition matrix of finite, stationary Markov chains. Let P be the transition matrix for such a Markov chain on a finite set of states, i.e., $P_{i,j}$ is the probability that the process goes from state i to state j in one time step. Then $P^n_{i,j}$ is the probability that the process moves from state i to state j in n steps. The standard proof of this fact is also similar to the proof given above.

Two graphs may have the same structure although node and edge "names" are different. For most purposes we do not want to distinguish between such graphs. Two graphs $G = (V, E)$ and $G' = (V', E')$ are **isomorphic** if there is a injective (one-to-one) function $\phi : V \to V'$ such that $[u, v] \in E$ iff $[\phi(u), \phi(v)] \in E'$.

One may produce new graphs by applying different operations on graphs; we here mention a couple of such operations. Consider a graph $G = (V, E)$ and let $U \subseteq V$ and $F \subseteq E$. The **subgraph** induced by U, denoted $G[U]$, is the graph $(U, E[U])$, where $E[U]$ consists of those edges in E with both endnodes in U. We then call $(U, E[U])$ a node-induced subgraph. If $F \subseteq E$ the graph $G[F] = (V, F)$ is called an edge-induced subgraph. The **complement** of a graph $G = (V, E)$ is the graph $\bar{G} = (V, \bar{E})$, where $\bar{E} = \{uv : uv \notin E\}$.

Some special graphs are often discussed. The **complete graph** on n nodes, denoted K_n, is the graph with n nodes where each pair of nodes are adjacent. Thus K_n has $n(n-1)/2$ edges. A **bipartite**) graph is a graph where the nodes may be divided into two sets V_1 and V_2 such that each edge has one endnode in $V_1$and the other endnode in V_2. More generally, the k**-partite graph** $K[n_1, \ldots, n_k]$ is the graph with node set consisting of (disjoint) sets, or color classes, V_i with $|V_i| = n_i$ for $i = 1, \ldots, k$ and where the endnodes of each edge belong to different color classes. Thus, a 2-partite graph is the same as a bipartite graph, see Fig.4.3. A k-partite graph is **complete** if each pair of nodes lying in different color classes are adjacent.

We now consider graphs where the edges have a certain direction. A **directed graph**, or **digraph**, is an ordered pair $D = (V, A)$ where V is a finite set of **nodes** (or **points**) and A is a finite set of *ordered* pairs of nodes. Each such ordered pair $e = (u, v)$ is called an **arc** (or **line, (directed) edge**)

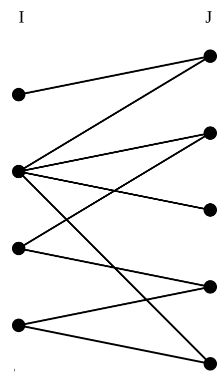

Figure 4.3: Bipartite graph

and u is called the **initial endnode**, or **tail**, of e and v is called the **terminal endnode**, or **head**, of e. One can define the concepts of **directed walk**, **path**, **circuit** and **cycle** as for graphs, but where each arc in the sequence is directed from the node that precedes it to the node that succeedes it. For instance, the following is a directed path from u to v with two arcs: $P : u, (u, w), w, (w, v), v$, see also Figure 4.4. For each node v we call the set of arcs with tail v the **outgoing star** of node v, and it is denoted by $\delta^+(v)$. Similarly, the set of arcs with head v is called the **ingoing star** of v and it is denoted by $\delta^-(v)$. More generally, if $S \subset V$, we let $\delta^+(S)$ denote the set of arcs with tail in S and head in $\bar{S} = V \setminus S$, and $\delta^-(S)$ is the set of arcs with tail in $\bar{S}$ and head in S. We let $d^+(v) = |\delta^+(v)|$ and $d^-(v) = |\delta^-(v)|$.

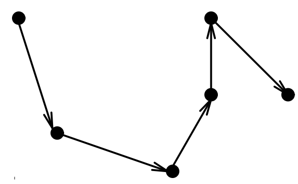

Figure 4.4: A directed path

We can represent a digraph by its **node-arc incidence matrix** $A \in \mathbb{R}^{n,m}$ where $a_{i,e} = -1$ (1) if i is the head (tail) of are e and $a_{i,j} = 0$ otherwise. Thus each column has exactly two nonzeros, one of these is 1 and the other -1.

Now, after so many (boring?) definitions, the time has come to give a result. It is quite natural present what is often called the **first theorem of graph theory**.

Theorem 4.2 *For any graph the degrees and the number of edges are related* *as follows*

$$\sum_{v \in V} d_v = 2m. \tag{4.1}$$

Proof. Recall that the degree d_v is equal to the number of edges in the star $\delta(v)$ of v. Thus, when we sum all these numbers we will count each edge twice (once from each end) and therefore the sum must be $2m$.

$\square$

Although this was a easy fact to prove, it is very useful. Let us give one interesting consequence.

Corollary 4.3 *In any graph there is an even number of nodes having odd degree.*

Proof. If the number of odd degree nodes were odd we would get an odd number on the left-hand-side of (4.1), a contradiction since $2m$ is even!

$\square$

This *parity argument* turns out to be very useful later as well, when we study combinatorial problems via linear programming. We next give a useful characterization of bipartite graphs in terms of the parity of its cycles.

Proposition 4.4 *A graph is bipartite if and only if it contains no odd cycles.*

Proof. By definition, a graph is bipartite iff we can color each node either red or blue (say) such that no edge has endnodes with the same color. Assume first that the graph G contains an odd cycle with nodes $v_1, \ldots, v_{2k+1}$, and we may assume that v_1 is colored red. Then v_2 must be blue, v_3 is red etc. In fact, all odd numbered nodes must be red and even numbered nodes blue. But then both v_1 and v_{2n+1} are red, and they are joined by an edge, which proves that no bicoloring is feasible, i.e., the graph is not bipartite.

Conversely, assume that the graph contains no odd cycles. Choose a start node v_1 and traverse the graph using breadth-first-search from v_1. Color the start node red, and in each iteration where, say, node v_i is processed give each of the nodes in $\delta(v_i)$ a different color than that of v_i. This process is well-defined (i.e., we never assign a different color to an already colored node) since all cycles are even, Thus the graph is bipartite.

$\square$

A basic graph property is connectivity. We say that two nodes u and v in a graph G are **connected** if G contains an uv-path. This gives rise to an **equivalence relation** on V. Let us write uCv if u and v are connected, and then this binary relation is an equivalence relation so

(i) uCu,

(ii) $uCv \Rightarrow vCu$, and

(iii) $(uCv \text{ and } vCw) \Rightarrow uCw$.

This equivalence relation gives rise to a partition of V into subsets $V_1, \ldots, V_p$ being the maximal connected subsets of V. These sets are called the **(connected) components** of G. Thus, two nodes are connected if and only if they are in the same component. Let $c(G)$ denote the number of components of G. A graph is called **connected** if $c(G) = 1$, i.e., if each pair of nodes is connected. Note that each isolated node is also a component.

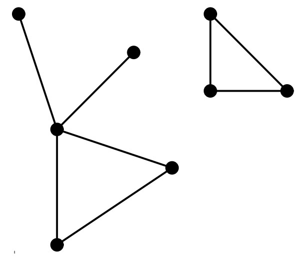

Figure 4.5: A graph with two connected components

What kind of graphs are minimally connected in the sense that if an edge is removed the connectedness is destroyed? These graphs constitute a very important class of graphs, called *trees*, which we study next. First, we make an important observation: a minimally connected graph can not contain a cycle, because if it did, an edge could be removed and the graph would still be connected. Recall that a forest is a graph without cycles. We define a **tree** as a connected forest, i.e., a connected graph without cycles. A **bridge** is an edge whose removal increases the number of components. Trees can be characterized in several ways. Remember that we have $n = |V|$ and $m = |E|$.

Proposition 4.5 *Let $T = (V, E)$ be a graph. Then the following statements are equivalent.*

(i) *T is a tree.*
(ii) *T is connected and each edge $e \in E$ is a bridge.*
(iii) *There is a unique path between each pair of nodes.*
(iv) *T is connected and $m = n - 1$.*
(v) *T is a forest and $m = n - 1$.*

The proof is left as an exercise.

We therefore see that the trees are precisely those graphs that are minimally connected. Another simple property of trees will be useful later. For a tree T we say that a node $v \in T$ is a **leaf** if v is incident to exactly one edge in T.

Corollary 4.6 *Any tree contains at least two leaves.*

Proof. Consider a tree $T = (V, E)$ with n nodes and m edges. Then $m = n - 1$, by Proposition 4.5. Any node v in a tree must have at least one incident edge so $d_v \geq 1$. From Theorem 4.2 we have that

$$\sum_{v \in T} d_v = 2m = 2(n-1). \tag{4.2}$$

Let no N be the number of nodes of degree 1. It follows from (4.2) that $2(n-1) = \sum_{v \in T} d_v \geq N + (n-N)2 = 2n - N$ and therefore $N \geq 2$ as desired.

$\square$

4.2 The shortest path problem

In this section we consider **weighted** graphs, that is, graphs with a nonnegative number associated with each edge. We shall study the problem of finding a shortest possible path between two given nodes in a weighted graph.

Let $G = (V, E)$ be a weighted graph with weight function w given by $w(e)$ (or w_e) for each $e \in E$. The **length**, or **weight**, $w(P)$ of a path P (relative to w) is defined by $w(P) = \sum_{e \in P} w_e$ where the sum is taken over all the edges in the path. (Note: a path may be considered as an edge-sequence, a node-sequence or a node/edge-sequence, depending on the context.) Based on w we obtain a concept of *distance* in the graph. Define $d_w(u, v)$ as the length of a shortest path taken over all uv-paths, i.e. $d_w(u, v) = \min\{w(P) : P \text{ is a } uv\text{-path}\}$. It is convenient here to define $d_w(u, v) = \infty$ if u and v are not connected. Similarly, we define $w(u, v) = \infty$ whenever $[u, v] \notin E$. If $w \geq 0$, d_w is a pseudometric, and when $w_e > 0$ for each e, then d_w becomes a metric and therefore (V, d) is a metric space.

The **shortest path problem** is the following optimization problem: given a weighted graph G with weight function w and two nodes u and v, find a uv-path of minimum length, i.e., a path P with $w(P) = d_w(u, v)$. We shall only consider nonnegative weights; the general case may also be solved efficiently, but by other algorithms. (When negative weights are allowed, there may be negative length cycles, so walks using such cycles may be shorter that paths; this fact requires special treatment.)

We next present the famous Dijkstra's algorithm for solving the shortest path problem. It turns out that the algorithm not only finds a shortest path from a node u_0 to a given node v, but it also finds a shortest path from u_0 to *all* other nodes, essentially without extra work.

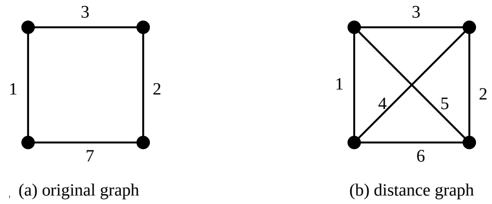

Figure 4.6: How weighted graphs give metric spaces

The underlying ideas may be explained as follows. Let u_0 be a fixed node and consider a node set S containing u_0. Let $\bar{S} = V \setminus S$ be the set complement of S. We define the **distance** from u_0 to $\bar{S}$ by $d(u_0, \bar{S}) = \min_{x \in \bar{S}} d(u_0, x)$. Important properties of the function d are given in the next lemma.

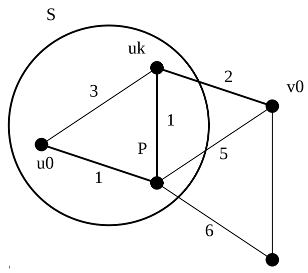

Figure 4.7: Illustration of the shortest path lemma

Lemma 4.7 *There is a node $v_0 \in \bar{S}$ and a u_0v_0-path $P : u_0, u_1, \ldots, u_k, v_0$ such that*

(i) $u_j \in S$ *for* $j \leq k$;

(ii) the path $u_0, \ldots, u_k$ *is a shortest* u_0u_k*-path in* G *and therefore also in* $G[S]$;

(iii) $w(P) = d(u_0, v_0) = d(u_0, \bar{S})$, *and finally*

$$d(u_0, v_0) = d(u_0, u_k) + w(u_k, v_0) = \min\{d(u_0, u) + w(u, v) : u \in S, v \in \bar{S}\}. \tag{4.3}$$

Proof. Confer with Figure 4.7 in this proof. Since V is finite, there is a node $v' \in \bar{S}$ such that $d(u_0, v') = d(u_0, \bar{S})$, and, furthermore, since the number of paths is finite, there must be a u_0v'-path P' with $w(P') = d(u_0, v') = d(u_0, \bar{S})$. Let $P' : u_0, u_1, \ldots, u_{k'}, v'$ and let v_0 be the first node in this sequence which is in $\bar{S}$. Then, by optimality of v', the edges between v_0 and v' must all have zero weight, and by defining P to be the subpath of P' from u_0 to v_0

we obtain $w(P) = w(P') = d(u_0, \bar{S})$. Then P and v satisfy the properties (i) and (iii) of the lemma. Furthermore, (ii) must hold, because if there were a shorter u_0u_k-path Q, then this path could not contain v_0 (this would contradict that v_0 is nearest node in $\bar{S}$ to u_0) and then the path obtained by augmenting Q by the edge u_kv_0 would be shorter than P, a contradiction. Thus, properties (i)–(iii) all hold and (4.3) also follows.

□

Based on this lemma we can solve our problem. Assume that we have found the distance from u_0 to all nodes in a set S. Initially we let $S = \{u_0\}$. Unless $S = V$, in which case we are done, Lemma 4.7 tells us that we can find a node $v_0 \in \bar{S}$ and determine its distance $d(u_0, v_0)$ from u_0. But then we have calculated the distance to one more node, and can proceed similarly with $S := S \cup \{v_0\}$. It remains to explain *how* to determine this new node v_0, and again Lemma 4.7 tells us the answer. We may choose $v_0 \in \bar{S}$ and also $u_k \in S$ as minimizers in (4.3), i.e., $d(u_0, u_k) + w(u_k, v_0) = \min\{d(u_0, u) + w(u, v) : u \in S, v \in \bar{S}\}$. In order to find this minimum we only need to consider nodes $u_k \in S$ and $v_0 \in \bar{S}$ for which $[u_k, v_0] \in E$ (otherwise $w(u_k, v_0) = \infty$), so we actually consider all edges in the **cut** $\delta(S) = \{[i, j] \in E : i \in S, j \in \bar{S}\}$. Finally, when we have found such nodes u_k and v_0, then a shortest u_0v_0-path is obtained by augmenting a shortest u_0u_k-path by the edge $[u_k, v_0]$ and the node v_0 (the first path was available from the previous step).

We therefore obtain the following algorithm in which each node v gets a label $(L(v), u)$ where $L(v)$ is the distance from u_0 to v and u is the predecessor of v on a shortest u_0v-path.

Dijkstra's shortest path algorithm.

Step 0. $S := \{u_0\}$, $L(u_0) := 0$, $L(v) = \infty$ *for each* $v \in V \setminus \{u_0\}$. *If* $|V| = 1$, *stop.*

Step 1. For each $v \notin S$ *do the following: if the minimum of* $L(u)+w(u, v)$ *for* $u \in S$ *and* $[u, v] \in E$ *is smaller than* $L(v)$, *then update* $L(v)$ *to this minimum number and let node* v *get the new label* $(L(v), u)$.

Step 2. Determine $\min_{v \in \bar{S}} L(v)$ *and let* v *be a node for which this minimum is attained.*

Step 3. Update S *by* $S := S \cup \{v\}$. *If* $S = V$, *then stop; otherwise return to Step 2.*

The correctness of this algorithm is a consequence of Lemma 4.7, and a detailed proof of this is left as an exercise. The algorithm has complexity $O(n^2)$; see Problem 4.4. We remark that if one is only interested in finding shortest paths from u_0 to each node in some set T of nodes, then one may terminate the algorithm whenever S contains T. This is possible since at the end of each iteration, the algorithm has found shortest paths to all nodes in S.

4.3 The minimum spanning tree problem

The **minimum spanning tree problem** is another basic combinatorial optimization problem, which arises in several applications. Let $G = (V, E)$ be a graph with nonegative weight function $w : E \to \mathbb{R}_+$. We assume that G is connected. The problem is to find a spanning tree of minimum weight. Here a spanning tree is a tree in G which spans all the nodes, i.e., each node of V is contained in the tree, and the weight of the tree is defined as the sum of the weights of its edges. An application of this problem is in the design of communication networks (telecommunication, computer, railway etc.) where a set of locations are to be connected by communication lines as cheaply as possible.

As a preparation for the discussion of an algorithm that solves the minimum spanning tree problem, we study graphs obtained from trees by adding an edge.

Proposition 4.8 *(i) If $T = (V, E)$ is a tree and $ij \notin E$, then the graph $(V, E \cup \{ij\})$ contains exactly one cycle C'.*

(ii) If C' is the unique cycle from (i) and $uv \in C' \setminus \{ij\}$, then the graph $(V, E \cup \{uv\} \setminus \{ij\})$ is also a tree.

Proof. (i) Any cycle that arises from T by adding the edge ij must consist of an ij-path in T plus ij. But since T is a tree it contains exactly one ij-path (see Proposition 4.5), so consequently the augmented graph has exactly one cycle.

(ii) Since the graph $(V, E \cup \{uv\} \setminus \{ij\})$ contains exactly one path between each pair of nodes, it must be a tree (again by Proposition 4.5).

□

We shall describe an extremely simple, and efficient, algorithm for solving this problem due to Kruskal. The algorithm is a **greedy algorithm** meaning that one gradually makes the locally best choices in the construction of a spanning tree. More precisely, in each iteration we add to the current partial solution another edge with property that it has the lowest possible weight among all those remaining edges that do not create a cycle when added. The remarkable fact is that such a simple idea gives an optimal solution, as proved below.

Kruskals algorithm.

Step 0. Order the edges according to nondecreasing weights, say that $w(e_1) \leq \ldots \leq w(e_m)$. Set $F := \emptyset$ and $k = 1$.

Step 1. If the graph $(V, F \cup \{e_k\})$ is acyclic, set $F := F \cup \{e_k\}$.

Step 2. If $|F| = n - 1$, then stop; (V, F) is a minimum weight spanning tree. Otherwise, set $k := k + 1$ and return to Step 1.

Since Step 1 and 2 require $O(m)$ steps, the main work of this algorithm is sorting the array of edge weights. This may be done in $O(m^2)$ steps using, e.g., the bubblesort algorithm. Note that, in Step 1 of Kruskal's algorithm, we have to check if the new graph $(V, F \cup \{e_k\})$ is acyclic. This can be done efficiently by maintaining a label for each node saying which component it belongs to. Initially, we label the nodes 1 to n, and, later, an edge may be added to F depending on the labels of its two endnodes. If these nodes have the same label, the edge can not be added as a cycle would be formed. Otherwise, the edge joins two components. We add this edge and also let all the nodes in the two components get the same label.

The correctness of Kruskal's algorithm is shown next.

Theorem 4.9 *For any weighted connected graph Kruskal's algorithm terminates with a minimum weight spanning tree.*

Proof. Let $T = \{e_1, \ldots, e_{n-1}\}$ (with edges chosen in that order) be the spanning tree at the termination of the algorithm, and assume that T is *not* optimal. Let M be a minimum weight spanning tree for which *the number of edges in common with T is maximum.* Let e_k be the first edge in T which is not in M (such an edge must exist as the nonoptimality of T implies that $M \neq T$). From Proposition 4.8 the graph $M \cup \{e_k\}$ contains a unique cycle, and, furthermore, this cycle must contain an edge $e \notin T$ (otherwise T would not be a tree). It also follows from Proposition 4.8 that $M' = M \cup \{e_k\} \setminus \{e\}$ is a tree. Since M is a minimum weight spanning tree and $w(M') = w(M) + w(e_k) - w(e)$, we must have $w(e_k) \geq w(e)$. But in the k'th iteration of Kruskal's algorithm the edge e_k was chosen in stead of e (and both were feasible choices; neither lead to a cycle) so we get $w(e_k) \leq w(e)$, and therefore $w(e_k) = w(e)$. It follows that M' is also a minimum weight spanning tree and it intersects T in one more edge compared to M; this contradicts our choice of M and therefore T must be optimal.

□

4.4 Flows and cuts

A number of problems in graphs involve flows and cuts. We introduce these concepts along with some basic properties.

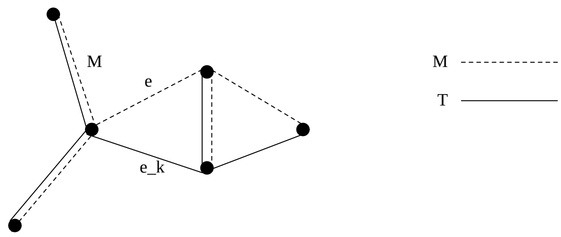

Figure 4.8: Proof of Kruskal's algorithm

We shall first formalize the notion of flow in a network. By a **network** we mean a four-tuple $\mathcal{N} = (V, E, d, b)$ where

(i) (V, E) is a directed graph (denoted D),

(ii) $d : E \to \mathbb{R}_+$ is a **capacity function** where $d_e = d(e) \geq 0$ denotes the capacity of arc e, and

(iii) $b : V \to \mathbb{R}$ is a **supply function** where $b_v = b(v)$ denotes the supply in node v.

We shall also assume that the newtork is connected when arc directions are ignored. Note that we allow negative values on b_v, in which case it may be viewed as a demand in that node.

We shall use the notation $x(F) := \sum_{e \in F} x_e$ whenever $F \subseteq E$ and $x = (x_e : e \in E)$ is a vector of variables, one for each arc e. A **flow** (arc-flow) in the network $\mathcal{N}$ is a vector $x \in \mathbb{R}^E$ that satisfies the following linear system

$$\begin{array}{lll} \text{(i)} & x(\delta^+(v)) - x(\delta^-(v)) = b_v & \text{for alle } v \in V \\ \text{(ii)} & 0 \leq x_e \leq d_e & \text{for all } e \in E. \end{array} \tag{4.4}$$

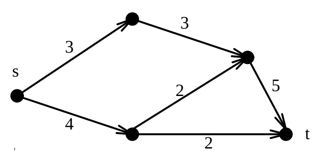

Figure 4.9: A flow of value 7 from s to t

In Figure 4.9 we indicate a flow for which $b_s = 7$, $b_t = -7$ and the other b_v's are zero. (The capacities are not drawn).

We call x_e, for $e = (i, j)$, the flow in arc $e = (i, j)$ and it should be interpreted as the magnitude of some "flow" entering the arc (i, j) at node i and leaving this arc at node j. The equation (4.4)(i) are **flow balance constraints** saying that the net flow out of node v, i.e., the difference between the total outflow $x(\delta^+(v))$ and the total inflow $x(\delta^-(v))$, must equal the supply b_v. Whenever $b_v = 0$, we call the corresponding constraint in (4.4)(i) **flow conservation**. The inequalities (4.4)(ii) are the **capacity constraints**, and note here that we only consider nonnegative flows. A special case is obtained whenever $b_v = 0$ for all $v \in V$, and then a flow is also called a **circulation**. Note that whenever $b_v = 0$, flow balance represents Kirchoffs's law in electricity; what goes in to a node, must also leave it.

We may represent the linear system (4.4) in matrix form by

$$Ax = b, \; 0 \leq x \leq d. \tag{4.5}$$

Here $A \in \mathbb{R}^{n,m}$ is the node-arc incidence matrix of D. Thus we see that the set P of flows, i.e., the solution set of (4.5), is a bounded polyhedron, i.e. a polytope (by Theorem 2.19). Note that P may be empty, and that a characterization of the nonemptiness of P may be found using Farkas' lemma.

We shall study the problem of finding a minimum cost flow in the next section, but here we shall consider a specialized flow called ***st*-flow**. Given two distinct nodes, a **source** s and a **sink** t in V, we call $x \in \mathbb{R}^E$ an st-flow if x satisfies

$$\begin{array}{lll} \text{(i)} & x(\delta^+(v)) - x(\delta^-(v)) = 0 & \text{for alle } v \in V \setminus \{s,t\}; \\ \text{(ii)} & \quad 0 \leq x_e \leq d_e & \text{for all } e \in E. \end{array} \tag{4.6}$$

An example is shown in Figure 4.9. We call the $f(x) := x(\delta^+(s))$ the **value** of the flow x, and note that it must be equal to $x(\delta^-(t))$ (why?). An st-flow may be viewed as a circulation in a certain graph associated with $D = (V, E)$. Without loss of generality we may assume that D has no arcs with head S and no arcs with tail t (this will be clear later). Define the **augmented graph** $\tilde{D} = (V, \tilde{E})$ by adding an "artificial" arc (t, s) to the arcs of D. Then a circulation $\tilde{x}$ in $\tilde{G}$ correspond to an st-flow through $\tilde{x}_e = x_e$ for each $e \in E$ and the flow value $f()$ equals the flow in the artificial arc (t, s).

As presented above the notion of flow, or st-flow, is expressed via flow in each arc. However, flows may also be represented in a different, more combinatorial way, in terms of paths and cycles. This is done through *flow-decomposition*, a very useful concept for, in particular, algorithmic developments.

Let $\mathcal{P}$ denote the set of all directed paths in D, and $\mathcal{C}$ denotes the set of all directed cycles in D. Let $P \in \mathcal{P}$ and assume that P is a st-path. Then P gives rise to a st-flow of value z given by $x = z\chi^P$, i.e., $x_e = z$ for each $e \in P$ and $x_e = 0$ otherwise. Similarly, each cycle $C \in \mathcal{C}$ induces a circulation $x = z\chi^C$. More generally, we can find an arc flow from a **path and cycle flow** function $g : \mathcal{P} \cup \mathcal{C} \to \mathbb{R}_+$; we call $g(P)$ and $g(C)$

the **flow** on path P and cycle C, respectively. We may view g as a vector in the space $PC := \mathbb{R}_+^{\mathcal{P}\cup\mathcal{C}}$ where each coordinate is a flow on some path or cycle. Define the linear transformation $T : PC_+ \to \mathbb{R}_+^E$ by $T(g) = x$ where $x = \sum_{P\in\mathcal{P}} g_P \chi^P + \sum_{C\in\mathcal{C}} g_C \chi^C$, i.e. $x_e = \sum_{P:e\in P} g_P + \sum_{C:e\in C} g_C$. It is clear that whenever $g \in PC_+$, then $x = T(g)$ is a an arc flow (for suitable capacity d and supply b). The next **flow decomposition theorem** says that the converse relation also holds, see [33].

Theorem 4.10 *Each arc flow $x \in \mathbb{R}^E$ may be decomposed as a path and cycle flow, that is, $x = T(g)$ for a suitable $g \in PC_+$. In fact, g may be chosen such that (i) if $g_P > 0$ for a path P then this path goes from a supply node to a demand node, (ii) at most $n + m$ paths and cycles have nonzero flow, and (iii) at most m cycles have nonzero flow.*

Proof. We shall describe an algorithm that successively determines paths and cycles and assigns numbers to these (flow values) and eventually ends up with the desired decomposition.

Let $x^0 = x$ and $b^0 = b$. We shall determine a path or a cycle with certain properties. Select, if any, a supply node s^0 (meaning that $x^0(\delta^+(s^0)) - x^0(\delta^-(s^0)) = b_v^0 > 0$). Then there is some adjacent node v_1 such that $x((s^0, v_1)) > 0$. If v_1 is a demand node (meaning that $x^0(\delta^+(v_1)) - x^0(\delta^-(v_1)) = b_v^0 < 0$), we terminate. Otherwise, we can find a new node v_2 with $x^0((v_1, v_2)) > 0$. In this way we proceeed until we either (i) determine a directed path P^0 from a supply node s^0 to a demand node t^0 such that $x_e^0 > 0$ for each $e \in P^0$, or (ii) we meet a node we have processed before so we have a directed cycle C with $x_e^0 > 0$ for each $e \in C$. Let H be the path or cycle determined in this way. ϵ^0 be the minimum value of x_e^0 for edges in H and, if $H = P$, b_{v_0}. Let $x^1 = x^0 - \epsilon^0\chi^H$ and define $b^1 = b^0$ if $H = C$ and if $H = P$ we let $b_{s^0}^1 = b_{s^0}^0 - \epsilon^0$, $b_{t^0}^1 = b_{t^0}^0 + \epsilon^0$ and $b_v^1 = b_v^0$ for all $v \neq s^0, t^0$. Then x^1 is a flow in the network (V, E, d, b^1).

Observe the crucial fact that *either the new flow has been reduced to zero in some arc or for some original supply node s we have obtained $b_s^0 = 0$*. We repeat the process described above until there are no more demand (or supply) nodes. Then we select, if any, a node with positive flow on some outgoing arc and can, as above, find a directed cycle with strictly positive flow. We then reduce the flow on this cycle and repeat this procedure until we end up with the zero flow. Thus we have found a sequence (x^k, b^k) for $k = 0, \ldots$ and eventually, for say $k = N$, we get $b^N = 0$ and then we terminate. Here the termination follows from the observation as we can find at most m cycles (in each case a flow on some arc is reduced to zero). Similarly, we must have $N \leq n + m$ since there are m arcs and at most n supply nodes. Furthermore, $x^N = 0$, so $0 = x^N = x - \sum_{j=0}^N \epsilon_j \chi^{H_j}$, and therefore we have found the flow decomposition $x = \sum_{j=0}^{N} \epsilon_j \chi^{H_j}$ with all the desired properties.

□

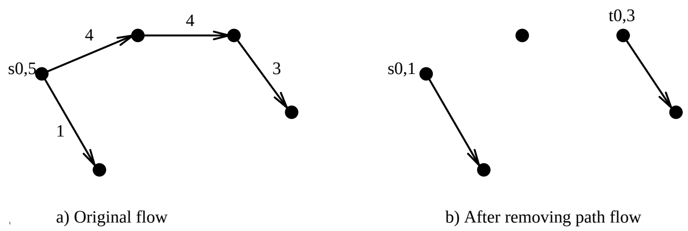

Figure 4.10: Finding flow decomposition

As a special case we apply flow decomposition to circulations.

Corollary 4.11 *A circulation can be decomposed as a cycle flow on at most m directed cycles.*

Proof. This is an immediate consequence of Theorem 4.10 since a circulation has no supply or demand nodes.

□

We now turn to a discussion of cuts. A (directed) cut in the digraph D is an arc set C of the form $C = \delta^+(S)$ where S is a nonempty node set not equal to V. Note that $\delta^+(S) = \delta^-(V \setminus \bar{S})$. Whenever $s, t \in V$, $s \neq t$ and the node set S satisfies $s \in S$, $t \notin S$, we call $\delta^+(S)$ an ***st*-cut**. When d is a capacity function defined on E, we call $d(C) = d(\delta^+(S)) = \sum_{e \in C} d_e$ the **capacity** of the cut $C = \delta^+(S)$. The basic property of an st-cut C is that each st-path P must contain at least one arc in C. More generally, we therefore have the following result.

Lemma 4.12 *The value of an st-flow x is no greater than the capacity of an st-cut C.*

Proof. Let $f(x)$ be the value of the st-flow x, so $f(x) := x(\delta^+(s))$, and let $S \subset V$ be such that $s \in S$, $t \notin S$ and let $C = \delta^+(S)$. Since x satisfies the flow conservation equations for all nodes in $S \setminus \{s\}$, we obtain $f(x) = \sum_{v \in S}[x(\delta^+(v)) - x(\delta^-(v))] = x(\delta^+(S)) - x(\delta^-(S)) \leq x(\delta^+(S)) \leq d(\delta^+(S)) = d(C)$. Here the last two inequalities were obtained from the capacity constraints (4.6)(ii).

□

We shall use this inequality in section 4.5 in order to determine a maximum st-flow.

4.5 Maximum flow and minimum cut

In this section we study two optimization problems in digraphs equipped with arc capacities: the maximum flow problem and the minimum cut problem. It is natural to treat these two problems in parallel as there are efficient algorithms that solve both problems simultaneously without additional work.

Let $D = (V, E)$ be a directed graph with nonnegative capacity function d defined on the arcs. Let s and t be two distinct nodes, called the source and the sink, respectively. The **maximum flow problem** is to find an st-flow with maximum value f. Note that this is actually an optimization problem with *continuous* variables, the flow x_e on each arc e. Thus, in contrast to the shortest path or spanning tree problems, we now have an infinite number of feasible solutions. It is therefore not obvious that a maximum flow really exists. It does, however, as can be seen by a compactness argument. Here we shall follow the presentation given in [23].

Lemma 4.13 *There exists a maximum st-flow x.*

Proof. The value of an st-flow x is $f(x) = x(\delta^+(s) = \sum_{e \in \delta^+(s)} x_e$ which is a continuous function of $x \in \mathbb{R}^E$. Furthermore, the feasible set of the max-flow problem is the bounded polyhedron P defined by the linear inequalities in (4.6). Thus P is a compact set (as each polyhedron is also closed), and by Weierstrass' theorem the maximum of the continuous function f is attained in P.

□

We remark that we obtain another proof of this result by polyhedral theory. In fact, since the value function is linear, the set of maximum flows is a nonempty face F of the polytope P. Thus F is also a polytope and each vertex in F is a maximum flow. This also means that the max flow problem is a linear programming problem and could be solved using e.g. a general linear programming algorithm. However, we shall next describe a faster combinatorial algorithm which is also efficient.

The **minimum st-cut problem** is to find an st-cut C in D with minimum capacity $d(C)$. This problem is closely connected to the max-flow problem since we obtain upper bounds on the flow value from st-cuts.

Lemma 4.14 *The maximum flow value is no greater than the minimum cut capacity.*

Proof. From Lemma 4.12 we have that $f(x) \leq d(C)$ whenever x is an st-flow of value $f(x)$ and C is an st-cut. By taking the maximum over all st-flows (and this maximum exists by Lemma 4.13), and then taking minimum over all cuts we obtain the desired result.

□

In fact, the inequality of the previous lemma is an equality, i.e., there exists an st-flow with value equal to some st-cut. This result, called the **max-flow min-cut theorem** is one of the most important results in combinatorial optimization and also combinatorics. We shall prove this theorem constructively. An illustration of a minimum cut in a digraph is given in Figure 4.11. It has capacity 5. Can you find an st-flow with value 5?

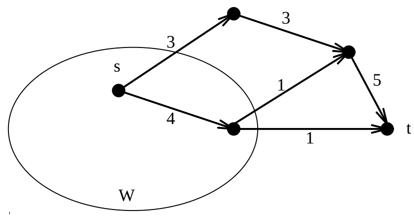

Figure 4.11: A minimum st-cut

The concept of an *augmenting path* is useful for modifying a flow into another flow with larger value. Let x be an st-flow. Let $v \in V$, $v \neq s$, and consider an sv-path P in the underlying undirected graph G associated with D. This graph is obtained by replacing each arc by an edge with the same end nodes. In D this path P consists of **forward arcs** and **backward arcs**; for the forward arcs the tail is closer than the head to s on P, while for backward arcs the head is closer. We let P^+ and P^- denote the set of forward and backward arcs, respectively. We say that such a sv-path P is x**-augmenting** to v if $x_e < d_e$ for each forward arc of P and $x_e > 0$ for each backward arc of P. If here $v = t$, we call P an x**-augmenting path**.

A characterization of whether an st-flow is maximum may be given in terms of augmenting paths.

Lemma 4.15 *An st-flow x is maximum if and only if there is no x-augmenting path.*

Proof. Let the *augmenting node set* $S(x)$ consist of s and all nodes v for which there exists an x-augmenting path.

Assume first that there is an augmenting path P, i.e., $t \in S(x)$. Define ϵ^+ as the minimum value among the numbers $d_e - x_e$ for e a forward arc of P, and ϵ^- as the minimum value of x_e for e among the backward arcs e of P, and finally let $\epsilon = \min\{\epsilon^+, \epsilon^-\}$. Then we have $\epsilon > 0$ as P is x-augmenting. Let $\bar{x}$ be obtained from x by defining $\bar{x}_e = x_e + \epsilon$ for $e \in P^+$, $\bar{x}_e = x_e - \epsilon$

for $e \in P^-$, and $\bar{x}_e = x_e$ for all other e. Then $\bar{x}$ is an st-flow with value $f(\bar{x}) = f(x) + \epsilon$, which proves that x is not a maximum flow.

Conversely, assume that there is no augmenting path, so $t \notin S(x)$. We shall prove that the st-cut $C(x) = \delta^+(S(x))$ has capacity equal to the value of the flow x. In fact, for each arc $e = (i, j) \in C(x)$ (meaning that $i \in S(x), j \notin S(x)$) we must have $x_{ij} = d_{ij}$ for otherwise we would have $j \in S(x)$. Furthermore, for each arc $e = (i, j)$ with $i \notin S(x), j \in S(x)$ we have $x_{ij} = 0$ (otherwise $i \in S(x)$). Thus the flow in each arc in the cut $C(x)$ is at its upper capacity while the flow in each arc in the reverse cut $\delta^-(S(x))$ is zero. From this (and flow conservation) $f(x) = \sum_{v \in S(x)} [x(\delta^+(v)) - x(\delta^-(v))] = x(\delta^+(S(x))) - x(\delta^-(S(x))) = x(\delta^+(S(x))) = d(\delta^+(S(x))) = d(C(x))$. Thus the value of the flow x equals the capacity of the cut $\delta(S(x))$. But by Lemma 4.14 it follows that x is a maximum st-flow, as desired, and also that $C(x)$ is a minimum st-cut.

□

From this result we obtain the max-flow min-cut theorem, discovered independently in 1956 by Ford and Fulkerson and by Elias, Feinstein and Shannon.

Theorem 4.16 *For any directed graph with arc capacity function and distinct nodes s and t the value of a maximum st-flow equals the minimum st-cut capacity.*

Proof. Let x be a maximum flow which is known to exist by Lemma 4.13. Then, according to Lemma 4.15, there is no augmenting path and the flow value $f(x)$ equals the capacity of the cut $C(x)$ as defined in the proof of that lemma. Then it follows from Lemma 4.14 that $C(x)$ is a minimum st-cut, and the theorem follows.

□

This theorem is an example of a **minmax theorem**: the maximum value of some function taken over a class of objects equals the minimum value of some other function for another class of objects. In several interesting cases such results can be obtained by combining linear programming duality with the concept of *total unimodularity.* We return to this in chapter 5, and, in fact, we shall show that the max-flow min-cut theorem may be obtained via these methods.

Based on augmenting paths we can give the following generic algorithm for solving the maximum flow problem. We use the notation introduced in connection with Lemma 4.15.

Generic max-flow algorithm. *Step 0. Let $x = 0$.*

Step 1. Determine the augmenting node set $S(x)$. If $t \notin S(x)$, terminate; the current st-flow x is optimal. Otherwise, let P be an augmenting path.

Step 2. Update the flow x by increasing the flow by ϵ on each forward arc of P, and reducing the flow by ϵ on each backward arc of P. Return to Step 1.

We remark that this algorithm may not terminate in case of irrational data. Whenever d is rational, we may replace the problem by one with integer capacities, simply by multiplying each capacity by some suitably large integer M. Then the problem is scaled and the optimal flow in the original problem is reconstructed by dividing flows by M.

A very important corollary of Theorem 4.16 is that there exists an *integral* maximum st-flow whenever d is integral. Recall that we say that a vector is integral whenever all its components are integral.

Corollary 4.17 *Whenever the capacity function d is integral, the generic maximum flow algorithm terminates with an integral flow, i.e. the flow on each arc is an integer.*

Proof. We note that in each iteration the flow augmentation ϵ is an integer. This follows from the fact that the first flow is integral and each of the magnitudes $d_e - x_e$ are integral, and the integrality of each flow follows from a simple induction argument.

$\square$

Note that we also obtain a solution of the minimum cut problem as a byproduct of the max-flow algorithm above. Again this relates to the minmax result of Theorem 4.16.

A very important corollary of the max-flow min-cut theorem is **Menger's theorem** (originally from 1927, see [27]) which is a central result in combinatorics involving connectivity in graphs. We say that a set of dipaths in a digraph are **arc-disjoint** if each arc belongs to at most one of these paths.

Theorem 4.18 *The maximum number of arc-disjoint st-dipaths in a directed graph equals the minimum number of arcs in an st-cut.*

Proof. Let each arc in the digraph have capacity 1. Then the capacity of a cut equals its number of arcs. Furthermore, it is easy to see that an st-flow x of integral value k may be written as a sum of unit flows on arc-disjoint st-dipaths $P_1, \ldots, P_k$: $x = \sum_{j=1}^{k} \chi^{P_j}$. Therefore a maximum st-flow corresponds to a maximum number of arc-disjoint st-dipaths. Thus the desired result follows from Theorem 4.16.

$\square$

We conclude this section by a few remarks concerning efficient implementations of the maximum flow algorithm. We consider a digraph $D = (V, A)$.

Data structures. In many graph algorithms one needs, typically within one or more loops, to find a path from some given node to some other node (or several nodes). It is therefore important to be able to determine the neighbor nodes of a given node fast. Except for small problems, this rules out using the adjacency matrix as a data structure (it often takes up too much storage as well). In stead one can use **adjacency lists** as sketched next. We describe this for digraphs; graphs may be handled similarly by replacing each edge by two opposite arcs. One introduces two data types, say type *node* and type *arc*. The nodes in the digraph is stored as a linked list of *nodes*. For each node V one also stores a linked list of *arcs*; this is the set of arcs having v as tail (initial endnode). Thus both the data types contain next-pointers, and in addition one typically has data fields for possible name, number and a field for storing a label (binary). The label may e.g. be used for marking the nodes visited in some search so that the same node is not visited (or processed) more than once.

Search. Say that we have a digraph respresented as in the previous paragraph, and that we want to find a st-path. This may be done using **breadth-first-search** as follows. Initially we give all nodes the label *unscanned*, and we shall manipulate a queue Q of nodes which initially is empty. First we visit all the neighbor nodes of s by traversing the adjacency list of s; each of these nodes are put into Q in the order visited, say that the first node is on top of the queue. We then relabel s as *scanned*. In the next step, we consider the node v_1 on top of Q and traverse its adjacency list as we did for s. We put each *unscanned* neighbor node v with $v \notin Q$ onto Q. Having finished this we label v_1 *scanned* and proceed with the next node on the top of Q in similar way. Eventually we either meet t as one of the neighbor nodes, in which case there is a st-path, or we terminate because the queue is empty, and then there is no st-dipath. Note the we can construct the desired st-path by simply marking each node being put onto the queue by its predecessor node (the node from which it was visited).

Max-flow implementation. We now return to the implementation of our generic maximum flow algorithm. First, we note how the problem of finding an augmenting path (which, as we recall, contained both forward and backward arcs) may be transformed to finding a directed st-path. Let x be an st-flow and define $A_f(x) = \{(i,j) : (i,j) \in A,\ x_{ij} < d_{ij}\}$ and $A_b(x) = \{(j,i) : (i,j) \in A,\ x_{ij} > 0\}$. The elements in A_f are called **forward arcs** and the elements in $A_b(x)$ are called **backward arcs**. Note that if the flow in (i,j) is strictly between its lower and upper bounds, then we have $(i,j) \in A_f(x)$ and $(j,i) \in A_b(x)$. The **augmenting digraph** is the digraph $D(x)$ with node set V and arc set $A(x) = A_f(x) \cup A_b(x)$. An *st*-path in $D(x)$ is called an **augmenting path** in $D(x)$. The key property is the one-to-one correspondence between augmenting dipaths in $D(x)$ and augmenting paths in G (as defined previously). Therefore, the max-flow algorithm consists in iteratively using e.g. breadth-first-search in $D(x)$ to find, if any, an augmenting dipath in $D(x)$, and the update the flow as explained before. The result is a new *st*-flow with larger value, and the new augmenting network is determined, and the process continues until no augmenting path can be found. In that case the current flow is maximum, and we also have labeled a set of nodes which induce a minimum cut.

It should be said that some care must be takes to make this into a polynomial algorithm. This can be done by making sure that one finds an augmenting path which is shortest possible; here length refers to the number of arcs in the path. This is accomplished by the breadth-first-search since this search determines consecutive layers of nodes in the network with respect to the distance from s.

For extensive discussions on max-flow algorithms and their implementation, we refer to [33].

4.6 Minimum cost network flows

In this section we treat the minimum cost network flow problem, a problem with many real-world applications and which also contains the maximum flow problem as a special case.

In Section 4.4 we defined a (network) flow in a network $\mathcal{N} = (V, E, d, b)$ as a feasible solution of the linear system $Ax = b$, $0 \leq x \leq d$, where A is the node-arc incidence matrix, b is the supply vector and d is the capacity vector. We now consider a linear cost associated with each flow x; the cost of flow x is $c^T x = \sum_{e \in E} c_e x_e$. The **minimum cost network flow problem** (MCNF) may be formulated as the following linear programming problem:

$$\min\{c^T x : Ax = b,\ 0 \leq x \leq d\}. \tag{4.7}$$

Since the MCNF problem is an LP problem, it can be solved with e.g., the general simplex method. However, one may improve this algorithm by exploiting the network structure of the coefficient matrix. Before we discuss these matters, it is useful to point out some applications of the MCNF problem.

Example 4.1 *The* **transportation problem** *may be presented as follows. Assume that we have a set of customers J where customer $j \in J$ has demand* *of s_j units of some product, say oil (and a unit may be one litre). In addition there is a set of suppliers I, and supplier $i \in I$ has available r_i litres of the same product. We assume that the cost of transporting x_{ij} litres of oil from supplier i to customer j is $c_{ij}x_{ij}$; so the cost is proportional to the amount transported. The question is: how should all the demands be met (satisfied) in order to minimize total costs and not exceeding the available supplies. We assume that the total supply is equal to the total demand, i.e., $\sum_{i \in I} r_i = \sum_{j \in J} s_j = T$. This transportation problem can be modeled as a MCNF problem by introducing a digraph D with node set $D = (I \cup J, E)$ where E contains the arcs (i, j), for each $i \in I$ and $j \in J$. The nodes in I and J corresponds to the suppliers and customers. We let the cost function be given by the unit cost c_{ij} on arc (i, j), for $i \in I$, $j \in J$. The supply function is $b_i = r_i$ for $i \in I$ and $b_j = s_j$ for $j \in J$. The capacities may be set to ∞ (or, if one prefers, a finite capacity $d_{ij} = M$ where M is some number e.g. greater than $\max_{i \in I} r_i$). Consider the MCNF problem in this network, and note that for feasible flow x the total flow into each node $j \in J$ must equal s_j, and the total flow out of node $i \in I$ must equal r_i. We see from this that the MCNF correctly models the transportation problem, and in the optimal solution x_{ij} tells us the amount of oil to be transported from supplier i to customer j.*

We remark that the transportation problem allows for some slight generalizations. First, it would be easy to introduce capacities on the amount of goods that can be transported from each supplier to each customer. We just add the proper upper capacity on the flow in arc (i, j) for $i \in I$ and $j \in J$. Secondly, we could also handle the situation where the total supply exceeds the total demand, so $\sum_{i \in I} r_i > \sum_{j \in J} s_j$. This can be done by adding an artificial customer node v with demand $\sum_{i \in I} r_i - \sum_{j \in J} s_j$ (and infinite capacity on the arc (v, t)), and we are back to the balanced situation. The opposite situation, where total supplies are not large enough to satisfy all demands is treated in an exercise.

Example 4.2 *The maximum flow problem can be viewed as a MCNF problem. Recall from Section 4.4 that finding a maximum st-flow in the digraph $D = (V, E)$ with capacity function d could be viewed as finding a circulation in the augmented graph $\tilde{D}$ with maximum flow through the augmented arc (t, s). Consider the network with digraph being this augmented graph, supply function $b = 0$ and capacities on all ordinary arcs ($e \in E$) given by d while we let the augmented arc (t, s) have infinite capacity (or, for those who dislike ∞, let the capacity be $c(\delta^+(s)$; the maximum flow cannot exceed this cut capacity). We let the cost function c be all zero, except again for the arc (t, s) where we let $c_{ts} = -1$. The MCNF problem in this network is therefore to* *find a circulation in the augmented graph with the flow in arc (t, s) as large as possible, i.e., we have the maximum flow problem.*

For many interesting applications of the MCNF problem we refer to [33].

We now proceed with a description of how the simplex algorithm may be adapted to a specialized algorithm for solving the MCNF problem. The first topic is to study the structure of basic feasible solutions of the LP problem 4.7. We shall assume that the digraph is connected (more precisely, the underlying graph where each arc is replaced by an undirected edge is connected). As usual n and m denotes the number of nodes and arcs, respectively.

Proposition 4.19 *We have that* $\text{rank}(A) = n - 1$*. Furthermore, if* $\tilde{A}$ *is the matrix obtained from* A *by deleting one row in* A *(no matter which one), then the rows of* $\tilde{A}$ *are linearly independent.*

Proof. Note first that we must have $m \geq n - 1$ (otherwise the underlying graph would be disconnected, see Proposition 4.5). The sum of all the row vectors in A equals the zero vector, so these row vectors are linearly dependent, and therefore $\text{rank}(A) \leq n - 1$. Let $\tilde{A}$ be obtained from A by deleting the row corresponding to some node i_0.

We shall use a kind of elimination procedure on a tree, where leaf after leaf is eliminated, and gradually we construct a lower triangular matrix with desired properties.

Let $S = (V, E[S])$ be a spanning tree in D (in the sense that the arc direction is ignored), and let $\tilde{A}_S \in \mathbb{R}^{n-1,n-1}$ be the submatrix of $\tilde{A}$ induced by the columns that correspond to arcs in S. Let $i_1 \neq i_0$ be a leaf in S; such a leaf exists according to Corollary 4.6 (S contains at least two leaves, so one must be distinct from i_0). The row in $\tilde{A}_S$ corresponding to i_1 is then a unit vector with a 1 in, say, column j_1 (so the remaining elements of that row are 0). Using row and column permutations on $\tilde{A}_S$ we move the (i_1, j_1)'th element to position $(1, 1)$ of this matrix, and then the first row and column both consist of zeros except for entry $(1, 1)$ which is 1.

Next, we delete node i_1 (and the incident arc j_1) from the tree S. The new tree has a leaf $i_2 \neq i_0$, and the i_2'th row in $\tilde{A}_S$ has at most two nonzeros, one corresponding to an arc j_2 in the new tree, and possibly one corresponing to j_1 (which happens if $j_1 = [i_1, i_2]$). We permute rows and columns so we get the i_2'th row as a new second row, and the column j_2 as a new second column in $\tilde{A}_S$. Now the modified matrix $\tilde{A}_S$ has the property that its principal submatrix consisting of the first two rows and columns is lower triangular with diagonal elements being 1 or -1. We repeat this procedure to the last tree, and eventually $\tilde{A}_S$ is modified into a matrix which is lower triangular with 1's and -1's on the diagonal. The determinant is therefore 1 or -1 and the matrix is nonsingular. This proves that the row-rank of $\tilde{A}$ is $n-1$. i.e., the rows are linearly independent, and therefore we also get that $\text{rank}(A) = n-1$.

□

In fact, the previous proof also leads to a description of the largest nonsingular submatrices of the node-arc incidence matrix A.

Proposition 4.20 *Choose a node $i_0 \in V$ (called root node), and let $\tilde{A}$ be the matrix obtained from the node-arc incidence matrix A by deleting the row corresponding to i_0. Then there is a one-to-one correspondence between the spanning trees in D and the bases in $\tilde{A}$ (i.e., nonsingular submatrices of dimension $(n-1) \times (n-1)$).*

Proof. The proof of Proposition 4.19 shows that every spanning tree in D gives rise to a basis in $\tilde{A}$. We shall prove the converse, that each basis arises in this way. So let $B \in \mathbb{R}^{n-1,n-1}$ be a basis in $\tilde{A}$, say that it corresponds to the $n-1$ columns $E' \subset E$. Then the columns of B are linearly independent, and this implies that also the corresponding columns in A are linearly independent.

Consider the subgraph $D' = (V, E')$. We claim that D' contains no cycle (in the underlying graph). For, assume that C is such a cycle with arcs $j_1, \ldots, j_t$. Choose an orientation of the cycle, so each arc is either a forward arc or a backward arc. Define $\lambda_i = 1$ if j_i is a forward arc, and $\lambda_i = -1$ if j_i is a backward arc, and let a_j denote the j'th column of A. Then we have that $\sum_{i=1}^{t} \lambda_i a_{j_i} = 0$ which contradicts that the columns in A corresponding to E' are linearly independent, and the claim follows.

Therefore D' is an acyclic graph with n nodes and $n-1$ arcs, and by Proposition 4.5 D' is a tree.

□

We are now prepared to describe the specialized simplex algorithm for the MCNF problem. Or, rather, in order to make things a bit easier, we assume that all upper bounds are ∞:

$$\min\{c^T x : Ax = b,\ x \geq 0\}. \tag{4.8}$$

This is therefore an LP problem on standard form. We remark that the MCNF problem with upper bounds on the flow can be treated similarly by an adoption of *the simplex algorithm with upper bounds* (where upper bounds are treated implicitly as lower bounds, and nonbasic variables are on one of the two bounds).

Let A and $\tilde{A}$ be as in Proposition 4.20 (for a chosen root node i_0). We shall apply the simplex algorithm to this "reduced problem":

$$\min\{c^T x : \tilde{A}x = \tilde{b},\ x \geq 0\}. \tag{4.9}$$

where $\tilde{b}$ is obtained from b by deleting the i_0'th coordinate. We first describe the basic solutions in (4.9), so consider a basis in $\tilde{A}$ and the corresponding spanning tree S (confer Proposition 4.20). How can we compute the corresponing basic solution x^S? Here the triangularity of the basis, seen in the proof of Proposition 4.19, makes this an easy task, in fact, it is a simple back-substitution as we know from the Gaussian elimiation algorithm for solving linear equations. First, all the nonbasic variables are zero, so for all $e \notin S$ we have $x^S_e = 0$. Let $i_1 \in V$ be a leaf of S, and let e_1 be the unique arc in S incident to i_1. Then we get from the flow balance equation in node i_1 that $x^S_{e_1}$ equals b_{i_1} (resp. $-b_{i_1}$) if i_1 is the tail (resp. head) of e_1. We then proceed similarly: consider the subtree $T \setminus \{i_1\}$, find a leaf and determine the flow in the unique tree-arc incident to the leaf based on flow balance and the flow in all the other arcs incident to i_1.

Example 4.3 *Consider the digraph D and the supply vector b shown in Figure 4.12 (a). A spanning tree S is indicated in (b) with the associated basic solution x^S. This solution may be found as follows. Let (for instance) w be the root node, and consider the leaf t of S. The flow in arcs (s, v), (v, w) and (u, t) are all zero (nonbasic). From flow balance in t we get that $x^S_{wt} = 5$. "Deleting" w we select another leaf, say v and find that $x^S_{uv} = 2$. Continuing in this manner we find the indicated solution.*

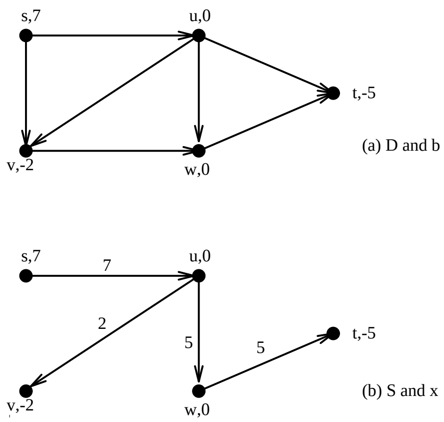

Figure 4.12: A basic solution in the network flow problem

We remark that we are free to use the root node as a leaf in the calculation of the basic solution, since flow balance holds in this node as well.

The basic solution we found in the example was in fact feasible, so x^S is a vertex of the polyhedron defined by the constraints $Ax = b$, $x \geq 0$. However, there are also nonfeasible basic solutions, i.e., basic solutions for which some arc flow is negative; try to find one in the example! Next, we explain how to determine dual basic solutions. Let B be a basis in $\tilde{A}$, so B corresponds to some spanning tree S in D. The dual solution y^S is the solution of $(y^S)^T B = c_B^T$. Here y^S contains a dual variable y_i for each $i \neq i_0$. However, it is convenient in the description of the algorithm to introduce a "dual variable" y_{i_0} associated with the root node, and e.g., let $y_{i_0} = 0$. Then $(y^S)^T B = c_B^T$ is equivalent to $y_i^S - y_j^S = c_{ij}$ for each $(i,j) \in T$. This system may also be solved using back-substitution due to the triangularity of B (more precisely, a suitable permutation of B is lower triangular). In fact, for each node v which is adjacent to i_0 in S we may determine y_v directly using the equation for the arc connecting i_0 and v and that $y_{i_0} = 0$. We may then proceed to consider further nodes similarly, and finally, as S is a spanning tree, we have determined all the dual variables.

The overall simplex algorithm for solving the MCNF problem (without upper bounds) is as follows:

The network simplex algorithm. *Step 0. Let x^0 be a basic feasible solution corresponding to a tree S.*

Step 1. Calculate the dual variables y_i^S, $i \in V$ by solving the equations $y_i^S - y_j^S = c_{ij}$ for each $(i,j) \in S$ (using $y_{i_0}^S = 0$). The reduced cost for the nonbasic variable x_{ij}, $(i,j) \notin S$ is then $\bar{c}_{ij} = c_{ij} - (y_i^S - y_j^S)$.

Step 2. If $\bar{c}_{ij} \geq 0$ for all $(i,j) \notin S$; stop, the current solution is optimal. Otherwise, go to Step 3.

Step 3. Choose a nonbasic arc $x_{i,j} \notin S$ with $\bar{c}_{ij} < 0$. Let C be the unique cycle formed by adding (i,j) to S. Adjust the flow by increasing the flow around the cycle C such that x_{ij} is increased as much as possible, until another arc $e \in C$, $e \neq (i,j)$ gets zero flow. Then update the basis tree to $T := T \setminus \{(i,j)\} \cup \{e\}$, and return to Step 1.

To start this algorithm we need a basic feasible solution which may be found by solving a maximum flow problem. Using suitable data structures (lists) the network simplex algorithm may be implemented so that it is significantly faster than the standard simplex algorithm on the same problem. For further implementation details, see e.g., [33], [25] or [2].

4.7 Exercises

Problem 4.1 *Give relations between the incidence matrix A_G of a graph and the incidence matrix of G's different subgraphs.*

Problem 4.2 *Prove Proposition 4.5.*

Problem 4.3 *Work out a correctness proof for Dijkstra's algorithm, based on Lemma 4.7.*

Problem 4.4 *Determine the complexity of Dijkstra's algorithm.*

Problem 4.5 *Implement Dijkstra's algorithm and test on a few examples.*

Problem 4.6 *Modify the greedy algorithm so that it solves the maximum weight spanning tree problem. Prove the correctness of the algorithm.*

Problem 4.7 *Determine a the complexity of Kruskal's algorithm.*

Problem 4.8 *Consider a weighted graph G (nonnegative weights) which may not be connected. Consider the problem of finding a maximum weight forest in G. What happens if you apply the greedy algorithm to this problem?*

Problem 4.9 *A* **matroid** *is a pair $(E, \mathcal{M})$ where E is a finite set and $\mathcal{M}$ is a class of subsets of E such that (i) $\emptyset \in \mathcal{M}$, (ii) if $X \in \mathcal{M}$ and $Y \subseteq X$, then $Y \in \mathcal{M}$, and (iii) if $X, Y \in \mathcal{M}$ with $|X| = |Y|+1$, there is an $x \in X \setminus Y$ such that $Y \cup \{x\} \in \mathcal{M}$. We call each $F \in \mathcal{M}$ a* **feasible set** *or an* **independent set***.*

Let now $G = (V, E)$ be a graph and let $\mathcal{M}$ consist of the edge set of all forests in G. Prove that $(E, \mathcal{M})$ is a matroid. We call this the **graphic matroid***.*

Next, let E be a finite set of vectors in $\mathbb{R}^n$, and let $\mathcal{M}$ consist of all the subsets of these vectors that are linearly independent. Prove that $(E, \mathcal{M})$ is a matroid. We call this the **linear matroid***.*

Let $(E, \mathcal{M})$ be a (general) matroid with nonnegative weights defined on E. Consider the problem of finding a maximum weight feasible (independent) set, i.e. max $\{w(F) : F \in \mathcal{M}\}$. Try to generalize the greedy algorithm to this problem. One can show that such an extension correctly solves the mentioned problem. Interprete the algorithm for the linear matroid. Do you think that the greedy algorithm also works (is correct) for all/other combinatorial optimization problems?

Problem 4.10 *Let $(E, \mathcal{M})$ be a matroid. A* **basis** *in this matroid is a maximal feasible set, i.e. a set $F \in \mathcal{M}$ such that $F \cup \{e\} \notin \mathcal{M}$ for all $e \in E \setminus F$. Prove that all bases have the same cardinality. Next, prove that a set system $(E, \mathcal{M})$ satisfying properties (i) and (ii) in our definition of a matroid in Problem 4.9 is in fact a matroid if and only if for all $A \subseteq E$ all maximal feasible subsets of A have the same cardinality. What are the bases for the graphic and the linear matroids? One final point: let B and B' be bases of* *a matroid and let $x \in B \setminus B'$. Prove that there is an $y \in B' \setminus B$ such that $(B \setminus \{x\}) \cup \{y\}$ is a basis. Can you interprete this result for the two mentioned matroids?*

Problem 4.11 *Consider a digraph $D = (V, E)$ and let s and t be distinct nodes in D. Assume that (i) $d^+(v) = d^-(v)$ for all $v \neq s, t$ and (ii) $d^+(s) = k + d^-(s)$ for some positive number k. Prove that D contains k arc-disjoint st-dipaths.*

Problem 4.12 *Figure out a small, but interesting (!) maximum flow example and solve the problem using our generic max-flow algorithm.*

Problem 4.13 *Try to find an example of a maximum flow problem where an "unlucky" choice of the augmenting paths leads to a high number of augmentations.*

Problem 4.14 *Figure out a small, but interesting (!) minimum cost network flow example and solve the problem by hand using the network simplex algorithm. Then use CPLEX to solve the same problem (with and without Netopt).*

Chapter 5

Combinatorial optimization and integral polyhedra

In polyhedral combinatorics one studies combinatorial problems by using concepts and results from convex analysis, polyhedral theory and linear programming.

One of the important contributions of polyhedral combinatorics is that it gives a general scheme for finding a combinatorial minmax relationship between "associated" combinatorial optimization problems. This procedure is based on combining results on integral polyhedra (i.e., polyhedra with integral vertices) with linear programming duality. These minmax results may often be used to develop a combinatorial algorithm for solving these optimization problems. In many cases it seems difficult, or impossible, to find such minmax relations, but the theoretical results may still lead to fast algorithms for solving interesting problem instances to optimality, or near-optimality. In the next chapter we discuss *polyhedral methods* based on the theory outlined here. Today, polyhedral combinatorics is a very active mathematical discipline and it may be viewed as a geometric study of certain classes of integral polyhedra.

For further reading in polyhedral combinatorics see [32], [29], [35] and [5].

5.1 A basic approach

We discuss a basic "work plan" in polyhedral combinatorics and illustrate with an example.

Consider the combinatorial optimization problem (CO) defined in Section 4. We have a class $\mathcal{F}$ of **feasible sets** each being a subset of a finite ground set E. The nonnegative weight function is w and the optimization problem is max $\{w(F) : F \in \mathcal{F}\}$.

The following approach is often used in polyhedral combinatorics, see [32]. The basic idea is to translate the (CO) problem into a linear programming problem in $\mathbb{R}^E$.

A basic approach:

1. *Represent each $F \in \mathcal{F}$ by its incidence vector χ^F in $\mathbb{R}^E$, and let P be the polytope defined as the convex hull of these incidence vectors. Sometimes one prefers to study the polyhedron being the vector sum of this polytope and the cone being the nonnegative orthant to this polytope).*

2. *Find a complete linear description of P.*

3. *Apply LP duality to obtain a minmax theorem.*

4. *Develop an algorithm using the minmax theorem as a stopping criterion.*

The most difficult part of this process is Step 2, so often one is only able to find partial descriptions of P (e.g., only a subclass of the facet defining inequalities are known). However, even such a partial description may be of great interest, both theoretically and practically.

We illustrate the approach by the problem of finding a **maximum weight forest** F in a given connected graph $G = (V, E)$ with weight function w; this example is from [32]. Note that if all weights are nonnegative this is equivalent to the spanning tree problem. With negative weights present, the optimal solution may be a forest and not a tree. In Chapter 4 we explained how the spanning tree problem can be solved in polynomial time by the greedy algorithm. The idea was to iteratively extend a forest by adding a minimum weight edge which does not introduce any cycles. This algorithm (see below) also solves the maximum weight forest problem, provided that we terminate whenever the next edge has nonpositive weight.

First we define the polytope of interest, so we let the **forest polytope** $F(G)$ be the convex hull of the incidence vectors of all forests in G. Jack Edmonds showed in [10] the following theorem which gives a complete linear description of $F(G)$.

Theorem 5.1 *$F(G) \subset \mathbb{R}^E$ is the solution set of the following linear system*

$$\begin{array}{lll} \text{(i)} & x_e \geq 0 & \text{for all } e \in E; \\ \text{(ii)} & x(E[S]) \leq |S| - 1 & \text{for all } S \subseteq V. \end{array} \qquad (5.1)$$

Proof. We follow the presentation of [32]. The idea of this proof, due to Edmonds, is to use the greedy algorithm on the LP dual to the problem of maximizing $c^T x$ subject to the linear system (5.1). Also, via complementary slackness, an optimal primal solution is constructed.

Let Q denote the polytope being the solution set of (5.1), and we shall prove that $Q = F(G)$. Note that the integral vectors in Q are precisely the incidence vector of forests since the inequalities (5.1)(ii) prevents cycles. By convexity this implies that $F(G) \subseteq Q$.

To prove the reverse inclusion, let $\bar{x}$ be a vertex of Q, i.e., this point is the *unique* optimal solution an LP problem max $\{c^T x : x \in Q\}$ for suitable $c \in \mathbb{R}^n$. The LP dual of this problem is

$$\begin{array}{lll} \text{min} & \sum_{S \subseteq V} y_S(|S|-1) & \\ \text{subject to} & & \\ \text{(i)} & \sum_{S:e \in E[S]} y_S \geq c_e; & \text{for all } e \in E; \\ \text{(ii)} & y_S \geq 0 & \text{for all } S \subseteq V. \end{array} \tag{5.2}$$

Consider the greedy algorithm applied to the primal problem, and assume that the edges found are $F = \{e_1, \ldots, e_s\}$ in that order. Thus, in the i'th iteration the edge e_i is added and it joins two components of the current solution and forms the new component V_i. (For instance, when $i = 1$ we have n components and $e_i = [u, v]$ joins the components $\{u\}$ and $\{v\}$, so $V_1 = \{u, v\}$.)

The dual solution y is defined next. We let $y_S = 0$ for all sets S not among the V_i, $i = 1, \ldots, s$. The values for the sets V_i are determined in the reverse order as follows. Let $y_{V_r} = c(e_r)$. Consider the primal greedy solution x' found above. The complementary slackness condition for edge e says, as $x'_{e_r} > 0$, that $\sum_{S:e_r \in E[S]} y_S = c(e_r)$. With our definition of y_{V_r} this equality already holds, and we are forced to define $y_S = 0$ for all other sets S that contain both endnodes of e_r. To define the remaining components of y, let, for each $j \in \{1, \ldots, r-1\}$, $I(j) = \{i : j+1 \leq i \leq r$ and both end nodes of f_j are in $V_i\}$ and define $y_{V_j} = c(e_j) - \sum_{i \in I(j)} y_{V_i}$ for $j = r-1, r-2, \ldots, 1$.

One can now check that y is dual feasible, x' is primal feasible and that the complementary slackness condition holds. Thus it follows from LP theory that both these solutions are optimal in the respective problems. But $\bar{x}$ was also optimal in the primal, and due to the uniqueness, we must have $\bar{x} = x'$, which proves that every vertex of Q is integral and we have $Q = F(G)$ as desired.

□

This proof idea has later been applied to show other theorems in polyhedral combinatorics. In fact, Edmonds showed the result above in the more general setting of matroid theory.

As mentioned it may be difficult to find a linear desription of P. However, it is usually easy to find a polyhedron whose integer points correspond to the feasible sets of interest. We shall describe a general technique for this, see [26]. Combining this with LP duality we also obtain problems that give bounds on the optimal value.

Consider again a general combinatorial optimization problem (CO) with ground set E, family $\mathcal{F}$ of feasible sets and nonnegative weight function w: min $w(F) : \{F \in \mathcal{F}\}$. Since w is nonnegative, we may assume that each $F \in \mathcal{F}$ is *minimal* with respect to set inclusion, i.e., it does not strictly contain another feasible set. Such a family $\mathcal{F}$ is called a **clutter**. We define the **blocker** $BL(\mathcal{F})$ of $\mathcal{F}$ as the class of subsets $B \subset E$ with $B \cap F$ nonempty for each $F \in \mathcal{F}$. Note that this definition applies to any set system (familiy of subsets of E) $\mathcal{F}$. Thus a set in $BL(\mathcal{F})$ intersects all the feasible sets. From the definition we see that $BL(\mathcal{F})$ is a clutter. A useful property is that

$$BL(BL(\mathcal{F})) = \mathcal{F}. \tag{5.3}$$

so $\mathcal{F}$ consists precisely of those sets blocking $BL(\mathcal{F})$. As an example, let $\mathcal{F}$ consist of the (edge sets of) st-paths in a graph $G = (V, E)$, for two distinct nodes s and t. Then $BL(\mathcal{F})$ consists of the edge-minimal st-cuts. Then (5.3) says that the st-paths are the minimal sets intersecting all st-cuts.

Define the polyhedron

$$P = \text{conv}(\{\chi^F : F \in \mathcal{F}\}) + \mathbb{R}^E_+. \tag{5.4}$$

so the (CO) problem may be viewed as min $\{w^T x : x \in P\}$. From (5.3) it follows that an *integer* linear programming formulation for (CO) is

$$\begin{array}{lll} \text{minimize} & w^T x & \\ \text{subject to} & & \\ \text{(i)} & x(B) \geq 1 & \text{for all } B \in BL(\mathcal{F}); \\ \text{(ii)} & x_e \geq 0 & \text{for all } e \in E; \\ \text{(iii)} & x_e \text{ integral} & \text{for all } e \in E. \end{array} \tag{5.5}$$

Let P^L be the polyhedron defined by (5.5)(i), (ii). It is easy to verify that $P^L_I = P$, i.e., the integer hull of P^L is P. In particular we therefore have that $P \subseteq P^L$, but the inclusion may be strict. If $P = P^L$, we say that $\mathcal{F}$ has the **max-flow min-cut** property, and later we shall study conditions that assure this. The name comes from the fact that the clutter (described above) of st-dipaths in a graph has the max-flow min-cut property.

The LP dual to the linear relaxation of problem (5.5) is

$$\begin{array}{lll} \text{maximize} & \sum_{B \in BL(\mathcal{F})} y_B & \\ \text{subject to} & & \\ \text{(i)} & \sum_{B \in BL(\mathcal{F}): e \in B} y_B \le w_e & \text{for all } e \in E; \\ \text{(ii)} & y_B \ge 0 & \text{for all } B \in BL(\mathcal{F}). \end{array} \tag{5.6}$$

Let D^L denote the polyhedron being the feasible region of (5.6). The following proposition contains important inequalities between values of different optimization problems related to $\mathcal{F}$.

Proposition 5.2 *The following inequalities hold*

$$\begin{array}{l} \min\{w(F) : F \in \mathcal{F}\} = \\ \min\{w^T x : x \in P\} = \\ \min\{w^T x : x \in P^L, \ \ x \textit{ integral}\} \ge \\ \min\{w^T x : x \in P^L\} = \\ \max\{\sum_{B \in BL(\mathcal{F})} y_B : y \in D^L\} \ge \\ \max\{\sum_{B \in BL(\mathcal{F})} y_B : y \in D^L, \ \ y \textit{ integral}\}. \end{array} \tag{5.7}$$

Proof. This result follows from the discussion above and LP duality.

□

We see that the last two problems in Proposition 5.2 give rise to lower bounds on $v(CO) = \max\{w^T x : x \in P\}$. For instance, if y is a feasible solution of one of the two mentioned problems, then $\sum_{B \in BL(\mathcal{F})} y_B$ is a lower bound on $v(CO)$. One may interprete the integer linear programming problems in (5.7) in terms of so-called hypergraphs. A **hypergraph** is an ordered pair $H = (E, \mathcal{F})$ where E is a finite set and $\mathcal{F}$ a set system on E (a finite family of subsets of E). The elements in E are then usually called **nodes** and each $F \in \mathcal{F}$ is an **hyperedge**. A graph is obtained when all hyperedges have cardinality two. The (CO) problem is to find an hyperedge of minimum weight, where the weight of a hyperedge is the sum of the weights of its elements. Alternatively, this problem is to find a minimum weight node cover for $BL(\mathcal{F})$: choose a minimum weight node subset ($F \subseteq E$) such that each hyperedge in $BL(\mathcal{F})$ is covered. In the last problem of (5.7) we choose an integer for each B lying in the blocker $BL(\mathcal{F})$ and the constraint says that the number of times a node $e \in E$ is selected must not exceed the "capacity" w_e. Thus this problem is an edge packing problem for the clutter $BL(\mathcal{F})$. The LP problems of (5.7) are fractional versions of these two combinatorial covering/packing problems. An interesting question to study is conditions that assure (or characterize) when one or both of the inequalities in (5.7) are equalities. We shall study this problem in Section 5.3, where we give such conditions in terms of certain determinant properties of a matrix (total unimodularity).

Note that if we have equalities throughout in (5.7), a combinatorial min-max theorem is obtained: *the minimum weight of a feasible set in $\mathcal{F}$ equals the maximum cardinality of an hyperedge packing w.r.t. w.* Several important combinatorial theorems arise in this way, for some examples see Section 5.4. For a thorough discussion of theoretical results and further examples, see [26].

5.2 Integral polyhedra and TDI systems

Let $P \subseteq \mathbb{R}^n$ be a nonempty polyhedron. We define its **integer hull** P_I by

$$P_I = \text{conv}(P \cap \mathbb{Z}^n) \tag{5.8}$$

so this is the convex hull of the intersection between P and the lattice $\mathbb{Z}^n$ of integral points. Note that P_I may be empty although P is not. If P is bounded, it contains a finite number of integral points, and therefore P_I is a polytope. By the finite basis theorem for polytopes (Theorem 2.19) it follows that P_I is a polyhedron. The next result (see [35]), says that the same conclusion holds even in the unbounded case, provided that P is a rational polyhedron (that is, defined by linear inequalities with rational data).

Theorem 5.3 *If P is a rational polyhedron, then P_I is a polyhedron. If P_I is nonempty,* $\text{char.cone}(P_I) = \text{char.cone}(P)$.

Proof. Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. According to the decomposition theorem for polyhedra we have that $P = Q + C$ where Q is a polytope and $C = \text{char.cone}(P) = \{x \in \mathbb{R}^n : Ax \leq 0\}$. Choose a finite set of *integral* generators $G = \{g_1, \ldots, g_m\}$ for C so $C = \text{cone}(G)$, see Problem 5.1 and consider the set

$$M = \{\sum_{j=1}^{m} \mu_j g_j : 0 \leq \mu_j \leq 1 \text{ for } j = 1, \ldots, m\}. \tag{5.9}$$

M is a polytope, it coincides with the convex hull of the vectors $\sum_{j=1}^{m} \mu_j g_j$ where $\mu_j \in \{0, 1\}$ for $j = 1, \ldots, m$.

We shall prove that

$$P_I = (Q + M)_I + C. \tag{5.10}$$

Let p be an integral vector in P, so $p = q + c$ for some $q \in Q$ and $c \in C$. Then $c = \sum_{j \leq m} \mu_j g_j = c' + c''$ where we define $c' = \sum_{j \leq m}(\mu_j - \lfloor \mu_j \rfloor) g_j$ and

$c'' = \sum_{j \le m} \lfloor \mu_j \rfloor g_j \in C$. Then c'' is integral and $c' \in M$ (as $0 \le \mu_j - \lfloor \mu_j \rfloor \le 1$). This gives $p = (q + c') + c'' \in (Q + M)_I + C$ because $q + c' \in Q + M$ and $q + c' = p - c''$ which is integral. We have therefore shown that $P_I \subseteq (Q + M)_I + C$. Furthermore, since $Q + M \subset P$ and $C = C_I$ we obtain $(Q + M)_I + C \subseteq P_I + C_I \subseteq (P + C)_I = P_I$. This proves (5.10).

The proof can now be completed by applying the decomposition theorem for polyhedra. In (5.10) $Q + M$ is a bounded polyhedron (i.e., a polytope) and therefore $(Q + M)_I$ is polytope. Thus, by (5.10), P_I is algebraic sum of a polytope and a convex cone which proves that it is a polyhedron. Furthermore, if P_I is nonempty, we also get char.cone(P_I) = char.cone(P) (from the uniqueness of polyhedral decomposition).

□

A polyhedron is called **integral** if $P = P_I$, i.e. it equals the convex hull of its integral vectors. (For convenience, we also say that the empty polyhedron is integral.) Integral polyhedra are interesting in connection with integer linear programming. In fact, we have in general that

$$\begin{aligned} &\max\{c^T x : x \in P, x \text{ is integral}\} = \\ &\max\{c^T x : x \in P_I\} \le \\ &\max\{c^T x : x \in P\}. \end{aligned} \tag{5.11}$$

If P is integral the inequality in (5.11) can be replaced by equality, and the values of the ILP and the corresponding LP coincide. In fact, among the optimal solutions of the LP problem max $\{c^T x : x \in P\}$ there is an integral one.

Some equivalent descriptions of integral polyhedra are listed next; the proof is left for an exercise.

Proposition 5.4 *The following conditions are all equivalent whenever $P \subseteq \mathbb{R}^n$ is a nonempty polyhedron.*

(i) P is integral.

(ii) Each minimal face of P contains an integral vector.

(iii) max $\{c^T x : x \in P\}$ is attained for an integral vector for each $c \in \mathbb{R}^n$ for which the maximum is finite.

Furthermore, if P is pointed (e.g., if $P \subseteq \mathbb{R}^n_+$), then P is integral if and only if each vertex is integral.

There are some further equivalent conditions describing the integrality of a polyhedron. One such interesting condition is that the optimal *value* is integral for any LP over the polyhedron with integral objective function. This result leads to the concept of total dual integrality which gives a method for proving that a polyhedron is integral.

Proposition 5.5 *Let P be a rational polyhedron in $\mathbb{R}^n$. Then P is integral if and only if* $\max\{c^T x : x \in P\}$ *is an integer for each $c \in \mathbb{Z}^n$ such that $v(LP)$ is finite.*

Proof. In order to simplify a bit, we only prove this result for the situation with P pointed. Assume that P is integral, and consider $\max\{c^T x : x \in P\}$ with $c \in \mathbb{Z}^n$ and assume that the optimal value is finite. Then, from Proposition 5.4 this LP has an optimal solution x^* which is integral. But then clearly the optimal value $c^T x^*$ is an integer, which proves the "only if" part.

We prove the converse implication by contradiction. So assume that P is not integral. As P is pointed this means that there is a vertex $\bar{x}$ of P with a fractional component $\bar{x}_j$. Therefore, there is an $\bar{c} \in \mathbb{R}^n$ such that $\bar{x}$ is the *unique* optimal solution of max $\{\bar{c}^T x : x \in P\}$. One can then show (see Problem 5.3) that there is an $\epsilon > 0$ such that for each $c' \in B(\bar{c}, \epsilon)$ $\bar{x}$ is the unique optimal solution of the problem max $\{(c')^T x : x \in P\}$. For a suitably large positive integer s we have that $d = \bar{c} + (1/s)e_j \in B(\bar{c}, \epsilon)$. Note that $\bar{x}$ is the optimal solution of the LP problem over P with objective function $sd = s\bar{c} + e_j$ and therefore its optimal value equals $sd^T \bar{x} = s\bar{c}^T \bar{x} + \bar{x}_j$. But then one of the two optimal values $sd^T \bar{x}$ and $s\bar{c}^T \bar{x}$ must be fractional (as $\bar{x}$ is fractional), i.e., there is an integer objective function such that the corresponding optimal value is fractional; a contradiction. This completes the proof.

□

We call a linear system $Ax \leq b$ **totally dual integral**, or **TDI** for short, if for all integral c with $\max\{c^T x : Ax \leq b\}$ finite, the dual LP problem $\min\{y^T b : y^T A = c^T, y \geq 0\}$ has an integral optimal solution. Note here that this definition concerns a given *linear system* and not the corresponding polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. In fact, a polyhedron may be represented by both a TDI system and another non-TDI system. The importance of the TDI property is seen from the next result.

Corollary 5.6 *Assume that the system $Ax \leq b$ is TDI and that b is integral. Then the associated polyhedron $P = \{x : Ax \leq b\}$ is integral.*

Proof. By definition of TDI the dual problem (D) $\min\{y^T b : y^T A = c^T, y \geq 0\}$ has an integer optimal solution y^* for each $c \in \mathbb{Z}^n$ with $\max\{c^T x : Ax \leq b\}$ finite. But as b is integral we get that the optimal value $v(D) = (y^*)^T b$ is also integral for such problems. By the LP duality theorem, this shows that $\max\{c^T x : x \in P\}$ is an integer for each $c \in \mathbb{Z}^n$ such that $v(LP)$ is finite, and P is integral by Proposition 5.5.

□

Note here that the fact that $Ax \le b$ is TDI is not sufficent to guarantee that $P = \{x : Ax \le b\}$ is integral, because with fractional b the optimal value of the dual problem may be fractional (although the optimal solution is integral). It can be shown that if $Ax \le b$ is a rational linear system, then there is an integer M such that the scaled system $(1/M)Ax \le (1/M)b$ is TDI. As a consequence, we see that each rational polyhedron has a TDI representation.

An example of a TDI system is

$$\begin{aligned} x(\delta^+(U)) &\ge 1 \quad \text{for all } U \subset V,\, r \in U; \\ x &\ge 0 \end{aligned} \tag{5.12}$$

where r is a given node in a directed graph $D = (V, E)$. The solution set of this system is the dominant of the convex hull of incidence vectors of r-arborescences in D. An r-**arborescence** is an arc set F such that the subgraph (V, F) contains a directed rt-path for each node $t \ne r$ and where each such node t has exactly one ingoing arc in F. We remark that these graphs are of interest in the design of certain transportation or communication networks.

5.3 Totally unimodular matrices

In the previous section we studied the integrality of a polyhedron in connection with TDI linear systems. The topic of the present section is to study a stronger property, namely for the matrix A alone, which assures the integrality of the associated polyhedron.

An $m \times n$ matrix A is called **totally unimodular**, **TU** for short, if the determinant of each submatrix is -1, 0 eller 1. Thus, in particular, a TU matrix has entrys being -1, 0 or 1. Such matrices arise naturally in connection with graphs, see Section 5.4.

A relation between integrality of a polyhedron and total unimodularity is given next.

Theorem 5.7 *Let $A \in \mathbb{R}^{m,n}$ be a TU matrix and $b \in \mathbb{R}^m$ an integral vector. Then the polyhedron $P = \{x \in \mathbb{R}^n : Ax \le b\}$ is integral.*

Proof. First we recall Cramer's rule for finding the inverse of a nonsingular matrix C: the (j, i)'th element of C^{-1} is given by $(-1)^{i+j}\det(C_{ij})/\det(C)$ where C_{ij} is obtained from C by deleting the i'th row and the j'th column. It follows that if C is integral and $\det(C)$ is either 1 or -1, then C^{-1} will be integral.

Let $F = \{x \in \mathbb{R}^n : A'x = b'\}$ be a minimal face of P (so $A'x \leq b'$ is a subsystem of $Ax \leq b$). We may assume that the m' equations in $A'x \leq b'$ (or the rows of A') are linearly independent (otherwise we could remove redundant constraints without changing the solution set). Then A' contains at least one $m' \times m'$ nonsingular submatrix B so (after permutation of columns) we may write $A = \begin{bmatrix} B & N \end{bmatrix}$. Therefore a vector x in F is given by $x_B = B^{-1}b$, $x_N = 0$ where x_B and x_N are the subvectors corresponding to B and N, respectively. But since B is a nonsingular submatrix of A and A is TU, it follows that B must be integral and its determinant is 1 or -1. Then, by Cramer's rule, B^{-1} is integral, and therefore x_B is integral. Thus F contains an integral vector x which proves that P is integral.

□

The TU property is preserved under a number of matrix operations, for instance

- transpose;
- augmenting with the identity matrix;
- deleting a row or column which is a coordinate vector;
- multiplying a row or column by -1;
- interchanging two rows;
- duplication of rows or columns;
- adding a row to another row.

We leave the proof of these facts for an exercise.

An important connection between integrality for dual LP problems and total unimodularity is discussed next.

Corollary 5.8 *Let $A \in \mathbb{R}^{m,n}$ be a TU matrix and $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ two integral vectors. Then each of the dual LP problems in the duality relation*

$$\max\{c^T x : Ax \leq b\} = \min\{y^T b : y^T A = c^T,\ y \geq 0\}. \tag{5.13}$$

has an integral optimal solution.

Proof. From Theorem 5.7 it follows that the primal polyhedron $\{x \in \mathbb{R}^n : Ax \leq b\}$ is integral. Therefore, by Proposition 5.4, the primal LP has an integral optimal solution. The dual LP problem may be viewed as the problem

$\min\ \{b^T y : Dy \leq d\}$ where D is given by $D = \begin{bmatrix} A^T \\ -A^T \\ -I \end{bmatrix}$. and $d = \begin{bmatrix} c \\ -c \\ 0 \end{bmatrix}$.
Note that D is obtained from A by using operations that preserve the TU property (see above), so D is TU. Since d is integral, the dual polyhedron is integral and the dual LP has an integral optimal solution.

□

Corollary 5.9 *Assume that $A \in \mathbb{R}^{m,n}$ is a TU matrix. Let b, b', d, d' be integral vectors with $b \leq b'$ and $d \leq d'$ where we allow components to be either $-\infty$ or ∞. Then the polyhedron $P = \{x \in \mathbb{R}^n : b' \leq Ax \leq b,\ d' \leq x \leq d\}$ is integral.*

Proof. We have that $P = \{x : Cx \leq c\}$ where

$$C = \begin{bmatrix} A \\ -A \\ I \\ -I \end{bmatrix}, \quad c = \begin{bmatrix} b \\ -b' \\ d \\ -d' \end{bmatrix}.$$

Note that whenever a component of b, b', d, d' is $-\infty$ or ∞, the corresponding constraint is dropped. Now, C is TU as it is obtained from A by TU preserving operations and also c is integral, so P is integral according to Theorem 5.7.

□

As we have seen, polyhedra defined by a TU matrix are integral. An interesting converse result due to Hoffman and Kruskal (1956) is given next without proof.

Theorem 5.10 *Suppose that A is an integral matrix. Then A is TU if and only if the polyhedron $\{x : Ax \leq b,\ x \geq 0\}$ is integral for every integral b.*

In order to determine if a matrix is TU the following criterion due to Ghouila-Houri (1962) may be useful. For a proof, see e.g. [29].

Proposition 5.11 *A matrix $A \in \mathbb{R}^{m,n}$ is TU if and only if for each $J \subseteq \{1, \ldots, n\}$ there is a partition J_1, J_2 of J (where J_1 or J_2 may be empty) such that*

$$|\sum_{j \in J_1} a_{ij} - \sum_{j \in J_2} a_{ij}| \leq 1 \text{ for } i = 1, \ldots, m. \tag{5.14}$$

5.4 Applications: network matrices and minmax theorems

We give some basic examples of TU matrices in connection with graphs, and derive important combinatorial minmax theorems.

Consider first an (undirected) graph $G = (V, E)$ and let $A_G \in \{0, 1\}^{V \times E}$ be its node-edge incidence matrix, i.e., the column of A_G corresponding to the edge $e = [u, v]$ has only two nonzeros, namely for the two rows corresponding to u and v. Note that since A_G is an incidence matrix (all elements are 0 or 1), it *may* be totally unimodular. The precise answer is that it is TU exactly when the graph contains no odd cycles.

Proposition 5.12 *A_G is TU if and only if G is bipartite.*

Proof. Let first G be bipartite with the two color classes I_1 and I_2. We shall then show that A_G is TU using Ghouila-Houri's TU characterization. Let I be a subset of the rows of A_G, and let $I'_1 = I \cap I_1$ and $I'_2 = I \cap I_2$. Let a be the vector obtained by summing the rows of A_G associated with I'_1 and subtracting the rows associated with I'_2. The component a_e of a correponding to an edge $e = [u, v]$ must be either 1 (if one of the endnodes is in I'_1 and the other is not in I'_2), -1 (the converse situation) or 0 (if both or none of the endnodes lie in I). This a is a vector with componenents -1,0 and 1 and therefore A_G is TU according to Proposition 5.11.

Conversely, assume that G is not bipartite. It follows from Proposition 4.4 that G has an odd cycle C. Let B be the square submatrix of A_G indiced by the rows and columns corresponding to the nodes and edges of C, respectively. With suitable permutations we transform this matrix into the circulant matrix $C_{2,t}$ where t is the length of the cycle. (This is a 0,1-matrix where the i'th row, for $i = 1, \ldots, t-1$ has two nonzeros (1's) namely in position i and $i+1$, while the two nonzeros (1's) of row t are in the first and last position). One can show that $|\det(C_{2,t})| = 2$ and it follows that A_G is not TU.

□

The previous result may be combined with Corollary 5.8 to get important combinatorial minmax theorems. We illustrate this for problems concerning packing and/or covering of nodes and edges of a graph. First, we give a more "symmetric" integrality/LP theorem derived from Corollary 5.8.

Corollary 5.13 *Let $A \in \mathbb{R}^{m,n}$ be a TU matrix and $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$ two integral vectors. Then each of the dual LP problems in the duality relation*

$$\max\{c^T x : Ax \le b,\ x \ge 0\} = \min\{y^T b : y^T A \ge c^T,\ y \ge 0\}. \tag{5.15}$$

has an integral optimal solution provided that the optimal values are finite.

Proof. We apply Corollary 5.8 to the matrix $\begin{bmatrix} A & -I \end{bmatrix}$ which is TU as A is TU. The primal problem becomes max $\{c^T x : Ax \leq b,\ x \geq 0\}$ as desired. The dual problem is

$$\begin{aligned}&\min\{y^T b : y^T A - z = c^T,\ y \geq 0,\ z \geq 0\} = \\ &\min\{y^T b : y^T A \geq c^T,\ y \geq 0\}.\end{aligned}$$

Note here that there is an optimal integral solution (y, z) of the first problem if and only if there is an optimal integral solution y of the second problem. The result then follows from the duality relation.

□

Let G be a bipartite graph, so by the previous proposition, its incidence matrix A_G is TU. Consider the LP duality relation (5.15) with A replaced by A_G and with $c = \mathbf{1}$ and $b = \mathbf{1}$. We then obtain

$$\max\{\mathbf{1}^T x : A_G x \leq \mathbf{1},\ x \geq 0\} = \min\{y^T \mathbf{1} : y^T A_G \geq \mathbf{1}^T,\ y \geq 0\}. \qquad (5.16)$$

We know that each of these two problems has an integral optimal solution and we interprete this combinatorially. First, we note that in the system $Ax \leq \mathbf{1},\ x \geq 0$ we have one variable x_e for each edge $e \in E$, and that x is a solution if and only if x is nonnegative and satisfies $x(\delta(v)) \leq 1$ for each $v \in V$. Thus an integer solution here must in fact be 0,1-valued, and it represents a matching in G. A **matching** in a graph is an edge subset such that no node is incident to more than one edge. Therefore, the maximization problem in (5.16) is to find a maximum cardinality matching in the bipratite graph G. This is a classical problem which is polynomially solvable by e.g. a combinatorial algorithm called the Hungarian algorithm. Now, we turn to the minimization problem, and observe that there is only one variable x_v for each node $v \in V$. An *integer* feasible solution of this problem assigns a nonnegative integer to each node in such way that each edge has an endnode with a strictly positive integer. Clearly, we may restrict our attention to such integers that are 0 or 1, and the the minimization problem becomes: find a **node cover** in G of minimum cardinality. A node cover is a subset V_0 of the nodes such that each edge has at least one endnode in V_0. Therefore, due to integrality, the relation (5.16) says that *the maximum cardinality of a matching in a bipartite graph equals the minimum cardinality of a node cover.* This result is known as the **König-Egervary theorem** and dates to 1931.

As a second application of Corollary 5.13 we study the effect of the choice $A = A_G^T$, i.e., the transpose of the node-edge incidence matrix of G. Again we use $c = \mathbf{1}$ and $b = \mathbf{1}$. We shall assume that G contains no isolated node (otherwise one of the problems studied would become infeasible). We then obtain (when we let the role of x and y be changed)

$$\max\{\mathbf{1}^T y : A_G^T y \leq \mathbf{1},\ y \geq 0\} = \min\{x^T \mathbf{1} : x^T A_G^T \geq \mathbf{1}^T,\ x \geq 0\}. \quad (5.17)$$

We recall the interpretations above and see that an integral feasible y in the maximization problem corresponds to a **node packing** (also called **independent set** or **stable set**). Thus this problem may be viewed as to find a maximum cardinality node packing in G. A node packing is a subset S of the node set such that no pair of nodes are adjacent. As remarked above in connection with the node cover problem, one may restrict the attention to 0-1 valued solutions in the minimization problem of (5.17). The problem therefore reduces to that of finding a minimum cardinality edge cover. An **edge cover** is a subset of the edge set such that each node is incident to one of the chosen edges. The interpretation of the minmax relation of (5.17) then becomes: *the maximum cardinality of a node packing in a bipartite graph equals the minimum cardinality of an edge cover.* This result is also due to König (1933) and is often called **König's covering theorem**.

Note that all of the four combinatorial problems discussed above are polynomially solvable in bipartite graphs. In fact, it follows from the results above that they may be solved by a polynomial algorithm for LP which finds optimal vertex solutions (and such algororithms do exist, e.g., based on the ellipsoid method). There are also purely combinatorial algorithms for each of these problems that are polynomial, see e.g. [23].

We next study problems in directed graphs. Let $D = (V, E)$ be a directed graph and let A_D be its node-arc incidence matrix.

The basic tool is the following result.

Proposition 5.14 *A_D is TU for each digraph D.*

Proof. We give an elegant and short proof of this fact due to Veblen and Franklin (1921), see [35].

We prove by induction that each subdeterminant is -1, 0 or 1. Assume that this is true for all submatrices of dimension t, and let N be a $t \times t$ submatrix of A_D. If N contains a column with all zeros, then $\det(N) = 0$. If a column of N contains exactly one nonzero, this number is either -1 or 1, and when we expand the determinant for this column we get that $\det(N) \in \{-1, 0, 1\}$ by the induction hypothesis. Finally, if each column of N contains two nonzeros, the sum of all row vectors is the zero vector, so the row vectors ar linearly dependent and $\det(N) = 0$.

□

Using this fact combined with Corollary 5.13 we obtain that

$$\begin{aligned}&\max\{c^T x : A_D x = 0,\ 0 \le x \le d,\ \ x \text{ integral}\} =\\ &\min\{y^T d : z^T A_D + y \ge c^T,\ y \ge 0,\ y, z \text{ integral}\}.\end{aligned} \tag{5.18}$$

We here assume that c and d are integral vectors and allow components of d to be ∞ (in which case the corresponding dual variable is not present). An interpretation may be given as follows. Recall that $x \in \mathbb{R}^E$ is called a circulation if $A_D x = 0$. Thus the primal problem is to find a nonnegative integral circulation in D obeying arc capacities d and maximizing the "profit" $\sum_{e \in E} c_e x_e$. In the dual problem we may interprete $z$as node variables z_v for $v \in V$ and then the constraints $z^T A_D + y \ge c^T$ says that $z_u - z_v + y_e \ge c_e$ for each arc $e = uv \in E$.

Consider next the special case when the profit function is all zero except for on one arc (t, s) where $c_{ts} = 1$. In addition we let $d_{ts} = \infty$. Then the circulation problem coincides with the maximum st-flow problem. Thus we see (due to Corollary 5.13) that the maximum flow problem has an integral optimal solution, so this gives a new proof of Corollary 4.17. Furthermore, (y, z) is feasible in the dual problem if and only if (y, z) is integral, y nonnegative and

$$\begin{array}{ll} z_u - z_v + y_e \ge 0 & \text{for all } e = (u, v) \ne (t, s);\\ z_t - z_s \ge 1 & \text{for all } e \ne (t, s) c_e. \end{array} \tag{5.19}$$

Note that the last inequality is due to the fact that $d_{ts} = \infty$. Therefore $z_t \ge z_s + 1$. Define the node set $W = \{v \in V : z_v \ge z_t\}$ and note that $t \in W$ while $s \notin W$. In addition we have for each $e = (u, v) \in \delta^-(W)$ that $z_v \ge z_t$, $z_u \le z_t - 1$ (due to integrality) so from feasibility we get $y_e \ge z_v - z_u \ge z_t - z_t + 1 = 1$. Since y is nonnegative we therefore obtain $y^T d \ge \sum_{e \in \delta^-(W)} y_e d_e \ge \sum_{e \in \delta^-(W)} d_e$. This proves that the optimal value of the dual problem is no less than the capacity $d(\delta^-(W))$ of the cut $\delta^-(W)$. On the other hand the value of the primal problem is not larger than $d(\delta^-(W)$. But from (5.18) the values of these two problems coincide, and it follows that there exists an st-cut with capacity equal to the maximum flow from s to t. Thus we have given a new proof of the max-flow min-cut theorem Theorem 4.16. Note that the proof in Section 4.5 used combinatorial ideas, while the present proof was based on total unimodularity of incidence matrices of digraphs combined with LP duality.

5.5 Exercises

Problem 5.1 *Let $C \subseteq \mathbb{R}^n$ be a convex cone. Show that there is a finite set G of rational vectors generating C so $C = \text{cone}(G)$.*

Problem 5.2 *Show that the set M as defined in (5.9) is equal to the convex hull of the vectors $\sum_{j=1}^{m} \mu_j g_j$ where $\mu_j \in \{0, 1\}$ for $j = 1, \ldots, m$.*

Problem 5.3 *Let $\bar{x}$ be a vertex of a polyhedron $P \subset \mathbb{R}^n$. Show that there is an $c \in \mathbb{R}^n$ and an $\epsilon > 0$ such that for each $c' \in B(c, \epsilon)$ $\bar{x}$ is the unique optimal solution of the LP problem max $\{(c')^T x : x \in P\}$.*

Problem 5.4 *Prove that the operations mentioned in the beginning of Section 5.3 all preserve the TU property.*

Problem 5.5 *Prove Proposition 5.4.*

Chapter 6

Methods

In this chapter we discuss several methods for actually solving (numerically) combinatorial optimization and integer linear programming problems. This is a very active research area and many interesting techniques are available. The general integer linear programming problem is *NP*-hard, so it is quite unlikely that one can find an efficient algorithm (polynomial running time) for this problem. However, the class of integer linear programs is very large and contains important subclasses with specific properties that algorithms may and should exploit. It is therefore important to have different basic algorithmic principles at hand in order to develop a suitable algorithm for a problem of interest. We shall give a brief presentation of some main techniques that are commonly used for solving integer linear programming and combinatorial optimization problems.

The main focus is on **polyhedral methods**, or **cutting plane methods**. These methods are based on approximating a certain "target" integral polyhedron P by simpler polyhedra containing P. But we also present some other methods, heuristics and Lagrangian relaxation, and for the latter we relate it to the polyhedral approach. Finally, as an important example, we consider the famous Traveling Salesman Problem.

A very good reference book for various methods in integer linear programming and combinatorial optimization is [29]. For algorithms in network flows, see [33].

6.1 From integer programming to linear programming

We consider a general integer linear programming problem (ILP) of the following form:

$$
\begin{array}{ll}
\text{maximize} & c^T x \\
\text{subject to} & \\
\text{(i)} & Ax \le b; \\
\text{(ii)} & x \text{ is integral.}
\end{array}
\tag{6.1}
$$

Let S be the feasible set of this problem, i.e.,

$$S = \{x \in \mathbb{Z}^n : Ax \le b\} = P \cap \mathbb{Z}^n \tag{6.2}$$

where $P = \{x \in \mathbb{R}^n : Ax \le b\}$. We are also interested in the **LP-relaxation** (LP) of (ILP) which is given by

$$\max\{c^T x : x \in P\}. \tag{6.3}$$

Clearly we have $v(ILP) \le v(LP)$ since $S \subseteq P$, but unfortunately this inequality may be strict. As a simple (and rather extreme) example, let n be a natural number and consider max $\{x_1 : x_1 \ge 0,\ x_2 \ge (1/n)x_1,\ x_2 \le 1 - (1/n)x_1,\ x \text{ is integral}\}$ (draw a picture!). The optimal value is zero and optimal solutions are $(0,0)$ and $(1,0)$. The (unique) optimal solution of the LP relaxation is $(n/2, 1/2)$ so the optimal value is $n/2$. Thus, the bound obtained from the LP relaxation is indeed very bad. As discussed below (using polyhedral theory) we can, at least theoretically, close this gap by adding other linear inequalities. In practice, it is usually impossible to close this gap completely, but we shall discuss principles that often lead to very good upper bounds on the optimal value. In addition to finding bounds like this we also need to explain how one can find feasible solutions that are near-optimal.

Recall the results in Section 5.2. The integer hull P_I of P is defined as the convex hull of the integer points in P, i.e., $P_I = \text{conv}(S)$. Since the objective function $c^T x$ is linear, (ILP) is equivalent to the optimization problem max $\{c^T x : x \in P_I\}$. The crucial fact is that this problem is a linear programming problem. This is a consequence of Theorem 5.3 which says that P_I is a rational polyhedron (recall that we assume that P is a rational polyhedron). Thus, there is a (rational) linear system $\bar{A}x \le \bar{b}$ such that $P_I = \{x \in \mathbb{R}^n : \bar{A}x \le \bar{b}\}$. Furthermore, the original problem (ILP) has been "reduced" to the linear programming problem

$$\max\{c^T x : \bar{A}x \le \bar{b}\}. \tag{6.4}$$

This might be confusing since we know that integer linear programming is *NP*-hard while linear programming is polynomial. However, our reduction merely says that there *exists* such a linear system $\bar{A}x \leq \bar{b}$, so, e.g., it may be much larger than the original system $Ax \leq b$. In fact, when we formulate combinatorial problems as integer programs these new systems $\bar{A}x \leq \bar{b}$ often (usually) grow exponentially as a function of the natural size of the original problem. For instance, for the matching problem in a graph with n nodes there are 2^n facet defining inequalities for the matching polytope P_I (the convex hull of incidence vectors of matchings). For the Traveling Salesman Problem the most common integer linear programming formulation contains even an exponential number of inequalities in the formulation (subtour elimination constraints) and several other, and larger (!) classes of facet defining inequalities are known for the Traveling Salesman Polytopes, see [22].

Thus one problem when using this reduction of (ILP) to (LP) for some practical problem is that the number of new inequalities may become very large. A more basic problem is *how to find* these inequalities, both in theory and practice.

Concerning the first question, it turns out that one may still optimize fast over a polyhedron although the number of inequalities is large (e.g., exponential relative to the input size). The main thing is how fast we can *separate* for these inequalities, i.e., finding for a given point a violated inequality of that class, or proving that no such inequality exists. Moreover, if this separation problem can be solved polynomially, then the optimization problem is also polynomial. This is a main result in the recent theory of linear programming over implicitly described polyhedra using the ellipsoid method, see [26].

The problem of *finding* additional inequalities will be discussed in the next section. These techniques are important because although we may not find a complete system $\bar{A}x \leq \bar{b}$ as above (defining P_I), even a partial system may produce good bounds on the optimal value and lead to good approximate solutions.

In this Chapter we omit some proofs of theorems in order to keep the length of the presentation limited. For proofs of similar results, confer [29] or [35].

6.2 Finding additional valid inequalities

We discuss some methods for finding classes of valid inequalities for a set of integral points. Ideally we would like to have methods for going from a polyhedron P to its integer hull P_I. We shall mention a theoretical result which says that this is indeed possible although the construction is far from practical. Thus we shall also discuss other general techniques that are applicable to any integer linear programming problem.

First we fix some terminology. Let S be some subset of $\mathbb{R}^n$ and let $a^T x \leq \alpha$ be an inequality with $a \neq 0$. We say that $a^T x \leq \alpha$ is **valid** for S if $S \subseteq H_{\leq}(a, \alpha) = \{x \in \mathbb{R}^n : a^T x \leq \alpha\}$, i.e., each point of S satisfies the inequality in question. We use the notation $a_{i,\cdot}$ to denote the i'th row of $A \in \mathbb{R}^{m,n}$ viewed as a column vector. Similarly, $a_{\cdot,j}$ denoted the j'th column of A.

The Chvátal-Gomory procedure.

Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ be a given polyhedron in $\mathbb{R}^n$ with A, b rational. We are interested in the problem of finding valid inequalities for the integer hull P_I. Clearly, each inequality in $Ax \leq b$ is valid for P_I, and the purpose is to provide methods for finding additional ones. The basic idea to be discussed is based on the simple fact that if an integral number x satisfies $x \leq \alpha$, then it also satisfies $x \leq \lfloor \alpha \rfloor$. This strengthening of the inequality is called **integer rounding**.

Geometrically, as P_I is spanned by the integer points in P, what we need is some procedure for "pushing" a hyperplane defined by an inequality in the system $Ax \leq b$ as far as possible towards P_I. Ideally, we would like to push until we meet a point in P_I, but in higher dimensions this may not be so easily achieved. What we can do in stead is to push the hyperplane until an integer point is met (although this point is outside P). For instance, if $x_1 + 2x_2 \leq 10/3$ is a valid inequality for P, then the "rounded inequality" $x_1 + 2x_2 \leq \lfloor 10/3 \rfloor = 3$ is also valid for P_I. Note that none of the parallel hyperplanes defined by the inequalities $x_1 + 2x_2 \leq \gamma$ contain integral points for $3 < \gamma \leq 10/3$. Two simple algebraic facts are that (i) multiplying a valid inequality by a positive number gives another (equivalent) valid inequality, and that (ii) the sum of valid inequalities is again a valid inequality. We remark that these properties may be expressed in terms of convexity as follows. Let $\bar{P}$ be the subset of $\mathbb{R}^{n+1}$ consisting of points (a, α) for which $a^T x \leq \alpha$ is a valid inequality for P. The mentioned algebraic properties of valid inequalities simply mean that $\bar{P}$ is a convex cone in $\mathbb{R}^{n+1}$. If we combine the rounding idea with the cone property of the valid inequalities we get the following procedure for finding a valid inequality. For simplicity, we assume that $P \subseteq \mathbb{R}^n_+$, i.e., that $x_j \geq 0$ for $j \leq n$ are valid inequalities for P. Multiply the i'th inequality $a_{i,\cdot}^T x \leq b_i$ by λ_i for each $i \leq m$ and sum all the inequalities. This gives the new inequality $(\sum_{i \leq m} \lambda_i a_{i,\cdot}^T) x \leq \sum_{i \leq m} \lambda_i b_i$. If we let $\lambda = (\lambda_1, \ldots, \lambda_m)^T$ this new inequality is $(\sum_{j \leq n} \lambda^T a_{\cdot,j}) x_j \leq \lambda^T b$. This inequality is redundant (as it is implied by $Ax \leq b$), but now we may apply integer rounding. Thus we see that $\sum_{j \leq n} \lfloor \lambda^T a_{\cdot,j} \rfloor x \leq \lambda^T b$ is valid for P as we assumed that each $x \in P$ satisfies $x \geq 0$. But if we insert an integral point x in P on the left-hand-side of this inequality we get an integer (all numbers are then integers). Thus we may round the right-hand-side down to the nearest integer and still have a valid inequality, namely

$$\sum_{j \leq n} \lfloor \lambda^T a_{\cdot,j} \rfloor x \leq \lfloor \lambda^T b \rfloor. \tag{6.5}$$

The procedure leading to (6.5) is called the **Chvátal-Gomory procedure** and we also call (6.5) a **Chvátal-Gomory inequality**.

Note that the Chvátal-Gomory procedure may be applied repeatedly and thereby generating gradually larges classes of valid inequalities. How far may we reach by doing this? A remarkable fact is that *every* valid inequality for P may be generated in this way (possibly by increasing the right-hand-side) provided that suitably many repetitions are taken. More specifically, for each linear system $Ax \leq b$ defining the polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ we can find a finite family of linear systems $A^{(k)}x \leq b^{(k)}$ for $k = 0, \ldots, t$ such that (i) $A^{(0)} = A$, $b^{(0)} = b$, (ii) each inequality in $A^{(k)}x \leq b^{(k)}$ is derived from $A^{(k-1)}x \leq b^{(k-1)}$ using the Chvátal-Gomory procedure for $k = 1, \ldots, t$, and (iii) $P_I = \{x \in \mathbb{R}^n : A^{(t)}x \leq b^{(t)}\}$. For a proof of this result, see [35] or [29].

As an example we consider the matching problem. A **matching** in a graph $G = (V, E)$ is a subset M of the edge set E such that $d_{(V,M)}(v) \leq 1$ for each node $v \in V$, i.e., each node is the endnode of at most one edge in M. One can check that the set of integral vectors satisfying $0 \leq x \leq 1$ and $x(\delta(v)) \leq 1$ for each $v \in V$ coincides with the set of incidence vectors to matchings in G. Let $Ax \leq b$ denote this linear system. It defines a polyhedron which is called the **fractional matching polytope**. We have that P_I is the **matching polytope**, i.e., the convex hull of the incidence vectors of perfect matchings. Let $S \subset V$ consist of an odd number of nodes, say $|S| = 2k + 1$. Then the inequality $x(E[S]) \leq k$ is clearly valid for P_I as each matching contains no more than k pairs of nodes in S. Thus, the validity is due to a simple combinatorial argument. However, this inequality may also be obtained using the Chvátal-Gomory procedure on the original system $Ax \leq b$. Consider the inequalities $x(\delta(v)) \leq 1$ for $v \in S$ and the inequalities $-x_e \leq 0$ for $e \in \delta(S)$. If we multiply each of these inequalities by $1/2$ and add them together, we get the (valid) inequality $x(E[S]) \leq k + 1/2$. By integer rounding we get the desired inequality $x(E[S]) \leq k$ which proves that this inequality may be obtained by the Chvátal-Gomory procedure applied to the system consisting of the degree inequalities and simple bounds. The matching polytope is completely described by the degree inequalities, simple bounds and the odd set inequalities $x(E[S]) \leq \lfloor |S|/2 \rfloor$ for $S \subseteq V$ and S odd. Thus, all the facet defining inequalities for the matching polytope are obtained by adding "one round of cutting planes" and we say that the original polyhedron P has Chvátal-Gomory-rank 1.

A feature of the Chvátal-Gomory procedure is that it may be affected by scaling of the inequalities. For instance, if $P = \{x \in \mathbb{R} : x \leq 3/2\}$ then $P_I = \{x \in \mathbb{R} : x \leq 1\}$ and the inequality $x \leq 1$ is obtained by integer rounding from $x \leq 3/2$. However, P is also the solution set of the (scaled) inequality $2x \leq 3$, and rounding directly on this inequality does not change the inequality. From this we realize that integer rounding should be preceded by a proper scaling of the inequality, i.e., dividing by the greatest common divisor of all the numbers involved. For a single inequality this produces the integer hull as the next result says. The proof is left to the exercises.

Proposition 6.1 *Let $P = \{x \in \mathbb{R}^n : \sum_{j=1}^n a_j x_j \leq \alpha\}$ where all the a_j's are integers. Let d be the greatest common divisor of $a_1, \ldots, a_n$. Then $P_I = \{x \in \mathbb{R}^n : \sum_{j=1}^n (a_j/d) x_j \leq \lfloor \alpha/d \rfloor\}$.*

Boolean implications.

Sometimes one can find new inequalities by detecting logical (boolean) implications of one or more constraints of the original system.

From Chapter 0 we recall the **knapsack problem** max $\{\sum_{j=1}^n c_j x_j : \sum_{j=1}^n a_j x_j \leq b,\ 0 \leq x \leq 1\}$ where all the data are positive integers. Let $P = \{x \in \mathbb{R}^n : \sum_{j=1}^n a_j x_j \leq b,\ 0 \leq x \leq 1\}$, so P_I is the **knapsack polytope**. As a specific example let $n = 3$, $a_1 = 3$, $a_2 = 3$, $a_3 = 2$ and $b = 7$. Let $C = \{1, 2, 3\}$ and note that $a(C) = \sum_{j \in C} a_j = 8 > b$. We call C a **dependent set** or **cover**. Since $a(C) > b$, no feasible integral solution in P can have all variables in C equal to 1. Thus $x(C) \leq |C| - 1$ is a valid inequality for P_I which is often called a **cover inequality**. These inequalities have proved to be very useful for solving several different combinatorial optimization problems since cover inequalities may be derived for individual constraints in the integer linear programming formulation. We also remark that for the polyhedron P above (the linear relaxation of the knapsack problem) all the vertices may be described in a simple way, see Problem 6.2.

Note that *any* linear inequality in 0-1 variables may be transformed to the situation just treated. A general linear inequality with rational data may be written (after suitable scaling) $\sum_{j \in J_1} a_j x_j + \sum_{j \in J_2} a_j x_j \leq b$ with a_j, $j \in J_1$ positive integers and a_j, $j \in J_2$ negative integers (j's with $a_j = 0$ are not of interest for the analysis). We make the affine transformation $z_j = 1 - x_j$ for $j \in J_2$, and the transformed inequality is $\sum_{j \in J_1} a_j x_j + \sum_{j \in J_2} (-a_j) z_j \leq b - \sum_{j \in J_2} a_j$. Here all coefficients are positive, so we may establish (e.g.)

cover inequalities as above and finally transform these back into new valid inequalities for the original problem.

Another type of constraint that is often met in applications is

$$\begin{array}{ll} \text{(i)} & \sum_{j \in N} y_j \leq nx; \\ \text{(ii)} & 0 \leq x \leq 1, \; 0 \leq y \leq 1; \\ \text{(iii)} & x \in \{0, 1\}. \end{array} \tag{6.6}$$

The logical contents of the constraints is that $x = 1$ whenever the sum of the continuous variables $y_j, j \in N$ is positive. However, this means that all the inequalities $y_j \leq x$ for $j \in N$ are also valid for the solution set of (6.6). By adding these new constraints we cut off fractional solutions (from the continuous relaxtion of (6.6)) like e.g., $y_1 = 1$, $y_j = 0$ for $j \in N \setminus \{1\}$, $x = 1/n$ since the inequality $y_1 \leq x$ is violated. For more about related mixed integer sets like the variable upper-bound flow models, confer [29].

Combinatorial implications.

For combinatorial polyhedra, i.e., polyhedra with vertices corresponding to some class of combinatorial objects, one may find valid inequalities by exploiting these combinatorial properties. The example with the matching polytope given in the previous paragraph fits into this framework. Here we give some other examples.

First, we consider the **set covering problem** (see e.g., [6]). This problem is of relevance to many applications. For instance, in airline crew scheduling each flight must be covered; in allocating student classes to rooms, each class must get a room, or in network design a number of capacity "bottlenecks" (like cuts) must be covered. Let I and J be the color classes of a bipartite graph G, so each edge in the edge set E joins some node in I to some node in J. Let c_j for $j \in J$ be given nonnegative weights. By a **cover** we understand a subset S of J such that each node in I is adjacent to at least one node in S. (Of course, we assume that G allows this to happen, i.e., each node in I is adjacent to some node in J). The set covering problem is to find a cover of minimum weight, where the weight $w(S)$ of a cover S is defined as $w(S) = \sum_{j \in S} w_j$. This problem is *NP*-hard. The polyhedral approach to this problem may start by introducing the **set covering polytope** P_{SC} as the convex hull of vectors χ^S where S is a cover. An integer linear programming formulation of the set covering problem is obtained by letting x_j indicate whether node j is in the cover to be determined. For $i \in I$ we let $\Gamma(i)$ denote the set of nodes in J that are adjacent to i.

$$
\begin{array}{lll}
\text{minimize} & \sum_{j \in J} c_j x_j & \\
\text{subject to} & & \\
\text{(i)} & \sum_{j \in \Gamma(i)} x_j \geq 1 & \text{for all } i \in I; \\
\text{(ii)} & 0 \leq x_j \leq 1 & \text{for all } j \in J; \\
\text{(iii)} & x \text{ is integral.} &
\end{array}
\tag{6.7}
$$

Thus the constraint (6.7)(i) assures that each node in I is covered. Let P be the solution set of (6.7)(i)–(ii), so the integer hull P_I of this polytope is precisely the set covering polytope P_{SC}. Due to the hardness of the set covering problem, it is to ambitious to find a complete linear description of P_I. However, in order to numerically solve practical set covering problems one may need to find some of these inequalities and add them to the description of P. In other words, the LP relaxation using P may give poor lower bounds on the true optimal value of the set covering instance of interest. For example, consider a graph with nodes $I = \{i_1, i_2, i_3\}$ and $J = \{j_1, j_2, j_3\}$, and with the six edges $[i_1, j_1], [i_1, j_2], [i_2, j_2], [i_2, j_3], [i_3, j_3], [i_3, j_1]$. Note that each node in J covers two consecutive nodes in I. Assume that the objective function is $c = (1, 1, 1)$. Then an (in fact, *the*) optimal solution of the LP relaxation is $\bar{x} = (1/2, 1/2, 1/2)$ with objective value 3/2. The optimal value of the integer program, and therefore the set covering problem, is 2. Thus, some valid inequality for P_{SC} is required to cut off this fractional vertex of P. Such an inequality may be deduced from a simple combinatorial argument: no single node in J can cover all the nodes in I, therefore at least two nodes in J must be chosen. Thus the inequality $x_1 + x_2 + x_3 \geq 2$ is valid for P_{SC} (as it holds for all vertices and then, by convexity, for all points of P_{SC}). Also, it is violated by $\bar{x}$. If we add this inequality to our current linear program and reoptimize we get an optimal integer solution, say $(1, 1, 0)$. The inequality we just identified actually belongs to a large class of valid, and often nonredundant, inequalities for set covering polytopes: the **odd cover inequalities**. It may seem that we only get a few inequalities in this way, but for a given graph G there may be many subgraphs that are isomorphic to the one of our example, and each of these produce several valid inequalities called **lifted odd cover inequalities**. The procedure involved is called **lifting** and makes it possible to find facets of a higher dimensional polytope via facets of lower dimensional ones (namely projections of the polytope of interest), see [29].

As another example we consider the **node packing problem**. A node packing (**independent set, stable set**) in a graph $G = (V, E)$ is a subset S of the nodes such that no pair of nodes is adjacent. The **node packing**

polytope (see [26]) is the convex hull P_{NP} of the incidence vectors of node packings in G. Note that this polytope depends on G. A binary vector $x \in \mathbb{R}^V$ is the incidence vector of a node packing iff $x_u + x_v \leq 1$ for each $[u, v] \in E$. Thus $P_{NP} = P_I$ where $P = \{x \in \mathbb{R}^V : 0 \leq x \leq 1, \; x_u + x_v \leq 1 \text{ for each } [u, v] \in E\}$. It can be shown that $P = P_I$ iff the graph G is bipartite. Thus, for general graphs further inequalities are needed to define the node packing polytope. For instance, consider a clique which is a complete subgraph, i.e., a subset V_0 of V such that $[u, v] \in E$ for all distinct $u, v \in V_0$. Clearly any node packing contains at most one node in such a clique, so the **clique inequality** $x(V_0) \leq 1$ is valid for P_{NP}. Note that this inequality is stronger than the inequalities $x_u + x_v \leq 1$ for $u, v \in V_0$. This means that each of these inequalities is implied by the clique inequality and the nonneativity constraints. Next, consider an odd cycle C with, say, $2k + 1$ nodes. Then at most every second node can lie in a node packing, so exploiting the parity property we get the valid inequality $x(C) \leq k$. A major research topic in polyhedral combinatorics is the study of those graphs for which the the clique constraints and the nonnegativity constraints are sufficient to describe the node packing polytope, see [29], [23]. Such graphs are called **perfect graphs**.

6.3 Relaxations and branch-and-bound

A relaxation of an optimization problem is obtained by enlarging the set of feasible solutions. This is done to get an optimization problem which is easier to solve and thereby obtain a bound on the optimal value of interest. We discuss relaxations in general and give a generic algorithm for integer linear programming problems based on that concept. Later we shall use relaxations in cutting plane algorithms and Lagrangian relaxation.

Consider a general optimization problem (Q) max $\{f(x) : x \in A\}$ where $A \subseteq \mathbb{R}^n$ and $f : A \to \mathbb{R}$. If $A \subseteq B \subseteq \mathbb{R}^n$ and $g : B \to \mathbb{R}$ is such that $g(x) \geq f(x)$ for each $x \in A$, we say that the optimization problem (R) max $\{g(x) : x \in B\}$ is a **relaxation** of (Q). We then have that $v(R) = \sup_{x \in B} g(x) \geq \sup_{x \in A} g(x) \geq \sup_{x \in A} f(x) = v(Q)$. Thus, by solving (R) in stead of (Q) we obtain an upper bound on the optimal value in (Q). We may also find an approximate optimal solution. For a real number $\epsilon \geq 0$ we say that a point $\bar{x} \in A$ is an **ϵ-optimal** solution of (Q) if $f(\bar{x}) \geq v(Q) - \epsilon$. Assume now that $\bar{x} \in A$ is such that $f(\bar{x}) \geq v(R) - \epsilon$. Then $\bar{x}$ is an ϵ-optimal solution of (Q) because $v(R) \geq v(Q)$. This provides a tool for estimating the quality of a feasible solution and also gives a possible stopping criterion. Note that one often uses relaxations where $f = g$ so only the feasible sets differ.

If (S) is a relaxation of (R) and (R) a relaxation of (Q), then (S) is also a relaxation of (Q) ("transitivity of relaxations"). We then say that (R) is a **finer relaxation** of (Q) than (S) is, and then we have $v(S) \geq v(R) \geq v(Q)$. Consider an integer linear programming problem (Q) max $\{c^T x : x \in P,\ x \text{ integral}\}$. The relaxations of interest in cutting plane algorithms are LP problems of the form max $\{c^T x : x \in P'\}$ where $P_I \subseteq P' \subseteq P$.

A generic relaxation algorithm for the integer linear programming problem (Q) max $\{c^T x : x \in P,\ x \text{ integral}\}$ may be described as follows. Again we let $S = P \cap \mathbb{Z}^n$. During the algorithm z_L and z_U are lower and upper bounds on $v(Q)$, respectively.

General relaxation algorithm.

Step 1. (Initialization) Set $z_L = -\infty$ and $z_U = \infty$ and choose an initial relaxation (R^0) max $\{f^0(x) : x \in S^0\}$ of (Q). Set $k = 0$.

Step 2. (Optimality test.) Solve the relaxation (R^k) and let x^k denote an optimal solution. If $x^k \in S$ and $c^T x^k = v(R^k)$ (remark: if $f^k(x) = c^T x$, then this last condition is automatically fullfilled), then x^k is an optimal solution of (Q). Otherwise, set $z_U = v(R^k)$.

Step 3. (Refinement.) If $x^k \in S$, set $z_L := \max\{z_L, c^T x^k\}$. Let (R^{k+1}) max $\{f^{k+1}(x) : x \in S^{k+1}\}$ be a finer relaxation of (Q) than R^k; set $k := k+1$ and go to Step 2.

Some suitable termination criterion should be added to this algorithm, e.g., by introducing a maximum number of iterations. Note that a straightforward modification of the algorithm finds an ϵ-optimal solution of (Q) in stead. An important property of the relaxation algorithm is that the upper bounds are monotonically decreasing, i.e., $z_U^k \geq z_U^{k+1}$ where the superscript indicates iteration number. Many specific algorithms fall into this relaxation framework by suitable spesification of how the refinement of the current relaxation is determined. Note that the relaxation algorithm could be applied to any optimization problem, not just integer programming.

Relaxations are often combined with an enumerative approach. To describe the idea we consider a 0-1 linear programming problem, i.e., an integer linear program where all variables are binary, say (Q) max $\{c^T x : x \in S\}$ where $S = P \cap \{0,1\}^n$ and $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. Note that S is a finite set. Let $S(u), u \in V$ be a partition of S, so these sets are pairwise disjoint and their union equals S. Then we have that

$$v(Q) = \max v(Q(u))$$

where $(Q(u))$ is the restricted problem max $\{c^T x : x \in S(u)\}$. Also, by solving the restricted problems and comparing we can find an optimal solution of (Q). A natural way of partitioning S is to fix variables to either 0 or 1. For instance, we can let S^0 (S^1) be those vectors in S with $x_1 = 0$ ($x_1 = 1$). Furthermore, one can establish partitions recursively by fixing new variables based on the previous partition. For instance, let $S^{0,0}$ ($S^{0,1}$) be those $x \in S$ with $x_1 = 0$ and $x_2 = 0$ ($x_1 = 0$ and $x_2 = 1$). For simplicity in the presentation we shall assume that there is a predetermined ordering of the variables which is followed when variables are fixed.

The recursive partitioning may be organized in an **enumeration tree** with nodes correponding to subsets of S and edges indicating subdivision. The nodes are partitioned into n layers; in the k'th layer the first k variables are fixed. Thus there are 2^k nodes in the k'th layer correpond to all possible ways of fixing the k first variables. For each node the subset $S(v)$ of S consists of those vectors in S that satisfy the variable fixing specified in that node. The restricted optimization problem $\max \{c^T x : x \in S(u)\}$ associated with node u will be denoted by $(Q(u))$. The single node v_r in layer 0 is called the **root node** and each of the nodes in the n'th layer are called bottom nodes. When u is a bottom node the problem $(Q(u))$ is trivial as all variables are fixed. The edges go between two consecutive layers. More precisely, a "mother node" on layer k have two adjacent nodes ("children") in layer $k + 1$ and they are obtained from the variable fixing in the mother node by fixing x_{k+1} to either 0 or 1. The bottom nodes represent a complete enumeration of all the feasible solutions in the problem (Q).

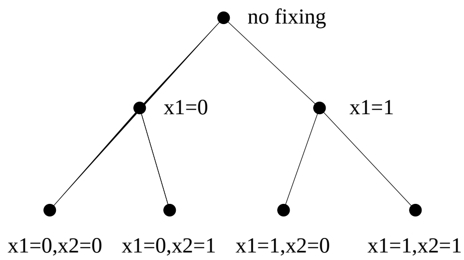

Figure 6.1: A branch-and-bound tree

Thus even for moderate values of n the enumeration tree is too large for practical purposes. The key idea is that we may only need to consider a small part of this tree in order to solve the problem. For a nonroot node u the tree contains a unique path $P(u)$ between u and the root node. For each nonroot node $w \in P(u)$ we say that u is *below* w or that w is *above* u. Under certain conditions a node u may be **pruned** which means that we need not solve any of the problems $(Q(u'))$ for nodes u' that are below u. This happens when we know that these problems can not improve on the current best solution. The following pruning criteria should be clear.

Proposition 6.2 *A node u in the enumeration tree may be pruned if one of the following conditions holds.*

(i) (Infeasibility.) $S(u)$ is empty.

(ii) (Optimality.) An optimal solution of $Q(u)$ is known.

(iii) (Value dominance.) $v(Q(u)) \leq v(Q)$.

One way of deciding if node u can be pruned is therefore to solve the restricted program $Q(u)$. This is usually not desirable as $Q(u)$ is a difficult (although smaller-dimensional) integer program. In stead one may solve a (much simpler) relaxation of $Q(u)$. This leads to the following pruning criteria. We let $R(u)$ denote a relaxation of $Q(u)$, and (for simplicity) assume that the objective functions of these problems coincide.

Proposition 6.3 *A node u in the enumeration tree may be pruned if one of the following conditions holds.*

(i) (Infeasibility) $R(u)$ is infeasible.

(ii) (Optimality.) We have an optimal solution $\bar{x}$ of $R(u)$ satisfying $\bar{x} \in S(u)$.

(iii) (Value dominance.) $v(R(u)) \leq z_L$ where z_L is the objective function for some point in S.

We are mainly interested in the case when all the relaxations $R(u)$ are LP relaxations, i.e., they are obtained by dropping the integrality constraints in the integer linear programming problems. Then we know from LP duality that the value $v(R(u))$ may be found by solving the LP dual $D(u)$ of $R(u)$ as we have $v(D(u)) = v(R(u))$ (provided that at least one of the problems is feasible). Therefore, if $D(u)$ is unbounded, then $R(u)$ must be infeasible and the node u may be pruned. Secondly, if we find a feasible solution $\bar{y}$ of $D(u)$ with (dual) objective value which is no greater than z_L, then we may prune due to value dominance (as this implies that $v(D(u)) \leq z_L$).

This leads to an enumerative algorithm based on linear programming. It consists in processing nodes (solving associated optimization problems) in the enumerative tree. We never process a node before all the nodes above that node have been processed. In the algorithm V_n is a list of nodes that remains to be processed. The algorithm is called **branch-and-bound** as we branch in the nodes and determine bounds on the optimal value in each of the nodes.

Branch-and-bound algorithm. *Step 1. (Initialization) Let $V_n = \{v_r\}$, $z_L = -\infty$ and $z_U = \infty$.*

Step 2. (Termination.) If $V_n = \emptyset$, the current best solution x^* is optimal; terminate.

Step 3. (Node selection and processing.) Select a node u in V_n and set $V_n := V_n \setminus \{u\}$. Solve the LP relaxation $R(u)$, possibly via solving the dual $D(u)$. Let $z(u)$ and $x(u)$ denote the optimal value and an optimal solution of $R(u)$, respectively.

Step 4. (Pruning.) (i) If $z(u) \leq z_L$, go to Step 2. (ii) If $x(u) \notin S(u)$, go to Step 5. (iii) If $x(u) \in S(u)$ and $c^T x(u) > z_L$, update the best solution by setting $x^* = x(u)$ and $z_L = c^T x(u)$. Update V_n by removing all nodes with $z_U(v) \leq z_L$. If $c^T x(u) = z_U$, go to Step 2; otherwise go to Step 5.

Step 5. (Division.) Add two new nodes u_0 and u_1 to V_n each being a child of node u such that $S(u_0)$ and $S(u_1)$ is a partition of $S(u)$. Let $z_U(u_0) = z_U(u_1) = z(u)$ and go to Step 2.

Division often consists in branching on a fractional variable. For instance, if the optimal solution found in node u has two fractional variables x_1, x_4 one selects one of these, say x_4, and introduces the new node u_0 with the additional constraint $x_4 = 0$ and another new node u_1 with the additional constraint $x_4 = 1$. There are other natural ways of introducing new partitions as well, see [29].

One important point is the strength of the LP relaxations. That is, if the LP (optimal value) bounds are too far away from the optimal integer value one cannot prune the enumeration tree and the algorithm becomes slow, maybe too slow for all practical purposes, see Exercise 6.3.

Two main issues in the development of branch-and-bound algorithms are *node selection* and *variable selection*.

Whenever we have solved the relaxation $R(u)$, and we do not terminate, we have to select the next node to be processed (also called the next active node). In this node selection problem several strategies exist, some are based on a priori rules and others are adaptive (depending on the calculations). For instance, a common strategy is *breadth-first-search plus backtracking* where one always chooses the next node as a child node and backtracks if the node is pruned. In the variable selection problem one decides how to make the partitioning that determines the (two or more) children problem. Empirically one knows that this choice affects the over-all speed of the algorithm, but still it is hard to find good rules for selecting "critical variables". A useful strategy is to predetermine some ordering of the variables based on the coupling in the constraints. For instance, the variables may fall into two classes such that fixing all the variables in the first class makes the remaining ones integral. In such a case it makes sense to branch on fractional variables in the first class whenever possible. Other possible techniques are discussed in [29].

6.4 Cutting plane algorithms

We discuss a general method for solving integer linear programming problems called a **cutting plane algorithm**. The idea is to add cutting planes to a linear program, i.e., valid inequalities, in order to approximate the underlying integer program.

Historically, an early version of this class of algorithms was the Gomory cutting plane algorithm from around 1960 (see [29] for a presentation of this work as well as references to the original papers). It is a finitely convergent algorithm that generates a new inequality from a fractional solution using the information of the linear programming tableau (basis description). The convergence speed of Gomory's algorithm is not good, so it is not practical for solving large problems. For integer programs coming from combinatorial optimization a main difficulty is that a huge number of constraints are needed to describe the convex hull of the integer points of interest. In a very important paper ([9]) by Dantzig, Fulkerson and Johnson in 1954 they solved some instances of the traveling salesman problem by adding just a few of the facet defining inequalities of the underlying polytope. They actually solved, to *proven optimality*, a TSP problem of finding the shortest round trip through 48 cities in the United States. This line of research was not persued at that time, probably because one did not see how to overcome the problem of the large number of inequalities and also one did not have mathematical theory for describing traveling salesman polytopes.

A new interest for cutting plane algorithms arose in the seventies and eighties with, in particular, the work of M. Padberg and M. Grötschel. This work was based on the work of Dantzig, Fulkerson and Johnson and also the recent new ellipsoid method for solving linear programming problems. A key feature of the ellipsoid method is that one does not need all the inequalities of a linear program explicitly, but may treat them implicitly by generating them "on the fly". More precisely, in each iteration of the ellipsoid method one must find, if any, a violated inequality for the current solution in order to proceed. Such an inequality is then used for "halving" an ellipsoid surrounding the feasible region, a new smaller ellipsoid is constructed and its center is the new candidate solution. Thus, it was observed that the ellipsoid method (discovered by Khachian in 1979) was excellent for handling problem with a large number of constraints. This was just the thing needed for integer programs!

The new approach to cutting plane algorithms consisted in finding theoretically a class of strong valid inequalities for an integral polytope *before* the computations started, and then constructing separation algorithms for testing whether some current solution violates one of these inequalities. Thus, the principle is to combine problem specific inequalities with the general framework of separation and optimization. We shall explain further how this is done.

Consider an integer linear programming problem (Q) max $\{c^T x : x \in P,\ x \text{ integral}\}$ where $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. Let S be the feasible solution, i.e., $S = P \cap \mathbb{Z}^n$. We shall only consider relaxations having the same objective function as (Q). In iteration k we consider a polyhedron P^k in $\mathbb{R}^n$ and these polyhedra are nested in the sense that

$$P_I \subseteq \ldots \subset P^{k+1} \subset P^k \subset \ldots \subset P^0 = P.$$

The polyhedron P^k may be viewed as the current approximation of the target polyhedron P_I. In each iteration we optimize over the current polyhedron and add, if needed, more inequalities to obtain the next polyhedron. For simplicity we assume that none of the LP's involved have unbounded solutions for the given objective function. Let Π be a (finite) class of valid inequalities for P_I that are known. Ideally, this class is a complete linear description of P_I, but, more frequently, it gives only a partial description of P_I.

Fractional cutting-plane (FCP) algorithm.

Step 1. (Initialization) Set $z_U = \infty$, $A^0 = A$, $b^0 = b$ and $k = 0$.

Step 2. (Optimization.) Solve the LP problem max $\{c^T x : A^k x \leq b^k\}$ and let x^k be an optimal solution. If x^k is integral, it is an optimal solution of (Q); terminate.

Step 3. (Separation.) Test if x^k satisfies all the inequalities in Π. If it does, terminate. Otherwise, let $A^{k+1}x \leq b^{k+1}$ be the system obtained by adding one or more violated inequalities in Π to the current system $A^k x \leq b^k$. Let $k := k + 1$ and go to Step 2.

The FCP algorithm is finite as the number of inequalities that can be added is finite. Note that, unless Π defines P_I, we are not guaranteed to find an *integer* solution by this algorithm. We may very well end up with a fractional optimal solution which then satisfies all the inequalities in the original system $Ax \leq b$ and in Π, but not all the inequalities needed to define P_I. A possibility is then to proceed by using branch and bound method with the last LP problem as the problem of the root node. If, however, we obtained an integral optimal solution in Step 2, we have found an optimal solution of (Q) (since it is optimal in a relaxation having the same objective function, see Section 6.3). In any case we have an upper bound on the optimal value of (Q): $v(Q) \leq c^T x^N$ where x^N is the final solution.

A natural theoretical measure of the success of the FCP algorithm for a problem instance is the **optimality gap** $c^T x^N - v(Q)$. Since $v(Q)$ is usually unknown (this is what we want to find!), one may in stead consider an upper bound Δ on the gap. The number Δ may then be used after the computations to evaluate the algorithm empirically. Another useful idea is to evaluate Δ in each iteration of the algorithm and terminate if Δ lies below some chosen level. This is often done as follows. In each iteration we use some algorithm (a heuristic) to produce a feasible solution $\hat{x}^k \in S$. Define $\Delta = c^T x^k - c^T \hat{x}^k$ as the desired upper bound on the optimality gap.

Consider the separation stage of the FCP (Step 3). For simplicity we identify the inequalities $\pi^T x \le \pi_0$ in Π with the vectors $(\pi, \pi_0) \in \mathbb{R}^{n+1}$. The separation problem is to check if the current solution $\bar{x}$ satisfies

$$\pi^T \bar{x} \le \pi_0 \text{ for all } (\pi, \pi_0) \in \Pi. \tag{6.8}$$

and, if it does not, find at least one violated inequality. This problem may be viewed as the optimization problem

$$\max\{\pi^T \bar{x} - \pi_0 : (\pi, \pi_0) \in \Pi\}. \tag{6.9}$$

The difficulty of this optimization problem is clearly dependent on the properties of the class Π. For instance, assume that Π together with $Ax \le b$ determines P_I, i.e., $P_I = \{x \in \mathbb{R}^n : Ax \le b,\ \pi^T x \le \pi_0 \text{ for all } (\pi, \pi_0) \in \Pi\}$. Then the the integer program (Q) is *NP*-hard (resp. polynomial) if and only iff the problem (6.9) is *NP*-hard (resp. polynomial). In practice, the class Π naturally falls into subclasses $\Pi^1, \Pi^2, \ldots$ where each class consists of inequalities having a similar structure. Them it may happen that the separation problem is easy for one class and difficult for another. An interesting point is that although the separation problem for a class of inequalities is *NP*-hard, there may exist a larger class of valid inequalities (i.e., containing the first class) for which the separation problem is polynomial. Ideally, one would like to find large classes of inequalities that both give rid to good bounds (on the optimal value) and also with a tractable separation problem. Of course, this may be difficult or impossible, but at least these two aspects should be taken into account in the construction of a cutting plane algorithm.

A remark on complete descriptions of integral polyhedra is in order. Usually, in practice, one can not find a complete description of P_I, but it is easy to find a valid integer linear programming formulation (this gives P). Possibly, one can strengthen this using some of the methods described in Section 6.2 and end up with an additional class of inequalities Π. A priori it is difficult to say much about the strength of the relaxtion obtained in this way, so it makes sense to implement a cutting plane algorithm and judge from some numerical experiments. If the lower bounds are bad for small-scale problems (that can be solved using branch-and-bound), one should look for other classes of inequalities. The fractional optimal solutions obtained may then be useful in "guessing" on such inequalities. In fact, looking at the structure of the fractional variables one may discover, e.g., a combinatorial property that all the integer points in P have, but such that the linear inequality that expresses this property is violated by the fractional solution. Such a empirical approach in solving specific problems makes sense, because this tends to lead to inequalities that are "relevant", i.e., they may be active in an optimal solution.

We comment on relations between a cutting plane algorithm and branch-and-bound. The simplest way of combining these methods is to use the last LP from FCP (assuming that it had a fractional optimal solution) as the root LP in the branch-and-bound algorithm. An alternative, and more powerful, approach is to integerate the cutting plane approach with branch-and-bound into what is known as **branch-and-cut**. Briefly, this approach is to follow a branch-and-bound scheme, but in each of the nodes of the enumeration tree one also generates cutting planes. Thus, one runs a FCP algorithm in the active node. Usually this is done by separation w.r.t. classes of inequalities that are valid for the main polyhedron of interest P_I, as this produces inequalities that are valid in all the nodes and they can therefore be used in the remaining process. [Remark: since the integer program in a tree-node contains variable fixing, there are further equalities and, possibly, inequalities that define that integer polyhedron. However, it is complicated to exploit this information constructively in the branch-and-cut, although it may be done in theory.]

A good introduction on some practical aspects of polyhedral methods is of [31] or [20]. There is a vast literature on the polyhedral approach to specific combinatorial problems, see e.g., [31], [17], [7], [30], [36].

6.5 Heuristics

A heuristic is an approximate algorithm, i.e., an algorithm that "solves" an optimization problem by finding a solution that may not be optimal, but hopefully good enough for some purpose. The idea is that what one gives away concering optimality shall be gained in terms of the speed of the algorithm. Since many combinatorial problems are *NP*-hard, heuristics are often used for such problems.

Typically a heuristic for a certain combinatorial optimization problem is designed by specializing general algorithmic principles to capture the specific structure of the problem studied. Consider a combinatorial optimization problem of the form (CO) max $\{w(F) : F \in \mathcal{F}\}$ where $\mathcal{F}$ is a class of feasible subsets of the finite ground set E and w is a nonnegative weight function defined on E. A heuristic for this problem may be divided into two stages: (i) finding an initial feasible solution, and (ii) improving the solution by local search techniques.

Finding an initial feasible solution. This may be difficult, depending on the problem characteristics. Many problems, however, has a certain monotonicity property that helps. We say that $\mathcal{F}$ is an **independence system** if each subset of a feasible set is also a feasible set. For instance, the set of matchings in a graph is an independence system. Another example is the set of all independent sets in a graph. We should remark that any set system (class of subsets of E) can be made into an independent set by adding all subsets of feasible sets; this is called **monotonization**. This transformation may, at least, be useful in the analysis of the problem. For instance, in the Traveling Salesman Problem, we may consider the class of edge sets being subsets of Hamilton cycles. Or in the Steiner problem (find a minimum weight tree, called a Steiner tree, that connects given terminal nodes in a graph) we can consider the subsets of Steiner trees as the feasible solutions.

For an independence system the **greedy algorithm** may be used to construct a feasible solution:

Greedy algorithm

Step 1. Order the elements in E in a sequence $e_1, \ldots, e_m$ such that $w(e_1) \geq \ldots \geq w(e_m)$. Set $F := \emptyset$.

Step 2. While $F \in \mathcal{F}$ add the next element in the sequence to F.

This algorithm terminates with a feasible solution, and it is simply called the **greedy solution**. It can be shown (see [32]) that this solution is optimal if and only if the independence system is also a matroid. This applies, for instance, to the spanning tree problem, see Chapter 4. For general independence systems, the greedy solution may be far from optimal so further improvements are needed, see below. The greedy idea may also be used for finding a feasible solution in an integer linear programming problem by first solving the LP relaxation and then systematically rounding fractional variables up or down (depending on the structure of the feasible region).

Improving the solution. Say that a feasible solution $F \in \mathcal{F}$ has been constructed. Then we may try to improve this solution by making "small" modifications that preserve feasibility, but increases the total weight. One way of doing this is to apply **k-interchanges** which is to replace k elements in F by k elements outside F provided that the new solution is feasible and better. This simple idea (with small adjustments) could be tailored to any combinatorial optimization problem. By applying such an improvement routine several times one gets a sequence of gradually better solutions. For the Traveling Salesman problem the k-interchange heuristic of Lin-Kernighan ([22]) has proved to be a very good heuristic (for $k = 2$ or 3), it usually gives near-optimal solutions withing reasonable times for interesting problems. For $k = 2$ one has a current tour (Hamilton cycle) T and swaps a pair of edges uv and wz in T with the new edges uw and vz when this gives a better tour. In this case one can test fast if the swap gives a new tour, while this may become too time-comsuming for larger values of k (because there are many ways of "glueing" together the different pieces of a tour obtained after the edge removal). The k-interchange idea is an example of **local search**. Briefly, local search consist in searching through some specified neighbourhood of the current solution for a better one. If one finds such a solution the search continues in the neighbourhood of the new solution etc. Eventually, one ends up with a solution that is locally optimal. Note that this notion of local optimality is relative to the chosen neighborhood system. Thus, different heuristics are distinguished by their neighborhood systems.

We also briefly mention another general idea that may lead to good heuristics for integer linear programming problems. It is based on complementary slackness. One then considers the LP relaxation (LP) max $\{c^T x : Ax \leq b\}$ of the integer program and its dual (D) min $\{y^T b : y^T A = c^T,\ y \geq 0\}$. Assume that we have a feasible solution y in (D); it may have been constructed using a greedy algorithm. By complementary slackness, assuming that y is optimal, we know that an optimal solution x in (LP) must satisfy $a_i^T x = b_i$ for all row indices i with $y_i > 0$. These constraints may guide us in choosing a primal solution x that pair up with y. And, of course, we make sure that the solution x is integral. Similarly, we may use complementary slackness to construct a dual solution y from a primal solution x. Therefore, we may combine ascent heuristics in the integer program or (LP), decent heuristics in (D) to get another heuristic for the integer program using dual information. One nice feature of this approach is that one obtains both upper and lower bounds on the optimal value. This idea has been very successful on the simple plant location problem, see [29] for a further elaboration of the details in this method.

Heuristics may be compared by a performance measure which is the worst possible ratio between the true optimal value and the value of the final solution found by the heuristic. This ideal measure can then be estimated to get some idea about the quality of the method.

6.6 Lagrangian relaxation

In Section 6.3 we discussed relaxations of optimization problems in a general setting. We here consider one specific type of relaxation that has turned out to be of great value in finding near-optimal solutions to (several) combinatorial optimization problems. The idea in Lagrangian relaxation is to exploit the underlying structure of an optimization problem in order to produce bounds on the optimal value.

Consider the 0-1 linear programming problem with feasible set $S = P \cap \mathbb{Z}^n$ where $P = \{x \in \mathbb{R}^n : Ax \leq b,\ 0 \leq x \leq 1\}$ and $A \in \mathbb{R}^{m,n}$. (The following development also works more generally, in fact for $S = P \cap X$ where X is any subset of $\mathbb{R}^n$). Assume that the system $Ax \leq b$ is split into two subsystems $A^1x \leq b^1$ and $A^2x \leq b^2$ where A^i has m^i rows and $m^1 + m^2 = m$. One can think of $A^2x \leq b^2$ as "complicating constraints" in the sense that if they were dropped an easier problem would be obtained. Thus we have $P = \{x \in \mathbb{R}^n : A^1x \leq b^1,\ A^2x \leq b^2,\ 0 \leq x \leq 1\}$. The 0-1 linear programming problem (Q) may be written as follows.

$$\begin{array}{ll} \text{max} & c^T x \\ \text{subject to} & \\ \text{(i)} & A^1 x \leq b^1; \\ \text{(ii)} & A^2 x \leq b^2; \\ \text{(iii)} & 0 \leq x \leq 1; \\ \text{(iii)} & x \text{ is integral.} \end{array} \tag{6.10}$$

The purpose of this constraint splitting is to open up for an associated and simpler 0-1 LP problem where the constraints $A^2x \leq b^2$ have been moved to the objective function with penalties. We consider the following problem $LR(\lambda)$

$$\begin{array}{ll} \text{max} & c^T x + \lambda^T(b^2 - A^2 x) \\ \text{subject to} & \\ \text{(i)} & A^1 x \leq b^1; \\ \text{(ii)} & 0 \leq x \leq 1; \\ \text{(iii)} & x \text{ is integral.} \end{array} \tag{6.11}$$

where $\lambda = (\lambda_1, \ldots, \lambda_{m^2})$ consists of nonnegative weights or "penalties", usually called the **Lagrangian multipliers**. Thus, in $LR(\lambda)$ a feasible point $\bar{x}$ may violate a constraint $a_i^T x \leq b_i$ in $A^2x \leq b^2$ but this increases the objective function by the amount of $\lambda_i(b_i - a_i^T \bar{x})$. On the negative side, we see that we get an "award" in the objective by satisfying an inequality strictly. This is an unavoidable problem when we want to maintain a linear objective function.

We call the problem $LR(\lambda)$ the **Lagrangian relaxation** (or **Lagrangian subproblem**) w.r.t. the constraints $A^2x \le b^2$. As the name indicates this Lagrangian relaxation is really a relaxation of (6.10) for any $\lambda \in \mathbb{R}^m_+$. To see this we note that the feasible region of the Lagrangian relaxation contains the feasible region of the original problem. In addition, if $\bar{x}$ is feasible in (6.10), then, in particular, we have $A^2\bar{x} \le b^2$ and therefore also $c^T\bar{x} + \lambda^T(b^2 - A^2\bar{x}) \ge c^T x$ as λ is nonnegative. Thus we obtain an upper bound on the optimal value of interest:

$$v(Q) \le v(LR(\lambda)).$$

Since this holds for all $\lambda \in \mathbb{R}^n_+$, the best upper bound obtained in this way is given by solving the so-called **Lagrangian dual problem** (LD) (w.r.t. $A^2x \le b^2$)

$$\min\{v(LR(\lambda)) : \lambda \ge 0\} \tag{6.12}$$

and we get the important inequalities

$$v(Q) \le v(LD) \le v(LR(\lambda)) \text{ for all } \lambda \in \mathbb{R}^n_+. \tag{6.13}$$

The Lagrangian dual problem may be viewed as a nondifferentiable convex minimization problem as $v(LR(\lambda))$ is a piecewise linear and convex function (it is the pointwise minimum of a finite number of affine functions). Algoritmically one tries to solve the Lagrangian dual problem by some kind of multiplier adjustment technique. The basic principle is to adjust the multiplier according to the current optimal solution x. If x violates the constraint, the penalty (multiplier) is increased, but if x satisfies the constraint, the penalty is decreased. Different ideas are used for deciding *how much* these adjustments should be, and for this good stategies are problem dependent. For a discussion of one such general technique, called the *subgradient method*, see [29].

We consider an application which illustrates the idea of Lagrangian relaxation.

The **degree-constrained spanning tree problem** (DCST) is to find a minimum weight spanning tree satisfying given degree constraints. More specifically, let w be a nonnegative weight function defined on the edges of a graph $G = (V, E)$ and let b_v for $v \in V$ be given positive integers. We want to find a spanning tree T satisfying the degree constraints $d_T(v) \le b_v$ for each $v \in V$ and with minimum total weight $w(T) = \sum_{e \in T} w_e$. (Of course, this problem is infeasible if the b_v's are "too small"). This problem is known to be *NP*-hard, see [12]. But we know that the spanning tree problem is tractable, i.e., polynomial, and this can be exploited as follows. It is possible to write down a linear system $A^1x \leq b^1$ with 0-1 solutions that correspond to the incidence vectors of sets $F \subseteq E$ such that (V, F) contains a spanning tree. Finding such a system is left as an exercise, but here we only need to know that the system exists. Then our problem (DCST) may be written as (6.10) with $c = -w$ and the system $A^2x \leq b^2$ being

$$x(\delta(v)) \leq b_v \quad \text{for all } v \in V. \tag{6.14}$$

The Lagrangian relaxation w.r.t. the degree constraints (6.14) is essentially a spanning tree problem. The objective function to be minimized in this problem is

$$w^T x + \sum_{v \in V} \lambda_v (x(\delta(v)) - b_v).$$

This means that the weight of a spanning tree T becomes (use $x = \chi^T$)

$$-\sum_{v \in V} \lambda_v b_v + \sum_{[u,v] \in T} (w_{uv} + \lambda_u + \lambda_v).$$

This objective will therefore tend to give spanning trees having low degrees. The Lagrangian relaxation can for each λ be solved by, e.g., Kruskal's algorithm. Thus, combined with a suitable multiplier technique we can solve the Lagrangian dual and obtain a lower bound on the optimal value of the DCST problem. Also, if we are lucky and find an optimal spanning tree in the final Lagrangian relaxation which satisfies all the degree constraints (6.14), then this solution is also an optimal solution of (6.14). Otherwise, one usually constructs a feasible spanning tree solution by some kind of heuristic method based on the last subproblem. This also produces a bound on the optimality gap.

We return to the general theory of Lagrangian relaxation. Consider again the Lagrangian relaxation w.r.t. $A^2x \leq b^2$ given in (6.11). The objective function $c^T x + \lambda^T(b^2 - A^2x) = (c^T - \lambda^T A^2)x + \lambda^T b^2$ is an affine function of x, i.e., a linear function $c(\lambda)x$ (with $c(\lambda) := c^T - \lambda^T A^2$) plus some constant. Since the constant may be removed from the optimization, the problem (6.11) consists in maximizing a linear function over the 0-1 vectors in the polyhedron defined by $A^1x \leq b^1$. As discussed in the introduction to this chapter we may convexify such a problem and obtain an equivalent LP problem

$$\max\{c(\lambda)^T x : x \in P_I^1\}$$

where P_I^1 is the integer hull of the polyhedron $P^1 = \{x \in \mathbb{R}^n : A^1x \leq b^1,\ 0 \leq x \leq 1\}$. Thus the Lagrangian relaxation corresponds to "integralization" with respect to the system $A^1x \leq b^1,\ 0 \leq x \leq 1$ and translating the objective function with some linear combination of the row vectors in A^2. To proceed, it follows from Motzkin's theorem that $P_I^1 = \mathrm{conv}(\{x^k : k \in K\})$ where x^k, $k \in K$ is a finite set of vectors in $\mathbb{R}^n$; these are the vertices of P_I^1. Therefore we obtain

$$
\begin{aligned}
&v(LD) = \min\{v(LR(\lambda)) : \lambda \geq 0\} = \\
&\min_{\lambda\geq 0}[\max_{x\in P_I^1}(c^T - \lambda^T A^2)x + \lambda^T b^2] = \\
&\min_{\lambda\geq 0}[\max_{k\in K}(c^T - \lambda^T A^2)x^k + \lambda^T b^2] = \\
&\min_{\lambda\geq 0}[\min\{\eta : \eta \geq (c^T - \lambda^T A^2)x^k + \lambda^T b^2,\ \text{for all } k \in K\}] = \\
&\min\{\eta : \lambda \geq 0,\ \eta + \lambda^T(A^2x^k - b^2) \geq c^T x^k,\ \text{for all } k \in K\} = \\
&\max\{c^T \sum_{k\in K} \mu^k x^k : A^2 \sum_{k\in K} \mu^k x^k \leq b^2 \sum_{k\in K} \mu^k x^k; \sum_{k\in K} \mu^k = 1; \mu \geq 0\} = \\
&\max\{c^T x : A^2 x \leq b^2,\ x \in P_I^1\}.
\end{aligned}
$$

The second last equality was obtained using linear programming duality. Note also the transformation used for converting the inner maximization problem into an LP problem of minimizing an upper bound. We have therefore shown the following result.

Theorem 6.4 *The Lagrangian dual problem (6.12) may be viewed as the dual of the LP problem* $\max\{c^T x : A^2 x \leq b^2,\ x \in P_I^1\}$. *In particular, the optimal values of these problems coincide.*

This result is the main result on Lagrangian relaxation. It says that the bound obtained from solving the Lagrangian dual equals the one obtained by the LP problem with feasible set is based on integralization only w.r.t. the constraints that are not relaxed. Define the three polytopes

$$
\begin{array}{lcl}
P_I^{1,2} & := & \{x \in \mathbb{R}^n : 0 \leq x \leq 1,\ A^1 x \leq b^1,\ A^2 x \leq b^2\}_I; \\
(P_I^1)^2 & := & \{x \in \mathbb{R}^n : 0 \leq x \leq 1,\ A^1 x \leq b^1\}_I \cap \{x \in \mathbb{R}^n : A^2 x \leq b^2\}; \\
P^{1,2} & := & \{x \in \mathbb{R}^n : 0 \leq x \leq 1,\ A^1 x \leq b^1,\ A^2 x \leq b^2\}.
\end{array}
\tag{6.15}
$$

Maximizing $c^T x$ over $P_I^{1,2}$ correponds to the original integer program (more precisely, transformed into an LP); maximizing over $(P_I^1)^2$ corresponds to solving the Lagrangian dual and, finally, maximizing over $P^{1,2}$ is simply the LP relaxation of the integer program (6.10). Let LP denote the last program. Since we have

$$P_I^{1,2} \subseteq (P_I^1)^2 \subseteq P^{1,2}$$

we get the following ordering of the optimal values in these optimization problems

$$
\begin{aligned}
&v(Q) = \max\{c^T x : P_I^{1,2}\} \leq \\
&\max\{c^T x : (P_I^1)^2\} = v(LD) \leq \\
&\max\{c^T x : P^{1,2}\} = v(LP).
\end{aligned}
$$

This means that the Lagrangian bound may improve on the bound coming from the LP relaxation. Note, however, an important consequence of Theorem 6.4 concerning the bounds.

Corollary 6.5 *If the polyhedron P^1 is integral, i.e., has integral vertices, then $v(LD) = v(LP)$.*

Thus, if integrality may be dropped in the Lagrangian subproblems (i.e., $\{x \in \mathbb{R}^n : A^1 x \leq b^1\}$ is integral), then we will not improve compared to the bound obtained by solving the original LP relaxation. Usually, in such cases, Lagrangian relaxation is not used unless it is viewed as more practical than solving the LP. However, if the polyhedron is not integral, then, depending on the objective function, the value of the Lagrangian dual will improve on the LP relaxation bound. Consequently, this should be taken into account when deciding on the splitting on the constraints, so as to make the Lagrangian subproblems "simple, but not too simple".

6.7 The Traveling Salesman Problem

In order to exemplify some of the principles and methods presented above, we discuss the Traveling Salesman Problem (TSP) in this section. Our presentation is very brief. A more detailed presentation is given in [29] and, of course, in the "TSP-book" [22].

The TSP has been studied a lot during the last 50 years by mathematicians and computer scientists. The problem is of interest in certain real-world applications, like vehichle routing and computer chip production, but it has also an attraction from a theoretical point of view. One reason is that it is easy to formulate, but difficult to solve!

The TSP problem is to find a shoutest trip through a given set of cities. More precisely, we have given an undirected graph $G = (V, E)$ with nonnegative weights (costs) on the edges: c_e, $e \in E$. A **tour** is a Hamilton cycle, i.e., a cycle going through all the nodes of G and passing through each node exactly once. The length of a tour is the sum of the weights of its edges. The problem is to find a shortest tour. The TSP is *NP*-hard, and even deciding if a graph contains a Hamilton tour is *NP*-complete. Even worse, the problem is also hard to approximate as it is *NP*-hard to solve the TSP with any given performance guarantee. That is, for *any* given positive number r it is "hard" to find a tour of length at most $(1 + r)$ times the optimal length. However, some special cases of the TSP are polynomially solvable, see [22].

Heuristics.

Many heuristics have been developed for the TSP. We present some basic ones. These heuristics may not work for all graphs as some kind of density of the edges is required, but here we just assume that such properties are present. We have already mentioned the k-interchange heuristic for improving a tour. Thus we here concentrate on methods for generating an initial tour.

First, we give some methods based on the greedy principle. The **greedy feasible** algorithm gradually builds up a tour by adding the cheapest edge to a partial solution F so as to maintain the properties that F is acyclic and each node in (V, F) has degree at most 2. The **nearest neighbor** heuristic extends a path $P : v_1, \ldots, v_j$ by a new node v_{j+1} with $[v_j, v_{j+1}]$ shortest possible. Initially, a single node is selected (e.g., endnode of a cheapest edge) and finally the tour is constructed by adding the edge $[v_{n-1}, v_1]$. (This last edge may have high cost!). In the **nearest insertion** algorithm one extends a cycle by instering a new node between two adjacent nodes on the current cycle. What all these heuristics have in common is that they are simple, but normally do not produce good solutions.

If the costs satisfy the **triangle inequality**

$$c_{uw} \leq c_{uv} + c_{vw}$$

many algorithms perform better. In particular, in this case, we have a more sophisticated heuristic called the **double spanning tree heuristic**. As a preparation we need some graph teory. An **euler tour** in a graph is a walk with the same start and endnode and containing each edge of the graph exactly once. If a graph contains an euler tour it is called **eulerian**. It was shown by Leonard Euler (in 1736, "the first paper of graph theory", [11]) that a graph is eulerian if and only if it is connected and each node has even degree.

The **double spanning tree heuristic** works as follows. First, we find a minimum weight spanning tree in G. Make a copy of each of the edges in a tree, so we get a graph G' (with multiple edges) having $2(n - 1)$ edges; G' is called a **double spanning tree**. Let each new edge have the same cost as its original edge. Note that G' is **eulerian** as it is connected and each node in V has even degree due to the edge copying. Now we can easily find an euler tour in the double spanning tree G'. Furthermore, from this euler tour we can construct a Hamilton tour by moving along the edges and avoid node repetitions by going to the next node. Note that such node jumps will never increase the length compared to going along the double spanning tree due to the triangle inequality. This leads to the following result.

Theorem 6.6 *If the weights of G are nonnegative and satisfy the triangle inequality, then a tour found by the double spanning tree heuristic is no longer than twice the length of a minimum spanning tree.*

We remark that a rather similar algorithm which also uses perfect matchings has the better performance guarantee of 3/2. (Note that this only applies to the case of nonnegative weights satisfying the triangle inequality; without this assumption, any algorithm "fails").

Integer linear programming formulation.

The following model is a valid integer linear programming formulation of the TSP:

$$
\begin{array}{lll}
\min & \sum_{e\in E} c_e x_e & \\
\text{subject to} & & \\
\text{(i)} & x(\delta(v)) = 2 & \text{for all } v \in V; \\
\text{(ii)} & x(\delta(W)) \geq 2; & \text{for all } W \subset V,\ W \neq \emptyset,\ W \neq V; \\
\text{(iii)} & 0 \leq x \leq 1; & \\
\text{(iv)} & x \text{ is integral.} &
\end{array}
\tag{6.16}
$$

The contraints (i), (iii) and (iv) assures that a solution x is of the form $x = \chi^F$ where $F \subset E$ and $d_{(V,F)}(v) = 2$ for each $v \in V$; such a set F is called a $\mathbf{2-}$**matching** (or 2-factor). Clearly, every tour is a 2-matching, so all these inequalities are valid. However, in general a 2-matching is a union of disjoint cycles, so it may not be a tour. The **2-connectivity inequalities** (ii) eliminate such a possibility, i.e., a 2-matching that satisfies (ii) is a tour (otherwise we could let W be the node set of one subtour, and we would get a violated inequality). From this it is not difficult to see that the feasible solutions in (6.16) are precisely the incidence vectors of tours, see Problem 6.4.

An equivalent set of constraints that can replace (6.16)(ii) is the set of **subtour elimination constraints**:

$$x(E[S]) \leq |S| - 1 \text{ for all } W \subset V,\ W \neq \emptyset,\ W \neq V.$$

These constraints were introduced in the 1954 paper by Dantzig, Fulkerson and Johnson [9]. Note that the number of 2-connectivity inequalities, or subtour elimination constraints, grows exponentially as a function of the number of nodes.

Relaxations.

A simple relaxation of (6.16) is obtained by removing the 2-connectivity constraints (6.16)(ii) and using the same objective function $c^T x$. This relaxation can be transformed into a matching problem in a bipartite graph (the assignment problem) and therefore solved efficiently. This relaxation may be combined with a suitable branch-and-bound scheme with partitioning that assures the 2-connectivity. This approach is called the **assignment problem/branch-and-bound algorithm**.

Consider the graph G and choose a node, say node 1. A **1-tree** in G is a set $F \cup \{e, f\}$ where $(V \setminus \{1\}, F)$ is a spanning tree and e and f are two edges incident to node 1. Observe that each Hamilton tour is a 1-tree, but the converse inclusion is false. The important property of 1-trees is that they are connected, and that the following characterization of tours is valid: F is a tour if and only if it is both a 2-matching and a 1-tree. Now, the incidence vectors of 1-trees are the feasible solutions of the following set of constraints:

$$\begin{array}{lll} \text{(i)} & x(\delta(v)) = 2 & \text{for } v = 1; \\ \text{(ii)} & x(E[W]) \leq |W| - 1 & \text{for all } W \subset V \setminus \{1\},\ |W| \geq 3; \\ \text{(iii)} & x(E) = |V|; & \\ \text{(iv)} & 0 \leq x \leq 1; & \\ \text{(v)} & x \text{ is integral.} & \end{array} \tag{6.17}$$

If we here add the degree constraints $x(\delta(v)) = 2$ for $v \in V \setminus \{1\}$, we get the (incidence vectors of) tours as solutions. In stead of doing this, we relax these degree constraints using Lagrangian relaxation. We introduce Lagrangian multipliers λ_v (that are unrestricted in sign, as we have equalities) where $\lambda_1 = 0$ and get the following Lagrangian relaxation:

$$2\sum_{v \in V} \lambda_v + \min\{\sum_{uv \in E} (c_{uv} - \lambda_u - \lambda_v)x_{uv} : x \text{ is the incidence vector of a 1-tree}\}.$$

Let $v_{1T}(\lambda)$ be the optimal value of this problem. The Lagrangian dual is then (LD) max $\{v_{1T}(\lambda) : \lambda \in \mathbb{R}^V, \lambda_1 = 0\}$ with optimal value $v(LD)$. Based on Corollary 6.5 it can be shown that z_{LD} equals the value of the linear programming relaxation of (6.16). To solve the Lagrangian subproblems one needs to find a minimum weight 1-tree. This is done by solving a minimum spanning tree problem in the subgraph $G \backslash \{1\}$ and then one finds the shortest pair of edges incident to node 1. The updating of the multipliers to solve the Lagrangian dual is done so that λ_v is increased if the degree of node v is 1 in the 1-tree, and decrease λ_v is the degree is larger than 2.

A polyhedral approach.

A polyhedral approach to the TSP may be based on the integer hull P_I of the polyhedron P defined by the linear inequalities in (6.16). Thus P_I, or P_{TSP} as we denote it, is the convex hull of the incidence vectors of tours, and it is called the **Traveling Salesman Polytope**. It can be shown that most of the inequalities defining P also define facets of P_{TSP}. We first describe some further facet defining inequalities and then briefy discuss some main components in a cutting plane algorithm for solving the TSP.

It is convenient to assume that G is the complete graph. This is a common technical assumption (often used in polyhedral combinatorics) which simplifies polyhedral arguments. Then the dimension of P_{TSP} is $m-n$ where $n = |V|$ and $m = |E| = n(n-1)/2$. (This is not trivial to show, but the easy thing is that $\dim(P_{TSP}) \leq m - n$ since the n degree inequalities are linearly independent.) We also assume that $n \geq 4$ (otherwise P_{TSP} is either empty or consists of one point only.) The subtour elimination constraints define facets for P_{TSP} whenever the node set W satisfies $2 \leq |W| \leq \lfloor n/2 \rfloor$, see e.g., [29] or [17].

If n is either 4 or 5, the Traveling Salesman Polytope is completely described by the trivial bounds, degree constraints and subtour elimination constraints. For larger number of nodes, other facets come into play. One such large class of inequalities is descibed next.

A **comb** in G is a class of sets H, T_i for $i \leq k$ all being subsets of the node set V and satisfying

- $H \cap T_i$ is nonempty for $i = 1, \ldots, k$;
- $T_i \setminus H$ is nonempty for $i = 1, \ldots, k$;
- the sets T_i for $i \leq k$ are pairwise disjoint;
- $k \geq 3$ is an odd number.

The set H is called the **handle** of the comb, and the T_i's are the **teeth**. Associated with each comb is a valid inequality for P_{TSP} which may be derived using the Chvátal-Gomory procedure as follows. Consider the valid

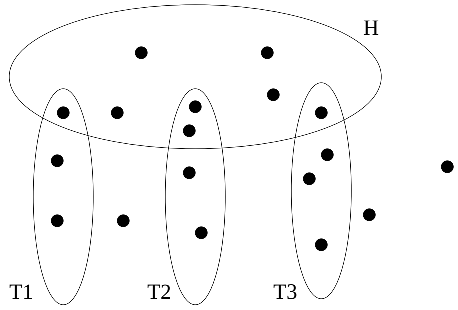

Figure 6.2: A comb

inequalities

$$\begin{array}{lll} \text{(i)} & x(\delta(v)) = 2 & \text{for all } v \in H; \\ \text{(ii)} & x(E[T_i]) \leq |T_i| - 1 & \text{for } i = 1, \ldots, k; \\ \text{(iii)} & x(E[T_i \setminus H]) \leq |T_i \setminus H| - 1 & \text{for } i = 1, \ldots, k; \\ \text{(iv)} & x(E[T_i \cap H]) \leq |T_i \cap H| - 1 & \text{for } i = 1, \ldots, k; \\ \text{(iv)} & -x_e \leq 0 & \text{for } e \in \delta(H) \setminus \cup_i E[T_i]. \end{array} \tag{6.18}$$

If we add all these inequalities and divide the result by 2, we get the inequality

$$x(E[H]) + \sum_{i=1}^{k} x(E[T_i]) \leq |H| + \sum_{i=1}^{k} (|T_i| - 1) - k/2.$$

Now, due to integrality since k is odd, we may round the right-hand-side down and still have a valid inequality for P_{TSP}. This gives the **comb inequality**

$$x(E[H]) + \sum_{i=1}^{k} x(E[T_i]) \leq |H| + \sum_{i=1}^{k} (|T_i| - 1) - (k+1)/2. \tag{6.19}$$

These inequalities were introduced by Grötschel and Padberg (1979) as a generalization of the **simple comb inequalities** found by Chvátal; simple combs are combs where each $H \cap T_i$ consists of one node. It was shown by Edmonds and Johnson that the solution set of (i) the simple bound inequalities ($0 \leq x \leq 1$), (ii) the degree constraints and (iii) the comb inequalities for which each tooth has cardinality two is precisely the convex hull of the incidence vectors of 2-matchings in G. This **2-matching polytope** is actually an interesting relaxation of P_{TSP}.

The comb inequalities may be generalized into the so-called clique-tree inequalities where more handles are allowed and these are organized in a tree-like fashion, see [18]. Furthermore, the clique inequalities is a subset of the bipartition inequalities; other inequalities are called star inequalities, binested inequalities and so on. In fact, a lot of work has been done on understanding the facial structure of Traveling Salesman Polytopes and a main goal is to get a unifying understanding of the structure, not just many strange classes of inequalities. A recent survey in this area is given in [28].

We now leave the presentation of valid inequalities and discuss briefly how these are useful in cutting plane algorithms for solving TSP's.

It is common to first solve a simple LP relaxation containing the simple bounds and the degree constraints. The optimal solution may violate some 2-connectivity constraint, so we need to separate for these. A very useful fact is that this problem reduces to a set of maximum flow problems so it can be solved in polynomial time. For a given solution $\bar{x}$ we consider the weighted graph with edge weights $\bar{x}_e$ for $e \in E$. Then all the 2-connectivity constraints hold, i.e.,

$$\bar{x}(\delta(W)) \geq 2$$

for node sets W if and only if the minimum cut weight (with weights as just described) is no smaller than 2. This can be checked by solving a minimum cut problem, or equivalenly a maximum flow problem, for each pair of distinct nodes in G. If possible, this algorithm also finds one or more violated inequalities, and then these are added to the LP problem to be solved in the next iteration. We mention that there are faster algorithms for solving the minimum cut problem than using $n(n-1)/2$ max flow calculations, for instance based on the so-called Gomory-Hu tree.

At present no polynomial separation algorithm is known for the comb or clique inequalities. However, for a fixed number of teeth, a polynomial algorithm was found recently. There is also a polynomial algorithm for a special subclass of the comb inequalities where $|H \cap T_i| = 1$ and $|T_i \cap H| = 1$ for each i (assuming that the point to be separated satisfies $0 \leq x \leq 1$ and all the degree constraints). This algorithm solves the so-called minimum odd cut-set problem based on minimum cut calculations. See [31] for a description of this algorithm.

In practice, one uses different heuristics for solving the separation problems approximately (except possibly for the subtour inequalities). This means that one is not guaranteed to find violated inequalities, but those found are of course violated. This is done because it may be difficult to find exact separation algorithms, or they are complicated to implement, or, finally, they may be too slow.

Note that although the number of facets for TSP polytopes is enourmous, we only need a suitable set of m linearly independent valid inequalities to prove the optimality of a certain tour (vertex of P_{TSP}). For instance, in 120 city TSP problem originating from cities in Germany was solved to optimality by a cutting plane algorithm where 13 LP were solved and the final LP contained 36 subtour and 60 comb inequalities, see [16]. The current "world record" was a TSP problem with more than 8100 nodes solved to optimality using a very sophisticated cutting plane algorithm (running i parallel on about 50 computers).

6.8 Exercises

Problem 6.1 *Give a proof of Proposition 6.1.*

Problem 6.2 *Consider the knapsack problem max* $\{\sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_j x_j \leq b,\ 0 \leq x \leq 1\}$ *where all the data are positive integers. Define the knapsack relaxation polyope by* $P = \{x \in \mathbb{R}^n : \sum_{j=1}^{n} a_j x_j \leq b,\ 0 \leq x \leq 1\}$. *Assume that the variables have been ordered such that* $c_1/a_1 \geq c_2/a_2 \geq \ldots \geq c_n/a_n$. *Try to guess an optimal solution and prove the optimality by considering the dual problem. Use this result to characterize all the vertices of* P. *What about the cover inequalities in relation to these vertices?*

Problem 6.3 *Consider the branch-and-bound algorithm. Consider a node* u *in the enumeration tree with* $v(R(u)) > v(Q)$ *where* Q *is the integer program and* $R(u)$ *is the LP relaxation in node* u. *Can we prune node* u*?*

Problem 6.4 *Prove, in detail, that (6.16) is a valid integer linear programming formulation of the TSP problem. Then do the same for the model obtained by replacing the cut inequalities by the subtour inequalities.*

Problem 6.5 *Try to figure out what the odd cover inequalities might be based on the example given for the set covering problem.*

Problem 6.6 *Consider the degree-constrained spanning tree problem. Find a valid integer linear programming formulation of this problem.*

Problem 6.7 *Consider the following problem. We shall decide location of sevice centers among a finite set of possible locations* I. *There is given a (finite) set* J *of customers, and each shall be connected to exactly one service centre. The cost of building a service centre at location* $i \in I$ *is* c_i *and the cost of connecting customer* j *to centre location* i *is* $d_{i,j}$. *The simple plant location problem is to decide in which locations service centres should be built and the connection of customers to centres so as to minimiize the total cost (design + connection cost). Model this problem as an integer linear* *programming problem. Figure out some data for a small example and solve the LP relaxation as well as the ILP on a computer using an optimization package (e.g., CPLEX).*

Problem 6.8 *Consider again the simple plant location problem from the previous problem. Suggest a Lagrangian relaxation algorithm for this problem. Discuss its properties (e.g., integrality).*

Problem 6.9 *Develop some simple heuristics for the simple plant location problem.*

Bibliography

[1] C. Berge. *Graphs and Hypergraphs.* North-Holland, Amsterdam, 1973.

[2] D.P. Bertsekas. *Linear network optimization: algorithms and codes.* MIT-Press, 1991.

[3] G. Chartrand and L. Lesniak. *Graphs and digraphs.* Wadsworth and Brooks, 1986.

[4] V. Chvatal. *Linear programming.* W.H. Freeman and Company, 1983.

[5] W. Cook and P.D. Seymour, editors. *Polyhedral combinatorics.* DIMACS Series in discrete mathematics and theoretical computer science. Americal Mathematical Society, 1989.

[6] G. Cornuejols and A. Sassano. On the 0, 1 facets of the set covering polytope. *Mathematical Programming*, 43:45–55, 1989.

[7] G. Dahl. Directed steiner problems with connectivity constraints. *Discrete Applied Mathematics*, 47:109–128, 1993.

[8] G.B. Dantzig. *Linear programming and extensions.* Princeton, 1963.

[9] G.B. Dantzig, D.R. Fulkerson, and S.M. Johnson. Solution of a large-scale traveling-salesman problem. *Operations Research*, 7:58–66, 1954.

[10] J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1:127–136, 1971.

[11] L. Euler. Solutio problematis ad geometriam situs pertinensis. *Commentarii Academiae Petropolitanae*, 8:128–140, 1736.

[12] M.R. Garey and D.S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness.* W.H. Freeman and company, 1979.

[13] D. Goldfarb and M.J. Todd. Linear programming. In Nemhauser et al., editor, *Optimization*, volume 1 of *Handbooks in Operations Research*, chapter 2, pages 73–170. North-Holland, 1989.

[14] M. Gondran and M. Minoux. *Graphs and algorithms.* Wiley, 1984.

[15] R. Grimaldi. *Discrete and combinatorial mathematics.* Addison Wesley, 1994.

[16] M. Grötschel. On the symmetric traveling salesman problem: solution of a 120-city problem. *Mathematical Programming Study*, 21:61–77, 1980.

[17] M. Grötschel and M.W. Padberg. Polyhedral theory. In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors, *The traveling salesman problem*, chapter 8, pages 251–361. Wiley, 1985.

[18] M. Grötschel and W.R. Pulleyblank. Clique tree inequalities and the symmetric travelling salesman problem. *Mathematics of Operations Research*, 11:537–569, 1986.

[19] J.-B. Hiriart-Urruty and C. Lemarechal. *Convex analysis and minimization algorithms.* Springer, 1993.

[20] K. Hoffman and M. Padberg. Lp-based combinatorial problem solving. *Annals of Operations Research*, 4:145–194, 1985.

[21] E.L. Lawler. *Combinatorial optimization: networks and matroids.* Holt, Rinehart and Winston, 1976.

[22] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors. *The Traveling Salesman Problem.* Wiley, 1985.

[23] L. Lovász and M. D. Plummer. *Matching theory.* North-Holland, 1986.

[24] D.G. Luenberger. *Linear and nonlinear programming.* Addison-Wesley, 1984.

[25] J.J. Jarvis M. Bazaraa and H.D. Sherali. *Linear programming and network flows.* Wiley, 1990.

[26] L. Lovász M. Grötschel and A. Schrijver. *Geometric algorithms and combinatorial optimization.* Springer, 1988.

[27] K. Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.

[28] D. Naddef. Handles and teeth in the symmetric traveling salesman polytope. In W. Cook and P.D. Seymour, editors, *Polyhedral combinatorics*, DIMACS Series in discrete mathematics and theoretical computer science, pages 61–74. Americal Mathematical Society, 1989.

[29] G. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*. Wiley, 1988.

[30] M. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60 – 100, 1991.

[31] M.W. Padberg and M. Grtschel. Polyhedral computations. In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors, *The traveling salesman problem*, chapter 9, pages 307–360. Wiley, 1985.

[32] W.R. Pulleyblank. Polyhedral combinatorics. In Nemhauser et al., editor, *Optimization*, volume 1 of *Handbooks in Operations Research and Management Science*, chapter 5, pages 371–446. North-Holland, 1989.

[33] J. B. Orlin R. K. Ahuja, T. L. Magnanti. *Network flows: theory, algorithms, and applications.* Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[34] R.T. Rockafellar. *Convex analysis.* Princeton, 1970.

[35] A. Schrijver. *Theory of linear and integer programming.* Wiley, Chichester, 1986.

[36] M. Stoer and G. Dahl. A polyhedral approach to multicommodity network design. *Numerische Mathematik*, 68:149–167, 1994.

[37] E. Torgersen. *Comparison of Statistical Experiments.* Cambridge University Press, Cambridge, 1992.

[38] R. Webster. *Convexity.* Oxford University Press, Oxford, 1994.

[39] G. Ziegler. *Lectures on polytopes.* Springer, 1994.

Index